Συγγραφική ομάδα: 23 Φοιτητές
Επιμέλεια: Γ.Παλαιγεωργίου
Τμήμα Μηχανικών Η/Υ, Δικτύων και Τηλεπικοινωνιών - Πανεπιστήμιο Θεσσαλίας

Η συγγραφική ομάδα: Αϊβαλής Στέφανος, Αϊβάτης Χρήστος, Αλεξίου Βασιλική, Βουρονίκου Ασημίνα, Γάκη Στυλιανή, Γιάκα Χρυσούλα, Γιαννόπουλος Μενέλαος, Γιωργαλλίδης Αντρέας, Γκανή Λουίζα, Ζησιού Ελένη, Ζυγούρα Βαΐα, Ιωαννίδης Σταύρος, Κράντα Μαριέτα, Λαμπρινίδη Αργυρώ, Μακέλη Κατερίνα, Παπαχαρισίου Κωνσταντίνος, Πασσάς Βιργίλιος, Σπανέλλης Δημήτριος, Σπανού Μαρία, Στεφανή Άννα, Τσαμαρδού Ηλιάνα, Τσιτσιγιάννης Μανώλης, Χαλκιάς Κωνσταντίνος

Επιμέλεια κειμένου – Συντονισμός έργου: Παλαιγεωργίου Γ.

Τμήμα Μηχανικών Η/Υ, Δικτύων και Τηλεπικοινωνιών
Πανεπιστήμιο Θεσσαλίας
Σεπτέμβριος 2011

Αγαπητοί μαθητές

Το παρόν βιβλίο είναι έργο συλλογικής δουλειάς από φοιτητές του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών & Δικτύων του Πανεπιστημίου Θεσσαλίας, γραμμένο με πολύ μεράκι και κόπο. Έχοντας περάσει κι εμείς από τα σχολικά θρανία έχουμε βιώσει το πόσο κουραστικό και βαρετό μπορεί να γίνει το να μαθαίνουμε συνεχώς καινούργια πράγματα χωρίς στην ουσία να χρησιμοποιούμε την δημιουργικότητά μας. Πόσο μάλλον σε ότι έχει σχέση με τον προγραμματισμό, που σχετίζεται με την επίλυση ενός προβλήματος με τον δικό μας μοναδικό τρόπο!

Όλοι εμείς λοιπόν, οι συγγραφείς του βιβλίου αυτού, σπουδάζοντας πλέον την επιστήμη των Η/Υ θέλουμε να μοιραστούμε την αγάπη μας για τον προγραμματισμό με εσάς με έναν ξεχωριστό τρόπο, όπως θα θέλαμε ίσως οι περισσότεροι από εμάς να τον είχαμε γνωρίσει. Αρχικά ζητάμε από εσάς να πιστέψετε στον εαυτό σας και στις προγραμματιστικές σας ικανότητες! Από κει και πέρα με υπομονή, επιμονή και μεθοδικότητα να είστε σίγουροι ότι στο τέλος του βιβλίου θα έχετε καταφέρει να προγραμματίζετε και εσείς τα δικά σας παιχνίδια! Αφήστε λοιπόν τη δημιουργικότητά σας ελεύθερη και διασκεδάστε με το ταξίδι σας προς την εκμάθηση του προγραμματισμού!

Καλή αρχή!

Αγαπητοί καθηγητές

Το βιβλίο «Δημιουργώ Παιχνίδια στο Kodu» δημιουργήθηκε στα πλαίσια του μαθήματος «Διδακτική της Πληροφορικής ΙΙ» που πραγματοποιήθηκε στο εαρινό εξάμηνο του ακαδημαϊκού έτους 2010 - 2011. Αρχικά η επιλογή του έργου μαζί με τον κ. Παλαιγεωργίου μας γέμισε όλους με χαμόγελα, καλή διάθεση αλλά και άγχος για το αν θα μπορούσαμε να ανταποκριθούμε σε αυτό το δύσκολο έργο. Έχοντας οι περισσότεροι από εμάς μια γεύση από την δυσκολία της διδασκαλίας, μέσα από το μάθημα «Διδακτική της Πληροφορικής Ι» του προηγούμενου εξαμήνου αρχίσαμε δειλά να οργανώνουμε το βιβλίο που διαβάζετε τώρα. Τα πράγματα αποδείχθηκαν όμως πολύ πιο δύσκολα από ότι περιμέναμε.

Πρώτα απ’ όλα έπρεπε να αφήσουμε πίσω μας τον ξύλινο τρόπο διδασκαλίας που είχαμε συνηθίσει μέχρι τώρα ενώ παράλληλα έπρεπε να έρθουμε αντιμέτωποι με τις δικές μας μαθησιακές δυσκολίες, που ομολογουμένως δεν ήταν και λίγες, ώστε να βρούμε τρόπους για να κάνουμε τα πράγματα πιο εύκολα για τους μικρούς μας φίλους. Όλα αυτά έπρεπε να γίνουν μέσα από επιτυχημένο συντονισμό 23 φοιτητών, 23 διαφορετικών προσωπικοτήτων. Όλοι εμείς είμαστε περήφανοι για το βιβλίο που σας παρουσιάζουμε και για αυτό ευελπιστούμε να αποτελέσει ένα σημαντικό βοήθημα στα χέρια σας, με σκοπό να προάγετε μια πιο δημιουργική μορφή της διδασκαλίας του προγραμματισμού στους μαθητές σας. Συγχωρέστε μας για τυχόν αδυναμίες και παραλείψεις μας, καθώς είναι η πρώτη εκδοχή του βιβλίου. Επιζητούμε τις παρατηρήσεις και τις διορθώσεις σας ώστε να το βελτιώσουμε! Ας μη ξεχνάμε άλλωστε ότι προορίζεται για παιδιά τα οποία αξίζουν το καλύτερο!

Καλή συνέχεια στο έργο σας!

Περιεχόμενα

Κεφάλαιο 1. Κάθε αρχή και…εύκολη

1.1 Σκοπός του βιβλίου

Καθημερινά χρησιμοποιούμε διάφορες υπολογιστικές εφαρμογές, όπως κειμενογράφους για να γράφουμε τα κείμενά μας και διαδικτυακούς περιηγητές για να εξερευνούμε το διαδίκτυο. Έχετε αναρωτηθεί πως δημιουργούνται οι εφαρμογές αυτές; Θα θέλατε να δημιουργήσετε τις δικές σας εφαρμογές; Σκοπός αυτού του βιβλίου είναι να σας εισάγει στον κόσμο του προγραμματισμού, ένα κόσμο που θα μας επιτρέψει να δημιουργούμε τα δικά μας προγράμματα με ένα, όμως, ιδιαιτέρως διασκεδαστικό και δημιουργικό τρόπο. Πως θα σας φαινόταν αν αντί για απλοί χρήστες ενός παιχνιδιού μπορούσατε να γίνετε οι κατασκευαστές του; Θα θέλατε να μπορείτε να προσδιορίσετε τη συμπεριφορά των ηρώων σας, πότε πυροβολούν, πότε χάνουν, ποια εμπόδια πρέπει να αποφύγουν, ποιους εχθρούς μπορούν να εξολοθρεύσουν ή πόσα νομίσματα πρέπει να συλλέξουν για να κερδίσουν;

Θα συζητήσουμε για όλα αυτά μέσα από τη χρήση της εφαρμογής προγραμματισμού MSKodu, ενός περιβάλλοντος προγραμματισμού που προσφέρετε δωρεάν από τα ερευνητικά εργαστήρια της Microsoft. Ένα περιβάλλον που δημιουργήθηκε για να εμπνεύσει μικρούς και μεγάλους ώστε να δημιουργήσουν εύκολα και γρήγορα συναρπαστικά παιχνίδια! Ο κεντρικός ήρωας, που θα κληθείτε να οδηγήσετε σε συναρπαστικές περιπέτειες, είναι ο Kodu που φαίνεται στη διπλανή εικόνα.

Στο τέλος της μελέτης του βιβλίου μας, θα είστε σε θέση να προγραμματίζετε τον ήρωα μας ώστε με τη βοήθεια του χρήστη να ανατινάζει κάστρα, να πυροβολεί τους αντιπάλους του, να τρώει μήλα, να κολυμπά, να κερδίζει πόντους και ζωές. Θα μπορείτε να φτιάξετε ένα παιχνίδι στο οποίο ο παίκτης θα παίζει ποδόσφαιρο σε ένα γήπεδο και θα προσπαθεί να σκοράρει ή ακόμη θα προσπαθεί να σώσει την αγαπημένη του Kodula περνώντας μέσα από χαράδρες με λάβα και ψηλά βουνά.

Καλωσορίσατε στον μαγικό κόσμο του προγραμματισμού με το MSKodu! Σας ευχόμαστε καλή διασκέδαση!

1.2 Πρόγραμμα και προγραμματισμός

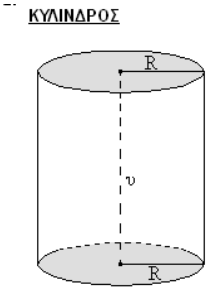
Τι είναι όμως το πρόγραμμα; Ας δούμε πως το ορίζει η βικιπαίδεια:

Στην επιστήμη υπολογιστών με τον όρο πρόγραμμα αναφερόμαστε σε μια συγκεκριμένη ακολουθία εντολών τις οποίες πρέπει να εκτελέσει ένας υπολογιστής για να παραγάγει το επιθυμητό για το χρήστη αποτέλεσμα.

Σας μπέρδεψε; Ας κάνουμε τα πράγματα πιο απλά: **Πρόγραμμα** είναι η ακολουθία των βημάτων που πρέπει να γίνουν για να λυθεί ένα πρόβλημα. Δυο είναι επομένως τα βασικά μας στοιχεία, **το πρόβλημα** και **ο τρόπος επίλυσής του**.

Τι προβλήματα συναντάμε στον κόσμο των υπολογιστών; Θέλουμε να παίζουμε παιχνίδια, να γράφουμε κείμενα, να επισκεπτόμαστε ιστοσελίδες στο διαδίκτυο, να κάνουμε σύνθετες πράξεις για να προβλέψουμε τον καιρό. **Προγραμματισμός**, λοιπόν, είναι η διαδικασία με την οποία προσπαθούμε να δώσουμε τις κατάλληλες οδηγίες στον υπολογιστή για το πώς θα λύσει τα συγκεκριμένα προβλήματα ενώ **προγράμματα** είναι τα τελικά σύνολα των εντολών που εμείς ως προγραμματιστές καταλήξαμε ότι λύνουν τα προβλήματά μας.

Όμως ο προγραμματισμός είναι μια γενική έννοια που δεν συναντάμε μόνο στους υπολογιστές αλλά και στην καθημερινή μας ζωή. Δημιουργούμε προγράμματα για να λύνουμε προβλήματα της καθημερινότητα μας.



Εσείς στο σχολείο καλείστε κυρίως να λύσετε μαθηματικά προβλήματα που περιλαμβάνουν υπολογισμούς και λογικές πράξεις, όπως π.χ. ο υπολογισμός του όγκου ενός κυλίνδρου. Πώς θα επιλύατε ένα αντίστοιχο πρόβλημα; Θα ακολουθούσατε μια διαδικασία δυο βημάτων.

Το πρώτο βήμα θα ήταν να υπολογίσετε το εμβαδόν της βάσης του κυλίνδρου που βρίσκεται από τον τύπο

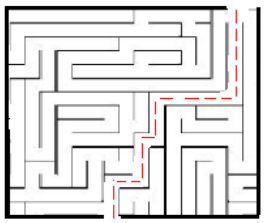
$$\pi \cdot r^2$$

Το δεύτερο βήμα σας θα ήταν να πολλαπλασιάσετε το εμβαδόν που βρήκατε με το ύψος του κυλίνδρου ώστε να έχετε ως αποτέλεσμα τον όγκο του κυλίνδρου.

$$V = E_\beta \cdot \upsilon = \pi R^2 \upsilon$$

Ένα άλλο πρόβλημα είναι και ο τρόπος που θα μπει μια τάξη 130 μαθητών σε λεωφορεία χωρητικότητας 50 ατόμων για να πάει εκδρομή με το σχολείο. Πως θα λύνατε αυτό το πρόβλημα; Θα ξεκινούσαμε υπολογίζοντας ποσά λεωφορεία θα χρειαστούν. Αφού το κάθε λεωφορείο χωράει 50 μαθητές και η τάξη έχει 130 μαθητές θα χρειαστούν 3 λεωφορεία για να σας μεταφέρουν. Δεύτερο βήμα μας θα είναι να βάλουμε τους μαθητές ισομερώς στα λεωφορεία. Έτσι διαιρούμε τον αριθμό των μαθητών με τον αριθμό των λεωφορείων που υπολογίσαμε στο πρώτο βήμα. Επειδή προκύπτει κάτι πάνω από 43, βάζουμε τους 43 πρώτους μαθητές στο πρώτο λεωφορείο. Έπειτα βάζουμε τους επόμενους 43 στο δεύτερο λεωφορείο και τέλος τους τελευταίους 44 μαθητές που περισσεύουν στο τρίτο λεωφορείο.

Για να λύσουμε τα παραπάνω προβλήματα ακολουθήσαμε μια σειρά από βήματα. Ο τρόπος που σκεπτόμαστε για να λύσουμε ένα πρόβλημα ονομάζεται αλγόριθμος. Κάθε πρόγραμμα "κρύβει" από πίσω του έναν αλγόριθμο. Όπως ίσως ήδη έχετε αντιληφθεί για να λύσουμε το πρόβλημα και να χτίσουμε σωστά τον αλγόριθμο πρέπει να κατανοήσουμε πλήρως το πρόβλημα, το περιβάλλον του και τις προϋποθέσεις που μας δίνονται.



Για να δούμε ακόμη ένα πρόβλημα, το πρόβλημα του λαβυρίνθου. Το ζητούμενο είναι ο ιππότης να βγει από τον λαβύρινθο και να σώσει την πριγκίπισσα. Για να λύσουμε το πρόβλημα έχουμε ως διαθέσιμες οδηγίες τις επιτρεπτές κινήσεις που μπορεί να κάνει ο ιππότης. Μπορεί να κινηθεί κατά ένα βήμα δεξιά, αριστερά, πάνω και κάτω αλλά δεν μπορεί να διαπερνά τους τοίχους. Πώς θα καθοδηγούσαμε λοιπόν τον ιππότη για να σώσει την αγαπημένη του;

Ένα σύνολο οδηγιών που θα έλυνε επιτυχώς το πρόβλημα είναι το ακόλουθο:

Ιππότη

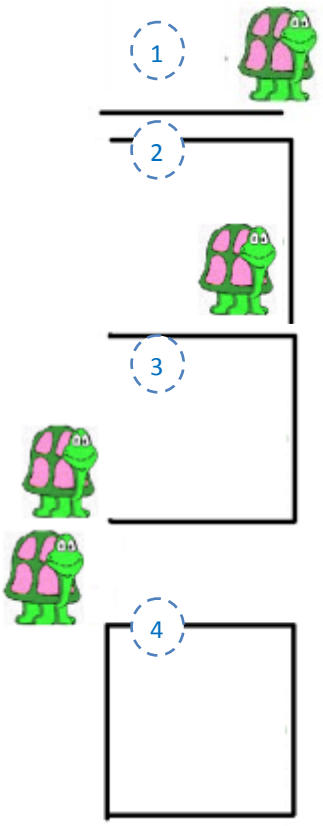
1. Κάνε πέντε βήματα προς τα κάτω
2. Κάνε τέσσερα βήματα προς τα δεξιά
3. Κάνε τέσσερα βήματα προς τα δεξιά
4. Κάνε ένα βήμα προς τα δεξιά
5. Κάνε δύο βήματα προς τα κάτω
6. Κάνε δύο βήματα προς τα δεξιά
7. Κάνε δύο βήματα προς τα κάτω

Να η πριγκίπισσά σου!

Ας δούμε άλλο ένα παράδειγμα. Έχουμε μια χελωνίτσα που αγαπά την ζωγραφική:

Ποιες οδηγίες πρέπει να της δώσουμε, για να ζωγραφίσει ένα τετράγωνο; Έστω ότι οι διαθέσιμες οδηγίες που μπορούμε να χρησιμοποιήσουμε είναι το σύνολο των κινήσεων που μπορεί να κάνει

- δεξιά, αριστερά, πάνω, κάτω - και ότι δεν μπορούμε να σηκώσουμε το μολύβι πριν τελειώσει το τετράγωνο.

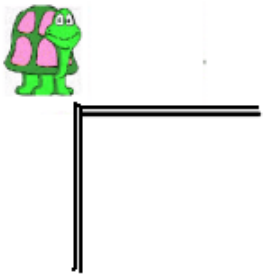


Οι οδηγίες θα μπορούσαν να είναι οι εξής :

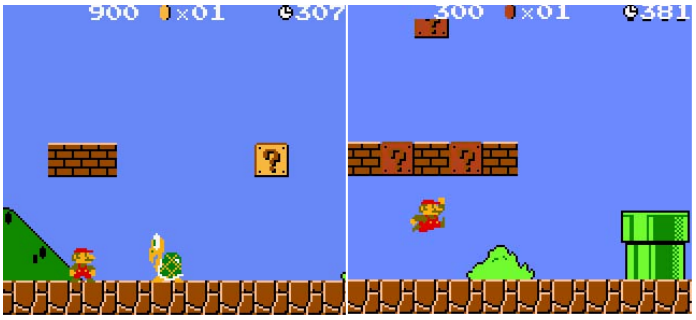
1. Τράβηξε μία γραμμή προς τα δεξιά
2. Τράβηξε μια ίδιου μήκους γραμμή προς τα κάτω
3. Τράβηξε μία ίδιου μήκους γραμμή προς τα αριστερά
4. Τράβηξε μία όμοια γραμμή προς τα πάνω

Τα βήματα που αποτελούν ένα πρόγραμμα λέγονται **εντολές-οδηγίες**. Το κλειδί λοιπόν για την λύση αυτών των προβλημάτων είναι η σωστή διατύπωση των *εντολών-οδηγιών*.

Οι εντολές αυτές πρέπει να ακολουθούνε μια λογική σειρά. Σκεφτείτε, τι θα συνέβαινε αν λέγαμε στην χελώνα να κάνει πρώτα το βήμα 3 και έπειτα το βήμα 2. Θα ζωγράφιζε τελικά η χελώνα το τετράγωνο; Προφανώς όχι. Θα προέκυπτε κάτι τέτοιο:



Σκεφτείτε τώρα τι συμβαίνει όταν παίζετε το ηλεκτρονικό παιχνίδι *Super Mario*. Εσείς πατάτε πλήκτρα, ο υπολογιστής αντιστοιχίζει τα πλήκτρα με τις εντολές του προγράμματος και κινεί ανάλογα τον Mario. Πατώντας το δεξί βελάκι, ο Mario προχωράει προς τα δεξιά, με το πλήκτρο κενό (space) χοροπηδάει, με το αριστερό βελάκι γυρίζει πίσω κλπ. Η κίνηση του Mario στον υπολογιστή είναι ένα πρόβλημα που καλείται να λύσει κάποιος προγραμματιστής.



Αυτό που πρέπει να γίνει απολύτως κατανοητό είναι ότι ο υπολογιστής δεν έχει νοημοσύνη. Οι εντολές που χρησιμοποιεί έχουν γραφτεί από κάποιον προγραμματιστή. Έτσι για κάθε πρόβλημα ο υπολογιστής εκτελεί μια σειρά εντολών που θα οδηγήσουν στην επίλυση του.

Φυσικά δεν είναι μόνο τα παιχνίδια, που η υλοποίησή τους βασίζεται πάνω σε αυτή την λογική. Ο διαδικτυακός περιηγητής που σίγουρα έχετε χρησιμοποιήσει για να επισκέπτεστε διάφορες ιστοσελίδες, το πρόγραμμα του ηλεκτρονικού ταχυδρομείου με το οποίο στέλνετε email σε έναν φίλο σας είναι μερικά από τα προγράμματα τα οποία δημιουργήθηκαν από τους προγραμματιστές. Τα προγράμματα αυτά μπορεί να σας φαίνονται πολύ δύσκολο για να δημιουργηθούν, αλλά η λογική που χρησιμοποιήθηκε για να κατασκευαστούν είναι παραπλήσια με τα προβλήματα που αναφέραμε προηγουμένως. Οι προγραμματιστές ανέλυσαν τα προβλήματα σε μικρότερα υποπροβλήματα, έφτιαξαν αλγόριθμους για να βρουν λύσεις σε αυτά και δημιούργησαν τις κατάλληλες ακολουθίες εντολών. Τα προγράμματα είναι μέσα στην ζωή μας ακόμα και αν δεν το καταλαβαίνουμε! Η διαδικασία σύνταξης προγραμμάτων ονομάζεται **προγραμματισμός**.

Ο προγραμματισμός είναι μια ταχύτατα εξελισσόμενη επιστήμη που μέσα από τα διάφορα επιτεύγματα που έχουν επιτύχει οι επαγγελματίες του κλάδου - οι προγραμματιστές - έχει συμβάλει κατά πολύ στην τεχνολογική πρόοδο.

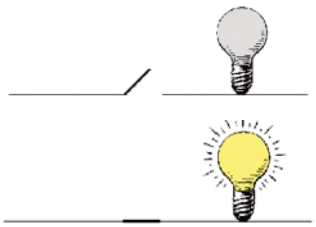
1.3 Πως επικοινωνούμε με τους υπολογιστές;

Και ποιες είναι οι οδηγίες-εντολές που καταλαβαίνει ένας υπολογιστής; Πίσω από όλα τα προηγούμενα παραδείγματα κρύβεται μια **γλώσσα προγραμματισμού**, δηλαδή ένας τρόπος με τον οποίο μιλάμε στο υπολογιστή και του διατυπώνουμε μια σειρά εντολών. Προηγουμένως, η γλώσσα προγραμματισμού της χελώνας περιλάμβανε εντολές οι οποίες μας επέτρεπαν να κινήσουμε την χελώνα δεξιά, αριστερά, πάνω, κάτω, ενώ στο παράδειγμα με τον ιππότη, ο ήρωάς μας ακολουθούσε μια παραπλήσια σειρά εντολών.

Διαλέγοντας το τελευταίο παράδειγμα, τι θα γινόταν στην περίπτωση που στη θέση του ιππότη είχαμε έναν υπολογιστή; Με ποιον τρόπο καταλαβαίνει ένας υπολογιστής την σειρά εντολών "Κάνε πέντε βήματα προς τα κάτω", "Κάνε τέσσερα βήματα προς τα δεξιά" κτλ που δώσαμε στον ιππότη; Και σίγουρα θα προβληματίζεστε για το πως είναι δυνατό να αντιληφθεί μια σειρά βημάτων ένας υπολογιστής αφού δεν έχει ούτε αυτιά για να ακούσει τις εντολές, ούτε μάτια για να δει ένα σχέδιο του λαβυρίνθου με την διαδρομή που θα πρέπει να ακολουθήσει;

Η απάντηση σε όλα τα παραπάνω ερωτήματα είναι πως ναι, ο υπολογιστής μπορεί να καταλάβει μια σειρά ενεργειών που θέλουμε να εκτελέσει. Αρκεί να του μιλήσουμε στην δική του γλώσσα, την μοναδική γλώσσα που καταλαβαίνει και ονομάζεται γλώσσα μηχανής. Ας μάθουμε λοιπόν, κάτι παραπάνω για την γλώσσα μηχανής.

Ανατρέχοντας στην ιστορία των υπολογιστών, βλέπουμε ότι οι πρώτοι υπολογιστές (από την δεκαετία του 1940) δημιουργήθηκαν, για να εκτελούν αριθμητικές πράξεις. Ωστόσο, οι κατασκευαστές της εποχής εκείνης ήρθαν αντιμέτωποι με ένα μεγάλο πρόβλημα. Τα αριθμητικά ψηφία (0, 1, 2, 3, 4, 5, 6, 7, 8 και 9) του δεκαδικού συστήματος ήταν πολλά για να απεικονιστούν με ψηφιακό τρόπο και η κατασκευή ενός τέτοιου υπολογιστή που βασιζόταν σε όλα αυτά τα ψηφία ήταν εξαιρετικά πολύπλοκη. Η λύση ήρθε με την χρήση ενός άλλου συστήματος αρίθμησης, πολύ απλούστερου, του δυαδικού, με το οποίο όλοι οι αριθμοί μπορούσαν να αναπαρασταθούν με τον συνδυασμό δύο καταστάσεων του 0 και 1. Οι δύο αυτές καταστάσεις διευκόλυναν τους κατασκευαστές γιατί τους έδωσε την δυνατότητα να αντιστοιχίσουν:



- την απουσία ρεύματος (που περνάει μέσα από τα κυκλώματα του υπολογιστή) με : **0**
- την παρουσία ρεύματος (που περνάει μέσα από τα κυκλώματα του υπολογιστή) με : **1**

και έτσι μπόρεσαν να δημιουργήσουν μια γλώσσα που είχε ένα απλό αλφάβητο με δυο μόνο στοιχεία, το 0 και το 1. Αν ήθελαν λοιπόν, να δώσουν μια απλή εντολή στον υπολογιστή π.χ. να προσθέσει το 5 + 6 και να εμφανίσει το αποτέλεσμα, έπρεπε να μετατρέψουν αυτήν την εντολή, σε μια γραμμή από 0 και 1. Η γλώσσα αυτή ονομάστηκε **γλώσσα μηχανής**. Ήταν όμως εξαιρετικά δύσκολη στη χρήση της! Οι προγραμματιστές σκέφτηκαν ότι ίσως θα μπορούσαν να δημιουργήσουν γλώσσες προγραμματισμού που είναι πιο φιλικές προς αυτούς, δηλαδή γλώσσες που περιέχουν εντολές-οδηγίες που είναι περισσότερο κατανοητές στον ανθρώπινο νου, ενώ στη συνέχεια οι εντολές αυτές θα μπορούσαν να «μεταφράζονται» σε γλώσσα μηχανής.

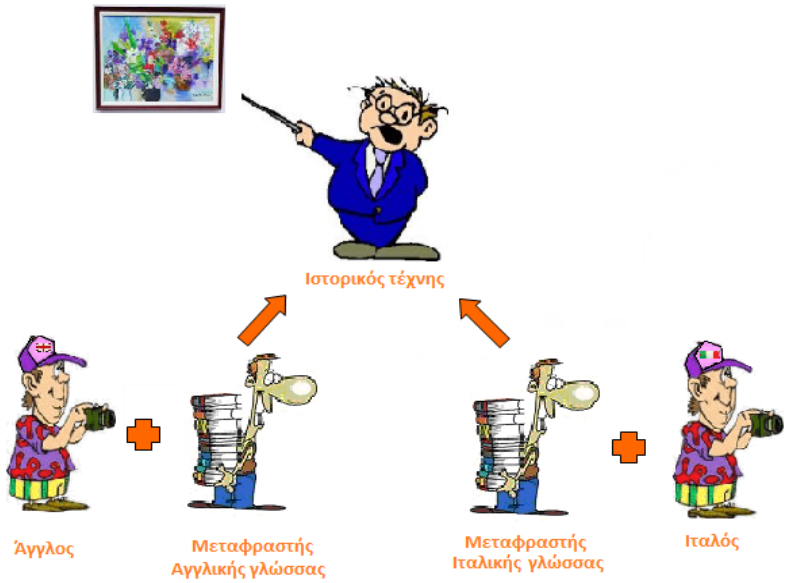
Έτσι λοιπόν, οι γλώσσες προγραμματισμού λειτουργούν όπως οι φυσικές γλώσσες (Ελληνικά, Αγγλικά, κτλ) που χρησιμοποιούν οι άνθρωποι για να επικοινωνούν μεταξύ τους. Κάθε μία έχει διαφορετικό λεξιλόγιο και συντακτικό αλλά όλες κατασκευάστηκαν για να ικανοποιούν τον ίδιο στόχο, δηλαδή την πιο "ανθρώπινη" επικοινωνία με τον υπολογιστή. Φυσικά τώρα αναρωτιέστε, πόσες γλώσσες υπάρχουν; Μια δεν είναι αρκετή; Ποια η διαφορά της μιας γλώσσας από την άλλη;

Σε αυτή την φάση δεν χρειάζεται να αγχώνεστε με όλα τα παραπάνω. Αρκεί να αναφέρουμε ότι υπάρχουν πολλές διαφορετικές γλώσσες και ότι ο λόγος ύπαρξής τους είναι ότι καμία δεν μπορεί να ικανοποιήσει τις πολύ διαφορετικές απαιτήσεις των ανθρώπων αλλά και τα ιδιαίτερα χαρακτηριστικά των διαφορετικών προβλημάτων. Κάθε μια από αυτές δημιουργήθηκε για να αντιμετωπίζει διαφορετικά προβλήματα (π.χ. μαθηματικά προβλήματα ή παιχνίδια με γραφικά) και καταλαβαίνετε πως όσο αναπτύσσεται η επιστήμη των υπολογιστών και αυξάνονται οι ανάγκες μας, τόσο θα κάνουν την εμφάνιση τους νέες γλώσσες προγραμματισμού.

Ήδη αναφέραμε πως οι γλώσσες προγραμματισμού αναλαμβάνουν να μετατρέψουν τις εντολές που παρέχουν, σε εντολές γλώσσας μηχανής. Την μετατροπή αυτή εκτελεί ένα πρόγραμμα που

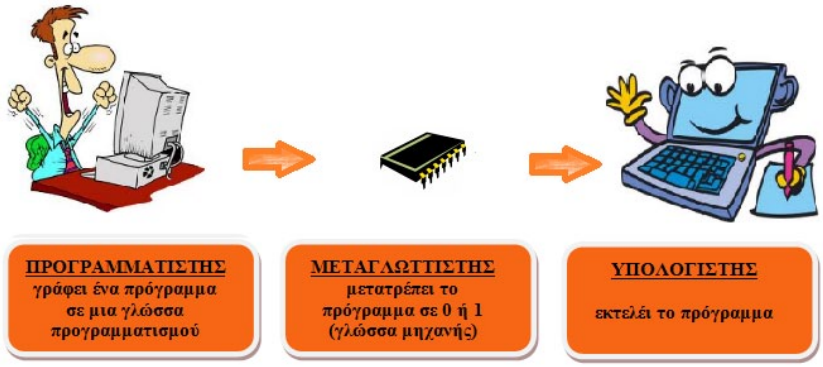
ονομάζεται **μεταγλωττιστής** και την λειτουργία του θα προσπαθήσουμε να εξηγήσουμε με το πιο κάτω παράδειγμα.

Κατά την επίσκεψή σας στην πινακοθήκη της πόλης σας, παρατηρείτε τους διάφορους τουρίστες που έρχονται να θαυμάσουν τα εκθέματα. Στην συνέχεια οι τουρίστες θέλουν να μάθουν περισσότερα για την ιστορία του κάθε εκθέματος και να θέσουν κάποια ερωτήματα στον ιστορικό τέχνης του μουσείου.



Κανένας από αυτούς δεν γνωρίζει την ελληνική γλώσσα, ωστόσο όλα τα γκρουπ των επισκεπτών έχουν μαζί τους από έναν ξεναγό ο οποίος μεταφέρει τα ερωτήματα στον ιστορικό τέχνης. Ο ξεναγός για παράδειγμα, από ένα γκρουπ Άγγλων θα μεταφράσει τα ερωτήματα από τα αγγλικά στα ελληνικά, ο Ιταλός ξεναγός θα μεταφράσει τα ερωτήματα από τα ιταλικά στα ελληνικά κ.ά.

Κατά ανάλογο τρόπο θα μπορούσαμε να πούμε ότι, όπως λειτουργούν οι διαφορετικές φυσικές γλώσσες (Αγγλικά, Ιταλικά, κτλ) που χρησιμοποιούν οι τουρίστες για να επικοινωνούν μεταξύ τους και να διατυπώνουν τα ερωτήματά τους, έτσι λειτουργούν και οι γλώσσες προγραμματισμού για να μπορέσουν οι άνθρωποι να επικοινωνήσουν με τον υπολογιστή. Όπως κάθε γκρουπ έχει τον δικό του ξεναγό, έτσι και κάθε γλώσσα προγραμματισμού έχει τον δικό της μεταγλωττιστή που μετατρέπει τις εντολές της γλώσσας προγραμματισμού σε εντολές της γλώσσας μηχανής. Τέλος, όπως ο ιστορικός τέχνης καταλαβαίνει μόνο την δική του γλώσσα και καμία άλλη, έτσι και ο υπολογιστής καταλαβαίνει μόνο τη γλώσσα μηχανής.



1.4 Γιατί να μάθω προγραμματισμό;

Για ποιον λόγο να μάθουμε προγραμματισμό; Έχουμε να κερδίσουμε κάτι ή είναι ένα αντικείμενο με το οποίο ασχολούνται μόνο ειδικοί στους υπολογιστές; Καταρχάς να αναφέρουμε πως εμείς με την βοήθεια του MSKodu θα μάθουμε προγραμματισμό για να κατασκευάζουμε τα δικά μας παιχνίδια! Άρα προβλέπεται *πολύ διασκέδαση*!

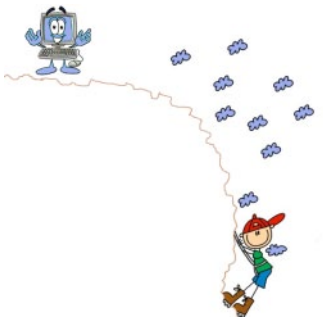
Επιπλέον θα ικανοποιήσετε την *περιέργεια* που έχετε για τον τρόπο δημιουργίας ενός παιχνιδιού και παράλληλα θα νιώσετε χαρά δημιουργώντας τα δικά σας έργα. Πέρα όμως από το να

κατασκευάζετε παιχνίδια, μπορείτε *να σκεφτείτε τον προγραμματισμό σαν ένα λευκό καμβά* πάνω στον οποίο θα μπορείτε να ζωγραφίσετε ότι θέλετε. Δηλαδή, θα είσαστε σε θέση να κατασκευάσετε ότι προγράμματα θέλετε χωρίς κανείς να περιορίσει τη δημιουργικότητά σας.

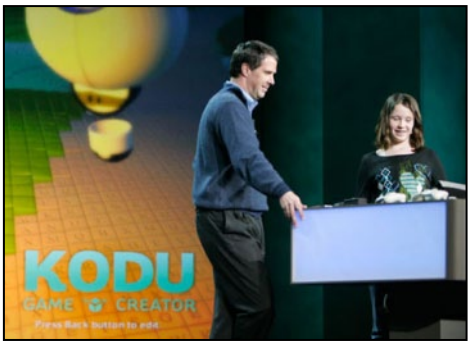
Χρησιμοποιώντας γλώσσες προγραμματισμού, θα μπορείτε να φτιάξετε, για παράδειγμα, ένα ηλεκτρονικό ημερολόγιο που θα κρατάει τα γενέθλια και τις γιορτές των φίλων σας και θα σας ενημερώνει όταν πλησιάζει κάθε μία. Θα μπορείτε να φτιάξετε ένα πρόγραμμα στο οποίο να εισάγετε τους βαθμούς των μαθημάτων σας και να σας υπολογίζει το μέσο όρο. Θα μπορούσατε να κατασκευάσετε, ακόμη και έναν ηλεκτρονικό διαχειριστή των οικονομικών σας, με την βοήθεια του οποίου να διαχειρίζεστε καλύτερα το εβδομαδιαίο χαρτζιλίκι σας. Ο προγραμματισμός λοιπόν μπορεί να *διευκολύνει* αλλά και να δώσει μια ευχάριστη νότα στην καθημερινότητά σας.

Ακόμη, θα κατανοήσετε πως όλες αυτές οι εφαρμογές με τις οποίες ασχολούμαστε καθημερινά δε δουλεύουν με ένα μαγικό τρόπο, αλλά αξιοποιούν απλές εντολές που καθορίζουν τη συμπεριφορά τους. Θα μπορέσετε για παράδειγμα να καταλάβετε το σκεπτικό ενός προγραμματιστή όταν γράφετε μια έκθεση στον κειμενογράφο, ή όταν ζωγραφίζετε ένα χαμογελαστό πρόσωπο στο πρόγραμμα ζωγραφικής. Κατανοώντας τον προγραμματισμό *θα σταματήσετε να αντιμετωπίζετε την τεχνολογία των υπολογιστών ως κάτι ακαταλαβίστικο* και *ίσως επικίνδυνο*. Θα χρησιμοποιείτε τους υπολογιστές με μεγαλύτερη αυτοπεποίθηση, ενώ κατασκευάζοντας τα δικά σας προγράμματα θα γίνετε *συν-δημιουργοί του μέλλοντος*, αφού θα διαπιστώσετε πως εσείς οι ίδιοι μπορείτε να δημιουργείτε τεχνολογία. Μην θεωρείτε πως είμαστε υπερβολικοί! Πολλοί συμμαθητές από διάφορα μέρη του πλανήτη κατάφεραν να κατασκευάσουν καταπληκτικά προγράμματα τα οποία εντυπωσίασαν ακόμα και τους πιο έμπειρους προγραμματιστές.

Φωτεινό παράδειγμα είναι η 12χρονη Sparrow. Το κορίτσι αυτό επιλέχθηκε από την ίδια την Microsoft, την εταιρία που δημιούργησε το MSKodu, για να παρουσιάσει το περιβάλλον του MSKodu στο κοινό. Η Sparrow κατά την επίδειξη δημιούργησε ένα τόσο συναρπαστικό παιχνίδι το οποίο δεν θαύμασε μόνο το κοινό αλλά και πολλοί καταξιωμένοι προγραμματιστές της Microsoft.



Αν θέλετε να δείτε όλη την επίδειξη, με την Sparrow να επιδεικνύει το παιχνίδι μπορείτε, ακολουθώντας τον σύνδεσμο: Επίδειξη του MSKodu. Στην εικόνα φαίνεται η 12χρονη Sparrow με τον προγραμματιστή της Microsoft, Robbie Bach.



Όμως τα οφέλη από την εκμάθηση του προγραμματισμού δεν σταματούν εδώ. Με τον προγραμματισμό *αναπτύσσεστε ακόμη και πνευματικά*. Αρκετές μελέτες έχουν δείξει ότι άτομα τα οποία έχουν μάθει να σκέφτονται με προγραμματιστικό τρόπο, έχουν την ικανότητα να λύνουν τα προβλήματα της καθημερινότητας πιο εύκολα, πιο γρήγορα, πιο αποδοτικά! Μαθαίνοντας να σκέφτεστε με αυτόν τον τρόπο, προβλέπετε καλύτερα τα αποτελέσματα που θα έχει ένας τρόπος επίλυσης ενός προβλήματος αλλά και τις εναλλακτικές επιλογές που πρέπει να ακολουθήσετε, σε περίπτωση που το αρχικό σχέδιο αποτύχει. Θετικές επιδράσεις του προγραμματισμού έχουν βρεθεί ακόμη και στον τομέα των μαθηματικών! Φανταζόμαστε ότι όλοι θέλετε να βελτιώσετε *την ταχύτητα και την αποτελεσματικότητα της σκέψης σας*.

Τέλος, το συναίσθημα που θα νιώσετε όταν το πρόγραμμά που έχετε δημιουργήσει λειτουργεί και παράγει τα σωστά αποτελέσματα, είναι κάτι που πραγματικά δεν συγκρίνεται! Είναι το συναίσθημα του δημιουργού, είναι το συναίσθημα του ορειβάτη που μόλις κατέκτησε την κορυφή. Θα σας γεμίσει αυτοπεποίθηση και χαρά.

Τι λέτε λοιπόν θέλετε να αρχίσουμε σιγά-σιγά την αναρρίχηση του προγραμματισμού;

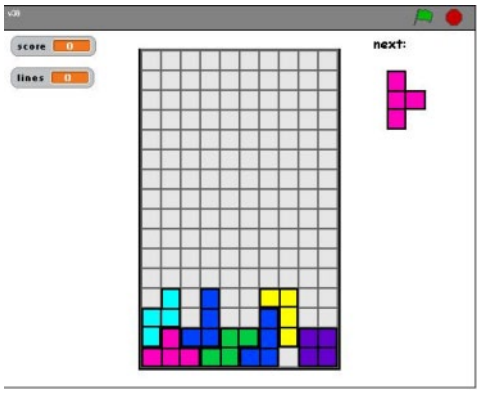
1.5 Προγραμματισμός και παιχνίδια – Πολλά περιβάλλοντα

Ήδη έχουμε αναφέρει ότι σκοπός αυτού του βιβλίου είναι να σας μάθει προγραμματισμό δημιουργώντας τα δικά σας παιχνίδια με τη βοήθεια του μικρού ήρωα του βιβλίου, του Kodu.

Εκτός όμως από τον Kodu θα μάθουμε και για άλλους χαρακτήρες, όπως τον *Μηχανάκια*, το *Υποβρύχιο*, το *Χελιδονόψαρο*, το *Κανόνι*, την *Χελώνα*. Θα έχουμε την δυνατότητα να προγραμματίσουμε την συμπεριφορά αυτών των χαρακτήρων επιλέγοντας να κάνουν διάφορες ενέργειες όπως, να πυροβολούν, να τρώνε μήλα, να ακούν ήχους, να κουβαλούν διάφορα αντικείμενα και να τα μετακινούν σε νέες θέσεις.

Το MSKodu δεν είναι το μόνο περιβάλλον με το οποίο μπορούμε να μάθουμε προγραμματισμό με διασκεδαστικό τρόπο. Εμάς μας ενθουσίασε και σας το παρουσιάζουμε. Σας προτρέπουμε όμως να δείτε και άλλα διαφορετικά προγραμματιστικά περιβάλλοντα που μας επιτρέπουν να κατασκευάσουμε τα δικά μας παιχνίδια, να κατασκευάζουμε ρομπότ, να παρουσιάσουμε ιστορίες και να δημιουργούμε ζωγραφιές. Ας δούμε συνοπτικά ορισμένα.

1.5.1 Scratch



Μέσα από το περιβάλλον του Scratch μπορείτε να φτιάξετε τις δικές σας ιστορίες και παιχνίδια εύκολα και γρήγορα ενώ θα κάνετε παράλληλα μια εισαγωγή στις βασικές έννοιες του προγραμματισμού! Με το Scratch μπορείτε να φτιάξετε το δικό σας pacman, το δικό σας tetris (δεξιά εικόνα) ή τον μοναδικό σας βυθό με μικρά και μεγάλα ψάρια. Με το Scratch δημιουργείτε πολύ εύκολα όλα αυτά τα παιχνίδια μιας και οι εντολές του απεικονίζονται με μικρά τουβλάκια. Αυτά τα τουβλάκια πρέπει να συνθέσετε μεταξύ τους ώστε να δημιουργήσετε όποιο πρόγραμμα έχετε φανταστεί. Κάθε νέο κόσμο που δημιουργείτε έχετε την δυνατότητα να τον βλέπετε άμεσα, να τον τροποποιείτε και να πειραματίζεστε πάνω σε αυτόν.

[http://scratch.mit.edu/][1]

1.5.2 Alice 3

Με το περιβάλλον Alice μπορείτε να φτιάξετε εικονικούς κόσμους τριών διαστάσεων μέσα στους οποίους δραστηριοποιούνται αντικείμενα που έχετε δημιουργήσει. Τα αντικείμενα αυτά μπορεί να είναι εξωγήινοι ή διαστημόπλοια, δράκοι ή δεινόσαυροι, ιππότες ή νεράιδες. Αν επιλέξετε το αντικείμενο που θα δημιουργήσετε να είναι άνθρωπος, τότε μπορείτε να του δώσετε χαρακτηριστικά όπως φύλο, χρώμα του δέρματος, ύψος, χρώμα των μαλλιών ή ματιών, να διαλέξετε ρούχα κτλ. Μπορείτε να δημιουργήσετε μια κοπέλα με καστανά μαλλιά και κορμάκι να κάνει πατινάζ σε μια παγωμένη λίμνη, έναν αστροναύτη να προσεληνώνεται ή να φτιάξετε έναν κόσμο με ιππότες, δράκους και μάγους όπως φαίνεται στις παρακάτω εικόνες:

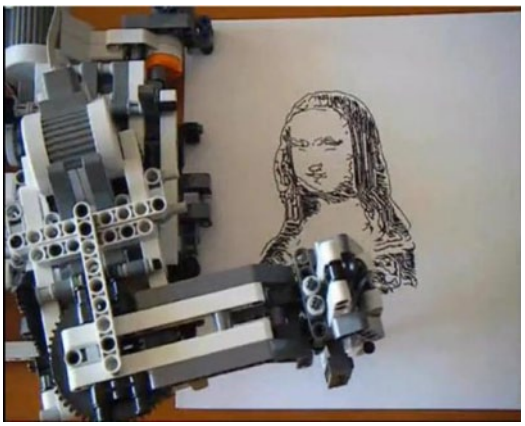
[1] Το αντίστοιχο βιβλίο μας για το Scratch στα ελληνικά, βρίσκεται στη διεύθυνση www.scratchpay.gr

Μπορείτε να δημιουργήσετε ακόμα και το γνωστό παραμύθι του κακού λύκου με τα τρία γουρουνάκια! Παράλληλα με την κατασκευή των κόσμων σας, θα μαθαίνετε προγραμματισμό, αφού έχετε την δυνατότητα να προγραμματίζετε τις ενέργειες που θα κάνουν οι χαρακτήρες σας καθώς και την συμπεριφορά τους όταν έρχονται σε επαφή με άλλους χαρακτήρες.

[http://www.alice.org/]

1.5.3 Lego Mindstorms

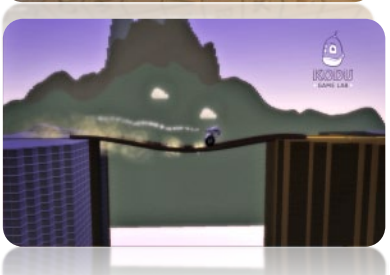
Τα Lego Mindstorms περιέχουν, τα ήδη γνωστά σε όλους, τουβλάκια Lego τα οποία συναρμολογούμε για να δημιουργήσουμε κάστρα, αυτοκίνητα, πόλεις και ότι άλλο προκύπτει από την φαντασία μας. Όμως με τα Lego Mindstorms δεν έχετε μόνο την δυνατότητα να δημιουργήσετε μια κατασκευή, αλλά μπορείτε να προγραμματίσετε την συμπεριφορά που θέλετε να έχει! Για παράδειγμα μπορείτε να κατασκευάσετε ένα σκυλάκι και να ορίσετε την συμπεριφορά του έτσι ώστε όταν ακούει τη φωνή σας να πλησιάζει προς το μέρος σας ή όταν αντιλαμβάνεται μια κίνηση λίγο πιο μακριά, να ανοίγει το στόμα δείχνοντας απειλητικά τα δόντια του. Εσείς, δηλαδή, προγραμματίζετε τη συμπεριφορά για το ρομποτικό σκυλάκι ώστε να έχει την αίσθηση του ήχου καθώς και την αντίληψη της απόστασης. Και φυσικά το σκυλάκι-ρομποτάκι μπορεί να αποκτήσει και άλλες αισθήσεις αν εσείς το επιθυμείτε. Μπορεί να έχει επιπλέον, την αίσθηση της αφής και του φωτός. Πιο σύνθετα παραδείγματα είναι η κατασκευή ρομπότ-ποδοσφαιριστή που προσπαθεί να βάλει γκολ σε ρομπότ- τερματοφύλακα, ή ρομποτάκια που παλεύουν σε μια μεσαιωνική μάχη για να σώσουν την πριγκίπισσα που βρίσκεται στον πύργο, ή ένα ρομποτάκι, σαν άλλος Leonardo da Vinci, να ζωγραφίζει την Μόνα Λίζα. Και όλα αυτά, προγραμματίζοντας με απλές εντολές.



1.6 Kodu και προγραμματισμός

Το MSKodu αρχικά είχε σχεδιαστεί ως εργαλείο μάθησης για τους νέους που χρησιμοποιούν το γνωστό σε εσάς Xbox 360 και κυκλοφόρησε επίσημα στις 7 Ιανουαρίου του 2009. Δύο χρόνια αργότερα, το MSKodu ήταν επιτυχία στο Xbox Live και το χρησιμοποιούσαν σε περισσότερα από 60 εκπαιδευτικά ιδρύματα σε όλη την υφήλιο για να εισάγουν τα παιδιά στον προγραμματισμό. Τώρα όμως είναι διαθέσιμο δωρεάν και για το PC.

Χρησιμοποιώντας τον ήρωα Kodu, μπορείτε να δημιουργήσετε απίθανα παιχνίδια και να προκαλέσετε τους γύρω σας να τα τερματίσουν!



Ας κάνουμε και μια βόλτα στα περιεχόμενα του βιβλίου.

Στο *κεφάλαιο 2* θα μάθετε πώς να εγκαταστήσετε το MSKodu και θα παίξετε τα πρώτα παιχνίδια σας! Στο *κεφάλαιο 3* ξεκινάτε από τόσο νωρίς να φτιάχνετε τα δικά σας παιχνίδια. Στο *κεφάλαιο 4* θα μάθετε να δημιουργείτε τους δικούς σας εντυπωσιακούς. Στα *κεφάλαια 5, 6 και 7* θα μελετήσουμε πως δημιουργούμε αντικείμενα, πως προγραμματίζουμε τη συμπεριφορά τους, πως τους επιτρέπουμε να αλληλεπιδρούν με τον χρήστη, με άλλα αντικείμενα και τον κόσμο του παιχνιδιού μας. Στο *κεφάλαιο 8* θα σας δείξουμε πως τα παιχνίδια σας μπορούν να αποκτήσουν στοιχεία όπως ζωές, σκορ, χρόνο,

ενέργεια. Στο *κεφάλαιο 9* θα συζητήσουμε κάποιες πιο σύνθετες προγραμματιστικές τεχνικές ώστε τα παιχνίδια σας να γίνουν πιο αληθοφανή! Στο *κεφάλαιο 10,* θα κάνουμε ένα μικρό διάλλειμα και θα σας παρουσιάσουμε βασικές αρχές της μηχανικής παιχνιδιών, δηλαδή των σχεδιαστικών επιλογών που πρέπει να λαμβάνουμε υπόψη ώστε να δημιουργούμε προκλητικά παιχνίδια. Τέλος, στα *κεφάλαια 11* και *12* θα εφαρμόσουμε όλες τις γνώσεις των προηγουμένων κεφαλαίων για να δημιουργήσουμε μαζί 2 ολοκληρωμένα παιχνίδια. Θυμηθείτε ότι ο μόνος περιορισμός στον προγραμματισμό είναι η φαντασία σας!

Περίληψη

Στο κεφάλαιο αυτό συζητήσαμε για την έννοια του προγραμματισμού, η οποία όπως αποδείχθηκε δεν είναι τελείως καινούρια καθώς, είτε το αντιλαμβανόμαστε είτε όχι, υπάρχει στην ζωή μας! Μέσα από παραδείγματα, είδαμε πως η καθημερινότητά μας αποτελείται από αλγορίθμους που δημιουργούμε για να λύσουμε τα προβλήματά μας, ενώ κατανοήσαμε περισσότερο την έννοια του προγράμματος τόσο στην ζωή μας όσο και σε σχέση με τους υπολογιστές. Στη συνέχεια μπορέσαμε να δούμε με ποιον τρόπο καταφέρνουμε να δίνουμε οδηγίες στον υπολογιστή και μιλήσαμε για τις γλώσσες προγραμματισμού. Επιπλέον πήραμε απαντήσεις για το τι όφελος έχουμε μαθαίνοντας προγραμματισμό και είδαμε πως στο βιβλίο αυτό, με την βοήθεια του ήρωά μας, του Kodu, θα μάθουμε προγραμματισμό φτιάχνοντας τα δικά μας παιχνίδια, δηλαδή, διασκεδάζοντας! Συζητήσαμε για εναλλακτικά προγραμματιστικά περιβάλλοντα με παρόμοιο χαρακτήρα με το MSKodu και τέλος ρίξαμε μια κλεφτή ματιά στα πιο σημαντικά στοιχεία που θα αναλυθούν στην συνέχεια. Σας προτείνουμε, λοιπόν να διαβάστε τα ακόλουθα κεφάλαια, να γνωρίσετε τις μαγικές δυνατότητες του Microsoft Kodu και να επιδοθείτε στην διερεύνηση της δημιουργικότητάς σας! Καλή μελέτη!

Ερωτήσεις

1. Περιγράψτε τι είναι ένα πρόγραμμα και ποια είναι τα βασικά του χαρακτηριστικά.

2. Ποιες είναι οι διαφορές μεταξύ του αλγορίθμου και του προγράμματος;

3. Ποια πιστεύετε ότι είναι τα οφέλη που μπορεί να σας προσφέρει ο προγραμματισμός στην καθημερινή σας ζωή;

4. Περιγράψτε καθημερινές σας δραστηριότητες στις οποίες χρησιμοποιείτε προγραμματιστικές εφαρμογές.

5. Ποιος πιστεύετε ότι είναι ο λόγος που δημιουργήθηκαν οι γλώσσες προγραμματισμού; Τι κοινό έχουν μεταξύ τους; Αρκεί μία γλώσσα προγραμματισμού;

Δραστηριότητες

1. Περιγράψτε τον αλγόριθμο 7 βημάτων, που θα δίνατε στην μητέρα σας για να φτιάξει το αγαπημένο σας κέικ!

- Στις οδηγίες που θα δώσετε, χρησιμοποιείστε τα παρακάτω: αλεύρι, μείγμα, αυγά, ζάχαρη, ρίξε, ανακάτεψε, βγάλε, σοκολάτα, φόρμα ψησίματος, βάλε, φούρνος, άφησε να ψηθεί 1 ώρα.

- Στην συνέχεια συγκρίνετε τον αλγόριθμό σας με τους αλγορίθμους που κατασκεύασαν οι υπόλοιποι συμμαθητές σας. Τι παρατηρείτε και σε ποιο συμπέρασμα καταλήγετε;

2. Ανακαλύψτε τις κρυμμένες λέξεις: προγραμματισμός, μεταγλώττιση, αλγόριθμος, γλώσσα, εκτέλεση, εντολή παιχνίδι.

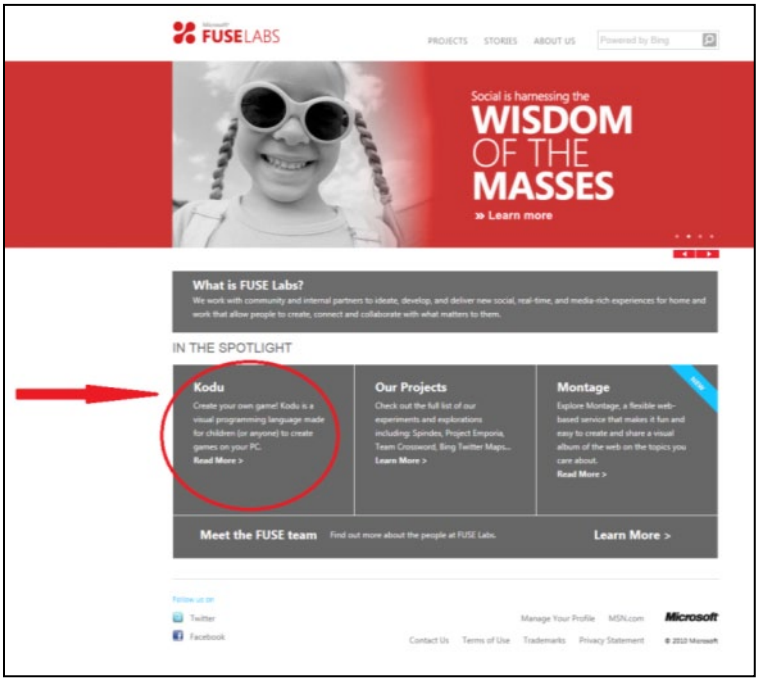
Π	Ω	Α	Ε	Τ	Υ	Γ	Λ	Ω	Σ	Σ	Α	Η	Ξ	Φ
Σ	Ρ	Δ	Ω	Μ	Δ	Ι	Ο	Π	Ρ	Φ	Ε	Ι	Χ	Β
Ψ	Ω	Ο	Α	Φ	Μ	Λ	Φ	Τ	Θ	Ρ	Δ	Κ	Ν	Μ
Σ	Φ	Ζ	Γ	Ψ	Ε	Κ	Τ	Ε	Λ	Ε	Σ	Η	Λ	Ρ
Ε	Ρ	Ν	Υ	Ρ	Τ	Θ	Χ	Ν	Μ	Δ	Τ	Φ	Σ	Α
Θ	Ε	Δ	Λ	Β	Α	Λ	Γ	Ο	Ρ	Ι	Θ	Μ	Ο	Σ
Γ	Ν	Θ	Α	Ξ	Γ	Μ	Ι	Υ	Υ	Θ	Η	Ι	Ρ	Ν
Δ	Τ	Η	Σ	Γ	Λ	Φ	Μ	Τ	Γ	Κ	Α	Ο	Τ	Κ
Β	Ο	Ξ	Δ	Κ	Ω	Ω	Π	Α	Ι	Χ	Ν	Ι	Δ	Ι
Ε	Λ	Φ	Γ	Η	Τ	Σ	Ε	Δ	Τ	Σ	Φ	Ε	Κ	Α
Υ	Η	Γ	Ξ	Β	Τ	Ο	Γ	Ζ	Κ	Ι	Λ	Α	Δ	Σ
Θ	Μ	Λ	Τ	Σ	Ι	Υ	Ξ	Η	Α	Μ	Σ	Σ	Γ	Ε
Π	Κ	Ε	Φ	Τ	Σ	Β	Φ	Ρ	Ν	Α	Ζ	Μ	Η	Θ
Η	Λ	Σ	Ε	Θ	Η	Ρ	Σ	Μ	Π	Ρ	Φ	Π	Ο	Ι
Ν	Τ	Ξ	Υ	Κ	Τ	Ι	Τ	Α	Θ	Μ	Η	Δ	Μ	Σ

Κεφάλαιο 2ο: Καλώς ήλθες στον κόσμο του MSKodu!

2.1 Εγκατάσταση

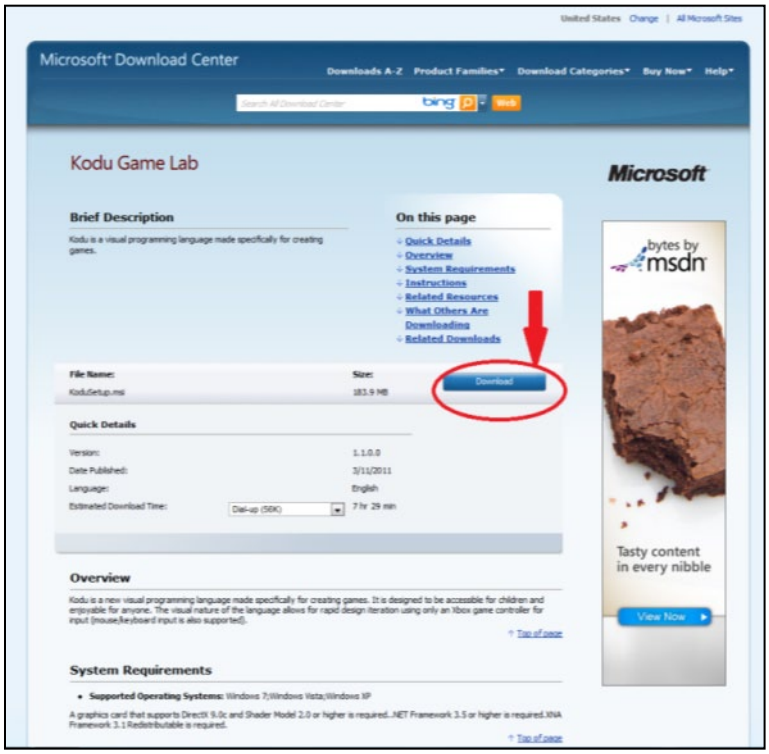
Ήρθε η ώρα να γνωρίσετε το «μυστηριώδες» MSKodu. Για να το κάνετε αυτό, πρέπει αρχικά να το κατεβάσετε και να το εγκαταστήσετε στον υπολογιστή σας! Ακολουθήστε μας βήμα προς βήμα.

Ας ξεκινήσουμε! Ανοίξτε ένα φυλλομετρητή και πληκτρολογείστε τη διεύθυνση: http://fuse.microsoft.com/, η οποία παραπέμπει στην επίσημη ιστοσελίδα των ερευνητικών εργαστηρίων της Microsoft, Fuse Labs, όπου έχει δημιουργηθεί το MSKodu. Παρατηρείστε ότι στην κάτω αριστερά γωνία υπάρχει ένα κουτάκι που αναφέρεται στο MSKodu. Κάντε κλικ πάνω σε αυτό το κουτάκι το οποίο θα σας μεταφέρει στη διεύθυνση http://fuse.microsoft.com/project/kodu.aspx.

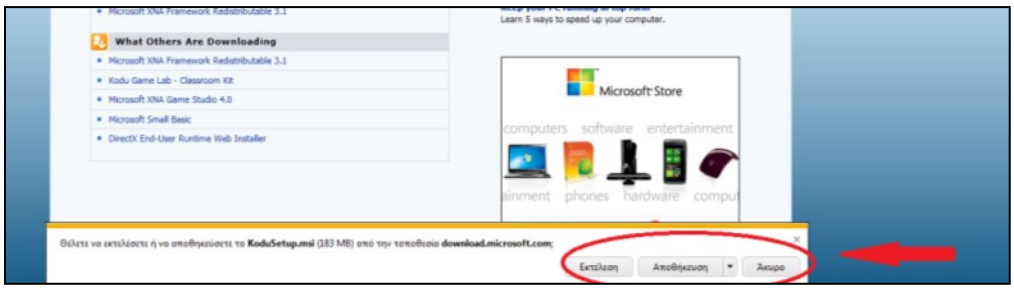


Η ιστοσελίδα που εμφανίζεται περιέχει οδηγίες, εκπαιδευτικά βίντεο, αντιπροσωπευτικές φωτογραφίες του περιβάλλοντος και πολλούς συνδέσμους για να εξοικειωθείτε με το MSKodu. Ας συνεχίσουμε όμως με την εγκατάσταση! Επιλέξτε το σύνδεσμο που βρίσκεται στο πάνω δεξιά τμήμα της οθόνης *Try it Now Download it here for free (Δοκίμασέ το! Κατέβασέ το δωρεάν)*.

Στη σελίδα που εμφανίζεται πατάμε πάνω στην επιλογή *Download (Λήψη)*.

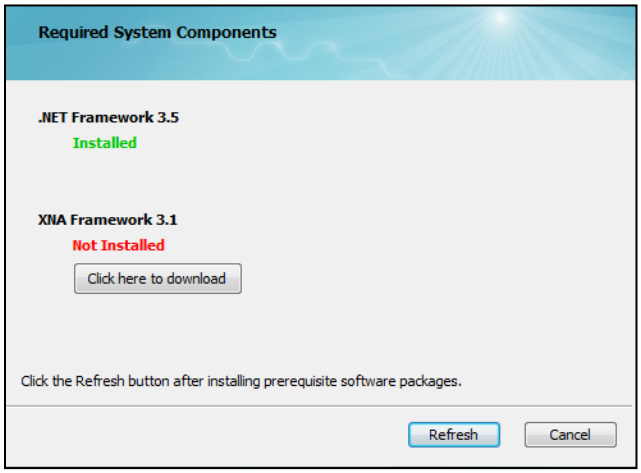


Εμφανίζεται ένα παράθυρο με τρεις επιλογές *Εκτέλεση*, *Αποθήκευση* και *Άκυρο*. Επιλέξτε *Εκτέλεση* ώστε το αρχείο εγκατάστασης να εκτελεστεί αυτόματα μετά τη λήψη του.

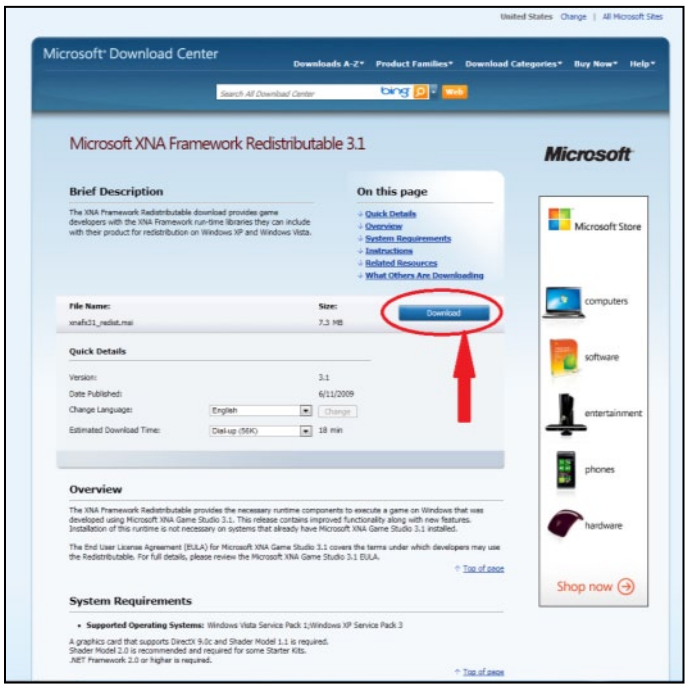


Στο νέο παράθυρο που εμφανίζεται, επιλέξτε *Εκτέλεση* για να γίνει η εγκατάσταση του MSKodu. Για να εκτελεστεί το MSKodu πρέπει να έχετε ήδη εγκατεστημένες στον υπολογιστή σας τις

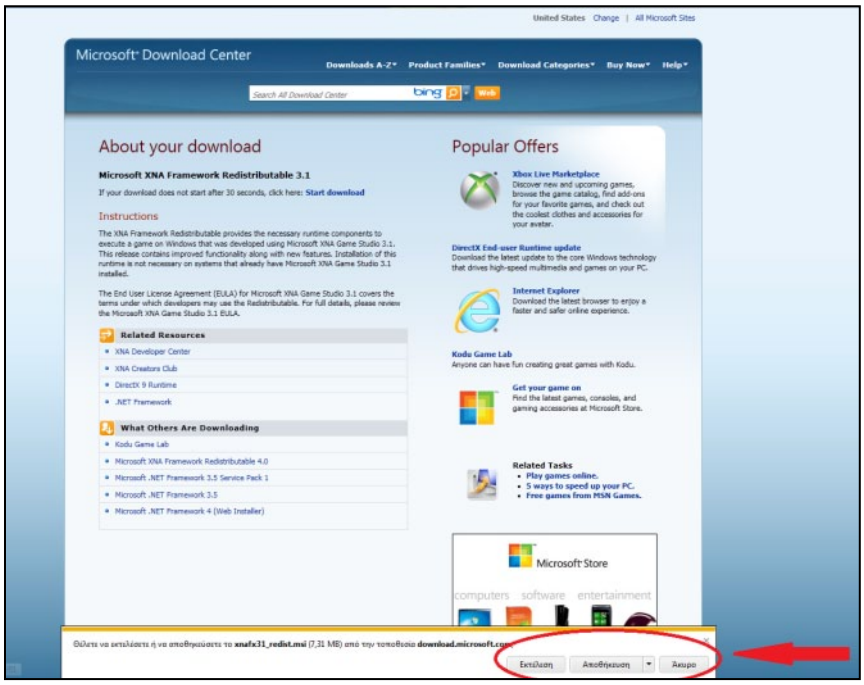
βιβλιοθήκες .NET Framework 3.5 και XNA Framework 3.1. Δε χρειάζεται να ανησυχείτε όμως, γιατί το πρόγραμμα εγκατάστασης ελέγχει αν υπάρχουν στον υπολογιστή σας και σας προτρέπει να εγκαταστήσετε ότι από τα δύο σας λείπει. Στο δικό μας σύστημα (όπως πολύ πιθανόν και στο δικό σας) λείπει η δεύτερη βιβλιοθήκη, οπότε θα πατήσουμε στο κουμπί *Click here to download (Πάτησε εδώ για να το κατεβάσεις)*. Αν έχετε ήδη αυτά τα προγράμματα μπορείτε να παραβλέψετε τις οδηγίες για την εγκατάστασή τους και να συνεχίσετε με την εγκατάσταση του MSKodu.



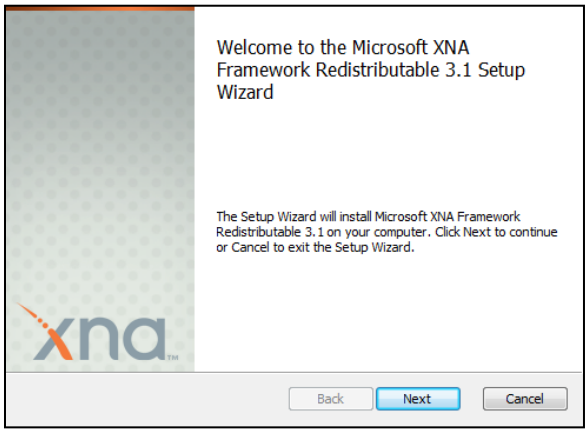
Αυτόματα ανοίγει ο προεπιλεγμένος περιηγητής σας στην παρακάτω ιστοσελίδα, στην οποία πρέπει πάλι να κάνετε κλικ στο *Download (Κατέβασε)* για να κατεβάσετε το αρχείο που χρειάζεστε.



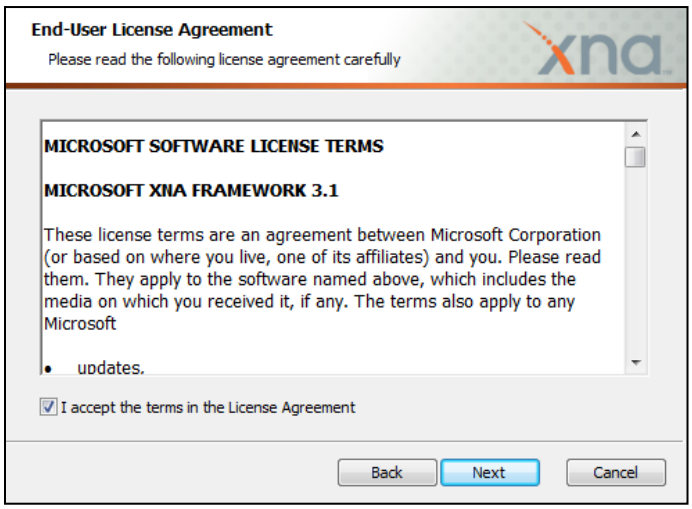
Μπορείτε, όπως αναφέρθηκε και παραπάνω, να επιλέξετε είτε *Εκτέλεση* είτε *Αποθήκευση* είτε *Ακύρωση* της λήψης του αρχείου. Για να γίνει λήψη και εγκατάσταση, επιλέξτε *Εκτέλεση*.



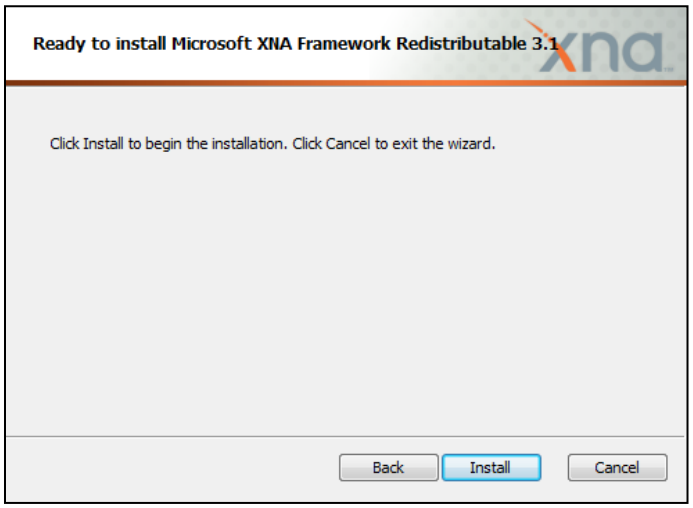
Το γνωρίζουμε ότι ανυπομονείτε να παίξετε με το MSKodu αλλά δυστυχώς πρέπει για άλλη μια φορά να περιμένετε μέχρι να ολοκληρωθεί η λήψη του νέου αρχείου. Επιλέξτε το κουμπί *Next (Επόμενο)* στο παράθυρο που ανοίγει από τον οδηγό εγκατάστασης.



Έπειτα κάντε κλικ στην ένδειξη *I accept the terms in the License agreement* (Αποδέχομαι τους όρους της άδειας χρήσης) και επιλέξτε για άλλη μια φορά το κουμπί *Next* (Επόμενο).

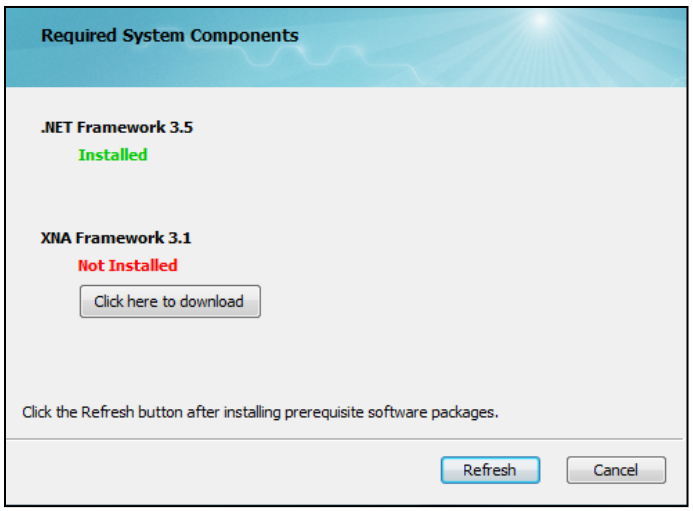


Για να ολοκληρωθεί η εγκατάσταση πρέπει να επιλέξετε την ένδειξη *Install (Εγκατέστησε)*.

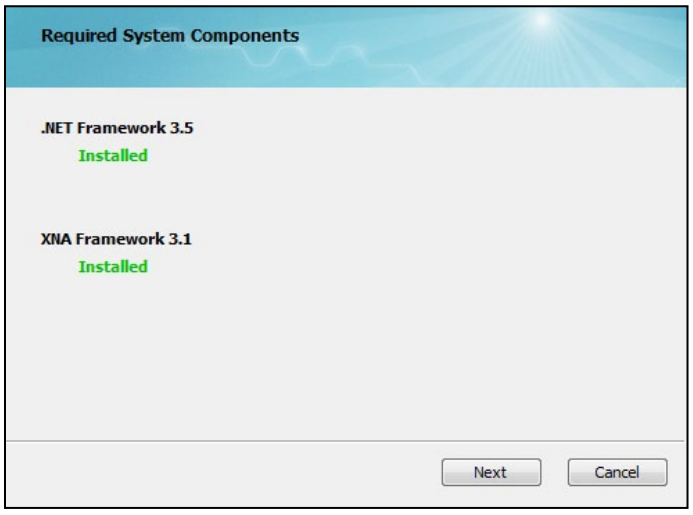


Το τελευταίο βήμα είναι να κάνετε κλικ στην ένδειξη *Finish (Τελείωσε)*.

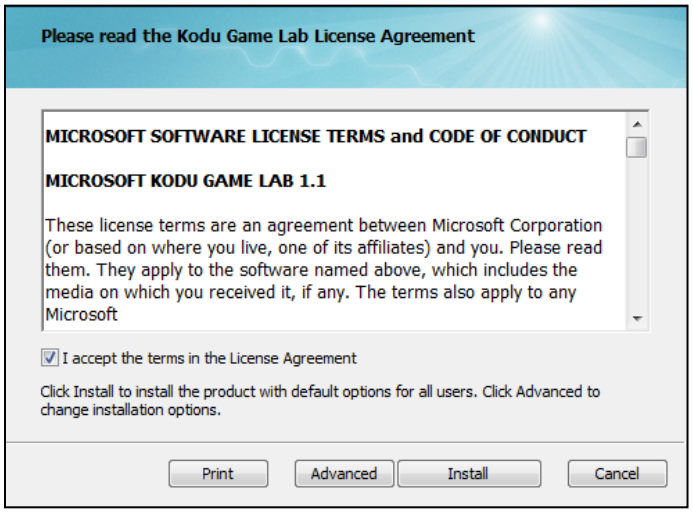
Αφού έχουμε κάνει μία μικρή παράκαμψη εγκαθιστώντας τη βιβλιοθήκη που έλειπε από το σύστημά μας, συνεχίζουμε στην αρχική μας εγκατάσταση πατώντας το κουμπί *Refresh (Ανανέωση)*.



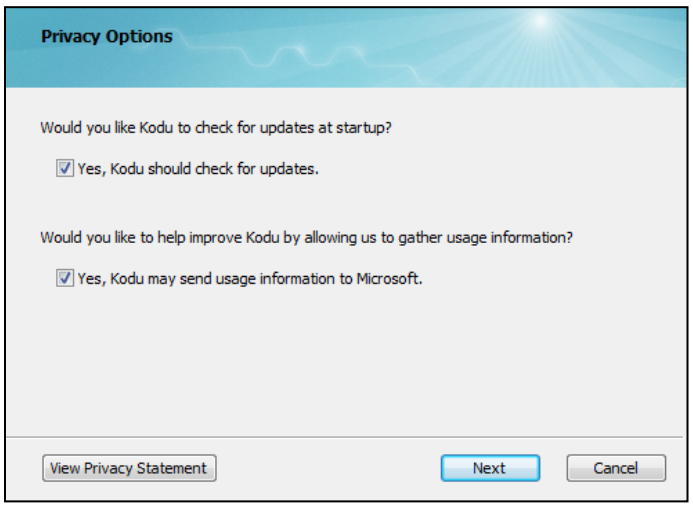
Παρατηρούμε ότι όλα τα προγράμματα που απαιτούνται είναι πλέον εγκατεστημένα στον υπολογιστή μας και έτσι μπορούμε πλέον να κάνουμε κλικ στην ένδειξη *Next (Επόμενο)*.



Για άλλη μια φορά πρέπει να κάνετε κλικ στην ένδειξη *I accept the terms in the License Agreement (Αποδέχομαι τους όρους στην άδεια χρήσης)* για να δηλώσετε ότι αποδέχεστε τους όρους της άδειας χρήσης του MSKodu. Στη συνέχεια πατήστε το *κουμπί Install (Εγκατέστησε)*.

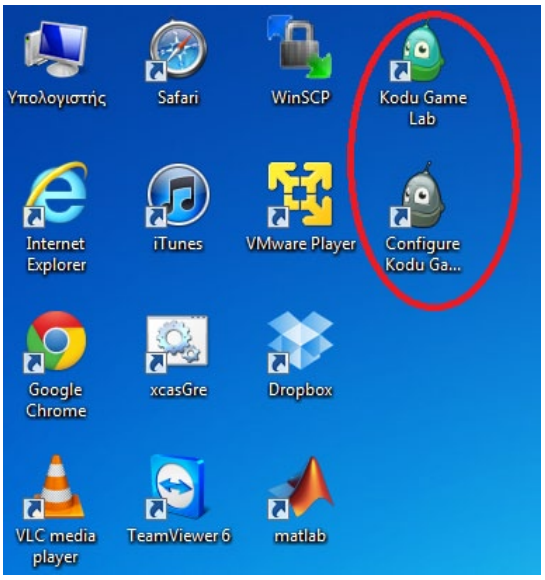


Στο παράθυρο που εμφανίζεται στην οθόνη σας υπάρχουν δύο ερωτήσεις: Η πρώτη αναφέρεται στο αν θέλετε να γίνεται αυτόματος έλεγχος για νέες ενημερώσεις του MSKodu ενώ η δεύτερη αφορά στην αποστολή πληροφοριών της χρήσης του προγράμματος στη Microsoft για τη βελτίωση του. Σίγουρα θέλουμε το πρόγραμμά μας να ενημερώνεται αυτόματα καθώς έτσι θα βελτιώνεται διαρκώς και θα επιλύονται τα προβλήματα που μπορεί να έχει. Καλό επίσης είναι να στέλνουμε στοιχεία της χρήσης μας για τη βελτίωση του MSKodu από τη στιγμή που τονίζεται η μη αποστολή προσωπικών μας πληροφοριών. Συνεχίζουμε πατώντας για τελευταία φορά το κουμπί *Next (Επόμενο)*.



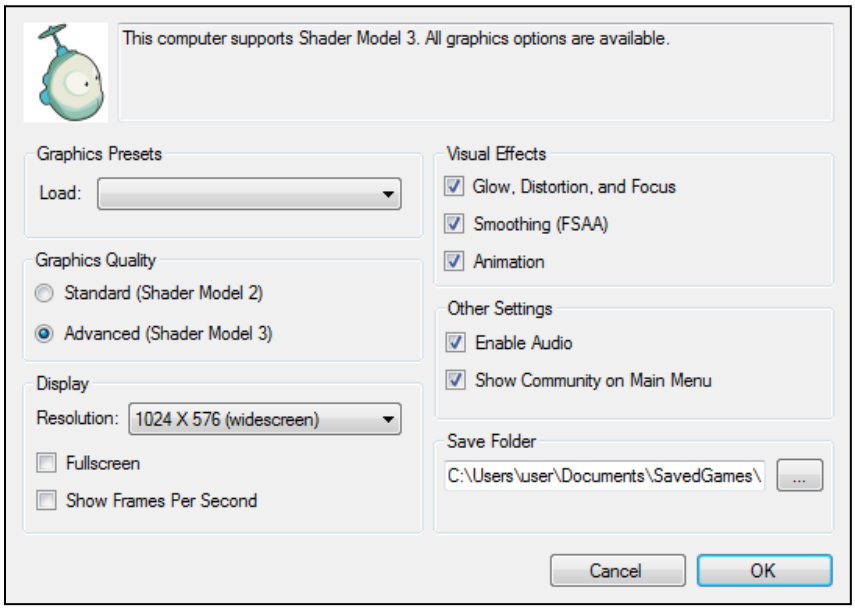
Μόλις η εγκατάσταση ολοκληρωθεί με επιτυχία πατάμε *Finish (Τελείωσε)*.

Ο οδηγός εγκατάστασης έχει δημιουργήσει στην επιφάνεια εργασίας σας δύο συντομεύσεις: Το Kodu Game Lab από όπου μπορείτε να τρέξετε το MSKodu και το Configure Kodu Game Lab με το οποίο μπορείτε να κάνετε ρυθμίσεις για τον τρόπο με τον οποίο εκτελείται το λογισμικό σας.



2.2 Ρυθμίσεις Kodu Game Lab

Ας ρυθμίσουμε το MSKodu πριν ξεκινήσουμε να σχεδιάζουμε και να παίζουμε παιχνίδια (δεν είναι απαραίτητο - μπορείτε να πάτε κατευθείαν στην επόμενη παράγραφο και να παίξετε το πρώτο σας παιχνίδι). Κάνοντας διπλό κλικ στη συντόμευση *Configure Kodu Game Lab (Ρύθμισε το Kodu)* θα εμφανιστεί το παράθυρο που βλέπετε στην επόμενη εικόνα:



Το παράθυρο αυτό σας δίνει την δυνατότητα να ρυθμίσετε κάποιες παραμέτρους του MSKodu, σύμφωνα με τις επιθυμίες σας αλλά και με βάση το υλικό (hardware) του υπολογιστή σας. Μπορείτε να παραβλέψετε εντελώς όλες τις ρυθμίσεις και να μπείτε κατευθείαν στο περιβάλλον. Καλό όμως είναι να έχουμε μια γενική εικόνα των ρυθμίσεων:

- Η επιλογή *Graphics Quality (Ποιότητα Γραφικών)* προσδιορίζει την ποιότητα των γραφικών του περιβάλλοντος βάσει δύο ρυθμίσεων: *Standard (Κανονικό-Shader Model 2)* και *Advanced (Προχωρημένο-Shader Model 3)*. Αν έχετε παλιά κάρτα γραφικών ίσως θα ήταν καλύτερο να επιλέξετε την Standard επιλογή που έχει λιγότερες απαιτήσεις από το σύστημά σας και θα σας επιτρέψει να χρησιμοποιήσετε το MSKodu χωρίς ιδιαίτερα προβλήματα (π.χ. καθυστερήσεις στα γραφικά). Όπως φαίνεται στο πάνω μέρος του παραθύρου, το MSKodu σας αναφέρει ποια μοντέλα γραφικών υποστηρίζει ο υπολογιστής σας.

- Στην επιλογή *Display (Εμφάνιση)* μπορείτε να καθορίσετε το μέγεθος του παραθύρου μέσα στο οποίο θα τρέχει το MSKodu (*επιλογή Resolution-Ανάλυση*). Μπορείτε να επιλέξετε αν θα καταλαμβάνει όλη την οθόνη σας (επιλογή *Fullscreen-Πλήρης Οθόνη*) ή ένα μικρότερο κομμάτι της (π.χ. 720 x 480 pixel) και το αν θα εμφανίζεται ο αριθμός των καρέ που δημιουργούνται ανά δευτερόλεπτο (*επιλογή Show Frames Per Second-Δείξε Αριθμό Πλαισίων Ανά Δευτερόλεπτο*). Το δεύτερο μάλλον δεν μας είναι ιδιαίτερα χρήσιμο και θα μπορούσαμε να το αφαιρέσουμε.

- Στην ομάδα ιδιοτήτων *Visual Effects (Οπτικά Εφέ)* μπορείτε να κάνετε επιλογές σχετικά με τα οπτικά εφέ που θέλετε να έχετε στα παιχνίδια σας. Θυμίζουμε ότι περισσότερες επιλογές, σημαίνουν μεγαλύτερη επιβάρυνση του υπολογιστή σας, αλλά και ελκυστικότερο οπτικό αποτέλεσμα.

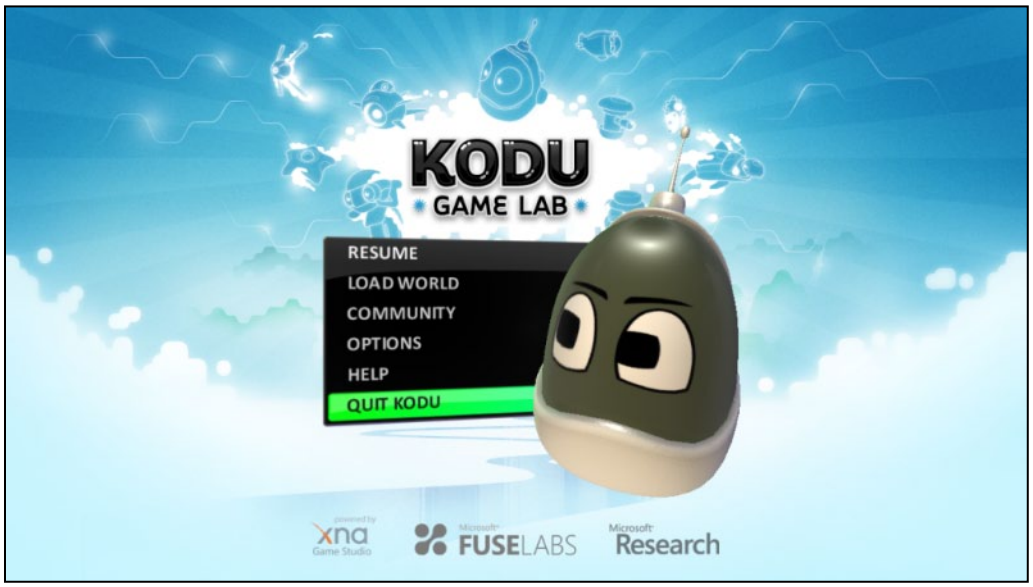
- Στην ομάδα ιδιοτήτων *Other Settings (Υπόλοιπες Ρυθμίσεις)* μπορείτε να προσδιορίσετε αν το MSKodu θα αναπαράγει ήχους (*επιλογή Enable Audio-Ενεργοποίηση Ήχων*) και αν θα εμφανίζεται η επιλογή *Κοινότητα (Community)* στο *Κεντρικό Μενού (Main Menu)* (*επιλογή Show Community on Main Menu*) ώστε να βλέπετε παιχνίδια-προγράμματα που έχουν δημιουργήσει άλλοι προγραμματιστές σαν εμάς από όλο τον κόσμο. Ας αφήσουμε όλα τα κουτάκια επιλεγμένα.

- Στο τελευταίο κουτάκι *Save Folder (Φάκελος Αποθήκευσης)* μπορείτε να καθορίσετε τον κατάλογο μέσα στον οποίο θα αποθηκεύονται τα παιχνίδια που δημιουργείτε εσείς ή που κατεβάζετε από την κοινότητα. Αν θέλετε να στέλνετε τα παιχνίδια που αναπτύσσετε στους φίλους σας, θα πρέπει να θυμάστε τη συγκεκριμένη επιλογή.

- Τέλος, το MSKodu για δική σας ευκολία διαθέτει τρία έτοιμα σετ συνολικών ρυθμίσεων από το κουτάκι *Graphics Presets (Προεπιλογές Γραφικών)* (Default, Standard, Advanced). Αφού ρυθμίσαμε εμείς τις επιλογές μας, δε χρειάζεται να επιλέξουμε προς το παρόν κάποια από αυτές. Αν το πρόγραμμα δεν εκτελείται ικανοποιητικά, θυμηθείτε να επισκεφτείτε και να αλλάξετε τις ρυθμίσεις εκτέλεσής του.

2.3 Παίζοντας τα πρώτα μου παιχνίδια

Έφτασε επιτέλους η στιγμή να γνωρίσετε για πρώτη φορά τον κόσμο του MSKodu! Ας υποθέσουμε ότι είστε πάλι στην επιφάνεια εργασίας σας και κάνετε διπλό κλικ στην συντόμευση *Kodu Game Lab*. Το λογισμικό αρχίζει να τρέχει και στο παράθυρο που ανοίγει εμφανίζεται το ***ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ (MAIN MENU)*** επιλογών.



Αν επιλέξετε ***ΦΟΡΤΩΣΗ ΚΟΣΜΟΥ (LOAD WORLD)*** θα εμφανιστούν όλα τα παιχνίδια του MSKodu που βρίσκονται στον υπολογιστή σας, ταξινομημένα σε καρτέλες.

Για να επιλέξετε ποιο από όλα τα παιχνίδια θέλετε να παίξετε, μπορείτε να μεταβείτε από το ένα στο άλλο χρησιμοποιώντας τα βελάκια που υπάρχουν δεξιά και αριστερά στο παράθυρο. Για να παίξετε ένα παιχνίδι, κάνετε κλικ πάνω του και στο μενού που εμφανίζεται δίπλα του πρέπει να πατήσετε *Παίξε (Play)* (ή αλλιώς μπορείτε να κάνετε διπλό κλικ εξ'αρχής πάνω στο παιχνίδι). Στο πάνω μέρος του παραθύρου εμφανίζονται 5 διαφορετικοί τρόποι φιλτραρίσματος των παιχνιδιών που είναι διαθέσιμα:

Οι κόσμοι μου (My Worlds), παιχνίδια που έχετε δημιουργήσει εσείς (προφανώς η λίστα αυτή θα είναι άδεια αρχικά),

Κατεβασμένα (Downloads), παιχνίδια που κατεβάσατε από την κοινότητα προγραμματιστών του MSKodu (και η λίστα αυτή θα είναι κενή αρχικά),

Μαθήματα (Lessons), μαθήματα που σας δείχνουν βήμα προς βήμα πως μπορείτε να δημιουργήσετε τα δικά σας παιχνίδια,

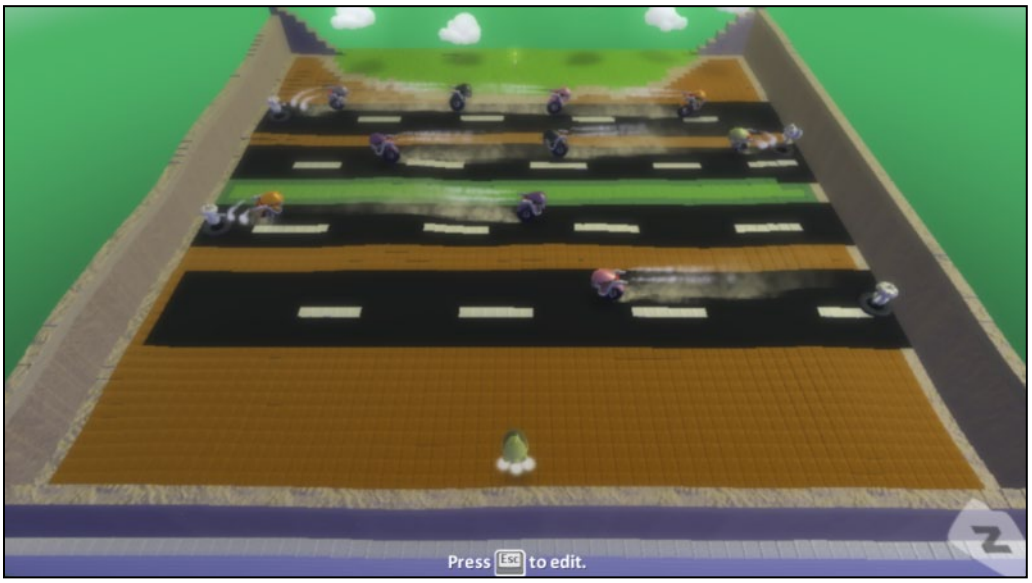
Παραδείγματα (Samples), επιλογή που εμφανίζει αντιπροσωπευτικά παραδείγματα παιχνιδιών που προσφέρονται μαζί με την εγκατάσταση του MSKodu, και

Όλα (All), επιλογή με την οποία παρουσιάζονται όλα τα παραπάνω μαζί.

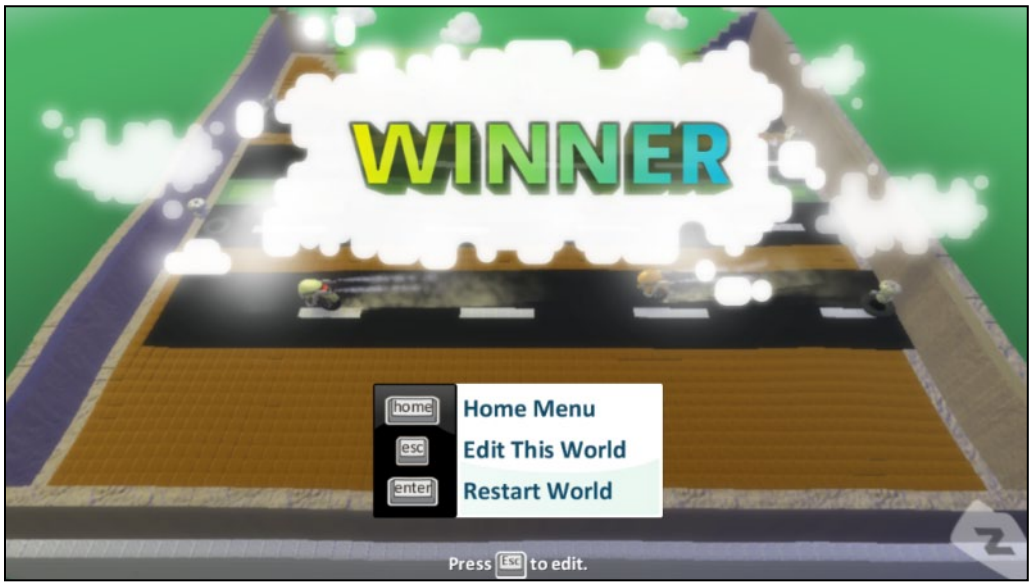
Ας παίξουμε πρώτα το παιχνίδι Roadkill v03. Το παιχνίδι αυτό μπορείτε να το βρείτε στην καρτέλα *Παραδείγματα (Samples)* ή ακόμη και στο συνοδευτικό υλικό του βιβλίου μας. Το μόνο που πρέπει να κάνετε είναι διπλό κλικ πάνω του και αυτό θα ανοίξει αμέσως με το MSKodu.

Σκεφτείτε ότι έχετε υπογλυκαιμία και τυχαία βρίσκεστε σε έναν κεντρικό δρόμο και θέλετε να περάσετε απέναντι για να πάτε στο περίπτερο να πάρετε μία σοκολάτα. Όμως ο δρόμος είναι γεμάτος από αυτοκίνητα και μηχανάκια που τρέχουν γρήγορα. Χμμμ... τι πρέπει να κάνετε άραγε; Μάλλον θα πρέπει να κοιτάξετε δεξιά και αριστερά, να προσέξετε πολύ και ύστερα να διασχίσετε το δρόμο για να φτάσετε στον προορισμό σας και να πάρετε την πολυπόθητη σοκολάτα! Αυτό λοιπόν συμβαίνει, κατά κάποιο τρόπο, και στο παιχνίδι μας! Πιο συγκεκριμένα, στόχος του παιχνιδιού είναι να μπορέσει ο πρωταγωνιστής να διασχίσει με ασφάλεια τους 4 δρόμους για να φτάσει στην απέναντι πλευρά επιτυχώς και να πάρει το έπαθλο. Το παιχνίδι διαδραματίζεται σε έναν ορθογώνιο κόσμο που περιτριγυρίζεται από ένα τείχος. Μέσα στον κόσμο αυτό είναι σχηματισμένοι 4 δρόμοι και επίσης 2 πλευρές, στη μία βρίσκεται ο πρωταγωνιστής και στην άλλη το έπαθλο. Ο πρωταγωνιστής είναι ο Kodu ο οποίος στην προσπάθειά του να πάρει το *Αστέρι (Star)*, που είναι το έπαθλο, έρχεται αντιμέτωπος όχι με έναν μόνο *Μηχανάκια (Cycle)*, αλλά με πολλούς, που δημιουργούνται από *τα Περισκόπια (Sticks)* και διασχίζουν το δρόμο ανενόχλητοι για να κάνουν τη δουλειά τους. Δυστυχώς, αν ακουμπήσουν τον Kodu τον σκοτώνουν! Με τα βέλη του πληκτρολογίου σας μπορείτε να μετακινήσετε τον Kodu μέσα στον κόσμο προς όλες τις κατευθύνσεις.

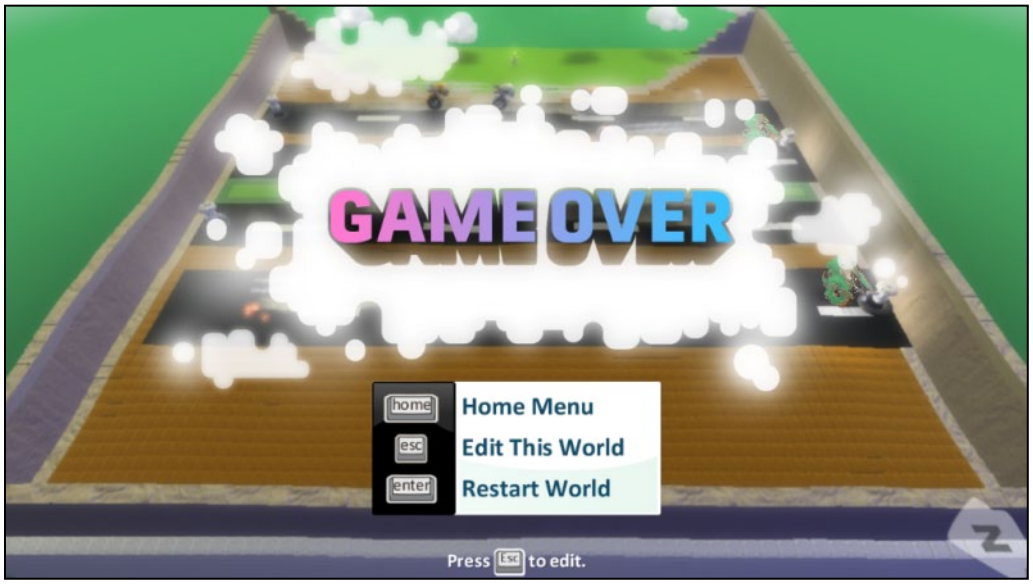
Για να αρχίσετε το παιχνίδι πατήστε Enter.



Μπορείτε να παρατηρήσετε ότι καθώς περνάτε περισσότερους δρόμους, οι επικίνδυνοι αντίπαλοί σας εμφανίζονται συχνότερα με αποτέλεσμα να γίνεται πιο δύσκολη η διέλευσή σας. Όταν περάσετε επιτυχώς στην άλλη πλευρά του κόσμου και ακουμπήσετε το *Αστέρι (Star)* γίνεστε ο νικητής του παιχνιδιού και το παιχνίδι τερματίζεται.



Αν όμως χάσετε τότε δυστυχώς εμφανίζεται η ακόλουθη οθόνη... *ΤΕΛΟΣ ΠΑΙΧΝΙΔΙΟΥ (GAME OVER)*!



Και στις δύο περιπτώσεις μπορείτε να παίξετε ξανά πατώντας Enter ή μπορείτε να μεταβείτε στο *Αρχικό Μενού (Home Menu)* πατώντας Home ή και να τροποποιήσετε το παιχνίδι πατώντας Escape.

Ένα ακόμα ενδιαφέρον παιχνίδι είναι το Racing World! v02 by Matt που έχουμε κατεβάσει από την κοινότητα προγραμματιστών του MSKodu και το οποίο βρίσκεται στο συνοδευτικό υλικό του βιβλίου. Απλά πατήστε διπλό κλικ πάνω του.

Το παιχνίδι αυτό είναι ένα κλασικό παιχνίδι *Αγώνα Ταχύτητας (Racing)*. Πρωταγωνιστές είναι ο Kodu και *Μηχανάκια (Cycles)*. Σκοπός του παιχνιδιού είναι να οδηγήσετε τον Kodu μέσα στον κόσμο με τα βέλη του πληκτρολογίου σας, ώστε να προσπεράσετε τα Μηχανάκια και να αποκτήσετε το μεγαλύτερο σκορ. Ο Kodu στην προσπάθειά του αυτή, έχει τη δυνατότητα να καθυστερήσει τα *Μηχανάκια (Cycles)* μαζεύοντας κόκκινα και πράσινα *Μήλα (Apples)* που του δίνουν τη δυνατότητα να εκτοξεύει πυραύλους. Η έναρξη σηματοδοτείται άμεσα με το άνοιγμα του παιχνιδιού. Στο πάνω δεξιά μέρος της οθόνης σας κρατούνται δύο σκορ: Το πρώτο σκορ είναι των ανταγωνιστών σας *(Cycles-Μηχανάκια)* και το δεύτερο είναι το δικό σας. Κάθε φορά που ο Kodu περνάει από την κόκκινη γραμμή παίρνει ένα βαθμό. Μια φορά σε όλη τη διάρκεια του αγώνα θα πάρει για το πέρασμά του από την κόκκινη γραμμή bonus 10 βαθμών. Αναλόγως όταν τα *Μηχανάκια (Cycles)* περνούν από την κόκκινη γραμμή προστίθενται 2 βαθμοί στο σκορ τους. Το παιχνίδι τερματίζει με νικητή αυτόν που θα μαζέψει πρώτος 20 βαθμούς.

3…2…1…GO!!

Είναι πολύ εύκολο να δημιουργήσετε και εσείς αντίστοιχα παιχνίδια, με τους δικούς σας κανόνες και τους δικούς σας πρωταγωνιστές. Είστε έτοιμοι;

Περίληψη

Συνοψίζοντας, σε αυτό το σύντομο κεφάλαιο είδαμε πώς μπορούμε να εγκαταστήσουμε το MSKodu, πώς μπορούμε να αλλάξουμε τις ρυθμίσεις του, ενώ εξοικειωθήκαμε με τις βασικές επιλογές μενού που χρειαζόμαστε για να φορτώνουμε παιχνίδια. Τέλος, παρουσιάστηκαν μία σειρά από παραδείγματα έτοιμων παιχνιδιών για να σας εμπνεύσουν και να σας εξάψουν την περιέργεια ώστε να ξεκινήσετε να δημιουργείτε τα δικά σας παιχνίδια.

Δραστηριότητες

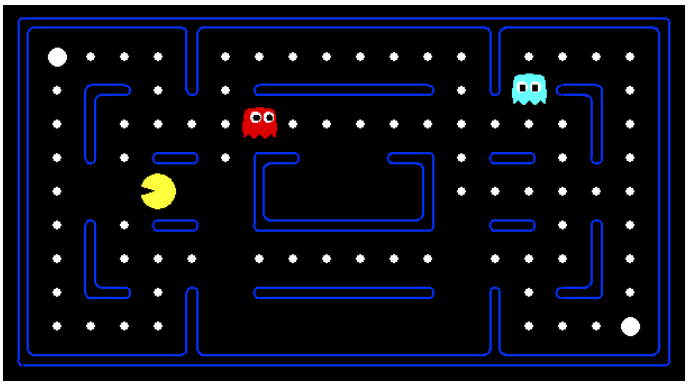
1. Μεταβείτε στο Configure Kodu Game Lab στην καρτέλα *Οπτικά Εφέ (Visual Effects)* και απενεργοποιήστε όλες τις επιλογές. Ύστερα παίξτε το παιχνίδι που περιγράφεται στα έτοιμα παραδείγματα Roadkill v03. Έπειτα ενεργοποιήστε πάλι όλες τις επιλογές και ξαναπαίξτε το. Τι παρατηρείτε; Ο υπολογιστής σας με ενεργοποιημένες αυτές τις επιλογές είναι πιο αργός ή όχι; Το αποτέλεσμα στην οθόνη σας έχει μεγάλη διαφορά πριν και μετά; Ρυθμίστε το MSKodu σύμφωνα με τις παρατηρήσεις σας.

2. Κατεβάστε από το μενού *ΚΟΙΝΟΤΗΤΑ (COMMUNITY)* του MSKodu το παιχνίδι KODU-s Greatest Quest v02, by HASCI. Παίξτε αυτό το παιχνίδι. Το παιχνίδι αυτό είναι αποτέλεσμα της δημιουργικής σκέψης ενός δεκατετράχρονου αγοριού. Προσπαθήστε να εμπνευστείτε από τον Hasci και να σκεφτείτε και εσείς ένα παιχνίδι που θα θέλατε να υλοποιήσετε. Ποιο θα ήταν το σενάριο και ποιος ο σκοπός του παιχνιδιού σας; Ποιοι οι πρωταγωνιστές του; Τι ενέργειες θα εκτελούσαν αυτοί;

Κεφάλαιο 3ο: Δημιουργώ το Πρώτο μου Παιχνίδι

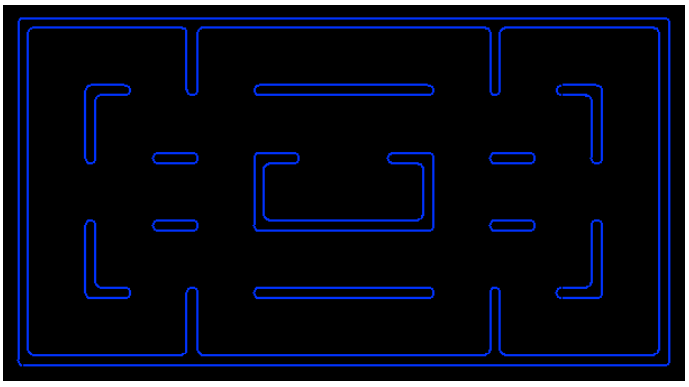
3.1 Γεγονοστραφής /Αντικειμενοστραφής Προγραμματισμός

Το MSKodu είναι ένα εργαλείο προγραμματισμού παιχνιδιών. Σας βοηθά, δηλαδή, παρέχοντας μια προγραμματιστική πλατφόρμα, να σχεδιάσετε παιχνίδια, να δημιουργήσετε τις δικές σας πίστες, τους ήρωες και τα αντικείμενά σας και να προγραμματίσετε τις συμπεριφορές τους. Ας ξεκινήσουμε όμως λίγο πιο γενικά, μελετώντας το σκεπτικό προγραμματισμού ενός διάσημου παιχνιδιού, του Pacman.



Φανταστείτε για μια στιγμή ότι εσείς είστε ο προγραμματιστής αυτού του παιχνιδιού. Ποια είναι τα στοιχεία του παιχνιδιού που έχετε να δημιουργήσετε; Στο παιχνίδι Pacman έχουμε μια πίστα που μοιάζει με μικρό λαβύρινθο, πάνω στην οποία υπάρχουν διάσπαρτες μικρές και μεγάλες άσπρες τελίτσες. Ένα μικρό κίτρινο προσωπάκι, ο pacman, πρέπει να "φάει" όλες τις τελίτσες πριν προλάβουν να τον ακουμπήσουν (και έτσι να τον σκοτώσουν) τα φαντάσματα που τον κυνηγούν. Όταν ο Pacman ακουμπήσει μια μικρή τελίτσα, τότε αυτή εξαφανίζεται. Όταν ο Pacman ακουμπήσει μία από τις μεγάλες τελίτσες τότε η μεγάλη τελίτσα εξαφανίζεται και ο Pacman κινείται πολύ πιο γρήγορα και μπορεί πλέον, να σκοτώσει αυτός τα φαντάσματα όταν τα ακουμπήσει. Ο παίκτης του παιχνιδιού πρέπει να κινεί τον Pacman προσεκτικά μέσα στο λαβύρινθο, χρησιμοποιώντας μόνο τα βέλη του πληκτρολογίου.

Παρατηρήστε ότι όλο το παιχνίδι εκτυλίσσεται μέσα σε μία πίστα που μοιάζει με λαβύρινθο. Άρα αρχικά θα έπρεπε να σχεδιάσουμε την πίστα μας.



Μέσα σε αυτή την πίστα υπάρχουν τα στοιχεία του παιχνιδιού ή αλλιώς τα αντικείμενα του παιχνιδιού. Ποια είναι τα αντικείμενα από τα οποία αποτελείται το παιχνίδι Pacman; Έχουμε τον Pacman, τα φαντάσματα, τις μικρές τελίτσες και τις μεγάλες τελίτσες. Κάθε ένα από αυτά τα αντικείμενα έχει κάποιες συμπεριφορές.

Μερικές **ΣΥΜΠΕΡΙΦΟΡΕΣ** που έχει ο pacman:

ΟΤΑΝ ακουμπήσει μια μεγάλη τελίτσα (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** ο Pacman αρχίζει να αναβοσβήνει και να τρέχει πολύ γρήγορα (**ΕΝΕΡΓΕΙΕΣ**).

Μία **ΣΥΜΠΕΡΙΦΟΡΑ** που έχει κάθε μικρή τελίτσα:

ΟΤΑΝ ακουμπήσει τον Pacman (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** αυτή εξαφανίζεται (**ΕΝΕΡΓΕΙΑ**).

Μία **ΣΥΜΠΕΡΙΦΟΡΑ** που έχει κάθε μεγάλη τελίτσα:

ΟΤΑΝ ακουμπήσει τον Pacman (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** αυτή εξαφανίζεται (**ΕΝΕΡΓΕΙΑ**).

Μία **ΣΥΜΠΕΡΙΦΟΡΑ** που έχει κάθε φάντασμα:

ΟΤΑΝ δει τον pacman (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** τον κυνηγά για να τον ακουμπήσει (**ΕΝΕΡΓΕΙΑ**).

Παρατηρήστε ότι κάθε **ΣΥΜΠΕΡΙΦΟΡΑ** που περιγράψαμε είναι ο συνδυασμός ενός **γεγονότος** και μιας **ενέργειας**.

Το **ΓΕΓΟΝΟΣ** αναφέρεται σε κάτι που συμβαίνει στον κόσμο και το οποίο μπορεί να γίνει αντιληπτό από το αντικείμενό μας (π.χ. να δει, να ακουμπήσει, κτλ). Μπορεί να γίνει αντιληπτό γιατί κάθε αντικείμενο διαθέτει κάποιους **αισθητήρες** (π.χ. αισθητήρα όρασης, αισθητήρα αφής, κτλ).

Η **ΕΝΕΡΓΕΙΑ** από την άλλη είναι μια δράση του αντικειμένου (π.χ. αναβοσβήνει, εξαφανίζεται, μετακινείται, κτλ).

Με αυτόν ακριβώς τον τρόπο θα προγραμματίσουμε τα παιχνίδια μας και στο MSKodu. Θα αποφασίζουμε ποια είναι τα αντικείμενα του παιχνιδιού μας και στη συνέχεια θα προσδιορίζουμε τις συμπεριφορές τους με τη μορφή γεγονότων και ενεργειών. Ο προγραμματισμός, ο οποίος βασίζεται στην ανίχνευση και δημιουργία αντικειμένων με κατάλληλες συμπεριφορές για την επίλυση των προβλημάτων μας, ονομάζεται **αντικειμενοστραφής προγραμματισμός.** Επιπλέον, ο προγραμματισμός ο οποίος επιτρέπει την απόδοση συμπεριφορών στα αντικείμενα βάσει των γεγονότων που μπορούν να αντιληφθούν (μέσω των αισθητήρων τους) λέγεται **γεγονοστραφής προγραμματισμός**.

Στο MSKodu θα έχετε την ευκαιρία να αξιοποιήσετε εύκολα και διασκεδαστικά και τα δύο αυτά στυλ προγραμματισμού, τα οποία συνεργάζονται για να κάνουν τη δουλειά του προγραμματιστή παιχνιδιών πιο εύκολη.

3.2 Ο Κόσμος, τα Αντικείμενα και οι Συμπεριφορές

Και στο MSKodu δυο στοιχεία θα μας απασχολούν κατά την ανάπτυξη ενός παιχνιδιού: ο **κόσμος** του και τα **αντικείμενά** του. Ο **κόσμος** ή η πίστα ενός παιχνιδιού είναι κάτι οικείο σε εσάς. Ο κόσμος του παιχνιδιού είναι το περιβάλλον μέσα στο οποίο εκτυλίσσεται το παιχνίδι, οι πεδιάδες, τα βουνά, οι λίμνες κτλ., το περιβάλλον μέσα στο οποίο κινούνται και ενεργούν οι ήρωές σας. Για παράδειγμα, αν παίζατε το παιχνίδι Star Wars ο κόσμος θα ήταν το διάστημα ή κάποιος πλανήτης.

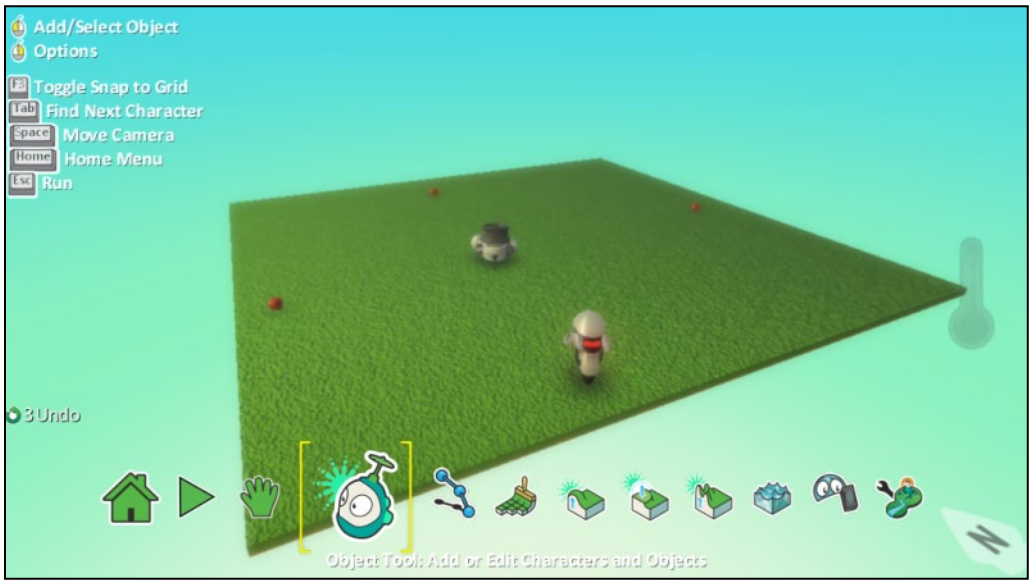
Αν παίζατε το παιχνίδι Titanic (ναι, υπάρχει!) ο κόσμος σας θα ήταν ένα πλοίο.

Στο MSKodu ένα από τα πράγματα που θα σας κάνουν να βάλετε τη φαντασία σας να δουλέψει είναι ότι εσείς θα σχεδιάζετε τον κόσμο του παιχνιδιού σας όπως θέλετε. Για παράδειγμα, μπορείτε να σχεδιάσετε έναν κόσμο ο οποίος θα είναι μια τεράστια λίμνη. Η μορφή που θα έχει ο κόσμος σας θα επηρεάσει την κίνηση του ήρωά σας και των υπόλοιπων αντικειμένων καθώς και την πλοκή του παιχνιδιού. Αν έχετε έναν κόσμο με λίμνες, τότε δεν μπορείτε να περιμένετε από τον ήρωά σας να κινείται γρήγορα μέσα σε αυτές.

Το δεύτερο βασικό στοιχείο ενός παιχνιδιού είναι τα **αντικείμενα** με τις **συμπεριφορές** τους. **Αντικείμενο** είναι οτιδήποτε έχει συμπεριφορές, οτιδήποτε δηλαδή εκτελεί κάποιες ενέργειες, παίζει κάποιο ενεργό ρόλο στο παιχνίδι. Όπως θα δείτε και εσείς, τα αντικείμενα που θα χρησιμοποιήσετε στο MSKodu είναι αντικείμενα που υπάρχουν και στον πραγματικό κόσμο. Μερικά από θα μας είναι οικεία όπως μια μπάλα, ένα πλοίο, μια ρουκέτα, ένα μηχανάκι, ένα αερόστατο κτλ., άλλα όχι και τόσο! Αν δημιουργήσετε έναν ήρωα στο MSKodu ο οποίος, όταν δει ένα κόκκινο αερόστατο, τότε να εκτοξεύει ένα πύραυλο προς αυτό, τότε θα έχετε δώσει μια **συμπεριφορά** στο αντικείμενο αερόστατο. Οι συμπεριφορές των αντικειμένων είναι αυτές που κάνουν ένα παιχνίδι στο MSKodu διασκεδαστικό, βαρετό, ρεαλιστικό ή όχι.

3.3 Το Παιχνίδι που θα δημιουργήσουμε

Να και μία πρώτη ιδέα για παιχνίδι:



Παρατηρήστε ότι στο κάτω δεξί μέρος της οθόνης του MSKodu παρουσιάζεται μια πυξίδα που μας βοηθά στο προσανατολισμό μας. Δείχνει στον Βορρά (North), γι' αυτό και το γράμμα 'N' μέσα στο αντίστοιχο σχήμα.!

Θα δημιουργήσουμε ένα τετράγωνο επίπεδο που θα είναι καταπράσινο με γρασίδι. Πάνω στο γρασίδι και συγκεκριμένα νότια της πίστας θα τοποθετήσουμε τον πρωταγωνιστή μας, που εδώ θα είναι ένας *Μηχανάκιας (Cycle)*. Με τα βέλη από το πληκτρολόγιο ο παίκτης θα μπορεί να κινεί τον *Μηχανάκια*. Στο κέντρο της πίστας θα βάλουμε ένα *Κανόνι (Cannon)* το οποίο όταν βλέπει τον *Μηχανάκια*, τότε θα τον κυνηγάει. Θα μου πείτε, και με το δίκιο σας, τι κανόνι είναι αυτό που μόνο θα κυνηγάει τον πρωταγωνιστή μας, προσπαθώντας να τον ακουμπήσει, χωρίς όμως να μπορεί να τον σκοτώσει; Δεν θα είχε περισσότερο νόημα αν έριχνε πυραύλους εναντίον του *Μηχανάκια*; Κάντε λίγη υπομονή μέχρι το τέλος του κεφαλαίου και θα γίνει και αυτό στη μία από τις δύο δραστηριότητες που σας έχουμε ετοιμάσει.

Επίσης, θα τοποθετήσουμε τρία *Μήλα (Apples)*, ένα βόρεια, ένα δυτικά και ένα ανατολικά. Στόχος του παίκτη θα είναι να κινήσει τον *Μηχανάκια* ώστε να ακουμπήσει και να φάει όλα τα *Μήλα*. Όταν τα *Μήλα* τελειώσουν, τότε το παιχνίδι θα ολοκληρώνεται με νίκη του παίκτη.

Δεν δοκιμάζετε να ανοίξετε και να παίξετε λίγο το παιχνίδι για να εξοικειωθείτε καλύτερα με την ιδέα του προτού αρχίσουμε να το δημιουργούμε; Αρκεί να πατήσετε διπλό κλικ στο αρχείο **[03_01.Kodu]** που βρίσκεται στο συνοδευτικό υλικό του βιβλίου και στη συνέχεια να πατήσετε την *Αρχή Παιχνιδιού (Play Game)* από την *Παλέτα Εργαλείων (Tool Palette).*

3.4 Πρώτα ας το σκεφτούμε

- Ποιος θα είναι ο κόσμος;

- Ποια θα είναι τα αντικείμενα και τι συμπεριφορές θα έχει το κάθε αντικείμενο;

Σκεφτείτε απλά: Κάθε αντικείμενο έχουμε πει ότι έχει συγκεκριμένες συμπεριφορές (θυμηθείτε τον pacman, τα φαντάσματα, τις μικρές τελίτσες και τις μεγάλες τελίτσες). Ο κόσμος-πίστα από την άλλη δεν έχει συμπεριφορά. Είναι η εικόνα στην οποία τοποθετούμε τα διάφορα αντικείμενα και στην οποία αυτά θα κινούνται (θυμηθείτε την εικόνα-λαβύρινθο στο Pacman). Άρα, το τετράγωνο επίπεδο με το γρασίδι θα αποτελεί τον κόσμο του παιχνιδιού.

Ο *Μηχανάκιας* από την άλλη θέλουμε να έχει συμπεριφορές. Συγκεκριμένα, τρεις συμπεριφορές:

1η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να κινείται στην αντίστοιχη κατεύθυνση πάνω στην πίστα (**ΕΝΕΡΓΕΙΑ**).

2η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* ακουμπήσει ένα *Μήλο* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τρώει το *Μήλο* (**ΕΝΕΡΓΕΙΑ**).

3η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* έχει φάει όλα τα μήλα (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τερματίζει το παιχνίδι με νίκη για τον παίκτη (**ΕΝΕΡΓΕΙΑ**).

Το *Κανόνι* θέλουμε και αυτό να έχει συμπεριφορές. Συγκεκριμένα, μία **ΣΥΜΠΕΡΙΦΟΡΑ**:

Θέλουμε

ΟΤΑΝ το *Κανόνι* βλέπει τον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τον κυνηγάει (**ΕΝΕΡΓΕΙΑ**).

Άρα, αφού το *Κανόνι* θέλουμε να έχει συμπεριφορά, θα είναι και αυτό ένα αντικείμενο.

Τα τρία *Μήλα* θα είναι αντικείμενα? Ναι, και αυτά θέλουμε να έχουν **ΣΥΜΠΕΡΙΦΟΡΑ**:

Θέλουμε:

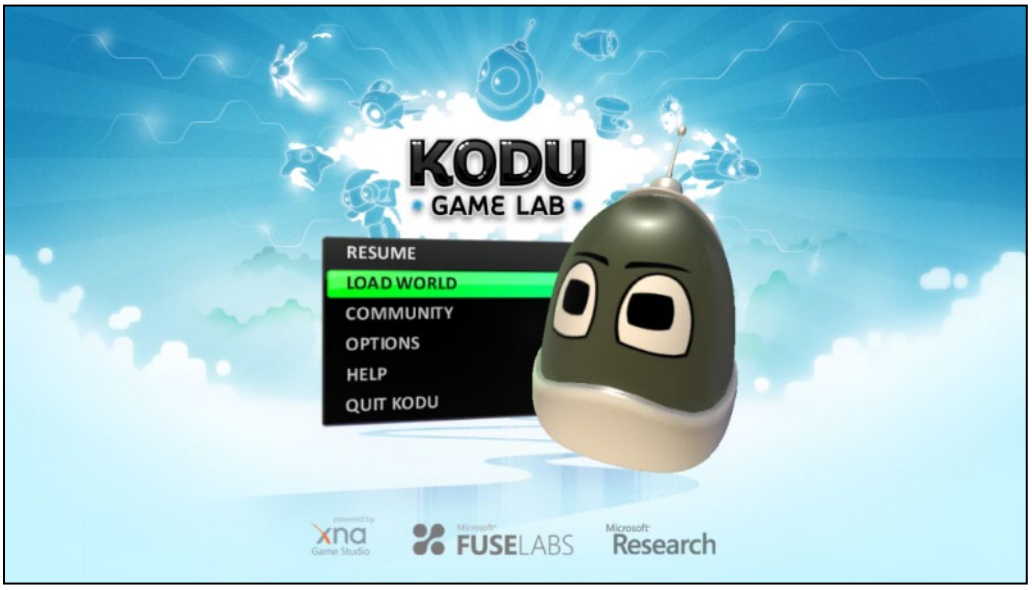
κάθε *Μήλο* **ΟΤΑΝ** ακουμπήσει με τον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να εξαφανίζεται (**ΕΝΕΡΓΕΙΑ**).

Άρα, και τα *Μήλα* θα είναι αντικείμενα.

Αφού λοιπόν διακρίναμε ποιος θα είναι ο κόσμος και ποια τα αντικείμενα του παιχνιδιού μας, ήρθε η ώρα να ανοίξουμε το περιβάλλον του MSKodu. Είναι σημαντικό να τονίσουμε ότι περιγράψαμε τις εντολές σε φυσική γλώσσα και όχι στη γλώσσα του MSKodu. Άραγε πόσο δύσκολο είναι να μετασχηματίσουμε αυτές τις εντολές σε εντολές του MSKodu;

3.5 Δημιουργώ τον Κόσμο

Το πρώτο πράγμα που θα φτιάξουμε είναι ο κόσμος. Λογικό, αφού χωρίς κόσμο, πού θα τοποθετούσαμε μετά τα διάφορα αντικείμενα, πού θα πατούσαν οι πρωταγωνιστές μας; Πηγαίνετε λοιπόν στο *ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ (MAIN MENU)* και από εκεί επιλέξτε *ΦΟΡΤΩΣΗ ΚΟΣΜΟΥ (LOAD WORLD)*, όπως φαίνεται και στην εικόνα παρακάτω:



Στη συνέχεια πατήστε την επιλογή *Όλοι (All)* ώστε να εμφανιστούν όλοι οι διαθέσιμοι κόσμοι που υπάρχουν. Εμείς είπαμε θέλουμε να φτιάξουμε από την αρχή έναν καινούργιο κόσμο, οπότε ψάξτε στους κόσμους αυτούς, χρησιμοποιώντας τα βέλη στο πληκτρολόγιο ή τη ροδέλα του ποντικιού, μέχρι να βρείτε την επιλογή ***Νέος Κόσμος (New World)***. Πατήστε *Παίξε (Play)* και περιμένετε να φορτώσει.

Θα πρέπει να σας εμφανίζετε ένας κενός κόσμος σαν αυτόν της επόμενης εικόνας.

Αντί να ψάξετε να βρείτε την επιλογή Νέος Κόσμος (New World), επιλέξτε όποιον κόσμο θέλετε, πατήστε Παίξε (Play) και περιμένετε να φορτώσει. Μόλις φορτώσει πατήστε το πλήκτρο Esc στο πληκτρολόγιο και από την Παλέτα Εργαλείων (Tool Palette) που θα εμφανιστεί κάτω πατήστε το Αρχικό Μενού (Home Menu) . Από εκεί επιλέξτε Νέος κενός κόσμος (New empty world).

Πλέον είμαστε έτοιμοι να δημιουργήσουμε! Οι επιλογές είναι πραγματικά πολλές για το πώς θα είναι ο κόσμος μας. Για το πρώτο μας παιχνίδι, είπαμε ότι θέλουμε ένα καταπράσινο από γρασίδι τοπίο. Στον κενό κόσμο πατήστε το πλήκτρο Esc ώστε να εμφανιστεί **η *Παλέτα Εργαλείων (Tool Palette)***, όπως φαίνεται στην παρακάτω εικόνα.

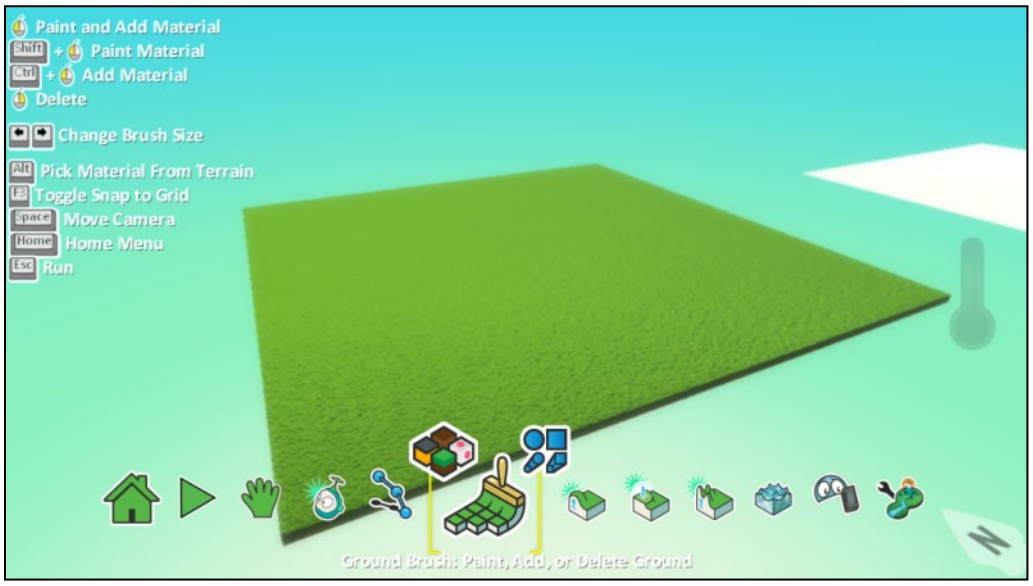
Αρχικά, θέλουμε να προσθέσουμε έδαφος. Κάπου στη μέση θα βρείτε τη ***Βούρτσα Εδάφους (Ground Brush)***. Πατήστε πάνω της. Θα δείτε ότι θα εμφανιστούν πάνω αριστερά και πάνω δεξιά της δύο άλλα εικονίδια.

Επιλέξτε το αριστερό (*επιλογή αλλαγής του υλικού*). Εμφανίζεται μια λίστα με διαφορετικούς τύπους εδάφους και μπορείτε να διαλέξετε τον τύπο εδάφους που θα προσθέσετε με τη *Βούρτσα*. Εμείς θέλουμε πράσινο με γρασίδι. Ψάξτε λοιπόν μέχρι να βρείτε το κατάλληλο έδαφος και απλά επιλέξτε το (έδαφος με νούμερο 15).

Τώρα μπορείτε να αρχίσετε να *Προσθέτετε Έδαφος (Add Ground)* στον κόσμο. Αρκεί να κάνετε αριστερό κλικ με το ποντίκι σε διάφορα σημεία του κόσμου. Προσπαθήστε να δημιουργήσετε την πρώτη σας πίστα.

Αν δεν σας πολυαρέσει το αποτέλεσμα, μην απογοητεύεστε! Αρχή είναι.

Μάλλον δεν είναι ιδιαίτερα ελκυστική. Μήπως θα ήταν καλύτερα να μεγαλώσετε πρώτα λίγο το μέγεθος της ***Βούρτσας Εδάφους (Ground Brush)*** και μετά να σχεδιάσετε την πίστα; Το MSKodu, για να σας βοηθήσει, σας δίνει τη δυνατότητα να μεγαλώσετε το μέγεθος της *Βούρτσας* πατώντας το δεξί βέλος του πληκτρολογίου και να μικρύνετε το μέγεθος της *Βούρτσας* πατώντας το αριστερό. Έτσι μπορείτε να προσθέσετε μεγάλα παρόμοια κομμάτια πίστας πιο γρήγορα. Συνεπώς μπορούμε ακόμη και με ένα μόνο κλικ να δημιουργήσουμε το ορθογώνιο γρασίδι που θέλουμε για την πίστα μας. Αν θέλετε να διαγράψετε κομμάτια πίστας, πατήστε το δεξί πλήκτρο του ποντικιού.



Είδατε πόσο απλό είναι; Θα μάθετε να χρησιμοποιείτε καλύτερα τη *Βούρτσα Εδάφους (Ground Brush)* καθώς και άλλα εργαλεία στο επόμενο κεφάλαιο. Έτσι θα μπορείτε να δημιουργείτε πιο πολύπλοκες πίστες με βουνά, κοιλάδες, λίμνες και ότι άλλο μπορείτε να φανταστείτε!

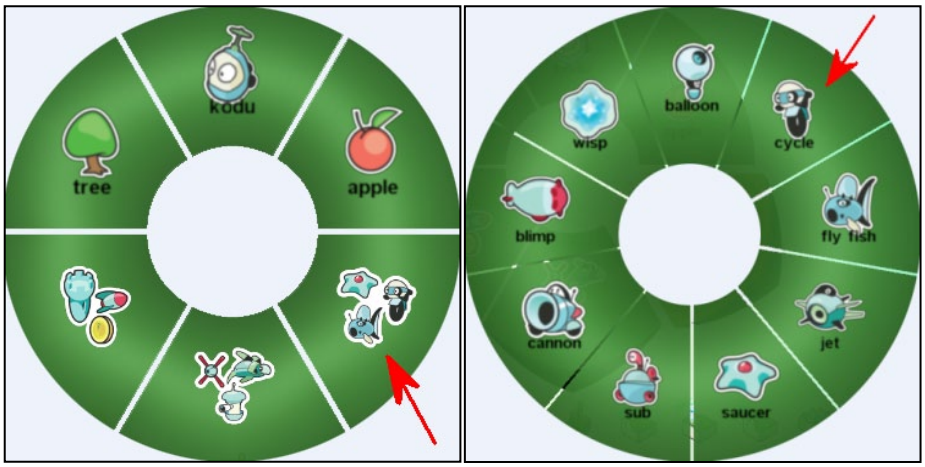
Τώρα, αφού ο κόσμος είναι έτοιμος, καιρός να εισάγουμε τα αντικείμενά μας.

3.6 Δημιουργώ τα Αντικείμενα

Μπορείτε να βρείτε όλα τα διαθέσιμα αντικείμενα στο MSKodu, αν επιλέξετε το *Εργαλείο Αντικειμένων (Object Tool)* από την παλέτα εργαλείων. Αφού το επιλέξετε, κάντε αριστερό κλικ με το ποντίκι σας σε κάποιο σημείο της πίστας και θα σας εμφανιστεί ένα μενού «πίτα» (όπως λέγεται!) με όλα τα αντικείμενα που μπορείτε να εισάγετε.

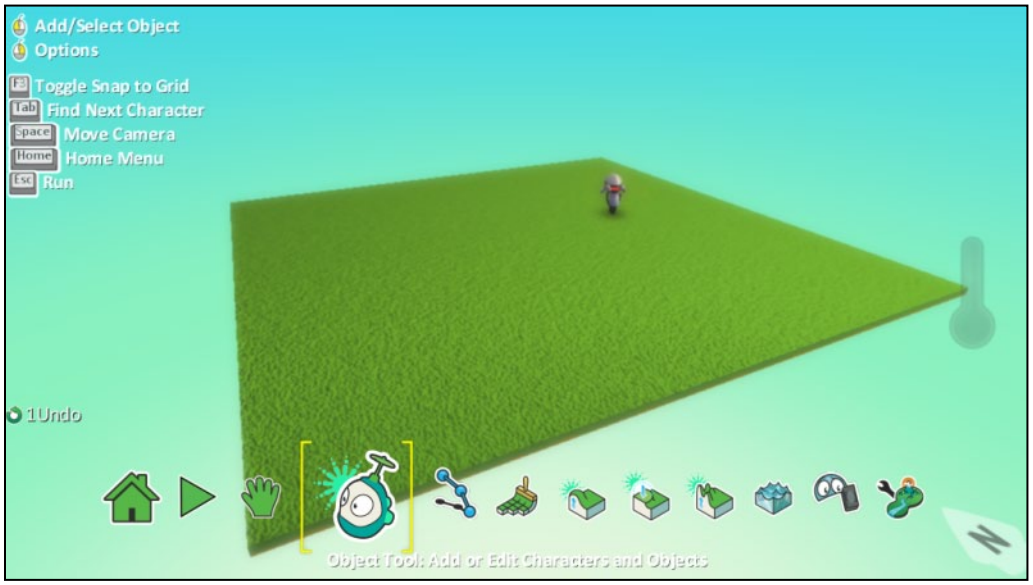
Η πρώτη πίτα έχει τρία αντικείμενα (ένα δέντρο, τον Kodu, και ένα μήλο) και τρεις κατηγορίες αντικειμένων που εμφανίζουν νέες πίτες και περιέχουν επιπλέον αντικείμενα. Ας ξεκινήσουμε με τον πρωταγωνιστή μας, τον *Μηχανάκια (Cycle)*!

Θα τον βρείτε εδώ:

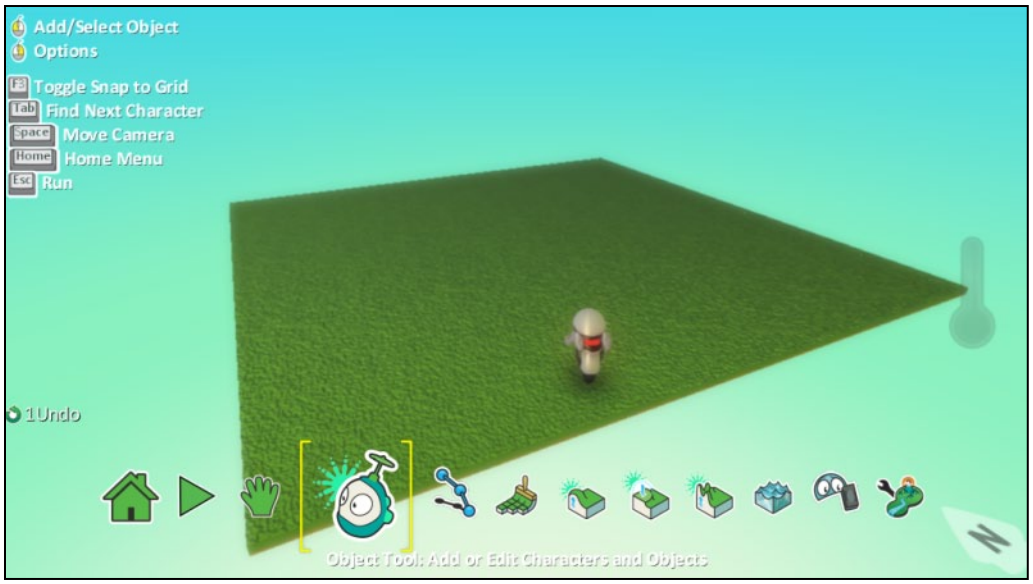


Επιλέξτε τον.

*Προσέξτε ότι όποτε δουλεύετε με αντικείμενα θα πρέπει να έχετε επιλέξει το **Εργαλείο Αντικειμένων (Object Tool)**, αλλιώς το MSKodu δεν θα ανταποκρίνεται αν κάνετε αριστερό ή δεξί κλικ με το ποντίκι σε κάποιο αντικείμενο!*



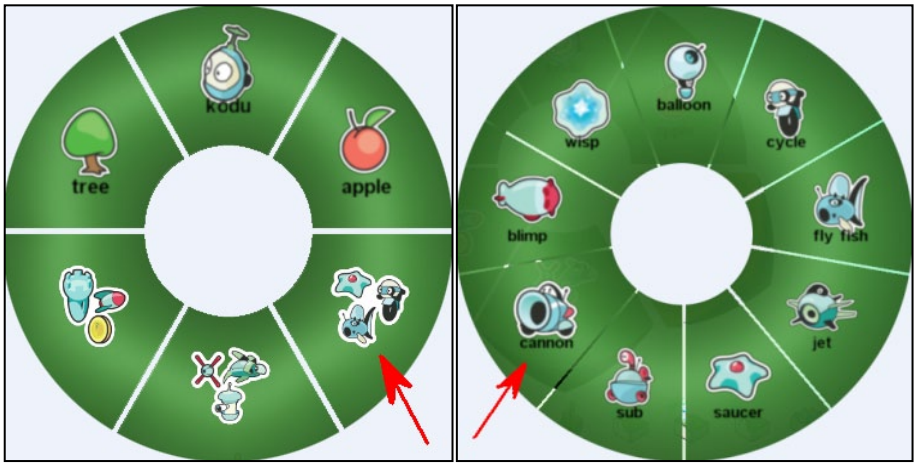
Είχαμε πει ότι θα τοποθετήσουμε τον *Μηχανάκια* νότια. Μετακινήστε λοιπόν τον *Μηχανάκια* που μόλις εισάγατε στη σωστή θέση. Πώς; Μπορείτε να αλλάζετε τη θέση των αντικειμένων κάνοντας απλώς αριστερό κλικ πάνω τους, κρατώντας πατημένο το αριστερό πλήκτρο του ποντικιού και μετακινώντας το ποντίκι στη θέση που θέλετε να μεταφερθεί το εκάστοτε αντικείμενο.



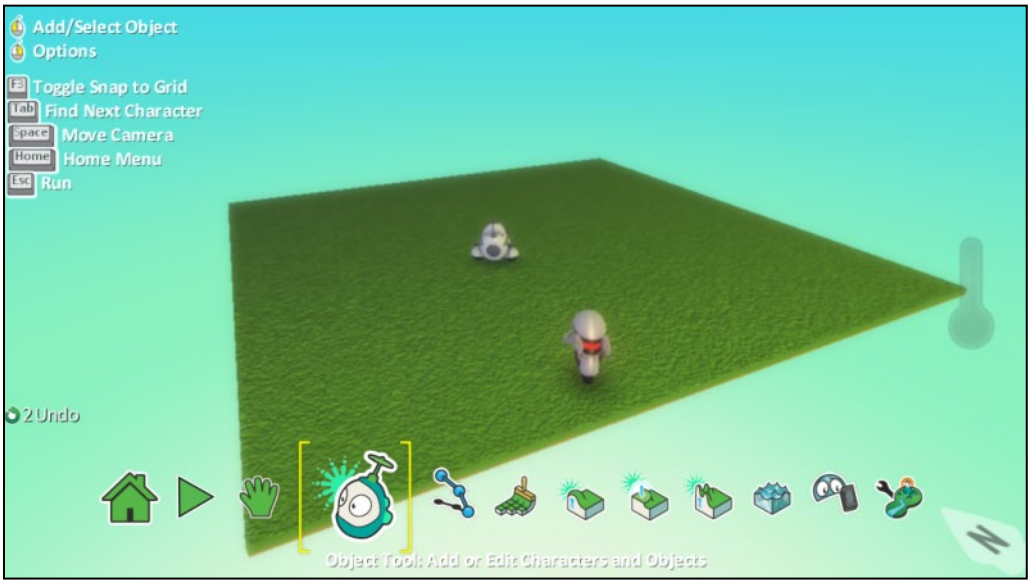
Πάμε για το επόμενο αντικείμενο. Μετακινήστε το ποντίκι ώστε ο δείκτης να βρεθεί στο κέντρο της πίστας και πατήστε το αριστερό πλήκτρο του ποντικιού, όπως και πριν, ώστε να εμφανιστούν πάλι τα αντικείμενα που μπορείτε να εισάγετε.

Επόμενο αντικείμενο για εισαγωγή είναι το *Κανόνι (Cannon)*!

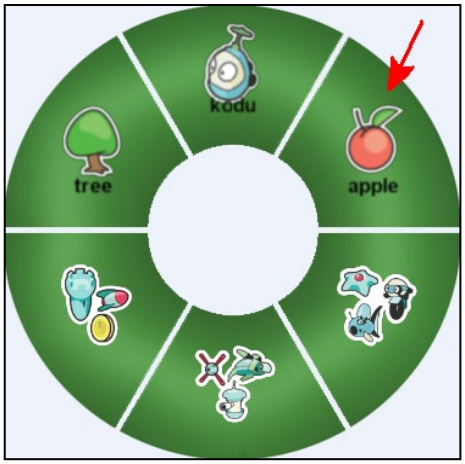
Θα το βρείτε εδώ:



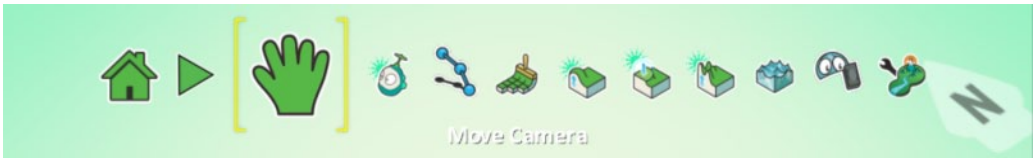
Επιλέξτε το.



Τέλος, θα εισάγουμε τα τρία *Μήλα (Apples)*, ένα βόρεια, ένα δυτικά και ένα ανατολικά. Και το *Μήλο (Apple)* είναι ένα αντικείμενο είπαμε. Θα το βρείτε εδώ:



Στο σημείο αυτό ίσως να αναρωτιέστε για το πώς μπορείτε να πλοηγηθείτε μέσα στον κόσμο που δημιουργείτε. Είστε υποχρεωμένοι να βλέπετε τον κόσμο μόνο από μια θέση; Πώς θα μπορούσατε να τον δείτε από διαφορετική οπτική γωνία; Κάθε στιγμή βλέπετε τον κόσμο του MSKodu μέσα από μια κάμερα. Και το MSKodu μας δίνει τη δυνατότητα να μετακινήσουμε την κάμερα από την επιλογή ***Μετακίνηση Κάμερας (Move Camera)*** που βρίσκεται στην *Παλέτα Εργαλείων (Tool Palette)* όπως φαίνεται από την επόμενη εικόνα.

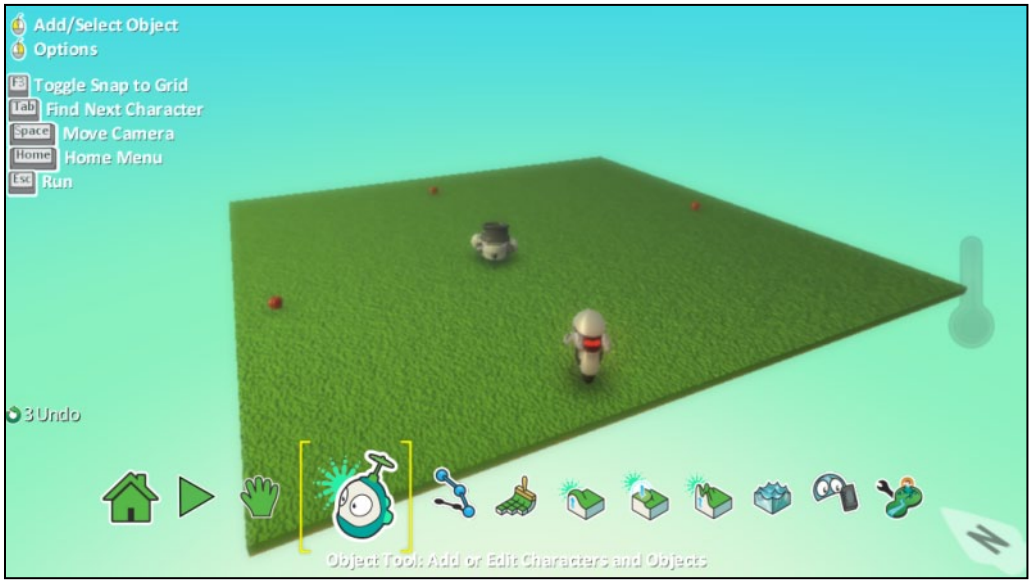


Επειδή όμως πολλές φορές θα θέλετε να μετακινήσετε την κάμερα του κόσμου ενώ πραγματοποιείτε μια άλλη εργασία, το MSKodu μας ενεργοποιεί όλες τις προηγούμενες επιλογές μετακίνησης με το ποντίκι απλώς πατώντας παράλληλα και το πλήκτρο Space.

Όταν έχετε ενεργοποιήσει το «χεράκι», τότε με το ποντίκι μπορείτε να μετακινηθείτε σε όποια θέση του κόσμου επιθυμείτε, από όποια οπτική γωνιά σας βολεύει.

Ενέργεια ποντικιού	**Κίνηση κάμερας**
Μετακίνηση ποντικιού με πατημένο το **αριστερό πλήκτρο**	η κάμερα μετακινείται ανάλογα προς τα βόρεια, νότια, ανατολικά και δυτικά της πίστας
Μετακίνηση ποντικιού με πατημένο το **δεξί πλήκτρο**	η κάμερα αλλάζει θέση στους τρεις άξονες, δηλαδή αλλάζει ύψος και περιστρέφει τον κόσμο μας
πλήκτρο κύλισης του ποντικιού	**αλλάζει η απόσταση** της κάμερας από τον κόσμο (zoom in –zoom out)

Θαυμάστε το δημιούργημά σας.



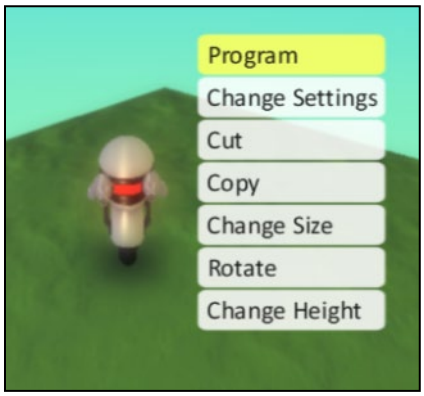
Ολοκληρώσαμε τον κόσμο-πίστα, τοποθετήσαμε και τα αντικείμενά μας, καιρός λοιπόν να προγραμματίσουμε! Καιρός δηλαδή να δώσουμε συγκεκριμένες συμπεριφορές στα αντικείμενά μας.

3.7 Δίνω Συμπεριφορές στα Αντικείμενα

Πατήστε την *Αρχή Παιχνιδιού (Play Game)* από την *Παλέτα Εργαλείων (Tool Palette)* για να παίξουμε το παιχνίδι που δημιουργήσαμε. Τι παρατηρείτε; Τι γίνεται; Ακριβώς! Δεν γίνεται ΤΙΠΟΤΑ απολύτως! Λογικό δεν είναι; Αφού δεν έχουμε δώσει καμιά συμπεριφορά στα αντικείμενα.

Για να προγραμματίσουμε ένα αντικείμενο πρέπει να πατήσουμε Esc για να εμφανιστεί η *Παλέτα Εργαλείων (Tool Palette)*, να επιλέξουμε το *Εργαλείο Αντικειμένων (Object Tool)* και να κάνουμε **δεξί κλικ** με το ποντίκι μας πάνω στο αντικείμενο στο οποίο θέλουμε να αποδώσουμε συμπεριφορά.

Ας ξεκινήσουμε με τον *Μηχανάκια (Cycle)*. Κάνουμε δεξί κλικ πάνω του. Από τις επιλογές που θα εμφανιστούν πατάμε στο ***Προγραμμάτισε (Program)***.



Θα παρατηρήσετε εκεί ότι η εντολή που μπορούμε να δημιουργήσουμε έχει συγκεκριμένη μορφή, που αποτελείται από δύο τμήματα:

ΟΤΑΝ (WHEN)… **ΚΑΝΕ** (DO)…

Σας θυμίζει κάτι; Μήπως τη συμπεριφορά των αντικειμένων στο παιχνίδι Pacman που είχαμε δει στην αρχή;

Μήπως επίσης σας θυμίζει τις συμπεριφορές των αντικειμένων του κόσμου μας, όπως αποφασίσαμε να τις προσδιορίσουμε στην παράγραφο 3.4;

Μηχανάκιας

Ας θυμηθούμε την πρώτη συμπεριφορά για τον *Μηχανάκια (Cycle):*

1η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να κινείται στην αντίστοιχη κατεύθυνση πάνω στην πίστα (**ΕΝΕΡΓΕΙΑ**).

Αισθητήρες

Λογικά αναρωτιέστε πώς είναι δυνατόν ο *Μηχανάκιας* ή ένα οποιοδήποτε αντικείμενο να αισθάνεται το πάτημα ενός πλήκτρου στο πληκτρολόγιο! Και όμως, κάθε αντικείμενο στο MSKodu έχει μία σειρά από αισθητήρες. Τα αντικείμενα μπορούν να αντιληφθούν για παράδειγμα ποιο πλήκτρο πατάει ο χρήστης στο πληκτρολόγιο και επομένως εμείς ως προγραμματιστές να προσδιορίσουμε αντίστοιχες ενέργειες για το αντικείμενο. Υπάρχουν αισθητήρες ακοής, αφής, όρασης κτλ που θα τους μελετήσουμε σιγά σιγά στα επόμενα κεφάλαια!

Για τη συγκεκριμένη συμπεριφορά θέλουμε να χρησιμοποιήσουμε τον **αισθητήρα αλληλεπίδρασης με το *Πληκτρολόγιο (Keyboard)***. Αυτό βεβαίως δεν μας είναι αρκετό αφού πρέπει να προσδιορίσουμε ποιο πλήκτρο του πληκτρολογίου θα προκαλεί την επιθυμητή ενέργεια. Κάθε αισθητήρας συνήθως συνοδεύεται από κάποια **προσδιοριστικά**, από στοιχεία προσδιορίζουν καλύτερα το γεγονός που θέλουμε να αντιληφθεί το αντικείμενό μας π.χ. όταν δει (αισθητήρας) ένα μήλο (προσδιοριστικό), ή όταν πατήσει (αισθητήρας) το space (προσδιοριστικό), ή όταν ακούσει (αισθητήρας) ένα αερόστατο (προσδιοριστικό).

Ενέργειες

Αφού τα αντικείμενά μας αντιληφθούν ένα γεγονός, θα πρέπει να εκτελέσουν κάποια ενέργεια. Κάθε αντικείμενο στο MSKodu έχει μια πληθώρα ενεργειών που μπορεί να εκτελέσει όπως να μιλήσει, να μετακινηθεί, παίξει ένα ήχο, να ρίξει έναν πύραυλο, να χρωματίσει κτλ.

Η ενέργεια που θέλουμε να εκτελεστεί με το πάτημα των πλήκτρων του πληκτρολογίου στο παράδειγμά μας, είναι η μετακίνηση του Μηχανάκια, δηλαδή η ***Κινήσου (Move)***.

Επιλέξτε λοιπόν *Προγραμμάτισε (Program)* στον *Μηχανάκια* και δημιουργήστε την εντολή:

Απλώς πατήστε πάνω στο "+" του ***ΟΤΑΝ (WHEN)*** και επιλέξτε τον αισθητήρα ***Πληκτρολόγιο (Keyboard)***

και μετά ξαναπατήστε στο "+" του ***ΟΤΑΝ (WHEN)*** και επιλέξτε το προσδιοριστικό του αισθητήρα, που εδώ θα είναι τα ***Βέλη (Arrows)***. Μετά πατήστε πάνω στο "+" του ***ΚΑΝΕ (DO)*** και επιλέξτε την ενέργεια ***Κινήσου (Move)***.

Έτοιμη η πρώτη εντολή μας! Δεν τη δοκιμάζουμε να δούμε αν δουλεύει όπως θέλουμε; Πατήστε Esc και επιλέξτε την ***Αρχή Παιχνιδιού (Play Game)*** από την *Παλέτα Εργαλείων (Tool Palette)*. Δοκιμάστε να κινήσετε τον *Μηχανάκια*. Όλα εντάξει; Ας προχωρήσουμε στη δεύτερη συμπεριφορά:

2η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* ακουμπήσει ένα *Μήλο* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τρώει το *Μήλο* (**ΕΝΕΡΓΕΙΑ**).

Τα αντικείμενα στο Kodu έχουν στη διάθεσή τους τον αισθητήρα ***Πέσεις Πάνω (bump)*** μπορούν δηλαδή να αντιληφθούν πότε πέφτουν πάνω σε ένα άλλο αντικείμενο. Ο αισθητήρας απαιτεί ως προσδιοριστικό το όνομα του άλλου αντικειμένου της σύγκρουσης, στη δική μας περίπτωση του μήλου.

Για να υλοποιήσετε αυτό το τμήμα της συμπεριφοράς, πατήστε πάνω στο "+" του ***ΟΤΑΝ (WHEN)*** της δεύτερης γραμμής και επιλέξτε τον αισθητήρα ***Πέσεις Πάνω (Bump)*** και μετά ξαναπατήστε στο "+" του ***ΟΤΑΝ (WHEN)*** και επιλέξτε ως προσδιοριστικό του αισθητήρα το ***Μήλο (Apple)*** που βρίσκεται στην κατηγορία *αντικείμενα (objects)*. Θα πρέπει να βλέπετε στην οθόνη σας κάτι τέτοιο:

Η ΕΝΕΡΓΕΙΑ που θέλουμε να εκτελέσει ο *Μηχανάκιας* όταν συμβεί το παραπάνω γεγονός είναι να φάει το μήλο. Απίστευτο αλλά το Kodu μας παρέχει μια ενέργεια ***Φάε (Eat)***, που εξαφανίζει το αντίστοιχο αντικείμενο. Προσέξτε ότι για να εκτελεστεί επιτυχημένα η ενέργεια δεν αρκεί μόνο το ***Φάε (Eat)***, αλλά να προσδιορίσουμε το τι θα φάει! Στη συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε το προσδιοριστικό ***Το (it)*** δηλαδή θα πούμε στον Kodu «Φάε Το». Αυτό σημαίνει ότι θα φάει το αντικείμενο με το οποίο έχει έρθει σε επαφή, άρα το προσδιοριστικό ενέργειας ***Το (it)*** αναφέρεται στο αντικείμενο ***Μήλο (Apple)***.

Για να καταγράψετε την ενέργεια, πατήστε πάνω στο "+" του ***ΚΑΝΕ (DO)*** και επιλέξτε την ενέργεια ***Φάε (Eat)***

και μετά ξαναπατήστε στο "+" του ***ΚΑΝΕ (DO)*** και επιλέξτε το προσδιοριστικό ενέργειας ***Το (It)***. Θα πρέπει να βλέπετε στην οθόνη σας τις εξής εντολές:

Δοκιμάστε αν θέλετε όπως και πριν με την *Αρχή Παιχνιδιού (Play Game)* να δείτε αν όλα δουλεύουν όπως θέλουμε. Ας προχωρήσουμε και στην τρίτη συμπεριφορά:

3η **ΣΥΜΠΕΡΙΦΟΡΑ**: Θέλουμε

ΟΤΑΝ ο *Μηχανάκιας* έχει φάει όλα τα μήλα (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τερματίζει το παιχνίδι με νίκη για τον παίκτη (**ΕΝΕΡΓΕΙΑ**).

Αρχικά, σύμφωνα με τα παραπάνω, θα θέλαμε ο *Μηχανάκιας* να έχει έναν αισθητήρα «τα έφαγε όλα» ώστε να χτίσουμε την εντολή που σκεφτήκαμε. Δυστυχώς όμως δεν υπάρχει τέτοιος αισθητήρας στο MSKodu. Είναι λογικό μια γλώσσα προγραμματισμού να έχει ένα συγκεκριμένο ρεπερτόριο εντολών και συνεπώς να μην υλοποιεί οποιαδήποτε σκέψη μας έρχεται στο μυαλό. Δουλειά του προγραμματιστή είναι να μεταφράζει τις σκέψεις του βάσει των εντολών που έχει στη διάθεσή του. Έτσι, πρέπει να βρούμε έναν τρόπο για να αντιληφθεί ο *Μηχανάκιας* ότι έφαγε όλα τα μήλα. Θυμηθείτε ότι υπάρχει ο αισθητήρας της όρασης! Αν ο *Μηχανάκιας* έχει φάει όλα τα μήλα, τότε δεν θα υπάρχουν πάνω στην πίστα μας άλλα, πράγμα που σημαίνει ότι δεν θα μπορεί να δει κανένα μήλο. Συνεπώς θα μπορούσαμε να μετασχηματίσουμε την προηγούμενη εντολή σε:

ΟΤΑΝ ο *Μηχανάκιας* δεν βλέπει μήλα (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τερματίζει το παιχνίδι με νίκη για τον παίκτη (**ΕΝΕΡΓΕΙΑ**).

Οι δυο εντολές είναι ισοδύναμες.

Το αντικείμενο *Μηχανάκιας* αντιλαμβάνεται το γεγονός αυτό, με τον αισθητήρα ***Βλέπεις (See)*** και το προσδιοριστικό ***Μήλο (Apple)***. Πως όμως θα δηλώσουμε άρνηση; Το MSKodu δέχεται την άρνηση ως τελευταίο προσδιοριστικό του εκάστοτε αισθητήρα, δηλαδή πρέπει να προσθέσουμε το ***Δεν (Not)*** στο τέλος του γεγονότος. Λίγο περίεργο δεν ακούγεται σαν φράση; "*Βλέπεις Μήλο Δεν*"; Με τον τρόπο αυτό όμως ολόκληρο το κομμάτι του *ΟΤΑΝ* που βρίσκεται πριν από το ***Δεν (Not)***, δηλαδή το "*Βλέπεις Μήλο*", γίνεται άρνηση. Εκεί δηλαδή που το γεγονός θα προκαλούνταν όταν το αντικείμενό μας θα έβλεπε *Μήλο*, με το *Δεν* στο τέλος το γεγονός προκαλείται όταν το αντικείμενό μας **δεν** βλέπει *Μήλο*, το ανάποδο δηλαδή.

Για να χτίσετε την τελευταία εντολή του *Μηχανάκια,* πατήστε πάνω στο "+" του ***ΟΤΑΝ (WHEN)*** και επιλέξτε τον αισθητήρα ***Βλέπεις (See)***

Μετά ξαναπατήστε στο "+" του ***ΟΤΑΝ (WHEN)*** και επιλέξτε το πρώτο προσδιοριστικό του αισθητήρα, που θέλουμε να είναι το ***Μήλο (Apple)*** - θα το βρείτε στην κατηγορία *αντικείμενα (objects)*. Ξαναπατήστε στο "+" του ***ΟΤΑΝ (WHEN)*** άλλη μια φορά και επιλέξτε το δεύτερο προσδιοριστικό του αισθητήρα, που θα είναι το ***Δεν (Not)***.

Θα πρέπει να βλέπετε κάτι τέτοιο:

Τέλος, η ενέργεια που θέλουμε να εκτελείται όταν δεν υπάρχουν μήλα στην πίστα, είναι να ανακηρυσσόμαστε νικητές. Το MSKodu μας παρέχει την ενέργεια ***Νίκη (Win)*** που τερματίζει το παιχνίδι και αναφέρει ότι είμαστε οι νικητές του παιχνιδιού. Πατήστε λοιπόν πάνω στο "+" του ***KANE (DO)*** και επιλέξτε την ενέργεια ***Νίκη (Win)***. Θα τη βρείτε στην κατηγορία (*Παιχνίδι - Game)* εδώ:

Οι εντολές του *Μηχανάκια* θα πρέπει να είναι οι εξής:

Δοκιμάστε ξανά το παιχνίδι για να δείτε αν όλα δουλεύουν όπως θέλουμε.

Κανόνι

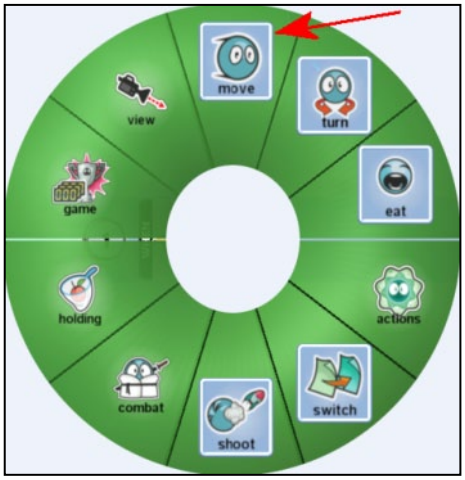
Συνεχίζουμε με το δεύτερο αντικείμενο του παιχνιδιού μας, το *Κανόνι (Cannon)*. Η βασική συμπεριφορά που θέλουμε να επιδείξει είναι:

ΟΤΑΝ το *Κανόνι* βλέπει τον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να τον κυνηγάει (**ΕΝΕΡΓΕΙΑ**).

Έχοντας την εμπειρία των προηγούμενων εντολών, τα πράγματα είναι πιο εύκολα. Πατάμε δεξί κλικ στο κανόνι και επιλέγουμε ***Προγραμμάτισε (Program)***. Το τμήμα του αισθητήρα της εντολής είναι εύκολο, καθώς χρησιμοποιήσαμε και προηγουμένως τον αισθητήρα ***Βλέπω (See)***. Δημιουργούμε το πρώτο μέρος της εντολής:

Το τμήμα της ενέργειας είναι λίγο δυσκολότερο γιατί για άλλη μια φορά δεν υπάρχει η ενέργεια που εκφράστηκε με προφορικό λόγο, δηλαδή δεν υπάρχει η ενέργεια «*κυνήγησε*». Πρέπει να μεταφράσουμε την εντολή «κυνήγησε» με βάσει τις διαθέσιμες εντολές του MSKodu. Μόλις δει το *Κανόνι* τον *Μηχανάκια*, θέλουμε να αρχίσει να μετακινείτε προς αυτόν. Μήπως η εντολή ***Κινήσου (Move)***, μπορεί να μας βοηθήσει;

Πατήστε πάνω στο "+" του ***ΚΑΝΕ (DO)*** και επιλέξτε την ενέργεια *Κινήσου (Move)*



Τι διαθέσιμα προσδιοριστικά έχει η εντολή Κινήσου ***Κινήσου (Move)***; Ξαναπατήστε στο "+" του ***ΚΑΝΕ (DO)*** και επιλέξτε το προσδιοριστικό ενέργειας ***Προς το μέρος του (Toward)***.

Παίξτε το παιχνίδι. Τι παρατηρείτε;

Το προσδιοριστικό ***Προς το μέρος του (Toward)*** καθόρισε ότι η κίνηση θα γίνεται προς το αντικείμενο που υπάρχει στον αισθητήρα, δηλαδή προς τον *Μηχανάκια*. Όπως θα παρατηρήσατε και από την αντίστοιχη πίτα, η εντολή κινήσου έχει πολλά ενδιαφέροντα προσδιοριστικά! Μην ανησυχείτε θα τα δούμε αναλυτικά σε επόμενα κεφάλαια.

Τα μήλα

Και, τέλος, έχουμε να προγραμματίσουμε τα τρία *Μήλα (Apples).* Η συμπεριφορά που θέλουμε να έχουμε είναι για κάθε *Μήλο*:

ΟΤΑΝ ακουμπήσει με τον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** να εξαφανίζεται (**ΕΝΕΡΓΕΙΑ**).

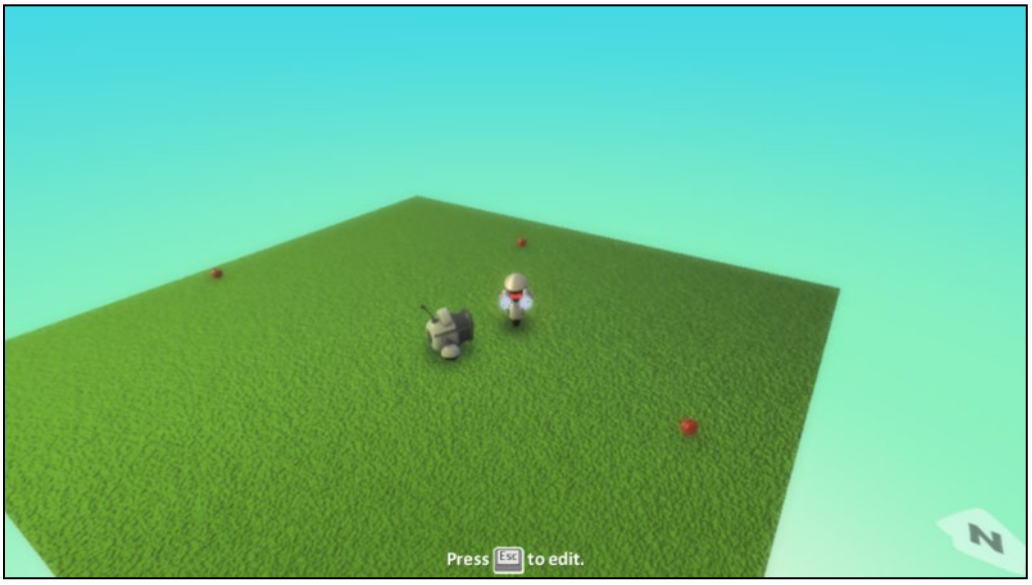
Για σκεφτείτε όμως, χρειάζεται να δώσουμε αντίστοιχη εντολή στα *Μήλα;*

Θυμηθείτε τη 2η συμπεριφορά του *Μηχανάκια* και την αντίστοιχη εντολή που γράψαμε στο MSKodu:

Ήδη με τη συμπεριφορά αυτή του *Μηχανάκια* έχουμε εξασφαλίσει ότι κάθε φορά που ο *Μηχανάκιας* ακουμπάει ένα *Μήλο* (ή ένα *Μήλο* ακουμπάει με τον *Μηχανάκια,* το ίδιο είναι) θα το τρώει, οπότε αυτό θα εξαφανίζεται.

Ήδη, δηλαδή, με τη συμπεριφορά αυτή του *Μηχανάκια* έχουμε πετύχει να δώσουμε και την επιθυμητή συμπεριφορά στα τρία *Μήλα*. Θα ήταν πλεονασμός να προσθέταμε τρεις ακόμη εντολές (μία για κάθε *Μήλο*) που να αντιστοιχούν στην παραπάνω συμπεριφορά κάθε *Μήλου*. Δεν χρειάζεται.

Κόσμος έτοιμος, αντικείμενα με συμπεριφορές έτοιμα. Με άλλα λόγια είμαστε έτοιμοι να δοκιμάσουμε το παιχνίδι μας. Πατήστε Esc και μετά ***Αρχή Παιχνιδιού (Play Game)***. Καλή διασκέδαση!



Περίληψη

Στο κεφάλαιο αυτό κάναμε μια γρήγορη περιήγηση σε όλα τα χαρακτηριστικά του MSKodu για να πάρουμε μια γεύση για αυτά που ακολουθούν. Πρέπει να θυμάστε ότι ουσιαστικά ο προγραμματισμός στο MSKodu είναι αντικειμενοστραφής αφού βασίζεται στη δημιουργία αντικειμένων και την απόδοση συμπεριφορών σε αυτά, αλλά και γεγονοστραφής, αφού οι συμπεριφορές είναι ενέργειες που εκτελούνται όταν συμβαίνουν ορισμένα γεγονότα. **ΣΥΜΠΕΡΙΦΟΡΑ** στο MSKodu είναι η **ΕΝΕΡΓΕΙΑ** που προγραμματίζουμε ένα αντικείμενο να κάνει όταν μέσω των **ΑΙΣΘΗΤΗΡΩΝ** που διαθέτει, αντιληφθεί κάποιο συγκεκριμένο **ΓΕΓΟΝΟΣ** στο περιβάλλον του. Δηλαδή στις εντολές του MSKodu, χρησιμοποιούμε *αισθητήρες* και *ενέργειες*. Είδαμε πώς να δημιουργούμε ένα απλό κόσμο και πώς να εισάγουμε τα αντικείμενα που θέλουμε και τέλος προγραμματίσαμε για πρώτη φορά δίνοντας συγκεκριμένες συμπεριφορές σε κάθε ένα αντικείμενο. Είμαστε έτοιμοι να φτιάξουμε πιο ελκυστικούς κόσμους!

Ερωτήσεις

1. Τι θα γινόταν στο παιχνίδι μας, αν στην εντολή που δώσαμε στο *Κανόνι (Cannon)* αφαιρούσαμε το προσδιοριστικό της ενέργειας *Κινήσου (Move)*, δηλαδή το *Προς το μέρος του (Toward)*;

2. Πως πιστεύετε (διαισθητικά) ότι λειτουργούν τα προσδιοριστικά της ενέργειας *Κινήσου (Move)* που δεν χρησιμοποιήσαμε;

3. Τι θα γινόταν αν στην τρίτη εντολή του *Μηχανάκια (Cycle)* ξεχνούσαμε να βάλουμε το *Δεν (Not)*;

4. Προσδιορίστε δύο συμπεριφορές του αυτοκινήτου-πρωταγωνιστή σε ένα παιχνίδι ράλι.

Δραστηριότητες

1. Είχαμε πει για το παιχνίδι που δημιουργήσαμε (όταν το σκεφτόμασταν στην αρχή) ότι μάλλον θα είχε περισσότερο νόημα αν το κανόνι έριχνε πυραύλους εναντίον του *Μηχανάκια*! Προσθέστε λοιπόν στο *Κανόνι (Cannon)* άλλη μία εντολή,

ΟΤΑΝ *Βλέπεις (See)* τον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** *Πυροβόλησε (Shoot) Πύραυλο (Missile)* (**ΕΝΕΡΓΕΙΑ**)

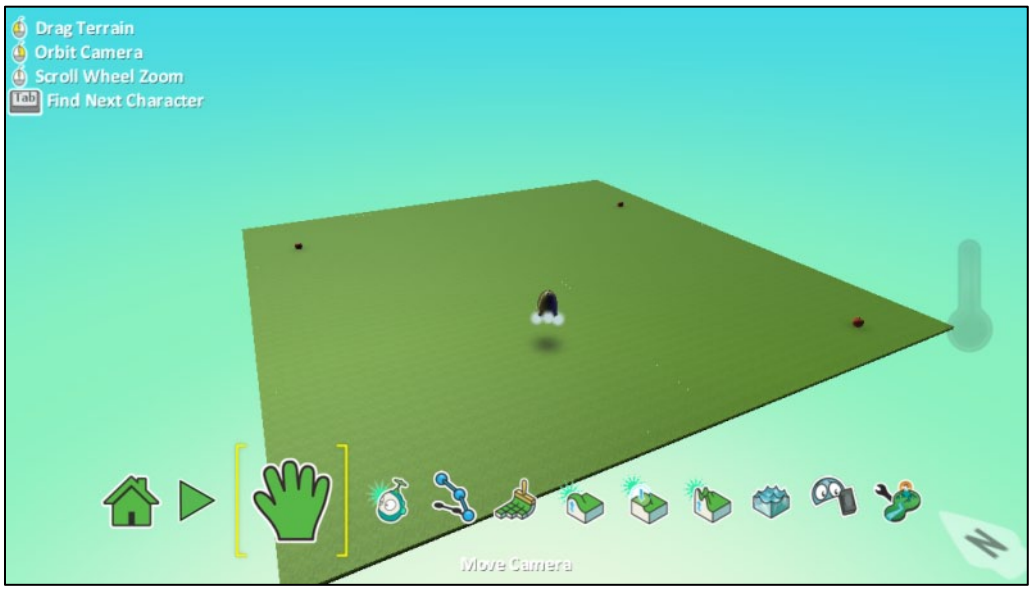
Η συγκεκριμένη λύση βρίσκεται στο αρχείο **[03_02.Kodu]**.

Πατήστε *Αρχή Παιχνιδιού (Play Game)* και δοκιμάστε να παίξετε. Λίγο αδύνατο να κερδίσει ο παίκτης, έτσι δεν είναι;! Δεν έχει πού να ξεφύγει ή που να κρυφτεί. Δοκιμάστε τότε κάτι άλλο: Αντί για την παραπάνω εντολή, προσθέστε μία άλλη εντολή στο *Κανόνι* που να αντιστοιχεί στην εξής **ΣΥΜΠΕΡΙΦΟΡΑ**:

ΟΤΑΝ *Πέσεις Πάνω (Bump)* στον *Μηχανάκια* (**ΓΕΓΟΝΟΣ**), **ΤΟΤΕ** *Φάε (Eat) Τον (It)* (**ΕΝΕΡΓΕΙΑ**)

Η συγκεκριμένη λύση βρίσκεται στο αρχείο **[03_03.Kodu]**). Πατήστε *Αρχή Παιχνιδιού (Play Game)* και δοκιμάστε να παίξετε. Δεν είναι πιο εύκολο τώρα να κερδίσει ο παίκτης;

2. Δημιουργήστε έναν πολύ απλό επίπεδο κόσμο και τοποθετήστε πάνω του τα εξής αντικείμενα, όπως φαίνονται και στην εικόνα παρακάτω: τον *Kodu* και 3 *Μήλα (Apples).*



Δώστε την εξής συμπεριφορά στον *Kodu*: ΟΤΑΝ βλέπει ένα μήλο ΤΟΤΕ να κινείται κατά πάνω του και να το τρώει. Δώστε αυτή τη συμπεριφορά, ώστε ο Kodu να κινείται από μόνος του προς τα μήλα μόλις πατήσετε *Αρχή Παιχνιδιού (Play Game).* Δε θα πρέπει, δηλαδή, να τον κινείτε εσείς με το πληκτρολόγιο.

Κεφάλαιο 4ο: Δημιούργησε τον κόσμο σου

4.1 Τα εργαλεία για τη δημιουργία κόσμων

Στο κεφάλαιο αυτό θα επικεντρωθούμε στο σχεδιασμό της πίστας που θα χρησιμοποιήσουμε στο παιχνίδι μας. Ο κόσμος θα υποδεχτεί στη συνέχεια τα αντικείμενα και πολλές από τις ιδιότητές του θα επηρεάσουν τη φύση, την αισθητική και το ενδιαφέρον των παιχνιδιών μας.

Ας ξεκινήσουμε λοιπόν με την περιγραφή των εργαλείων και ας αφήσουμε τη φαντασία μας ελεύθερη να δημιουργήσει τα πιο απίστευτα σκηνικά!

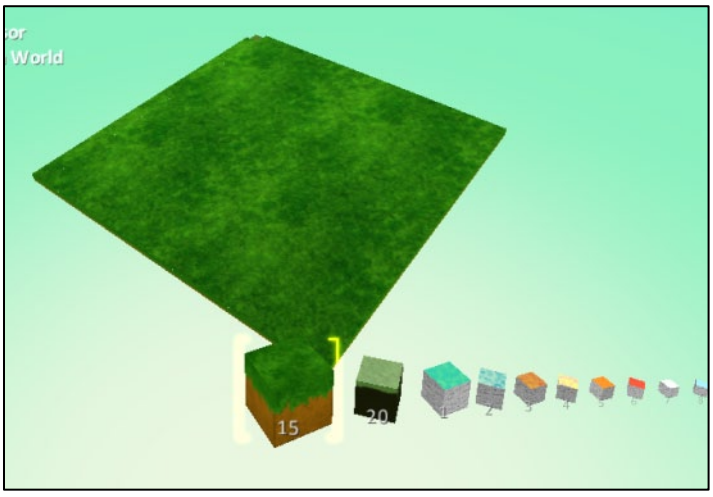
4.1.1 Βούρτσα Εδάφους (Ground Brush)

Πριν όμως προχωρήσουμε παρακάτω ίσως θα πρέπει να υπενθυμίσουμε τη διαδικασία φόρτωσης ενός άδειου κόσμου. Από το ***ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ (MAIN MENU)*** επιλέγουμε ***ΦΟΡΤΩΣΗ ΚΟΣΜΟΥ (LOAD WORLD)*** και στη συνέχεια χρησιμοποιώντας τα βελάκια βρίσκουμε τον ***Άδειο Κόσμο (Empty World)***. Πατάμε το ***Παίξε (Play)*** και περιμένουμε να φορτώσει ο κόσμος μας. Στη συνέχεια πατάμε Escape και εμφανίζεται η παλέτα εργαλείων για την κατασκευή του παιχνιδιού μας.

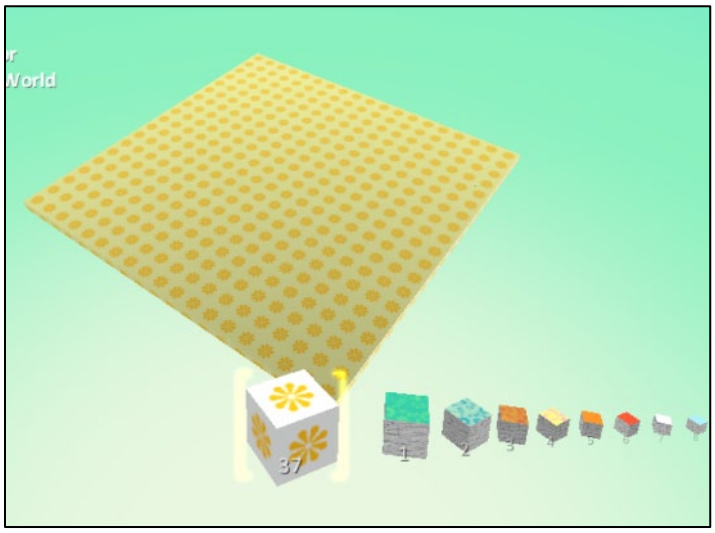
Πρώτο βήμα για τη δημιουργία μιας πίστας είναι η διαμόρφωση του εδάφους. Αρχικά επιλέγουμε το εργαλείο ***Βούρτσα Εδάφους: Προσθέστε Έδαφος/Διαγράψτε Έδαφος (Ground Brush: Paint, add or delete ground)***.

Όπως έχουμε αναφέρει και στο κεφάλαιο 3, πάνω αριστερά και δεξιά της *Βούρτσας (Brush)* εμφανίζονται δυο επιλογές που αφορούν το είδος και το σχήμα του εδάφους που επιθυμούμε να εισάγουμε.

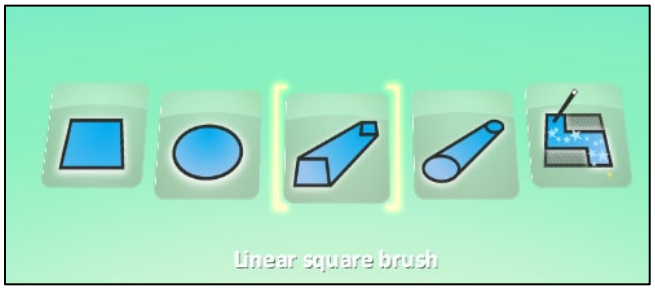
Με την επιλογή ***αλλαγής του υλικού του εδάφους*** μπορούμε να διαλέξουμε το χρώμα και την αίσθηση που θέλουμε το έδαφός μας να αποτυπώνει. Κάθε υλικό που προσφέρεται για δημιουργία εδάφους έχει ένα χαρακτηριστικό αριθμό που το προσδιορίζει. Για παράδειγμα όπως φαίνεται και στις εικόνες που ακολουθούν μπορούμε να επιλέξουμε το έδαφος να μοιάζει με γρασίδι (νούμερο 15).



Ή μπορούμε να δημιουργήσουμε ένα έδαφος γεμάτο λουλούδια (νούμερο 37)!

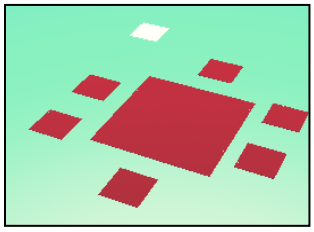


Με την επιλογή ***αλλαγής σχήματος βούρτσας*** καθορίζουμε το σχήμα της βούρτσας με την οποία δημιουργούμε το έδαφος. Εδώ υπάρχουν οι εξής επιλογές:



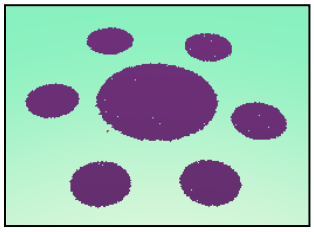
Τετράγωνη βούρτσα (Square brush):

Είναι το προεπιλεγμένο σχήμα της βούρτσας μας και μας επιτρέπει να προσθέτουμε **τετράγωνα εδάφους** στον κόσμο μας όπως φαίνονται και στον παρακάτω κόσμο.



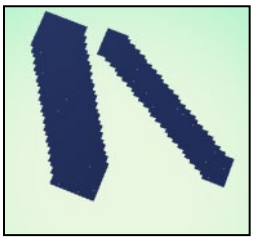
Στρογγυλή βούρτσα (Hard round brush):

Μας δίνει τη δυνατότητα να προσθέσουμε **κύκλους εδάφους** στον κόσμο μας, όπως φαίνονται και στον παρακάτω κόσμο.



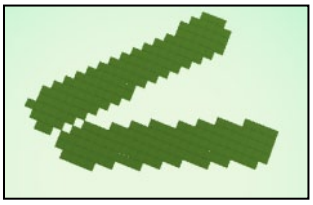
Τετράγωνη βούρτσα γραμμής (Linear square brush):

Επιτρέπει τη δημιουργία **ευθειών γραμμών εδάφους με τετραγωνισμένες άκρες**. Αρκεί να πατήσουμε το αριστερό πλήκτρο του ποντικιού και να σύρουμε το ποντίκι μέχρι το σημείο που επιθυμούμε.



Στρογγυλή βούρτσα γραμμής (Linear hard round brush):

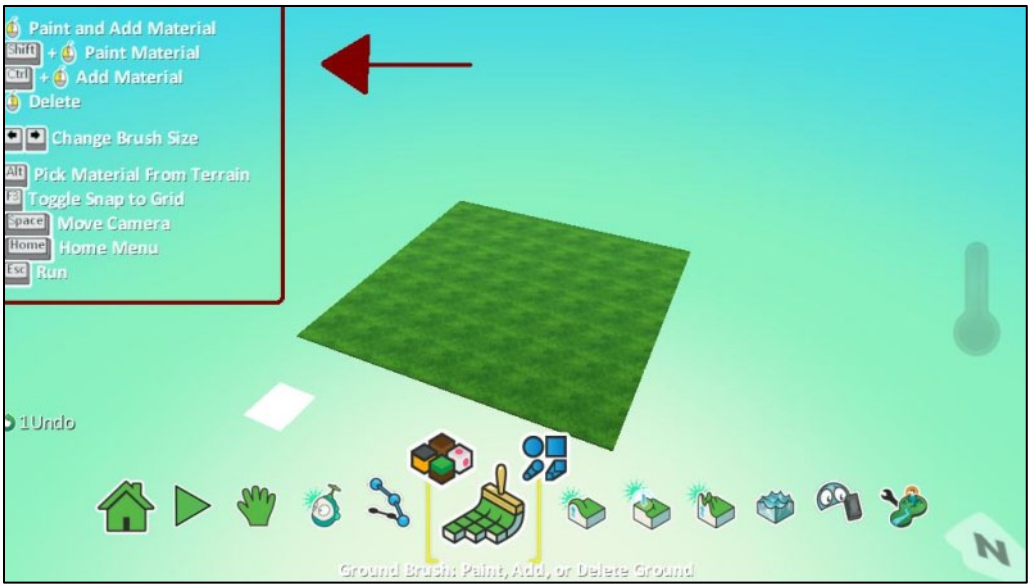
Ομοίως, μπορούμε να δημιουργήσουμε **ευθείες γραμμές με κυκλικές άκρες,** όμως, αυτή τη φορά.



Μαγική βούρτσα (Magic brush):

Μία πολύ χρήσιμη επιλογή που θα την εξετάσουμε εκτενέστερα στη συνέχεια. Αρκεί εδώ να αναφέρουμε ότι **επιλέγει ολόκληρες περιοχές εδάφους ανάλογα με το υλικό που έχει χρησιμοποιηθεί** και **συνδυάζεται με όλα τα υπόλοιπα εργαλεία δημιουργίας της πίστας**.

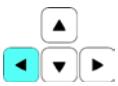
Παρατηρούμε τώρα στο πάνω αριστερά μέρος της εφαρμογής μας ότι εμφανίζονται κάποιοι συνδυασμοί πλήκτρων που μας επιτρέπουν να υλοποιήσουμε επιπρόσθετες λειτουργίες για τη δημιουργία του κόσμου μας. Ας προσπαθήσουμε παρακάτω να τους εξηγήσουμε και να τους κατανοήσουμε στην πράξη.



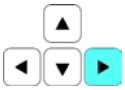
Πρόσθεση εδάφους

Αφαίρεση εδάφους

Ξεκινάμε λοιπόν με τα κλικ του ποντικιού. Με αριστερό κλικ προσθέτουμε έδαφος στον κόσμο μας ενώ με δεξί κλικ αφαιρούμε κομμάτι εδάφους από το ήδη υπάρχον. Δοκιμάστε να προσθέσετε υλικά διαφορετικών τύπων και χρησιμοποιώντας διαφορετικές βούρτσες.



Μείωση μεγέθους βούρτσας

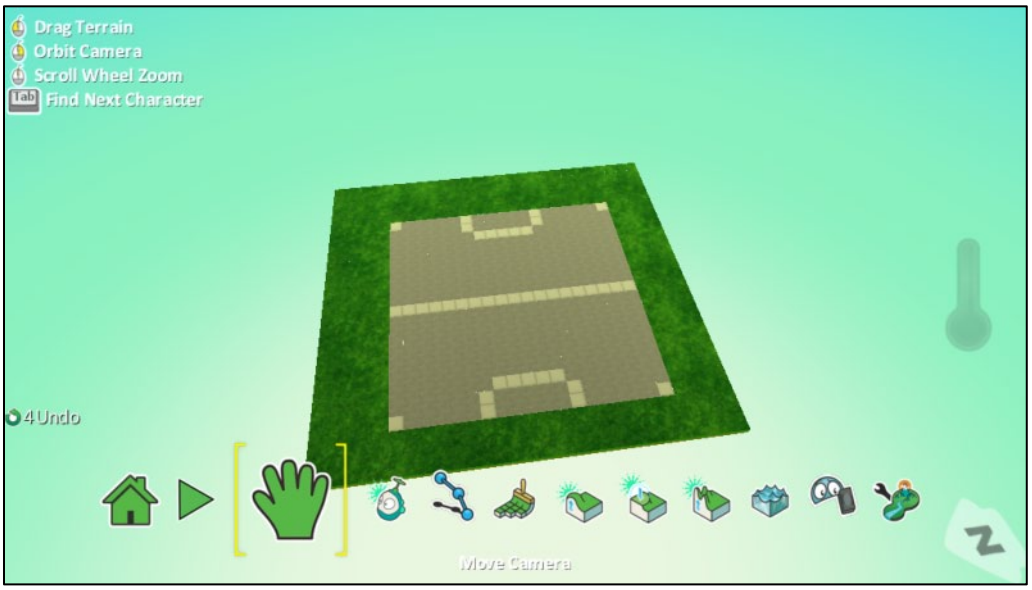


Αύξηση μεγέθους βούρτσας

Επιπλέον, χρησιμοποιώντας το πληκτρολόγιο σε συνδυασμό με το ποντίκι, μπορούμε να εκτελέσουμε πιο σύνθετες λειτουργίες. Φανταστείτε για παράδειγμα ότι θέλουμε να προσθέσουμε έδαφος ενός διαφορετικού τύπου στον κόσμο μας χωρίς όμως να επηρεαστεί το ήδη υπάρχον έδαφος. Κανονικά, όπου πατάμε με τη βούρτσα, αποτυπώνεται ο επιλεγμένος τύπος εδάφους. Πατώντας όμως Ctrl και αριστερό κλικ όμως, προσθέτουμε έδαφος χωρίς να αλλάζουμε τις προηγουμενες δημιουργίες μας. Σκεφτείτε για παράδειγμα πώς μπορούμε να προσθέσουμε γρασίδι γύρω από ένα γήπεδο ποδοσφαίρου από χώμα που έχουμε ήδη φτιάξει. Το γρασίδι θέλουμε να λειτουργεί ως περίγραμμα, επομένως αυξάνουμε το μέγεθος της ***Βούρτσας Εδάφους (Ground Brush)*** πατώντας το αριστερό βέλος από το πληκτρολόγιο ώστε να υπερκαλύψει το

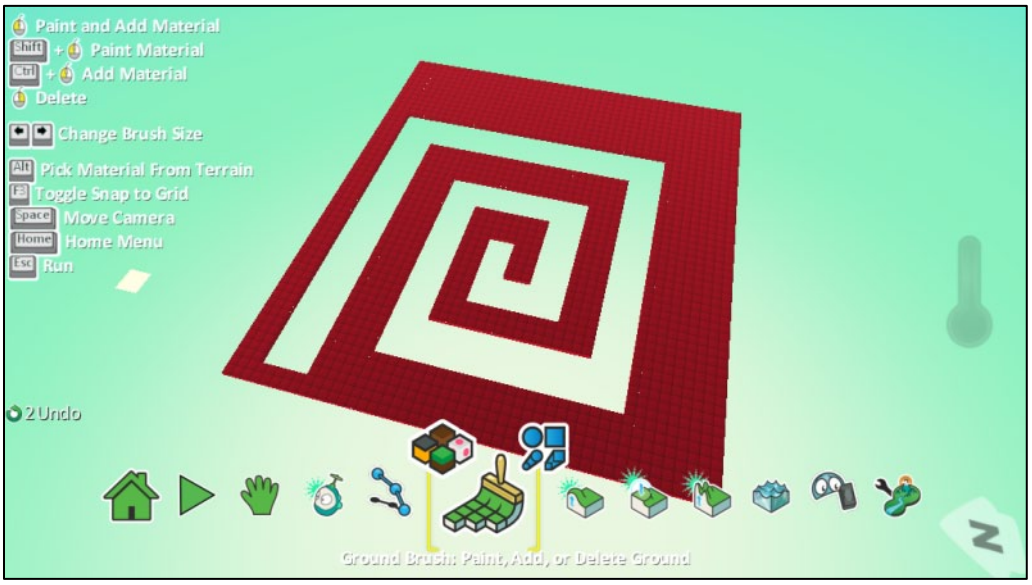
γήπεδο. Πατάμε Ctrl και μια φορά το αριστερό κλικ του ποντικιού δημιουργήσαμε το περίγραμμά μας! Η παρακάτω εικόνα δείχνει το αποτέλεσμα.

Πρόσθεση εδάφους χωρίς να επηρεάζεται το προηγούμενο



Αλλαγή υλικού εδάφους

Το MSKodu μας λύνει τα χέρια και για την περίπτωση που έχουμε δώσει μορφή στο έδαφος και αποφασίσουμε τελικά ότι θέλουμε να αλλάξουμε το υλικό που χρησιμοποιήσαμε. Η μία λύση θα ήταν να δημιουργήσουμε από την αρχή το έδαφος μας. Φαίνεται πολύ κουραστικό! Μπορούμε λοιπόν, χωρίς να καταστρέψουμε το έδαφος, με Shift και αριστερό κλικ του ποντικιού να γεμίσουμε ουσιαστικά το έδαφος με ένα καινούριο υλικό! Έτσι, έναν αρχικά κόκκινο λαβύρινθο μπορούμε με ευκολία να τον μετατρέψουμε σε έναν κίτρινο! Το σημαντικό είναι με τη χρήση του Shift δεν προσθέτουμε ούτε αφαιρούμε έδαφος, αλλάζουμε μόνο τον τύπο του εδάφους.



Δειγματοληψία εδάφους

Μία ακόμη σημαντική δυνατότητα που μας προσφέρει το MSKodu είναι η γνωστή από το πρόγραμμα της ζωγραφικής, *δειγματοληψία*. Έχοντας πατημένο το πλήκτρο Alt και με αριστερό κλικ του ποντικιού μπορούμε ανά πάσα στιγμή να επιλέξουμε για τη βούρτσα μας, ένα από τα ήδη υπάρχοντα υλικά εδάφους του κόσμου. Είναι πιο εύκολο λοιπόν σε έναν κόσμο με πολλά υλικά να διορθώνουμε κάποιες λεπτομέρειες χωρίς να είμαστε υποχρεωμένοι να ψάχνουμε όλα τα είδη εδάφους ένα προς ένα από την αντίστοιχη λίστα.

Εμφάνιση συντεταγμένων δείκτη ποντικιού

Τέλος, για να είμαστε ακόμη πιο ακριβείς στο σχεδιασμό μας, πατώντας το πλήκτρο F3 του πληκτρολογίου έχουμε τη δυνατότητα να ανακτούμε τις συντεταγμένες κάθε σημείου στον κόσμο μας καθώς εμφανίζονται στο κάτω μέρος του παραθύρου της εφαρμογής μας.

Συγκεντρωτικός Πίνακας

Δεξί κλικ ποντικιού	Πρόσθεση εδάφους στον κόσμο
Αριστερό κλικ ποντικιού	Διαγραφή εδάφους
Shift + αριστερό κλικ ποντικιού	Αλλαγή του υλικού του εδάφους σε ένα ήδη υπάρχον έδαφος
Ctrl + αριστερό κλικ ποντικιού	Πρόσθεση εδάφους στον κόσμο χωρίς να επηρεάζεται το ήδη υπάρχον έδαφος
Αριστερό βέλος	Μείωση του μεγέθους της βούρτσας
Δεξί βέλος	Αύξηση του μεγέθους της βούρτσας
Alt + αριστερό κλικ ποντικιού	Επιλογή ενός υλικού εδάφους από τα ήδη υπάρχοντα στον κόσμο
F3	Εμφάνιση συντεταγμένων

4.1.2 Λόφοι και Κοιλάδες (Up/Down)

Ήρθε η ώρα να διαμορφώσουμε το έδαφός μας στην τρίτη διάσταση, να του δώσουμε δηλαδή ύψος! Αυτό το στάδιο διαμόρφωσης του κόσμου είναι πολύ σημαντικό αφού επηρεάζει το οπτικό πεδίο των χαρακτήρων του παιχνιδιού, όπως επίσης και την ικανότητά τους να περιπλανηθούν πάνω σε αυτό (άλλα αντικείμενα είναι ικανά να σκαρφαλώνουν σε ένα βουνό και άλλα όχι).

Αρχικά, επιλέγοντας το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***, παρατηρούμε ότι όπως και με το προηγούμενο εργαλείο, ο δείκτης του ποντικιού συνοδεύεται από μια περιοχή εφαρμογής της συγκεκριμένης λειτουργίας, υπάρχει δηλαδή μια αντίστοιχη βούρτσα. Επιπλέον εμφανίζεται μια επιλογή που μας επιτρέπει να αλλάξουμε τη μορφή της βούρτσας όπως ακριβώς και με το εργαλείο ***Βούρτσα Εδάφους (Brush)***. Δεν μας δίνεται, όμως, η δυνατότητα επιλογής τύπου εδάφους! Γιατί; Γιατί πολύ απλά το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** μπορεί να χρησιμοποιηθεί μόνο πάνω σε υπάρχον έδαφος, η λειτουργία του είναι να δίνει ύψος σε ένα τμήμα του κόσμου που ήδη υπάρχει. Άρα αν θέλουμε να δημιουργήσουμε ένα βουνό σε έναν κενό κόσμο, πρώτα πρέπει να χρησιμοποιήσουμε τη ***Βούρτσα Εδάφους (Brush)*** για να

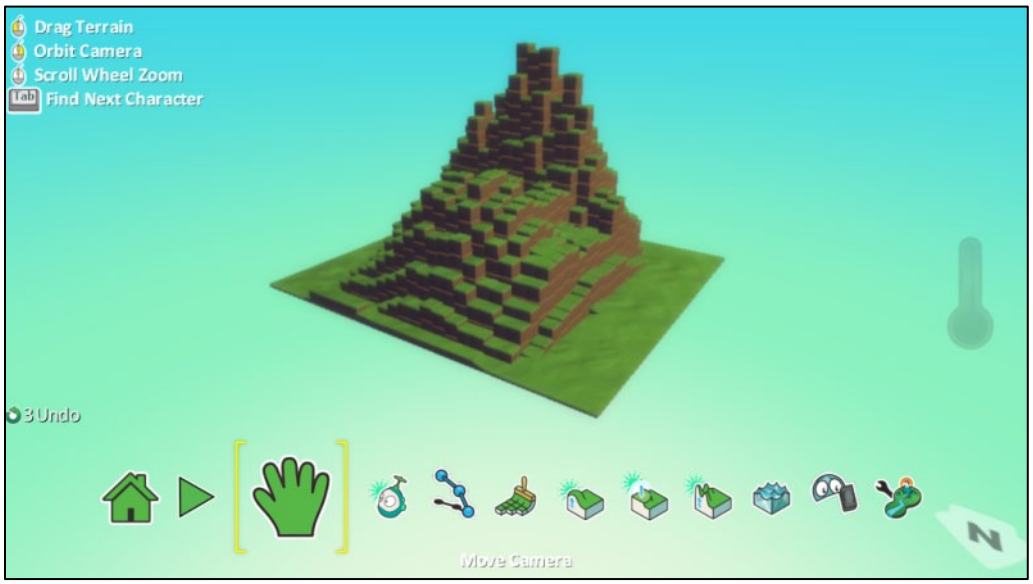
Αύξηση ύψους

Μείωση ύψους

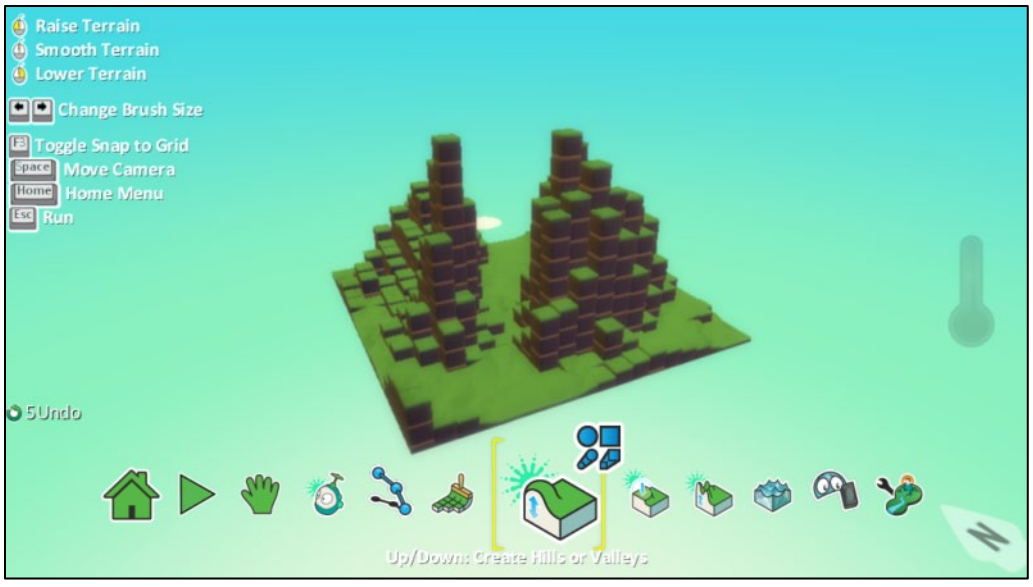
δημιουργήσουμε τη βάση του και στη συνέχεια να χρησιμοποιήσουμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** για να δώσουμε ύψος στο κομμάτι της επιφάνειας που επιθυμούμε να γίνει λόφος. Και σε αυτό το εργαλείο η βούρτσα αλλάζει μέγεθος με το δεξί (αυξάνεται) και το αριστερό (μειώνεται) βελάκι από το πληκτρολόγιο.

Είμαστε έτοιμοι, αρκεί ένα αριστερό κλικ με το ποντίκι και παρατηρούμε ότι η επιφάνεια της βούρτσας μας αρχίζει να αποκτά ύψος. Όσο περισσότερα κλικ τόσο ψηλότερος ο λόφος (μπορείτε να κρατήσετε συνεχόμενα πατημένο το αριστερό κλικ του ποντικιού). Με δεξί κλικ γίνεται η αντίστροφη διαδικασία, δηλαδή μειώνεται το ύψος του λόφου και αν παρατείνουμε το πάτημα παρατηρούμε ότι ο λόφος θα εξαφανιστεί!

Προσπαθήστε να δημιουργήστε το αποτέλεσμα της επόμενης εικόνα, είναι πολύ εύκολο.



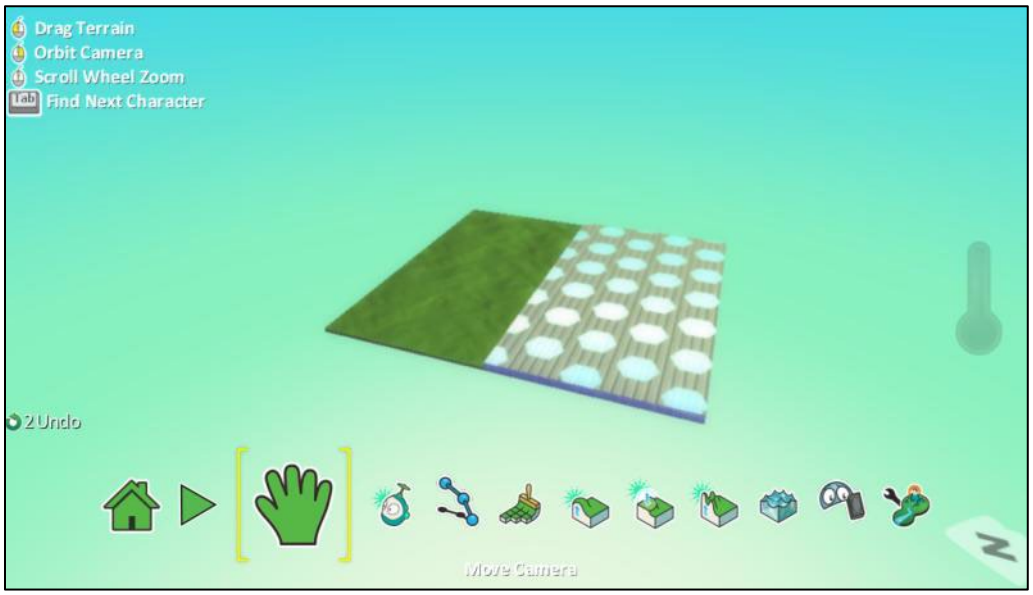
Τώρα μπορούμε στο λοφώδες έδαφος που δημιουργήσαμε να δοκιμάσουμε το ίδιο εργαλείο αλλά πατώντας δεξί κλικ αυτή τη φορά. Επιλέξτε ένα αρκετά μικρότερο μέγεθος βούρτσας ώστε το τελικό αποτέλεσμα να είναι παρόμοιο με αυτό της επόμενης εικόνας. Θα μειώσετε δηλαδή το ύψος ενός μικρού κομματιού του λόφου!



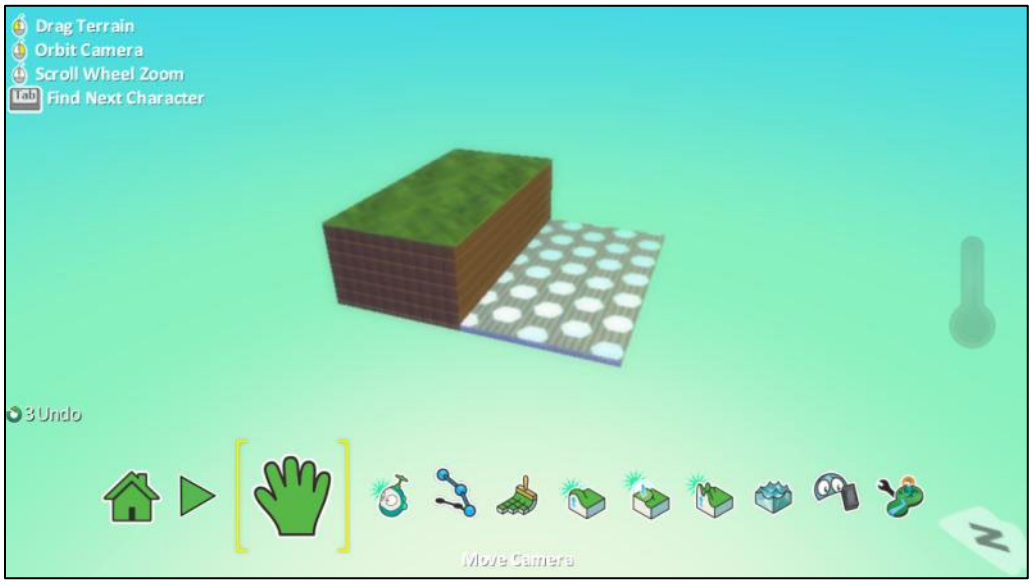
Τι είναι όμως η ***Μαγική Βούρτσα (Magic Brush)*** που είδαμε στην επιλογή σχήματος της βούρτσας και τι το διαφορετικό μπορεί να κάνει με τη χρήση του εργαλείου *Λόφοι και* Κοιλάδες (*Up/Down*);

Αυτό που κάνει τη μαγική βούρτσα ξεχωριστή είναι ότι μας δίνει τη δυνατότητα να αυξήσουμε ή να μειώσουμε το ύψος περιοχών του κόσμου μας, που διαχωρίζονται βάσει του υλικού τους. Ας δούμε ένα παράδειγμα:

Αρχικά ο κόσμος μας μπορεί να αποτελείται από δύο υλικά, όπως φαίνεται παρακάτω.

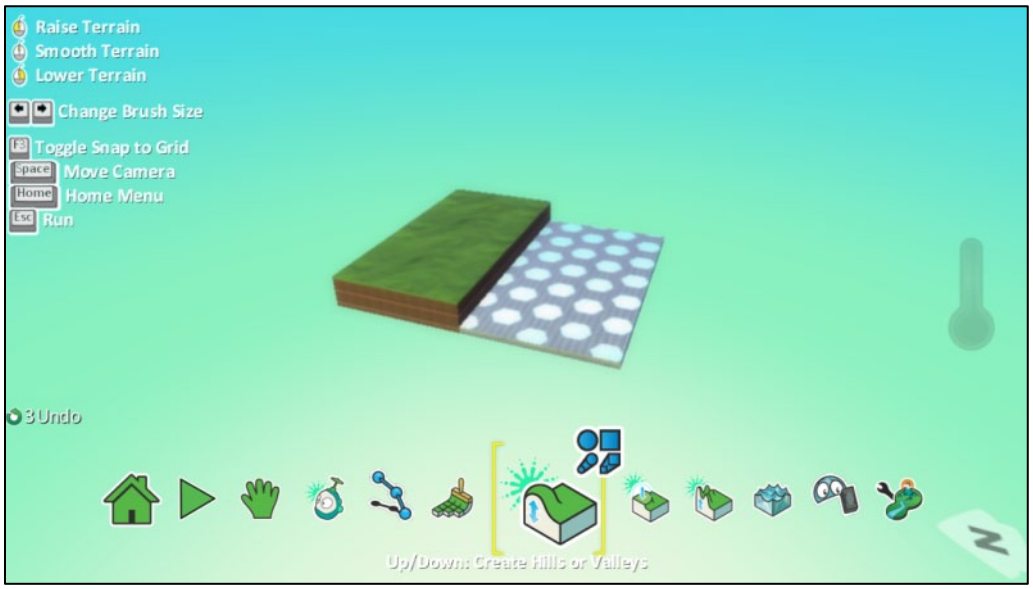


Επιλέγοντας το εργαλείο *Λόφοι και* Κοιλάδες (*Up/Down*) και τη μαγική βούρτσα, παρατηρούμε ότι ανάλογα με το σημείο που βρίσκεται το ποντίκι μας είτε θα αναβοσβήνει η περιοχή με το γρασίδι είτε η περιοχή με το άλλο υλικό. Όπως καταλαβαίνετε με τη ***Μαγική Βούρτσα (Magic Brush)*** δεν θα επεξεργαζόμαστε μεμονωμένα σημεία του κόσμου μας αλλά ολόκληρες περιοχές που οριοθετούνται βάση υλικού. Πατώντας επαναλαμβανόμενα αριστερό κλικ στο γρασίδι το αποτέλεσμα θα είναι το παρακάτω.



Τώρα αν στην ίδια περιοχή (στο γρασίδι στην προκειμένη περίπτωση) κάνουμε δεξί κλικ, το ύψος θα μειωθεί.

Άρα, τι θα κάνατε αν θέλατε να αλλάξετε το υλικό του εδάφους σας στην περίπτωση του λαβυρίνθου που παρουσιάστηκε παραπάνω; Απλά θα επιλέγατε το επιθυμητό έδαφος, θα επιλέγατε τη Μαγική Βούρτσα και με ένα κλικ, θα είχατε τελειώσει!



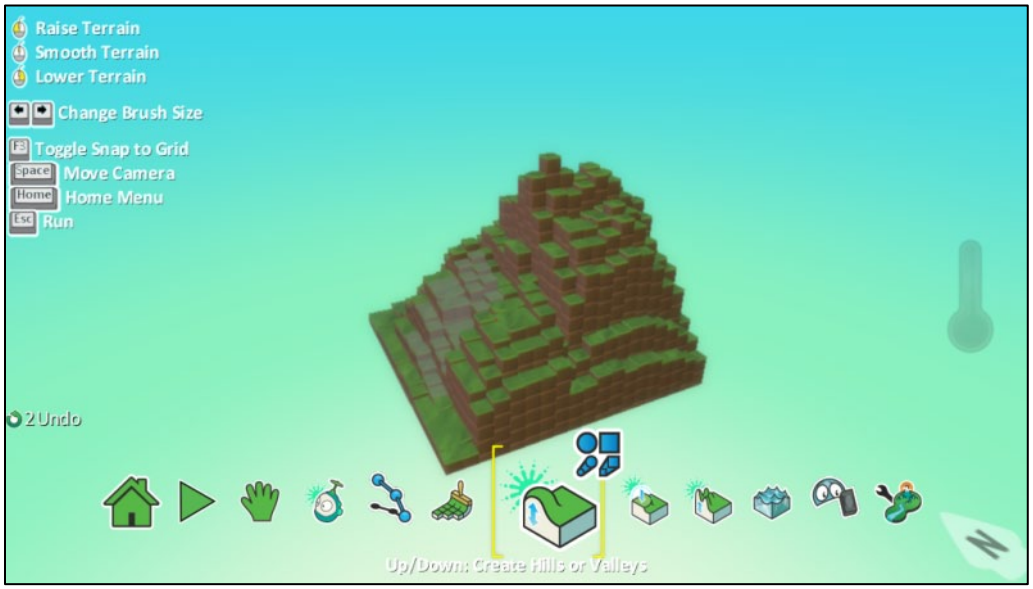
Άρα η μαγική βούρτσα είναι ένα εργαλείο που μας επιτρέπει να εκτελέσουμε ενέργειες όχι βάσει τους σχήματος της βούρτσας αλλά βάσει των περιοχών που οριοθετούνται από το υλικό του εδάφους.

4.1.3 Λείανση Εδάφους (Flatten)

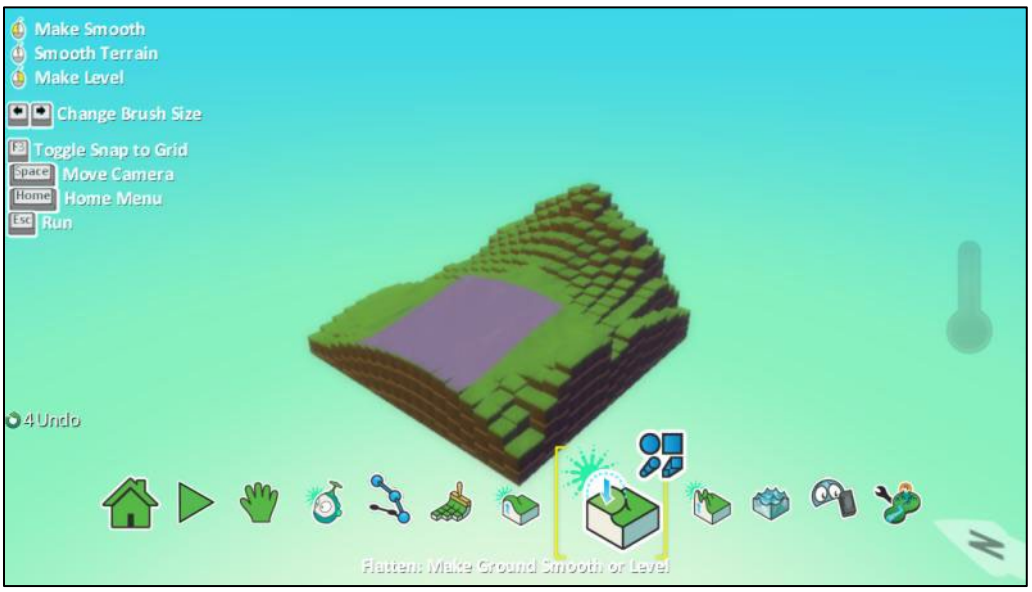
Παρατηρήσατε ότι το έδαφος που δημιουργείται με το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** είναι λίγο ανώμαλο; Το MSKodu μας δίνει την δυνατότητα να εξομαλύνουμε την επιφάνεια του εδάφους με το εργαλείο ***Λείανση Εδάφους: Κάντε το Έδαφος Λείο ή Επίπεδο (Flatten: Make Ground Smooth or Level)***. Κάνοντας κλικ πάνω σε αυτό το εργαλείο παρατηρούμε ότι εμφανίζεται η γνωστή βούρτσα που αναβοσβήνει στην οθόνη και η οποία προσδιορίζει την επιφάνεια του εδάφους πάνω στην οποία θα εφαρμόσουμε τη λειτουργία. Μπορούμε να αλλάξουμε το σχήμα ή και το μέγεθος της βούρτσας με τον ίδιο τρόπο που αναφέραμε προηγουμένως.

Το μόνο που μας μένει είναι ένα αριστερό κλικ και θα δούμε τις υψομετρικές διαφορές του εδάφους να εξομαλύνονται και το έδαφος να λειαίνεται. Αν στην περιοχή που διαμορφώνουμε δεν υπάρχει λόφος ή βουνό, προφανώς δεν θα υπάρχει αποτέλεσμα, δηλαδή δεν θα αλλάξει τίποτα στην επιφάνεια του κόσμου μας. Ας δούμε ένα παράδειγμα.

Δημιουργούμε έναν νέο άδειο κόσμο στον οποίο και φτιάχνουμε ένα λόφο με το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***. Παρατηρούμε ότι ο λόφος μας αποτελείται από μικρά τετράγωνα μέρη του εδάφους σε διαφορετικά ύψη το καθένα. Παρακάτω βλέπουμε το αρχικό βουνό/λόφο που υπάρχει στον κόσμο μας και τις “αιχμηρές” άκρες του:



Επιλέγοντας το εργαλείο ***Λείανση Εδάφους (Flatten)***, είμαστε έτοιμοι να ομαλοποιήσουμε το έδαφος. Κάνουμε αριστερό κλικ στις περιοχές που θέλουμε και με λίγη υπομονή, αφού χρειάζεται να επιμείνουμε λίγο σε ένα σημείο για να λειάνουμε πλήρως το έδαφος, θα μπορούσαμε να πάρουμε το παρακάτω αποτέλεσμα.

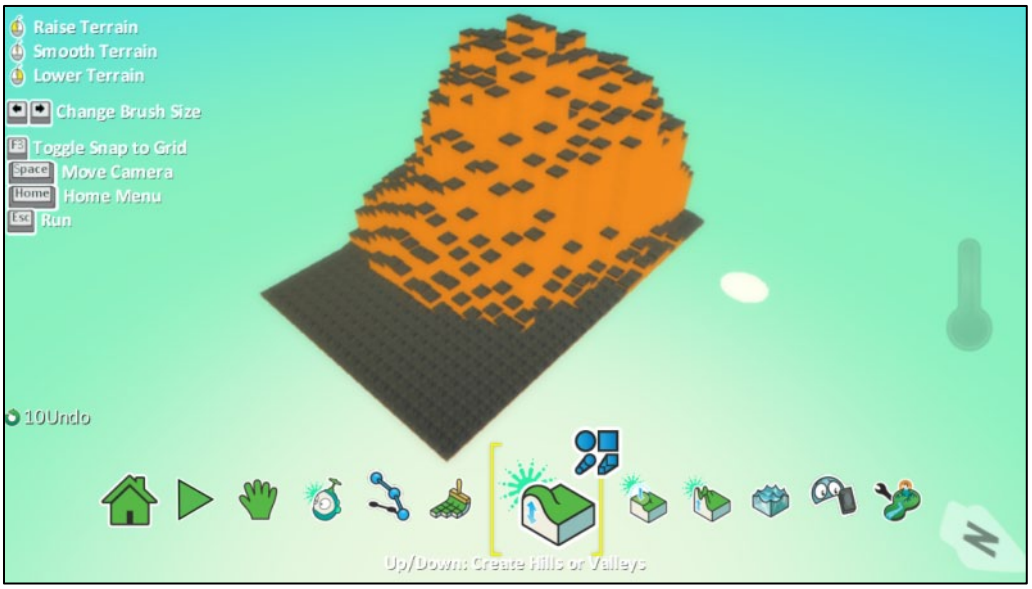


Λείανση εδάφους

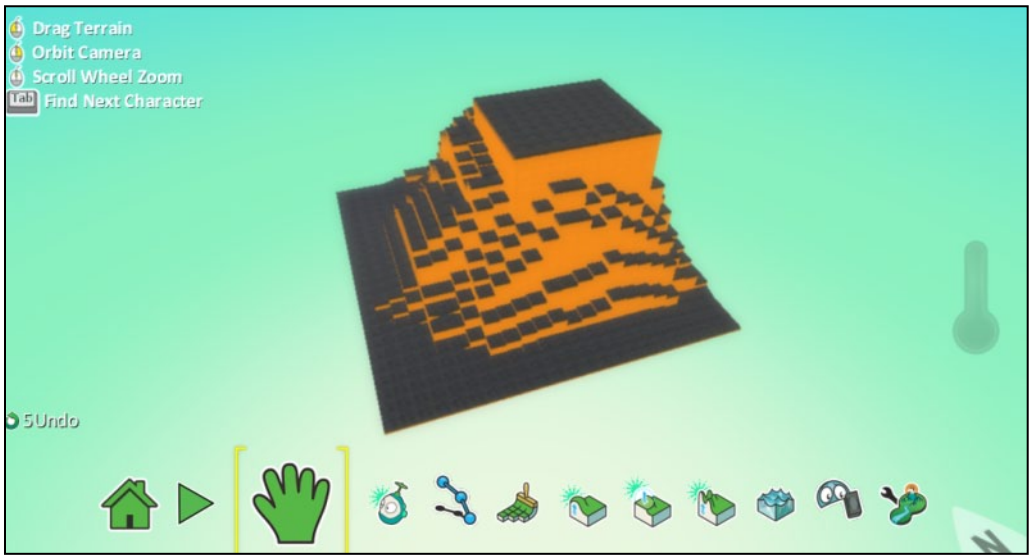
Εξομάλυνση βάσει σημείου

Με δεξί κλικ, το εργαλείο ***Λείανση Εδάφους (Flatten)*** λειτουργεί λίγο διαφορετικά. Εξομαλύνει τις υψομετρικές διαφορές των σημείων που βρίσκονται κάτω από την επιφάνεια της βούρτσας χρησιμοποιώντας όμως ως βάση το σημείο ακριβώς κάτω από το οποίο βρίσκεται ο δείκτης του ποντικιού! Στα χαμηλότερα σημεία κάτω από τη βούρτσα δίνει ύψος ενώ αφαιρεί ύψος από τα υψηλότερα σημεία σε σχέση με το επιλεγμένο σημείο. Αυτό μπορεί να συνεχιστεί μέχρι να δημιουργηθεί ένα επίπεδο με ύψος που καθορίζεται από το σημείο που βρίσκεται ο δείκτης του ποντικιού.

Έστω, λοιπόν ότι έχουμε δημιουργήσει τον επόμενο κόσμο.



Στη συνέχεια επιλέγουμε το εργαλείο ***Λείανση εδάφους (Flatten)***. Αν τοποθετήσουμε το δείκτη του ποντικιού στο υψηλότερο σημείο του λόφου και πατήσουμε επαναλαμβανόμενα δεξί κλικ τότε σταδιακά όλα το σημεία της περιοχής κάτω από τη βούρτσα θα έρθουν στο ίδιο ύψος!

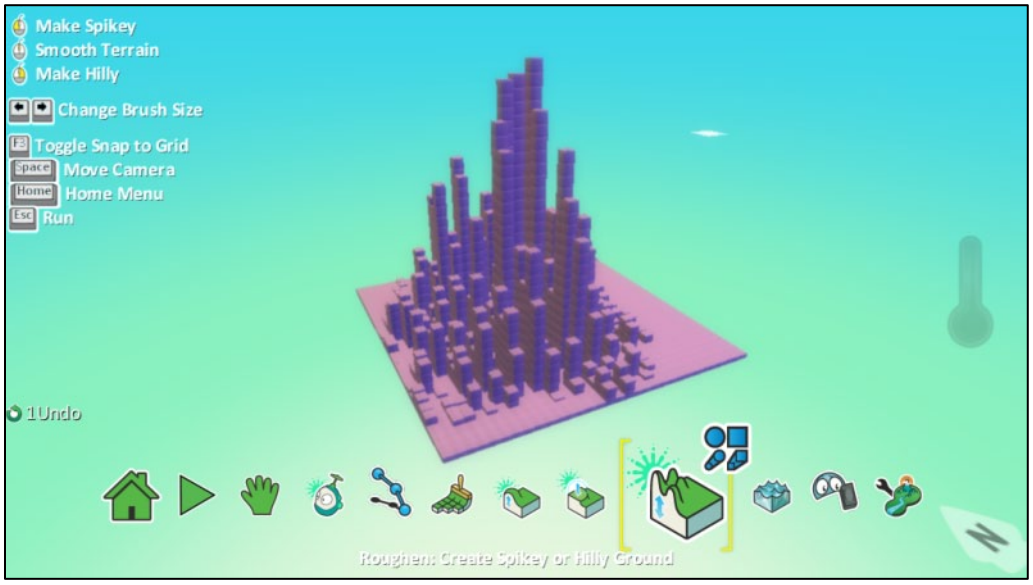


Η χρήση του συγκεκριμένου εργαλείου μπορεί να σας δυσκολεύει λίγο προς το παρόν όμως θα παρατηρήσετε στην πορεία ότι το εργαλείο ***Λείανση εδάφους (Flatten)*** είναι πολύ χρήσιμο.

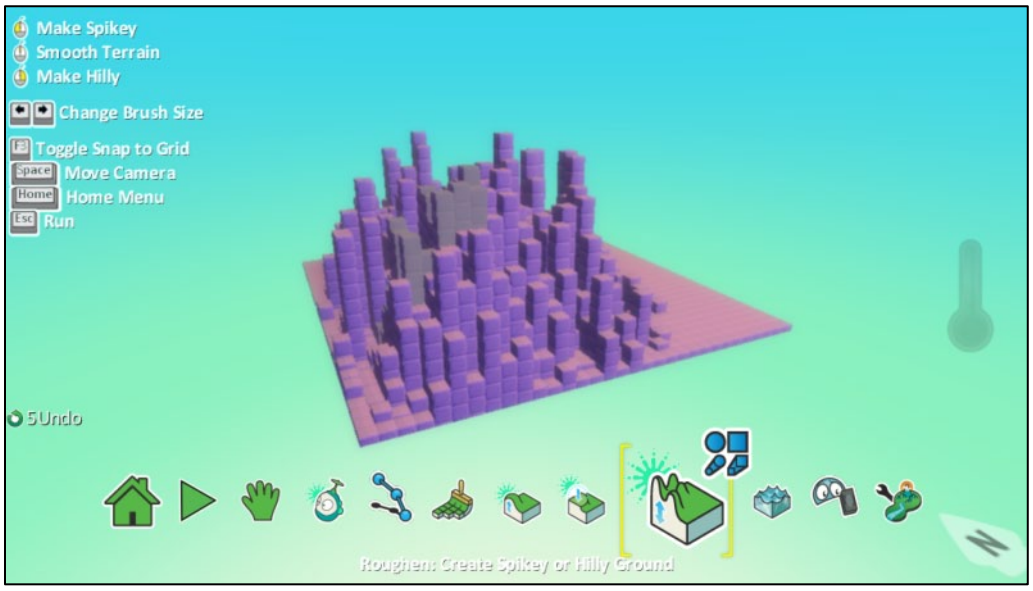
4.1.4 Σκλήρυνση Επιφάνειας (Roughen)

Με το εργαλείο ***Σκλήρυνση Επιφάνειας: Δημιουργήστε Μυτερό ή Λοφώδες Έδαφος (Roughen: Create Spikey or Hilly Ground)*** δημιουργούμε υψώματα με μεγάλες και ακαθόριστες υψομετρικές διαφορές που δίνουν την αίσθηση περισσότερο μιας απόκρημνης ακανθώδους άγριας περιοχής παρά ενός βουνού ή ενός λόφου που δημιουργούνται με το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***.

Έτσι αν σε έναν νέο κόσμο λοιπόν χρησιμοποιήσουμε το εργαλείο ***Σκλήρυνση Επιφάνειας (Roughen)***, θα δημιουργηθεί ένα αποτέλεσμα σαν αυτό της επόμενης εικόνας:

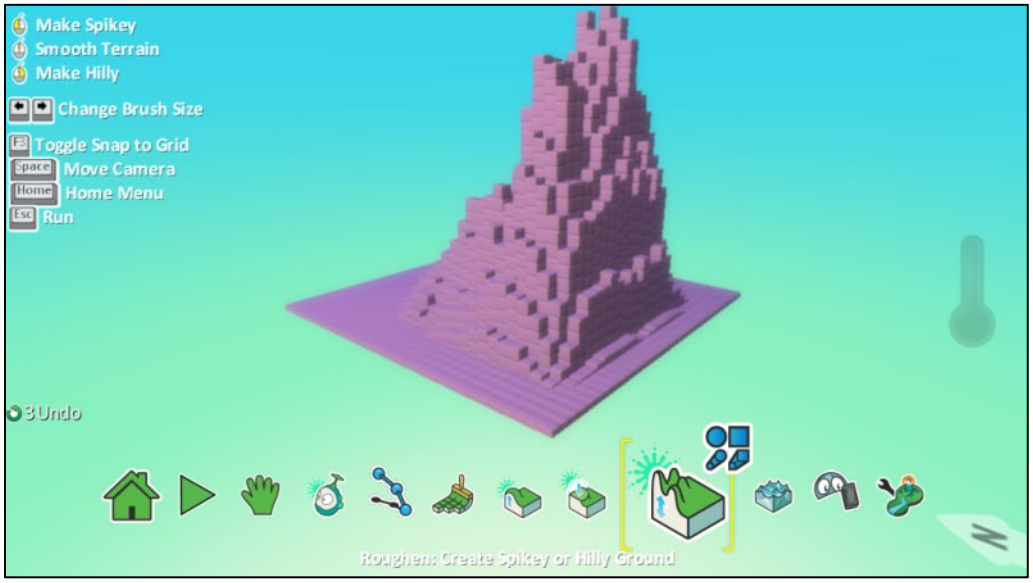


Αν τώρα στο έδαφος που ήδη δημιουργήσαμε, χρησιμοποιήσουμε το εργαλείο ***Σκλήρυνση Επιφάνειας (Roughen)*** αλλά κάνοντας αρκετά κλικ στη ροδέλα του ποντικιού, η διαμόρφωση που θα υποστεί ο κόσμος μας θα είναι η παρακάτω.



Ουσιαστικά δηλαδή, πατώντας κλικ στη ροδέλα του ποντικιού, ομαλοποιούμε τις υψομετρικές διαφορές των αγκαθωτών σημείων. Η επιφάνεια ακόμα θεωρείται αγκαθωτή, αλλά πλέον υπάρχει μια ομαλή διακύμανση στο ύψος. Ας δούμε τώρα τι μπορούμε να κάνουμε χρησιμοποιώντας το συγκεκριμένο εργαλείο σε συνδυασμό με δεξί κλικ.

Πατώντας δεξί κλικ παρατηρούμε ότι δημιουργείται ένα αγκαθωτό ύψωμα που έχει ωστόσο τη μορφή λόφου, σε αντίθεση με το ακαθόριστο σχήμα που δημιουργείται με αριστερό κλικ όπως είδαμε παραπάνω. Στην εικόνα φαίνεται το αποτέλεσμα της διαμόρφωσης που θα υποστεί η επιφάνεια.



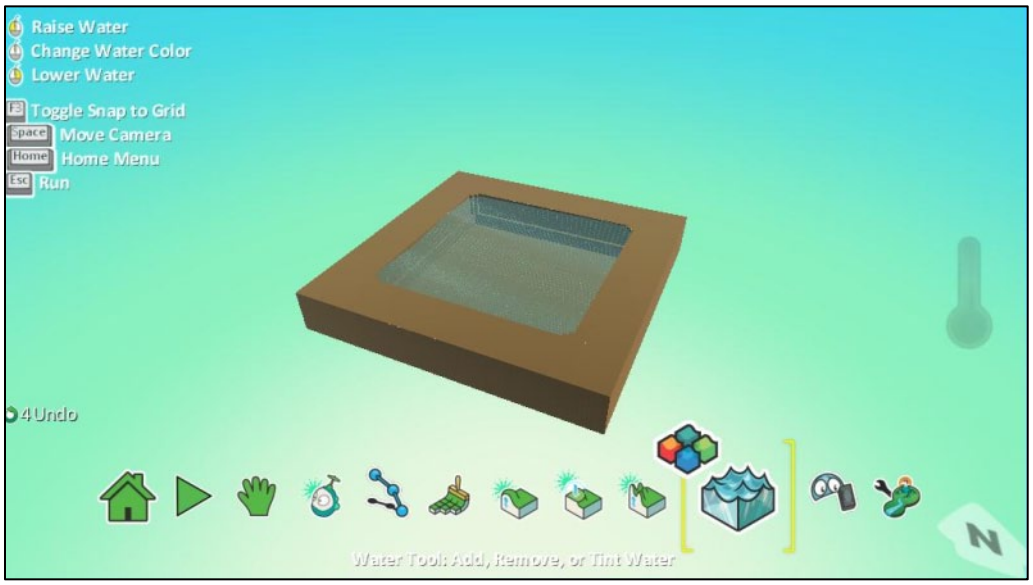
4.1.5 Εργαλείο Νερού (Water tool)

Ο κόσμος μας πλέον περιλαμβάνει λόφους, βουνά, πεδιάδες, υψώματα. Ήρθε η ώρα να αποκτήσει λίμνες και ποτάμια!

Με το ***Εργαλείο Νερού: Προσθέστε, Αφαιρέστε ή Χρωματίστε Νερό (Water tool: Add, Remove, or Tint Water)*** μπορούμε να εισάγουμε νερό σε περιοχές του κόσμου μας. Αν πατήσετε πάνω στο εργαλείο, θα διαπιστώσετε ότι έχουμε τη δυνατότητα να επιλέξουμε το χρώμα και την αίσθηση του υγρού στοιχείου που επιθυμούμε. Μπορείτε δηλαδή να προσθέσετε νερό κοκκινο-κίτρινου χρώματος (κάτι σαν λάβα, αριθμός 7) ή ακόμη και μαύρο νερό (αριθμός 8).

Ανακαλώντας όσα έχουμε μάθει στις προηγούμενες ενότητες, θα προσπαθήσουμε να δημιουργήσουμε μια υποδοχή/περιοχή στην οποία θα εισάγουμε νερό με αποτέλεσμα να δημιουργηθεί μια λίμνη όπως φαίνεται στο παρακάτω σχήμα.

Επειδή όμως πολλές φορές θα θέλετε να μετακινήσετε την κάμερα του κόσμου ενώ πραγματοποιείτε μια άλλη εργασία, το MSKodu μας ενεργοποιεί όλες τις προηγούμενες επιλογές μετακίνησεις με το ποντίκι απλώς πατώντας παράλληλα και το πλήκτρο Space.



1. Επιλέγουμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** και στη συνέχεια από τα σχήματα τη ***Μαγική Βούρτσα (Magic Brush)***.

2. Με αριστερό κλικ δίνουμε ύψος σε ολόκληρο τον κόσμο.

3. Με το δεξί βέλος του πληκτρολογίου μειώνουμε το μέγεθος της βούρτσας.

4. Με δεξί κλικ στο εσωτερικό της περιοχής της υπερυψωμένης επιφάνειας, δημιουργούμε ένα "βαθούλωμα" που θα υποδεχθεί στη συνέχεια το νερό.

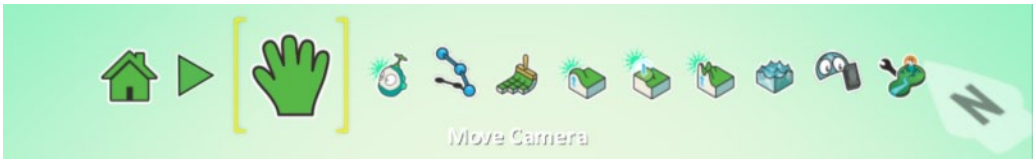
5. Τέλος, με το ***Εργαλείο Νερού (Water tool)*** και αριστερό κλικ γεμίζουμε το "βαθούλωμα" με τον τύπο νερού που επιθυμούμε. Ίσως θα πρέπει να πατήσουμε πολλές φορές το εργαλείο νερού μέχρι να φτάσουμε στο επίπεδο που θέλουμε… κι αν ξεχειλίσει; Τότε, με δεξί κλικ αφαιρούμε μέρος από το υγρό που βάλαμε!

Είναι σημαντικό να τονίσουμε ότι το νερό διαχέεται σε όλο τον κόσμο με βάση τις φυσικές ιδιότητες του κόσμου, όπως συμβαίνει και στην πραγματικότητα. Αν έχετε δημιουργήσει μια κλειστή περιοχή και πατήσετε αριστερό κλικ μέσα σε αυτή, τότε το νερό θα μένει μόνο εκεί. Αν όμως υπερχειλίσετε την περιοχή αυτή, τότε το νερό θα καταλάβει όλο τον κόσμο σας!

4.1.6 Χειρίζομαι την κάμερα για να δημιουργήσω τις πίστες

Σε αυτό το σημείο και πριν προχωρήσουμε στη δημιουργία πιο πολύπλοκων κόσμων είναι σημαντικό να επαναλάβουμε τους τρόπους με τους οποίους μπορούμε να αλλάξουμε τη θέση της κάμερας ώστε να βλέπουμε την πίστα μας από διαφορετικές οπτικές γωνίες. Μεγάλοι και σύνθετοι κόσμοι απαιτούν να δουλεύουμε σε διαφορετικά σημεία της πίστας και από διαφορετικές σκοπιές.

Υπενθυμίζουμε λοιπόν ότι για να μετακινήσουμε την κάμερα πρέπει να επιλέξουμε το εργαλείο ***Μετακίνηση Κάμερας (Move Camera)*** που βρίσκεται στην *Παλέτα Εργαλείων (Tool Palette)* όπως φαίνεται από την επόμενη εικόνα.

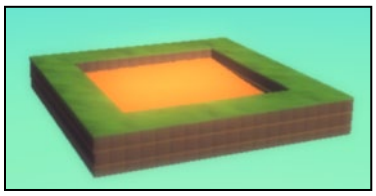


Οι βασικές λειτουργίες που μπορούμε να κάνουμε με το συγκεκριμένο εργαλείο επιλεγμένο:

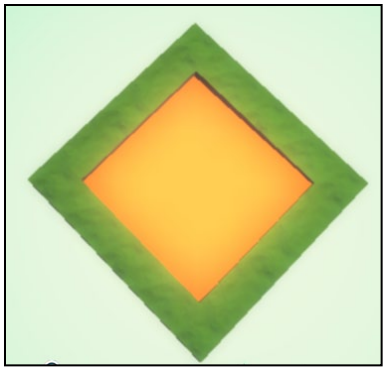
Ενέργεια ποντικιού	Κίνηση κάμερας
Μετακίνηση ποντικιού με πατημένο το **αριστερό πλήκτρο**	η κάμερα μετακινείται αντίστοιχα προς τα βόρεια, νότια, ανατολικά και δυτικά της πίστας
Μετακίνηση ποντικιού με πατημένο το **δεξί πλήκτρο**	η κάμερα αλλάζει θέση στους τρεις άξονες, δηλαδή αλλάζει ύψος και περιστρέφεται στον κόσμο μας
Πλήκτρο κύλισης του ποντικιού	**αλλάζει η απόσταση** της κάμερας από τον κόσμο (zoom in –zoom out)
Πλήκτρο κενό (space) και τα προηγούμενα	Ενεργοποιείται το εργαλείο Μετακίνησης της Κάμερας για όσο διάστημα πατάμε το πλήκτρο κενό (space) και μπορούμε με το ποντίκι να κάνουμε τις προηγούμενες ενέργειες. Μόλις αφήσουμε το space, επιλέγεται το εργαλείο που χρησιμοποιήσουμε αμέσως πριν.

Με τη χρήση της κάμερας, μπορούμε να δούμε τον κόσμο από διαφορετικές σκοπιές, όπως φαίνεται και στις επόμενες εικόνες.

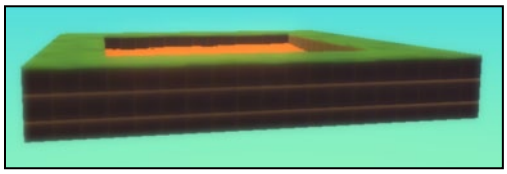
Αρχική θέση κάμερας:



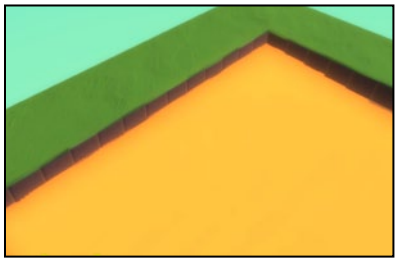
Δεξί κλικ του ποντικιού και σύρσιμο για να δούμε την κάτοψη του κόσμου μας:



Δεξί κλικ του ποντικιού και σύρσιμο για να δούμε από πλάγια τον κόσμο μας:



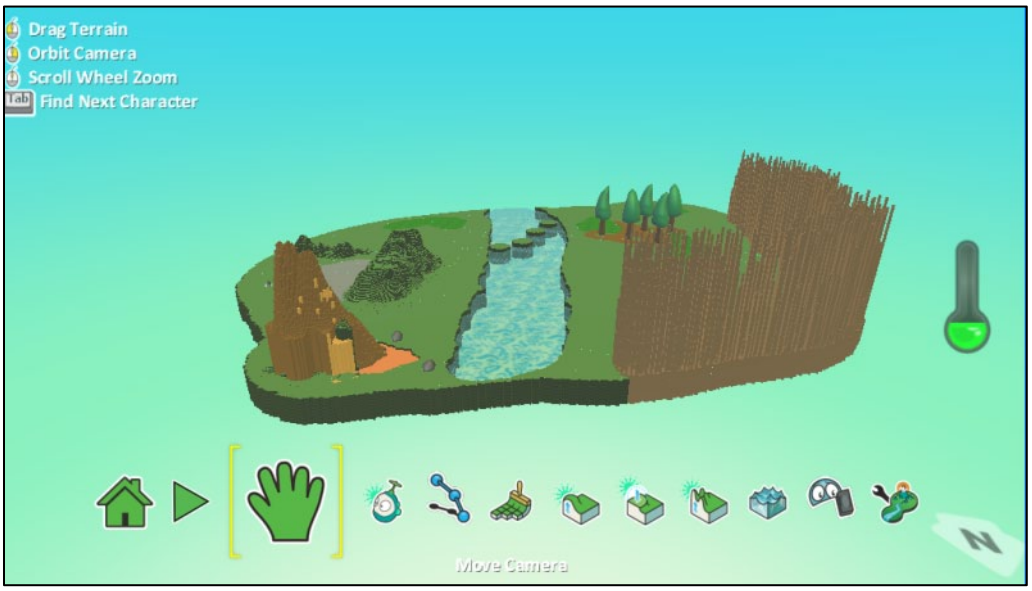
Χρήση ροδέλας του ποντικιού για να ζουμάρουμε:



4.2 Δημιουργώ πίστες παιχνιδιών

Σε αυτό το σημείο, έχοντας περιγράψει όλα τα εργαλεία και τις δυνατότητες του MSKodu, ήρθε η στιγμή να δημιουργήσουμε νέες πιο σύνθετες πίστες. Μπορούμε να φτιάξουμε αρκετά πολύπλοκους και μεγάλους κόσμους αρκεί να έχουμε φαντασία και να συνδυάσουμε με τον κατάλληλο τρόπο τα εργαλεία που μας δίνονται! Είναι πολύ σημαντικό να έχουμε κατανοήσει πλήρως τι κάνει κάθε εργαλείο ώστε να το χρησιμοποιούμε σωστά και να καταφέρουμε τελικά να δημιουργήσουμε τον κόσμο που έχουμε σκεφτεί.

Έστω ότι θέλουμε να φτιάξουμε μία πίστα που να ανταποκρίνεται σε ένα παιχνίδι κρυμμένου θησαυρού. Τι θα πρέπει να περιέχει η πίστα; Ίσως το παιχνίδι εκτυλίσσεται σε μια περιοχή με λόφους, χαράδρες και επικίνδυνες βουνοπλαγιές ή ο ήρωας να πρέπει να περάσει μέσα από ένα βαθύ ποτάμι και ίσως κάπου εκεί κοντά να υπάρχει και ένα ηφαίστειο. Αντιλαμβανόμαστε λοιπόν ότι σίγουρα η πίστα πρέπει να έχει μια ποικιλομορφία εδάφους που να αποδίδει τη δυσκολία που θα αντιμετωπίσει ο ήρωας μέχρι να βρει το χαμένο θησαυρό. Πάμε λοιπόν να εξετάσουμε ένα απλό παράδειγμα που θα μας βοηθήσει να αντιληφθούμε τη μορφή που θα πρέπει να έχει η πίστα μας.

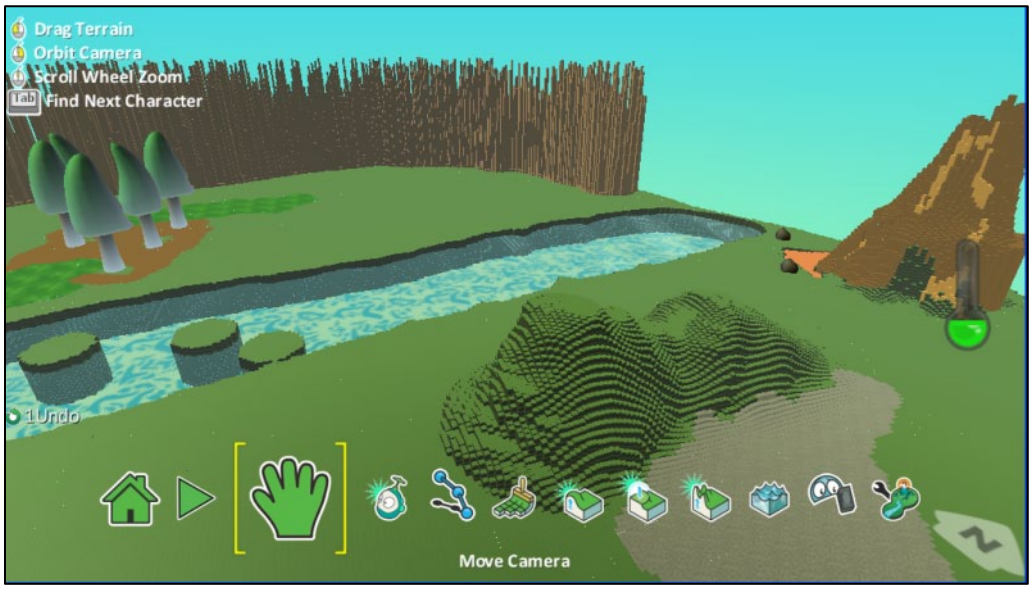


Δείτε το παράδειγμα *04_01.kodu*

Πίστα 1: Φαίνονται με μεγάλη ευκρίνεια τα υλικά και οι λειτουργίες που χρησιμοποιήσαμε. Ο ήρωας πρέπει να ψάξει σε όλη την πίστα για να βρει το θησαυρό!

Εάν παρατηρήσουμε την πίστα που φαίνεται στην παραπάνω εικόνα, πρέπει να είμαστε σε θέση να καταλάβουμε ότι έχουν χρησιμοποιηθεί το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***, το εργαλείο ***Λείανση εδάφους (Flatten)***, το εργαλείο ***Εργαλείο Νερού (Water tool)*** και το εργαλείο ***Σκλήρυνση Επιφάνειας (Roughen)***.

Μια διαφορετική θέση της κάμερας ίσως σας βοηθήσει περισσότερο να καταλάβετε τα εργαλεία που εφαρμόστηκαν!

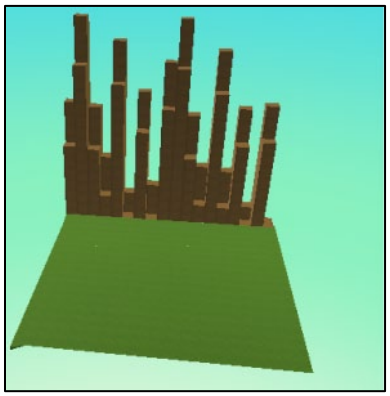


Επίσης, διακρίνουμε και μερικά δέντρα που καλύπτουν ένα μεγάλο μέρος της πίστας και βρίσκονται τοποθετημένα δίπλα στο ποτάμι. Τα δέντρα είναι αντικείμενα του MSKodu που θα μελετήσουμε εκτενέστερα σε επόμενο κεφάλαιο, εδώ τα χρησιμοποιούμε ως στατικά αντικείμενα χωρίς συμπεριφορές που ομορφαίνουν τον κόσμο μας.

Ας ξεκινήσουμε να δημιουργήσουμε μια παραπλήσια πίστα!

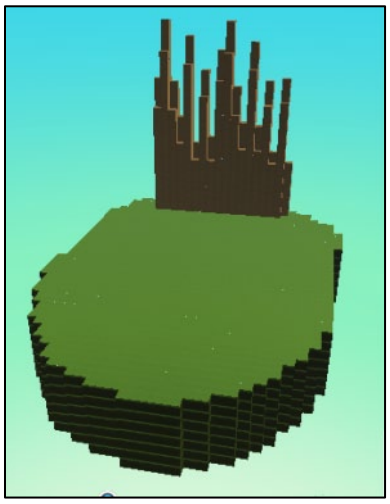
Αρχικά, για ακόμη μία φορά φορτώνουμε έναν άδειο κόσμο. Στο πρώτο βήμα της δημιουργίας της πίστας μας, θα χρησιμοποιήσουμε υλικό που να μοιάζει με χώμα και θα εφαρμόσουμε το εργαλείο ***Σκλήρυνση Επιφάνειας (Roughen)*** πατώντας αριστερό κλικ στη μία πλευρά της πίστας μας για να δώσουμε την αίσθηση ενός ψηλού αγκαθωτού λόφου.

Βήμα 1



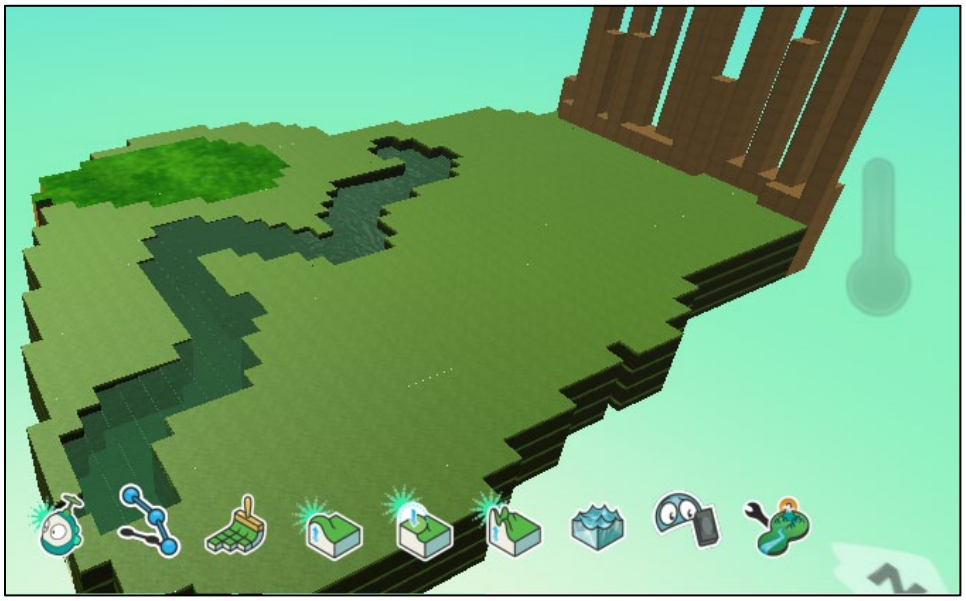
Στη συνέχεια μεγαλώνουμε την πίστα προσθέτοντας επιπλέον έδαφος κυκλικά στο ήδη υπάρχον. Για να το πετύχουμε, επιλέγουμε σχήμα βούρτσας ***Στρογγυλή βούρτσα (Hard round brush)*** και σχετικά μεγάλο μέγεθος. Εφόσον η πίστα μας έχει μεγαλώσει, επιλέγουμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** και τη ***Μαγική Βούρτσα (Magic Brush)*** και το εφαρμόζουμε σε όλη την έκταση της πίστας για να της δώσουμε ύψος.

Βήμα 2



Στο βήμα 3 θα επιχειρήσουμε να προσθέσουμε στην πίστα μας ένα ποτάμι που θα την διασχίζει. Πρέπει να σχεδιάσουμε αρχικά με άλλο υλικό το περίγραμμα που θέλουμε να έχει το ποτάμι χρησιμοποιώντας την ***Βούρτσα Εδάφους (Ground Brush)***. Στη συνέχεια με το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** και με επιλεγμένη τη ***Μαγική Βούρτσα (Magic Brush)*** πατάμε δεξί κλικ στο ποτάμι ώστε να δημιουργηθεί η κοίτη του. Για να ολοκληρώσουμε το ποτάμι μας, απλά πατάμε αριστερό κλικ με το ***Εργαλείο Νερού (Water Tool)*** και γεμίζουμε την κοίτη του με νερό.

Βήμα 3



Όπως φαίνεται και στην επόμενη εικόνα που δείχνει το στιγμιότυπο για το βήμα 4, μπορούμε με τα εργαλεία ***Λόφοι και Κοιλάδες (Up/Down)*** και ***Λείανση εδάφους (Flatten)*** να δημιουργήσουμε μικρούς λόφους, μικρά υψώματα που θα πλαισιώνουν το ποτάμι. Θέλει λίγο δουλειά ακόμη, λίγα δέντρα, λίγοι λόφοι ή ακόμη και αγκαθωτές κορυφές που απλώνονται στη μία άκρη της πίστας!

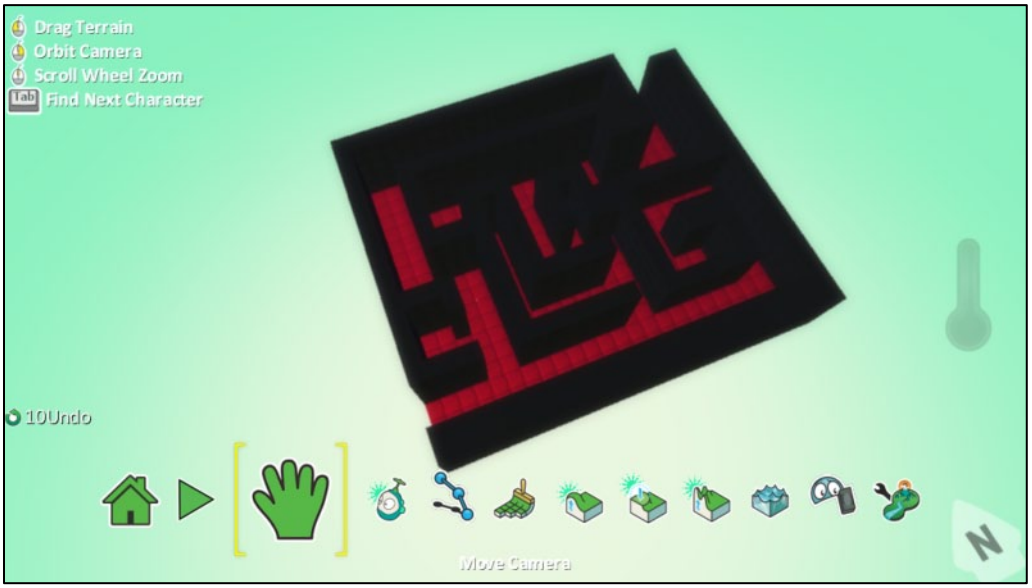
Βήμα 4

Δείτε το παράδειγμα *04_02.kodu*

Καταφέραμε λοιπόν μέσα σε λίγη ώρα να δημιουργήσουμε μια ενδιαφέρουσα πίστα παιχνιδιού. Φανταστείτε τι μπορούμε να κάνουμε αν αφιερώσουμε λίγη προσοχή στις λεπτομέρειες και εξοικειωθούμε περισσότερο με τα εργαλεία του MSKodu. Αλήθεια, τι άλλα παραδείγματα κόσμων μπορείτε να σκεφτείτε; Πώς φαντάζεστε ότι πρέπει να είναι μια ελκυστική πίστα για εσάς;

4.2.1 Επικίνδυνες διαδρομές

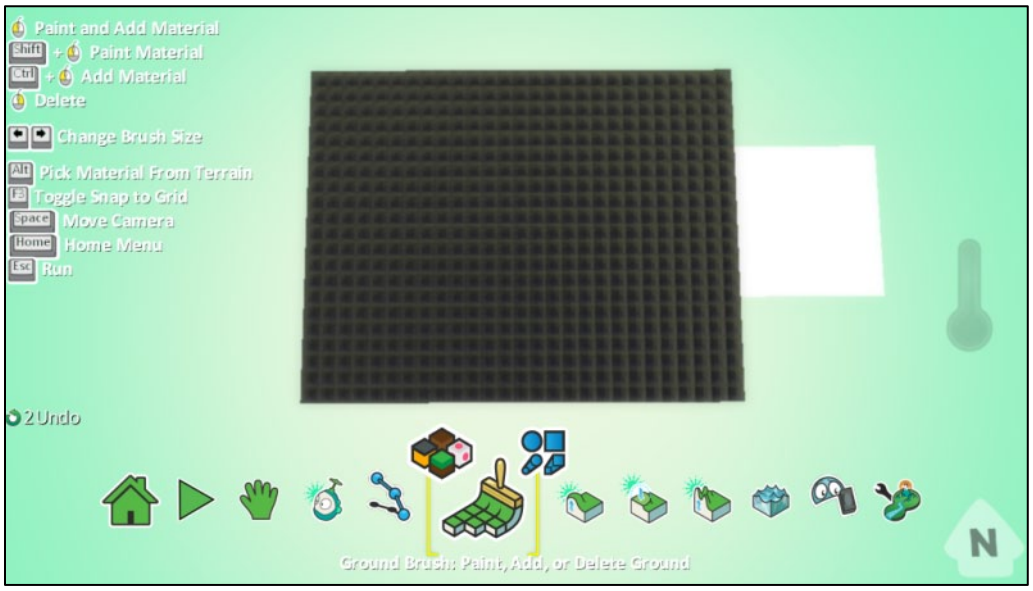
Ας συνεχίσουμε με τη δημιουργία ενός λαβύρινθου, όπως τον βλέπουμε στην επόμενη εικόνα.



Δείτε το παράδειγμα *04_03.kodu*

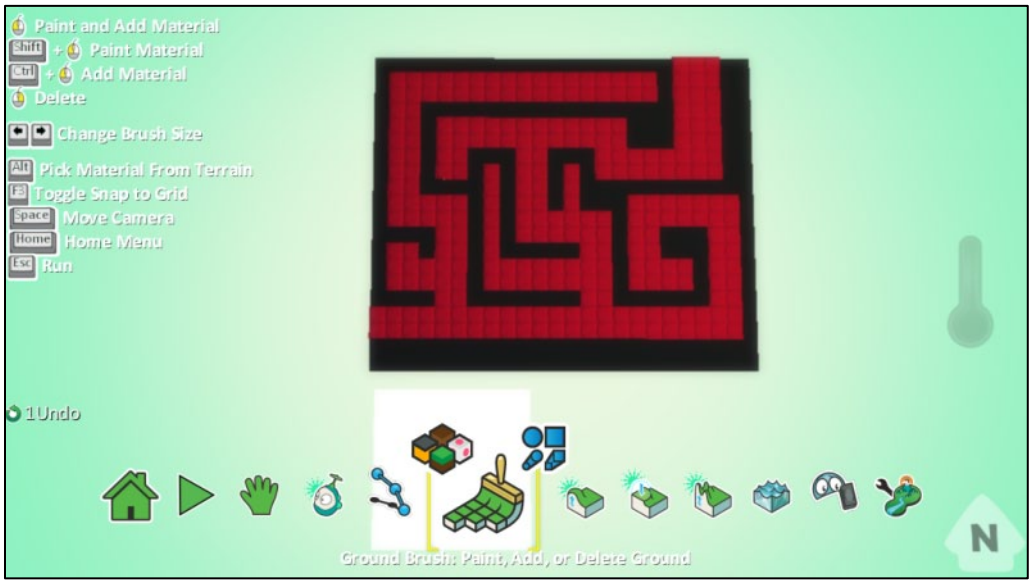
Εδώ πρέπει να πούμε ότι πιθανότατα δεν υπάρχει ένας και μοναδικός τρόπος για να δημιουργήσουμε αυτόν τον λαβύρινθο. Μπορούμε να ακολουθήσουμε διαφορετικές τεχνικές για να υλοποιήσουμε οτιδήποτε έχουμε φανταστεί. Εμείς θα συζητήσουμε για έναν από αυτούς.

Επιλέγοντας το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** και την ***επιλογή αλλαγής υλικού*** αρχικά δημιουργούμε ένα επίπεδο έδαφος με το υλικό με νούμερο 25.



Στη συνέχεια επιλέγουμε ένα διαφορετικό υλικό και δημιουργούμε διαδρομές στον αρχικό επίπεδο κόσμο μας έτσι ώστε τουλάχιστον μια από αυτές να συνδέει την είσοδο με την έξοδο, ενώ οι υπόλοιπες απλά να μπερδεύουν τον παίκτη.

Για να δημιουργήσουμε τις διαδρομές πρέπει να επιλέξουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** και στη συνέχεια να επιλέξουμε υλικό που είναι διαφορετικό από το αρχικό (π.χ. νούμερο 46). Σειρά έχει το σχήμα της βούρτσας που θα πρέπει να είναι μάλλον η ***τετράγωνη βούρτσα γραμμής (Linear square brush)*** και το μέγεθος της βούρτσας που θα πρέπει να είναι σχετικά μικρό. Έτσι, κάνοντας αριστερό κλικ στις περιοχές που θέλουμε να δημιουργήσουμε τις διαδρομές, το αποτέλεσμα είναι το παρακάτω.



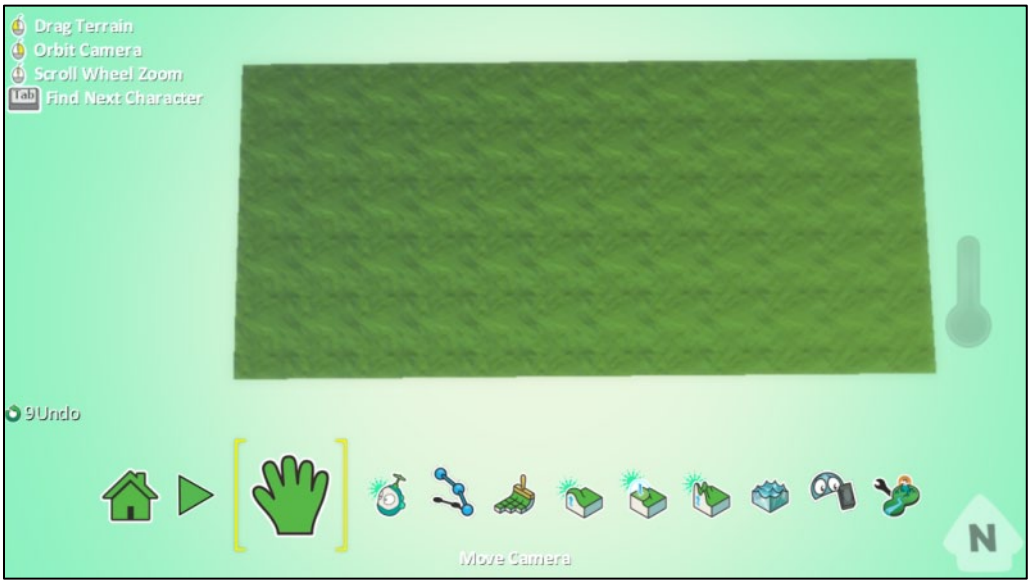
Τέλος, χρησιμοποιώντας τη μαγική βούρτσα του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)***, και πατώντας αριστερό κλικ υψώνουμε το αρχικό υλικό και ο λαβύρινθος είναι έτοιμος!

4.2.2 Αγώνες ταχύτητας

Ας δούμε τώρα τι κόσμο θα μπορούσαμε να δημιουργήσουμε για ένα παιχνίδι αγώνων ταχύτητας. Τι θα λέγατε να προσπαθήσουμε να φτιάξουμε τον κόσμο που φαίνεται στην παρακάτω εικόνα;

Δείτε το παράδειγμα *04_04.kodu*

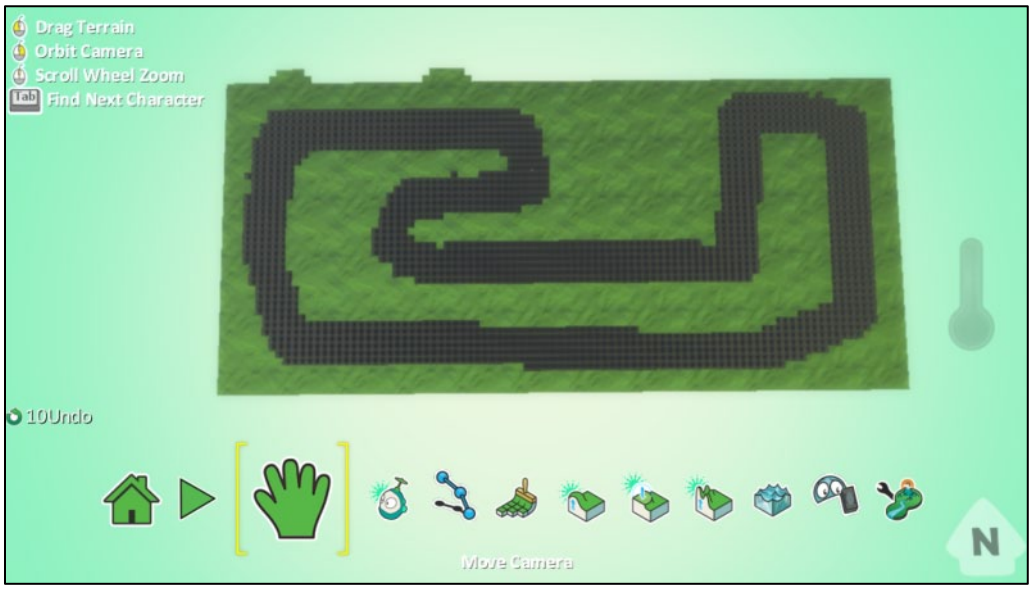
Ξεκινάμε δημιουργώντας με το υλικό με νούμερο 20 μια μεγάλη ορθογώνια πίστα.



Το επόμενο βήμα είναι να δημιουργήσουμε τη διαδρομή μέσα στην οποία θα κινούνται οι χαρακτήρες του παιχνιδιού μας.

Επιλέγουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, ορίζουμε ως υλικό δρόμου το υλικό με νούμερο 25 και με τα βέλη του πληκτρολογίου καθορίζουμε το μέγεθος της βούρτσας ανάλογα με το πόσο φαρδύς θέλουμε να είναι ο δρόμος που θα σχεδιάσουμε.

Αφού έχει γίνει η απαραίτητη προεπιλογή σχήματος βούρτσας (προτείνεται η ***Στρογγυλή βούρτσα (Hard round brush)*** για τις στροφές και η ***τετράγωνη βούρτσα γραμμής (Linear square brush)*** για τις ευθείες) με αριστερό κλικ επάνω στον αρχικό κόσμο σχεδιάζουμε τη διαδρομή όπως φαίνεται παρακάτω.



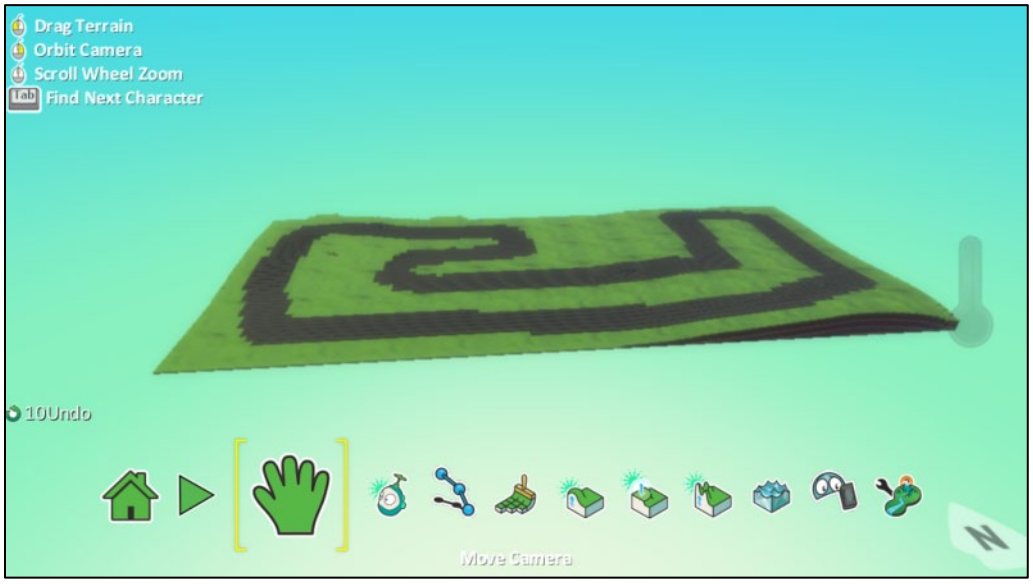
Τώρα είναι η κατάλληλη στιγμή να διαμορφώσουμε το έδαφος, ώστε αν και λείο να έχει διαφορετικό υψόμετρο σε ορισμένα σημεία. Για να πετύχουμε κάτι τέτοιο πρέπει να συνδυάσουμε σωστά τη χρήση δύο εργαλείων, του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)*** και του εργαλείου ***Λείανση εδάφους (Flatten)***.

Πρώτα, με τη βούρτσα του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)*** κάνουμε αριστερό κλικ πάνω στις περιοχές στις οποίες θέλουμε να δημιουργήσουμε ύψωμα. Ωστόσο, θα χρειαστεί να προσέξουμε το χειρισμό του εργαλείου, αφού δεν θέλουμε να δημιουργήσουμε ψηλούς λόφους, αλλά μια μικρή υψομετρική διαφορά. Το καλύτερο είναι να χρησιμοποιήσουμε τα βέλη του πληκτρολογίου για να ορίσουμε σχετικά μεγάλο μέγεθος βούρτσας και να κάνουμε κλικ πολύ μικρής διάρκειας ώστε να έχουμε καλύτερο έλεγχο στο αποτέλεσμα του εργαλείου.

Στη συνέχεια χρησιμοποιούμε το εργαλείο ***Λείανση εδάφους (Flatten)*** ώστε να λειάνουμε όλες τις περιοχές στις οποίες προηγουμένως εφαρμόσαμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***. Για δική σας διευκόλυνση και για καλύτερο αποτέλεσμα προτείνεται η βούρτσα του εργαλείου *Λείανση εδάφους (Flatten)* να έχει σχετικά μεγάλο μέγεθος και να καλύπτει αρκετά μεγάλη επιφάνεια επίπεδου εδάφους.

Μια δοκιμή θα σας κάνει να εξοικειωθείτε περισσότερο με το εργαλείο αυτό και να καταλάβετε τις δυνατότητές του.

Στην εικόνα φαίνεται το αποτέλεσμα της διαμόρφωσης του εδάφους.



Τώρα θα μπορούσαμε να προσθέσουμε στις «επικίνδυνες» στροφές προστατευτικά ρείθρα. Επιλέγουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, το υλικό με νούμερο 16, αρκετά

μικρό μέγεθος βούρτσας και με αριστερό κλικ αρχίζουμε να "καλύπτουμε" τις στροφές όπως φαίνεται παρακάτω.

Μπορούμε να υψώσουμε λίγο τα ρείθρα χρησιμοποιώντας το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***, και τη ***Μαγική Βούρτσα (Magic Brush)*** και πατώντας αριστερό κλικ του ποντικιού.

Στη συνέχεια θα φτιάξουμε το χώρο στάθμευσης των οχημάτων/χαρακτήρων που θα έχει το παιχνίδι. Με το υλικό που έχουμε σχεδιάσει τις διαδρομές (νούμερο 25) δημιουργούμε μια επέκταση του εδάφους. Μπορούμε στο εσωτερικό της να ζωγραφίσουμε κάθετες γραμμές (χρησιμοποιώντας πολύ μικρό μέγεθος βούρτσας και το υλικό με νούμερο 16) ώστε να διαχωρίζουμε τις θέσεις παρκινγκ!

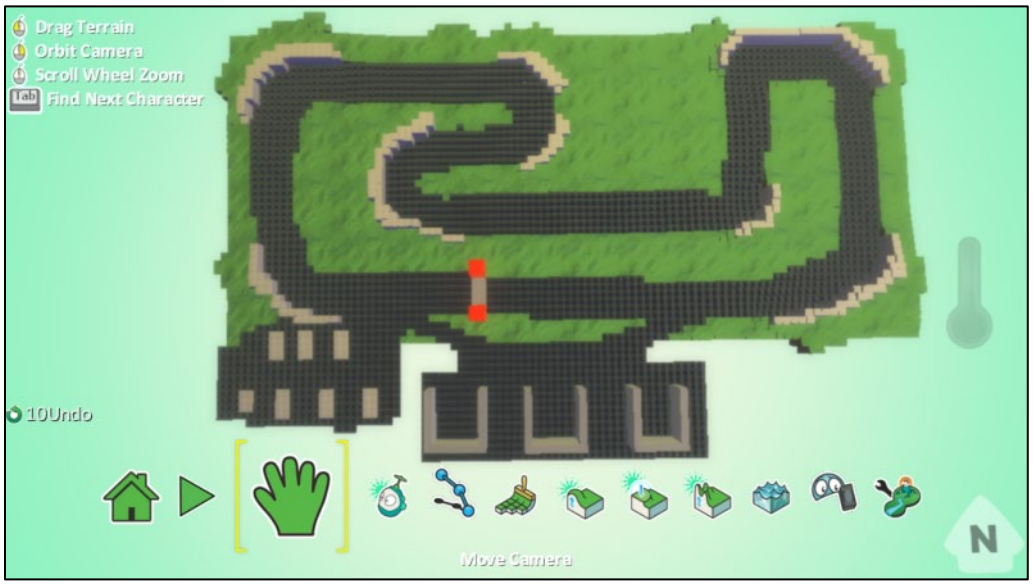
Ακόμα, μπορούμε να δημιουργήσουμε ένα σημείο στο οποίο τα οχήματα/χαρακτήρες του παιχνιδιού είτε θα επισκευάζονται είτε θα ανεφοδιάζονται. Επιλέγουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** και με το υλικό που έχουμε σχεδιάσει το δρόμο (μπορούμε να χρησιμοποιήσουμε τη ***δειγματοληψία*** πατώντας Alt και αριστερό κλικ) θα φτιάξουμε μια επιπλέον επέκταση του κόσμου μας όπως αυτή που φαίνεται παρακάτω. Το μέγεθος της βούρτσας θα πρέπει να είναι μικρό για τους δρόμους εισόδου και εξόδου από την κυρίως διαδρομή, ενώ θα πρέπει να είναι σχετικά μεγάλο για τη δημιουργία της υπόλοιπης περιοχής. Επίσης μπορούμε να σχεδιάσουμε τρία σχήματα όπως αυτά που φαίνονται στην εικόνα, επιλέγοντας τη ***Βούρτσα Εδάφους (Ground Brush)***, το υλικό με νούμερο 16, και ***Γραμμική τετράγωνη βούρτσα (Linear square brush)*** πολύ μικρού μεγέθους. Τα σχήματα αυτά μπορούμε να τα ανυψώσουμε χρησιμοποιώντας τη μαγική βούρτσα του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)***, σε συνδυασμό με αριστερό κλικ.

Οι τελευταίες πινελιές στο έργο μας θα μπουν με το σχεδιασμό της γραμμής εκκίνησης/τερματισμού καθώς και τους πλαϊνούς πυλώνες. Για τη γραμμή εκκίνησης/τερματισμού επιλέγουμε στο εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** το υλικό με νούμερο *16*, σχήμα ***Γραμμική τετράγωνη Βούρτσα (Linear square brush)*** και καθορίζουμε από τα βέλη του πληκτρολογίου πάρα πολύ μικρό μέγεθος. Στη συνέχεια κάνουμε αριστερό κλικ από το ένα άκρο του δρόμου στο άλλο άκρο προσπαθώντας να δημιουργήσουμε μια κάθετη γραμμή.

Για να σχεδιάσουμε τους πλαϊνούς πυλώνες αρκεί να επιλέξουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, *το υλικό με νούμερο* 118 και τετράγωνη βούρτσα της οποίας το μέγεθος θα πρέπει να είναι πάρα πολύ μικρό. Αριστερά και δεξιά του δόμου κάνουμε αριστερό κλικ ώστε να

δημιουργηθούν δύο τετράγωνα σχήματα τα οποία στη συνέχεια θα ανυψώσουμε. Για την ανύψωση χρησιμοποιούμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***, επιλέγουμε τη **μαγική βούρτσα** και πατάμε αριστερό κλικ επάνω στα δύο τετράγωνα σχήματα που δημιουργήσαμε.

Σε αυτό το σημείο η πίστα αγώνων ολοκληρώθηκε.



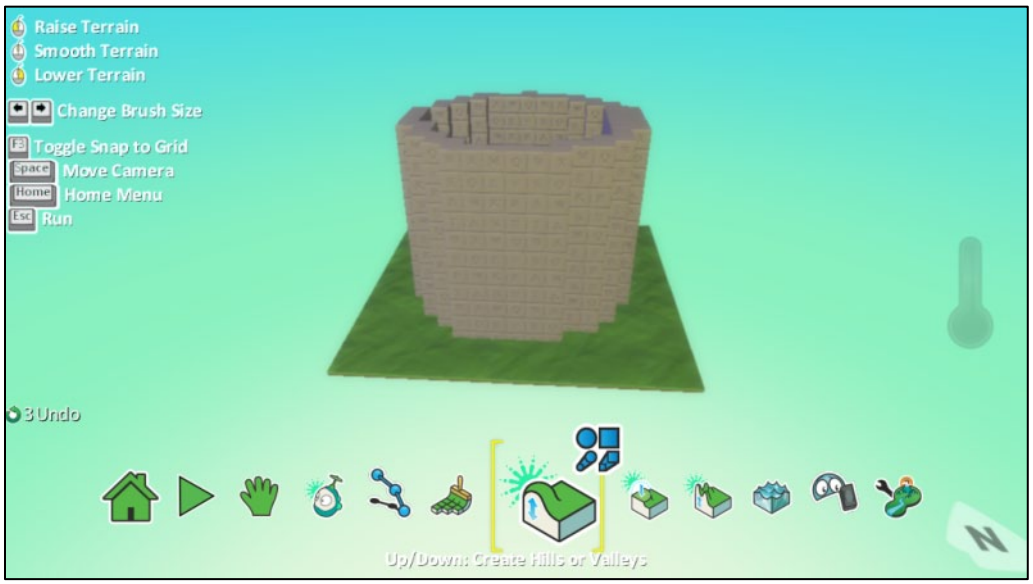
4.2.3 Κάστρα και πολιορκητές

Ας δούμε τώρα μια πίστα βγαλμένη από παραμύθι, με κάστρα, πολεμίστρες, ακόμη και τάφρο!

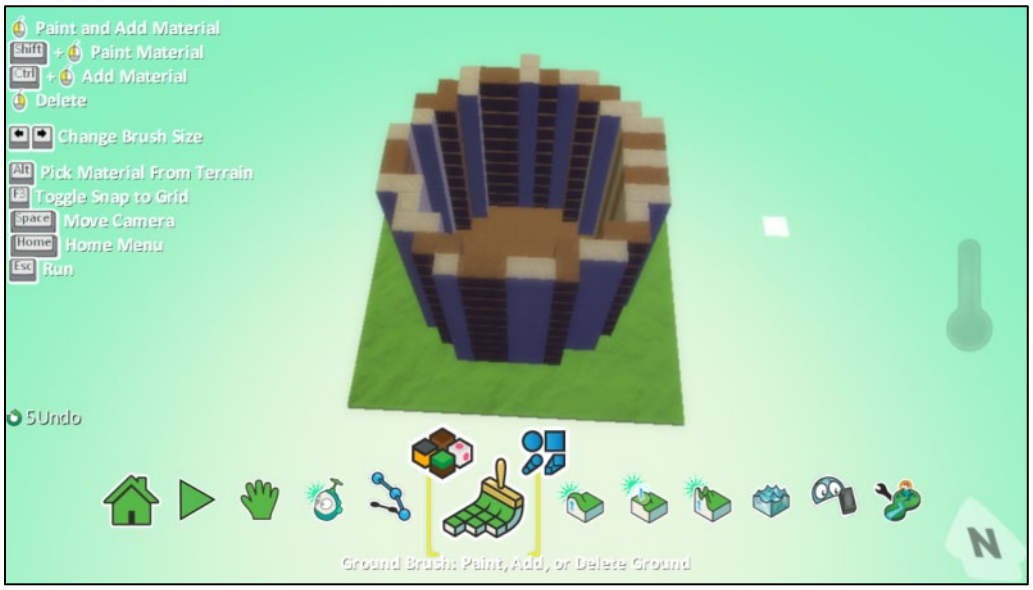
Πώς μπορούμε λοιπόν να δημιουργήσουμε ένα κάστρο παρόμοιο με αυτό της παραπάνω εικόνας; Είναι τόσο δύσκολο όσο φαίνεται να φτιάξουμε τις πολεμίστρες ή την τάφρο γύρω από το κάστρο; Ας προσπαθήσουμε λοιπόν.

Αρχικά, σε ένα επίπεδο έδαφος θα πρέπει να σχεδιάσουμε έναν κύκλο από διαφορετικό υλικό. Επιλέγουμε λοιπόν το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, το υλικό με νούμερο 14 και σχήμα ***Σκληρή στρογγυλή βούρτσα (Hard round brush)*** με σχετικά μεγάλο μέγεθος (θυμίζουμε ότι το μέγεθος της βούρτσας το καθορίζουμε από τα βέλη του πληκτρολογίου). Στη συνέχεια, επάνω στον αρχικό επίπεδο κόσμο κάνουμε αριστερό κλικ ώστε να μια δημιουργηθεί μια κυκλική επιφάνεια. Επιλέγουμε και πάλι το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, το υλικό 78 και σχήμα ***Σκληρή στρογγυλή βούρτσα (Hard round brush)*** με μέγεθος μικρότερο από αυτό του πρώτου κύκλου που σχηματίσαμε. Στο εσωτερικό του πρώτου κύκλου κάνουμε αριστερό κλικ ώστε να δημιουργηθεί ο δεύτερος κύκλος από το υλικό με νούμερο 78. Ωστόσο, χρειάζεται προσοχή ώστε οι δύο κύκλοι που σχηματίσαμε να είναι κατά προσέγγιση ομόκεντροι όπως φαίνεται και στην εικόνα:

Στη συνέχεια, χρησιμοποιώντας τη **μαγική βούρτσα** υψώνουμε τον εξωτερικό κύκλο και το κάστρο μας είναι σχεδόν έτοιμο, όπως φαίνεται και στην εικόνα:

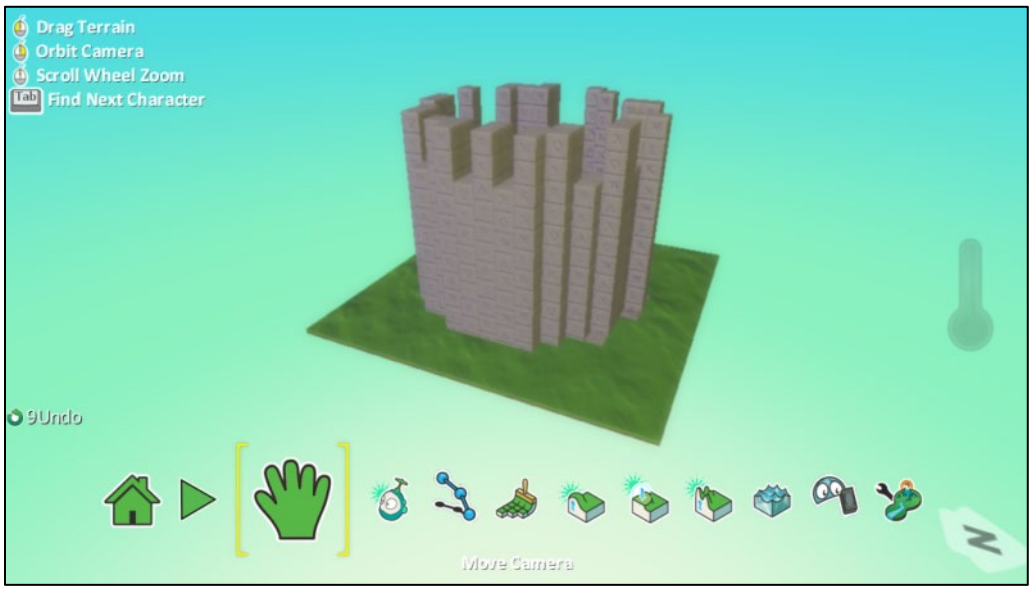


Το μόνο που μένει πια είναι να δημιουργήσουμε τις πολεμίστρες! Για να το κάνουμε αυτό μπορούμε με το υλικό που δημιουργήσαμε τον εσωτερικό κύκλο, δηλαδή με το υλικό με νούμερο 78, να ζωγραφίσουμε κάποια μέρη του τοίχους, όπως βλέπουμε στην επόμενη εικόνα. Θα χρειαστεί να μικρύνουμε τη βούρτσα με τα βέλη του πληκτρολογίου. Επίσης, θα πρέπει να είμαστε προσεκτικοί για να μη χρωματίζουμε άλλα μέρη εκτός του κάστρου. Καλό να μετακινήσετε την κάμερα ώστε να βλέπετε την κάτοψη της πίστας σας.



Με τη ***μαγική βούρτσα*** κατεβάζουμε σε ύψος τα μέρη που ζωγραφίσαμε με το υλικό νούμερο 78 (εδώ μας εξυπηρετεί το γεγονός ότι διαλέξαμε το ίδιο υλικό με αυτό του εσωτερικού κύκλου, οπότε κατεβαίνει το ύψος όλων των πολεμίστρων ταυτόχρονα!). Οι πολεμίστρες μας δημιουργήθηκαν!

Τέλος, μπορούμε να ζωγραφίσουμε τα τμήματα του τοίχους των πολεμιστρών με το αρχικό τους υλικό (νούμερο 14) και το κάστρο πλέον περιμένει τους πολεμιστές!

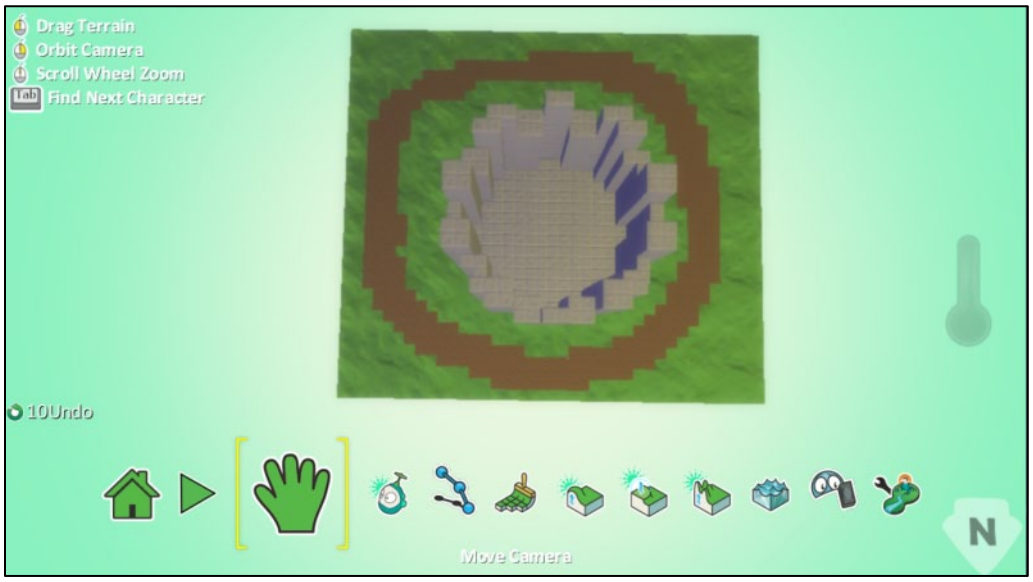


Θα μπορούσαμε σε αυτό το σημείο να προσθέσουμε ένα ποτάμι γύρω από το κάστρο μας.

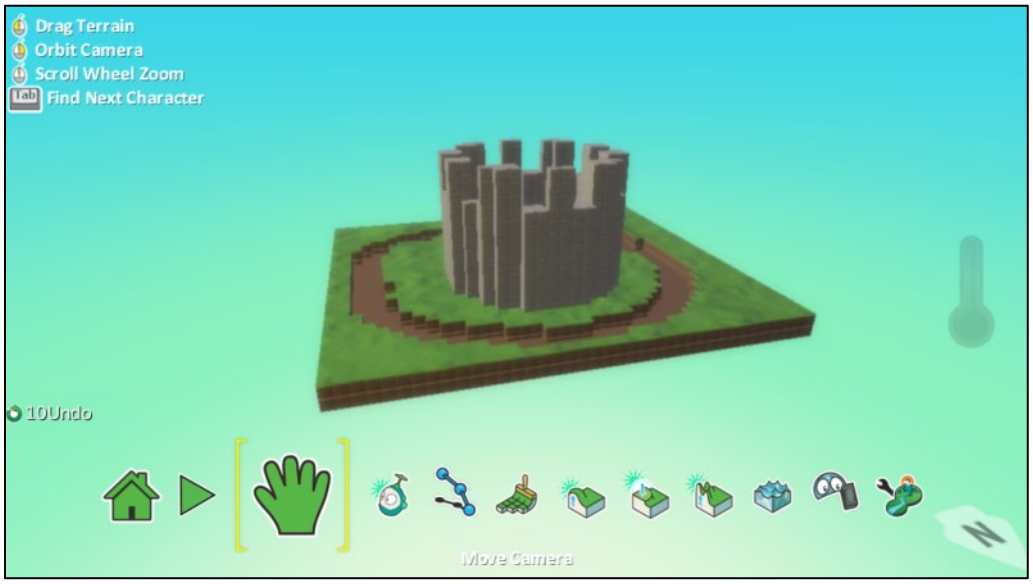
Αρχικά, θα πρέπει να ελέγξουμε αν αρκεί το έδαφος γύρω από το κάστρο ή αν θα πρέπει να αυξήσουμε το μέγεθος της πίστας μας, ώστε να υπάρχει χώρος για το ποτάμι.

Αν χρειαστεί επιπλέον έδαφος, χρησιμοποιώντας το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** και επιλέγοντας το κατάλληλο έδαφος, το σχήμα (προτιμάται η ***Γραμμική τετράγωνη βούρτσα (Linear square brush)*** ως πιο εύχρηστη σε αυτή την περίπτωση) και το μέγεθος της βούρτσας, προσθέτουμε νέα τμήματα εδάφους.

Στη συνέχεια, επιλέγουμε ένα άλλο υλικό (για παράδειγμα το υλικό με νούμερο 18), διαλέγουμε τη ***στρογγυλή βούρτσα***, καθορίζουμε το μέγεθος που θέλουμε να έχει και σε συνδυασμό με κατάλληλο χειρισμό της κάμερας σχεδιάζουμε έναν κυκλικό περίγραμμα γύρω από το κάστρο, όπως φαίνεται στην εικόνα.



Θα πρέπει τώρα να υψώσουμε το έδαφος που μοιάζει με γρασίδι, ώστε να δημιουργηθεί η περιοχή χαμηλότερου υψώματος στην οποία θα προσθέσουμε νερό. Χρησιμοποιώντας το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)***, επιλέγουμε τη μαγική βούρτσα και πατώντας αριστερό κλικ στις περιοχές με το γρασίδι παρατηρούμε ότι αυτές ανυψώνονται όπως φαίνεται στην επόμενη εικόνα:



Το μόνο που μένει τώρα να προσθέσουμε είναι το νερό! Επιλέγουμε το ***Εργαλείο Νερού (Water Tool)*** και κάνουμε αριστερό κλικ στη βυθισμένη περιοχή που δημιουργήσαμε, όσες φορές χρειαστεί, προσέχοντας όμως να μην ξεχειλίσει!

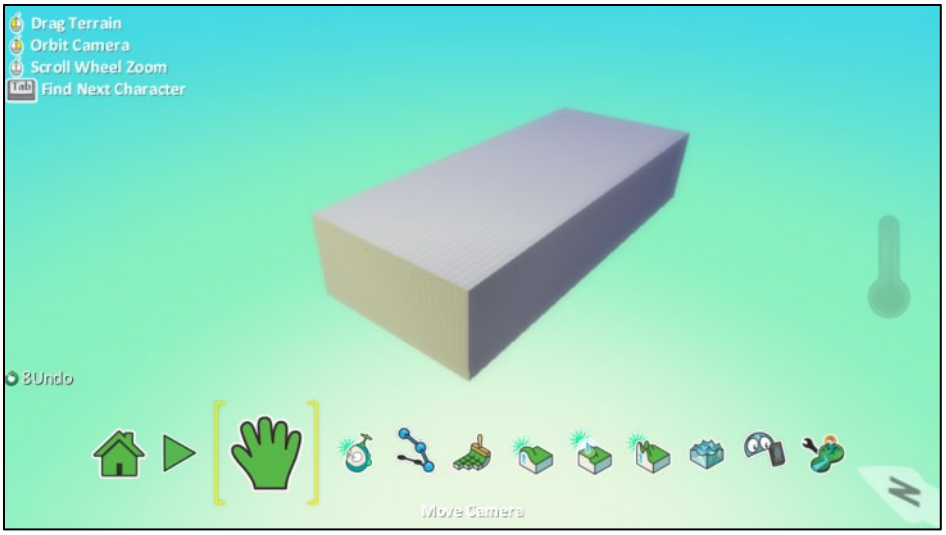
Το τελικό αποτέλεσμα είναι το παρακάτω.

Δείτε το παράδειγμα
04_05.kodu

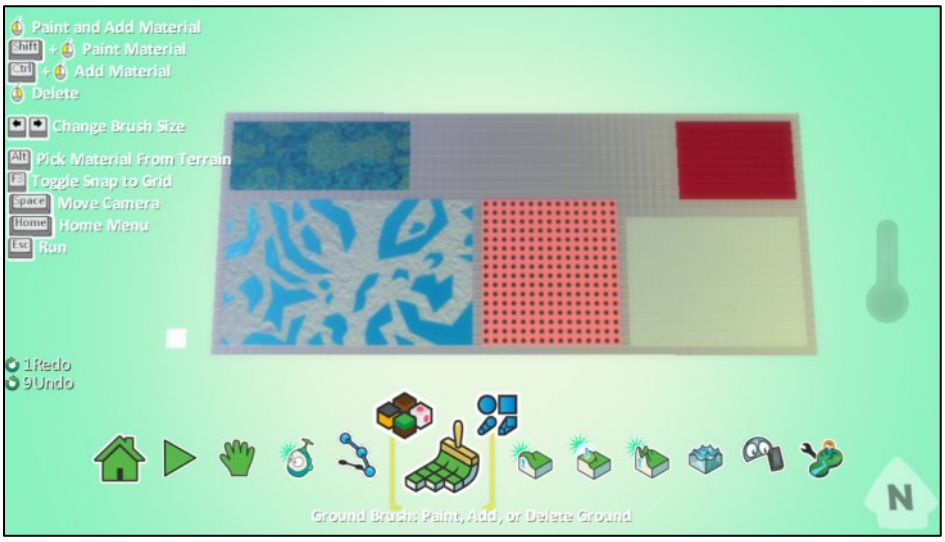
4.2.4 Το σπίτι του Kodu

Τι θα λέγατε να δημιουργήσουμε ένα ολόκληρο σπίτι για το παιχνίδι μας!

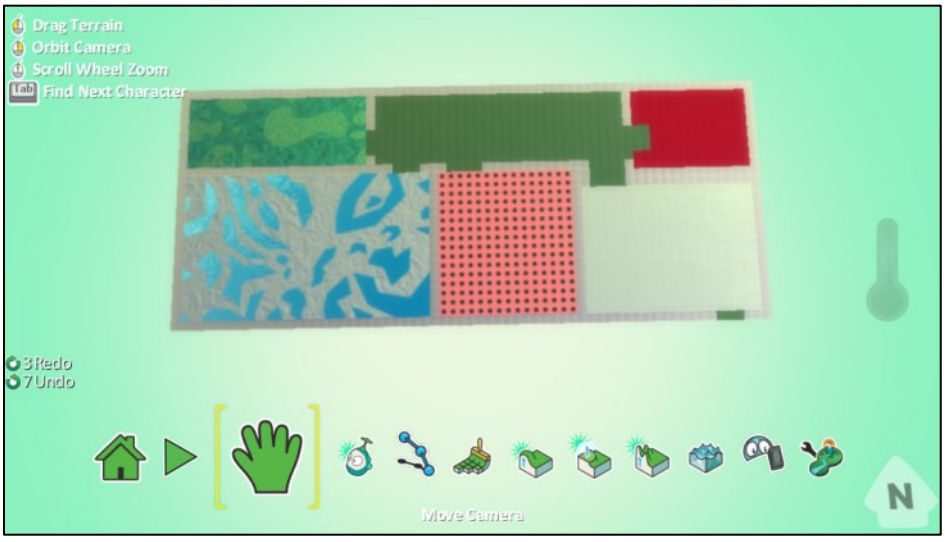
Αρχικά δημιουργούμε ένα αρκετά μεγάλο επίπεδο έδαφος χρησιμοποιώντας το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)*** και το υλικό με νούμερο 27. Στη συνέχεια υψώνουμε αρχικά το επίπεδο έδαφος με το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** και τη ***μαγική βούρτσα*** κάνοντας αρκετά αριστερά κλικ. Το αποτέλεσμα θα είναι το παρακάτω:



Σε αυτό το σημείο, μετακινούμε την κάμερα ώστε να βλέπουμε την κάτοψη του «κτιρίου» μας και σχηματίζουμε στο πάνω μέρος του τη μορφή των δωματίων του σπιτιού, χρησιμοποιώντας τα υλικά που επιθυμούμε, όπως φαίνεται στην εικόνα.

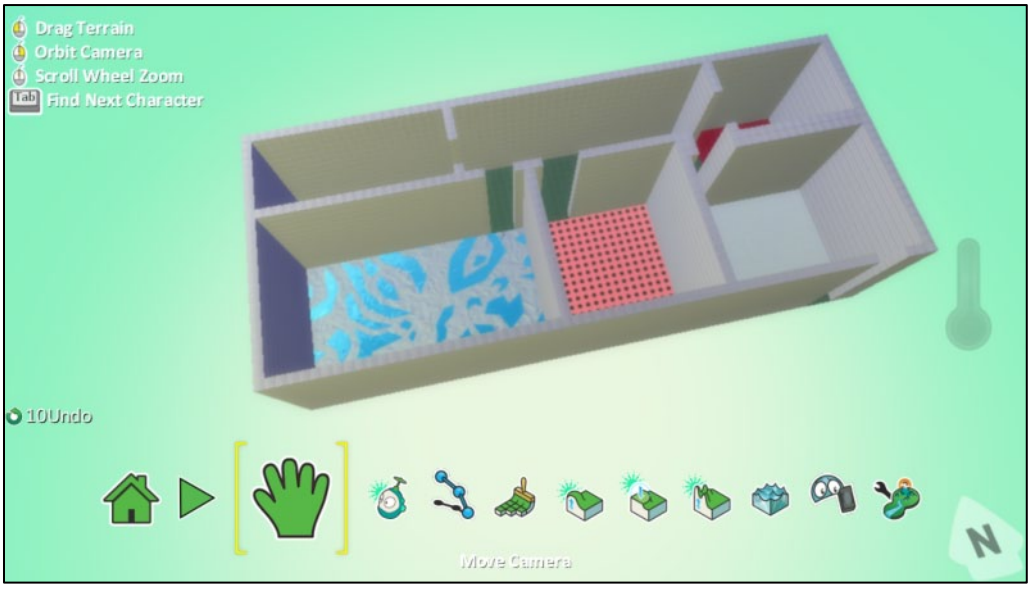


Ωστόσο, πρέπει να προσέξουμε αν θέλουμε το σπίτι μας να έχει έναν διάδρομο με πόρτες που συνδέουν τα δωμάτια. Για το σχηματισμό του διαδρόμου και των πορτών, απλά πρέπει να φέρουμε σε επαφή τα διαφορετικά υλικά των δωματίων.



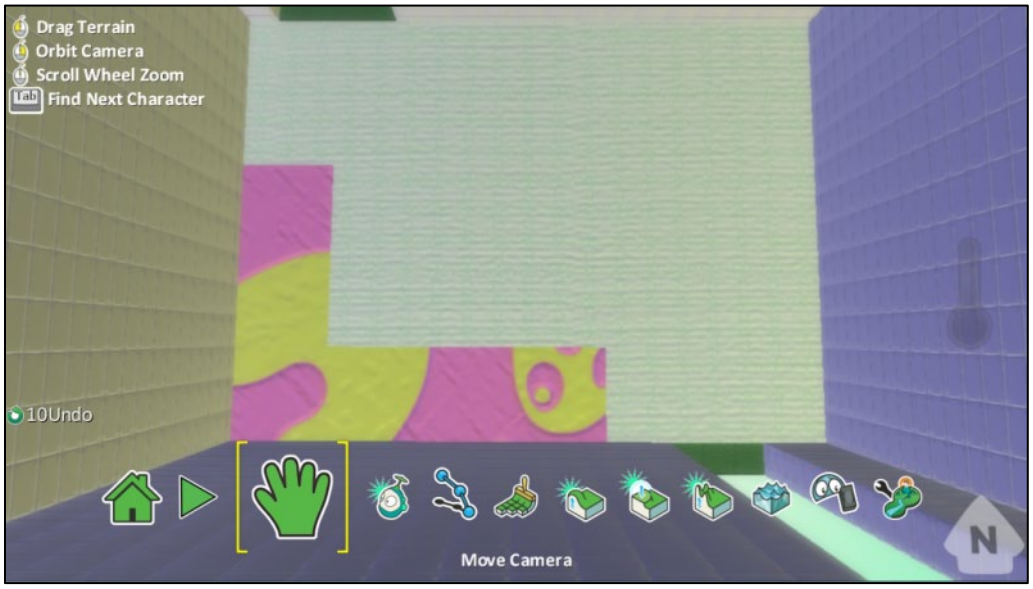
Σε πίστες με διαφορετικούς τύπους εδάφους, μη ξεχνάτε ότι μπορείτε να διορθώνετε εύκολα τα λάθη σας χρησιμοποιώντας το πλήκτρο Alt σε συνδυασμό με το δεξί κλικ (δειγματοληψία) για να επιλέξετε έναν τύπο εδάφους που ήδη υπάρχει.

Χρησιμοποιώντας τη ***μαγική βούρτσα*** του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)*** και κάνοντας δεξί κλικ κατεβάζουμε το ύψος των περιοχών που αντιστοιχούν στα δωμάτια και δημιουργούμε ένα αποτέλεσμα σαν αυτό:

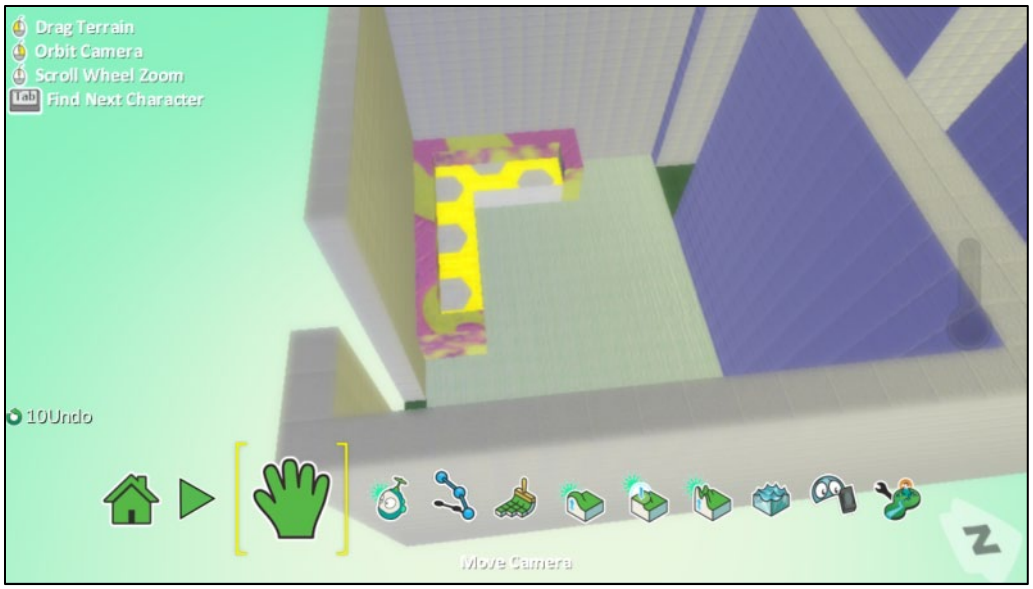


Δεν τελειώσαμε όμως! Ας σχεδιάσουμε τώρα το καθιστικό! Στις εικόνες φαίνονται τα βήματα για τη δημιουργία ενός καναπέ.

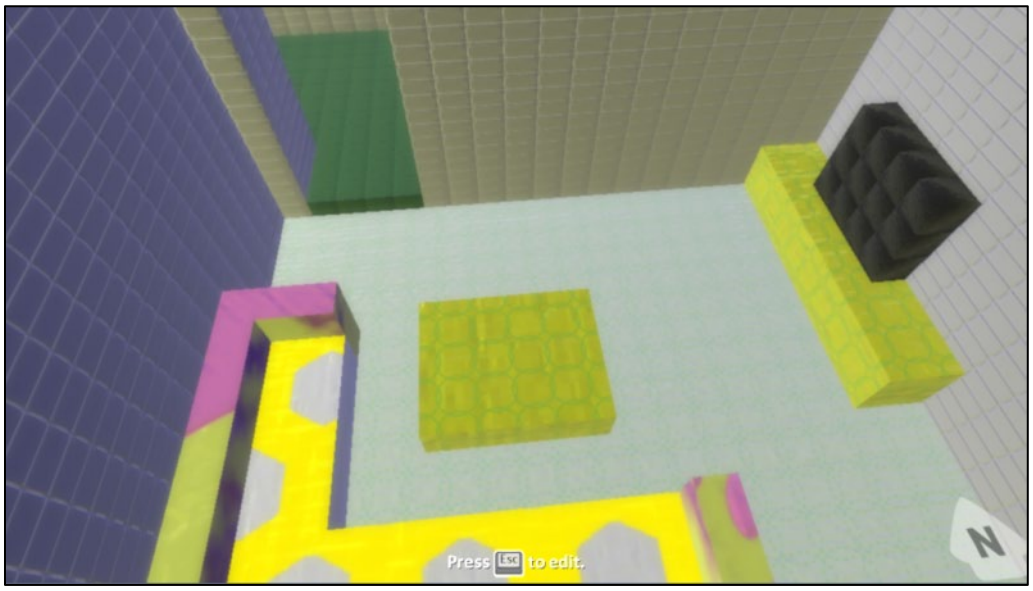
Αρχικά, μετακινούμε την κάμερα και κάνουμε zoom στο δωμάτιο που θέλουμε να διαμορφώσουμε. Στη συνέχεια σχεδιάζουμε με το υλικό που θέλουμε το σχήμα του καναπέ. Στο συγκεκριμένο παράδειγμα, επιλέγουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush)***, ***σχήμα Γραμμική τετράγωνη βούρτσα (Linear square brush)*** και μέγεθος σχετικά μικρό. Σε αυτό το σημείο θα πρέπει να προσέξουμε ώστε να μην επικαλύψουμε με το υλικό του καναπέ τους πλαϊνούς τοίχους. Το επόμενο βήμα είναι να χρησιμοποιήσουμε το εργαλείο ***Λόφοι και Κοιλάδες (Up/Down)*** και την αντίστοιχη ***μαγική βούρτσα*** ώστε, κάνοντας λίγα αριστερά κλικ στον καναπέ που έχουμε σχεδιάσει, να του δώσουμε ύψος.



Με ένα άλλο υλικό σχεδιάζουμε το μέρος που θέλουμε να χαμηλώσει σε ύψος και χρησιμοποιούμε και εδώ τη μαγική βούρτσα αλλά με δεξί κλικ αυτή τη φορά για να πετύχουμε το παρακάτω αποτέλεσμα.

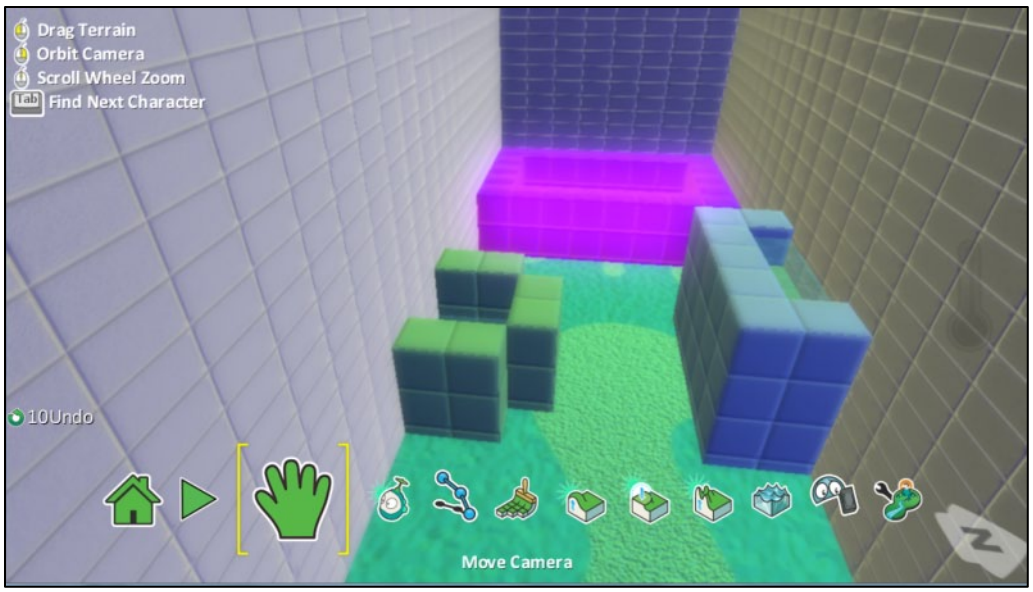


Σειρά έχει ένα τραπεζάκι και μια τηλεόραση.



Ας προσπαθήσουμε να φτιάξουμε το μπάνιο του σπιτιού!

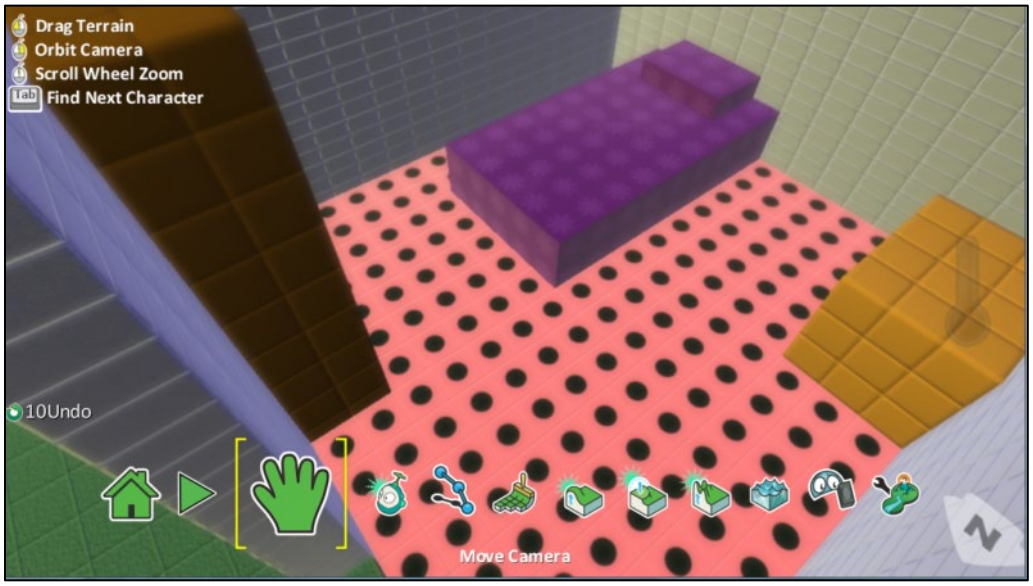
Όπως και προηγουμένως, θα πρέπει να μετακινήσουμε κατάλληλα την κάμερα ώστε να επικεντρωθούμε στο χώρο που θα διαμορφώσουμε. Επιλέγουμε το ***εργαλείο Βούρτσα Εδάφους (Ground Brush)*** και αφού καθορίσουμε ένα υλικό διαφορετικό από το «πάτωμα» του δωματίου, με αριστερό κλικ δημιουργούμε ορθογώνιες περιοχές που θα παίξουν το ρόλο της μπανιέρας και της βρύσης. Στη συνέχεια, με τη ***μαγική βούρτσα*** του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)*** και πατώντας αριστερό κλικ ανυψώνουμε τα αρχικά ορθογώνια σχήματα. Το επόμενο βήμα είναι να επιλέξουμε το εργαλείο ***Βούρτσα Εδάφους (Ground Brush),***και με ένα πάρα πολύ μικρό μέγεθος βούρτσας και ένα διαφορετικό υλικό επικαλύπτουμε τα μέρη στα οποία θα μειώσουμε το ύψος ώστε να δημιουργηθεί η αίσθηση της μπανιέρας και του νιπτήρα. Αφού έχουμε σχεδιάσει τις περιοχές που πρόκειται να "βυθίσουμε", χρησιμοποιούμε τη ***μαγική βούρτσα*** του εργαλείου ***Λόφοι και Κοιλάδες (Up/Down)*** κάνοντας δεξί κλικ μικρής διάρκειας. Τέλος μπορούμε να χρησιμοποιήσουμε το ***Εργαλείο Νερού (Water Tool)*** ώστε να γεμίσουμε με νερό την μπανιέρα και το νιπτήρα. Το τελικό αποτέλεσμα φαίνεται παρακάτω:



Όπως παρατηρήσατε, έχουμε χρησιμοποιήσει τα εργαλεία ***Βούρτσα Εδάφους (Ground Brush)*** και ***Λόφοι και Κοιλάδες (Up/Down)***. Μόνο με αυτά λοιπόν μπορούμε να φτιάξουμε και όλα τα υπόλοιπα δωμάτια του σπιτιού, τα οποία είναι:

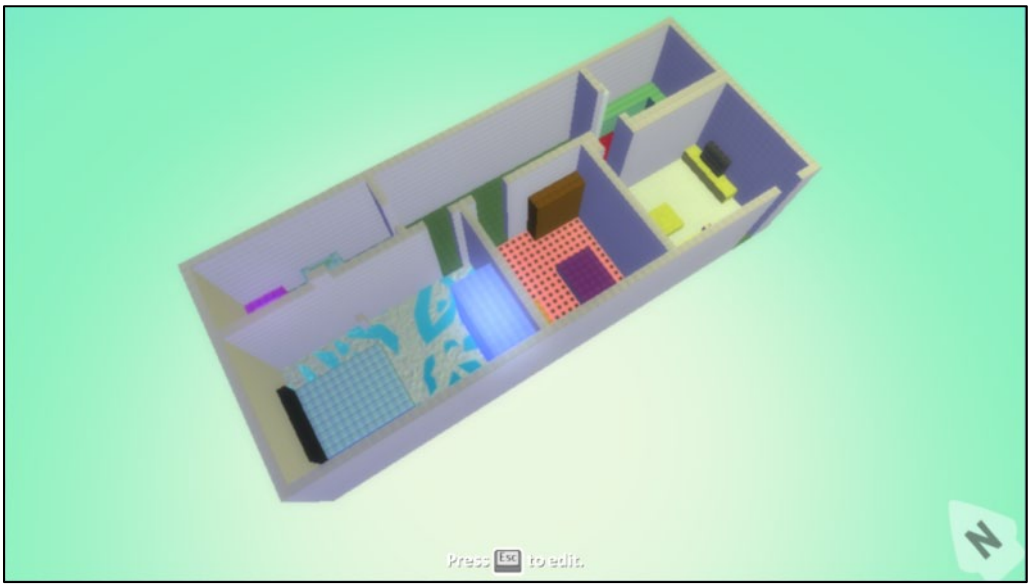
η κουζίνα,

ένα παιδικό δωμάτιο,



και ένα ακόμα υπνοδωμάτιο.

Το σπίτι του Kodu είναι έτοιμο και μια πανοραμική εικόνα του είναι η παρακάτω:



Δείτε το παράδειγμα
04_06.kodu

Περίληψη

Στο κεφάλαιο αυτό, γνωρίσαμε τα βασικά εργαλεία που παρέχει το MSKodu για τη δημιουργία κόσμων και πιο συγκεκριμένα τα εργαλεία ***Βούρτσα Εδάφους (Ground Brush)***, ***Λόφοι και Κοιλάδες (Up/Down)***, ***Λείανση Εδάφους (Flatten)***, ***Σκλήρυνση Επιφάνειας (Roughen)***, ***Εργαλείο Νερού (Water tool)***. Μελετήσαμε τη χρήση και το συνδυασμό αυτών των εργαλείων για τη δημιουργία ολοκληρωμένων κόσμων που περιέχουν λίμνες, κάστρα, σπίτια, μέχρι και τουαλέτα ☺. Είναι αυτονόητο ότι ο κόσμος επηρεάζει δραματικά την πλοκή των παιχνιδιών καθώς μπορεί να κρύβει εκπλήξεις (π.χ. στην περίπτωση ενός λαβυρίνθου), δυσκολίες (π.χ. αν ο χαρακτήρας πρέπει να περάσει πάνω από ποτάμια) ή μυστήρια (π.χ. καλά κρυμμένα αντικείμενα). Οι κόσμοι είναι όμως εξαιρετικά σημαντικοί και για έναν ακόμη λόγο: στο MSKodu τα αντικείμενα έχουν αισθητήρες με τους οποίους αντιλαμβάνονται τον τύπο του εδάφους πάνω στον οποίο βρίσκονται και μπορούν να ενεργούν ανάλογα!

Ερωτήσεις

1. Με ποιο συνδυασμό πληκτρολογίου και ποντικιού μπορούμε:

   α. να αλλάξουμε το υλικό σε μια συγκεκριμένη υπάρχουσα περιοχή χωρίς να προσθέσουμε έδαφος η να αλλοιώσουμε το σχήμα αυτής;

   β. να βρούμε το ακριβές υλικό που χρησιμοποιήθηκε σε ένα κομμάτι του κόσμου μας χωρίς να χρειαστεί να ψάξουμε όλες τις επιλογές υλικού;

   γ. να προσθέσουμε έδαφος στον κόσμο μας χωρίς να καλύψουμε με το νέο υλικό που επιλέξαμε κάποια ήδη υπάρχουσα περιοχή του κόσμου μας;

2. Αναφέρετε ονομαστικά τα πέντε (5) εργαλεία εδάφους που εξετάσαμε και περιγράψτε εν συντομία μια εφαρμογή που έχει το καθένα στη δημιουργία μιας πίστας.

3. Περιγράψτε για τις λειτουργίες που πραγματοποιούν τα επόμενα τρία (3) εργαλεία πατώντας αριστερό και δεξί κλικ του ποντικιού.

   α. Βούρτσα Εδάφους: Προσθέστε Έδαφος/Διαγράψτε Έδαφος (Ground Brush: Paint, add or delete ground)

   β. Λόφοι και Κοιλάδες: Δημιουργήστε Λόφους ή Κοιλάδες (Up/Down: Create Hills or Valleys)

   γ. Λείανση Εδάφους: Κάντε το Έδαφος Λείο ή Επίπεδο (Flatten: Make Ground Smooth or Level)

4. Ποια εργαλεία εδάφους και ποια επιλογή σχήματος βούρτσας θα προτείνατε να χρησιμοποιήσουμε για να δημιουργήσουμε μία πισίνα;

Δραστηριότητες

1. Φτιάξτε ένα γήπεδο ποδοσφαίρου ή ένα γήπεδο βόλεϊ που δεξιά και αριστερά του θα υπάρχει γρασίδι και κερκίδες για τους θεατές.

2. Δημιουργήστε ένα δωμάτιο που θα αναπαριστά το πραγματικό υπνοδωμάτιο του σπιτιού σας. Συμβουλευτείτε την ενότητα 4.2.4 του κεφαλαίου για να φτιάξετε τα έπιπλα και τους τοίχους του δωματίου.

Κεφάλαιο 5ο: Συμπεριφέρομαι

Σε αυτό το κεφάλαιο θα δώσουμε έμφαση πως μπορείτε να προγραμματίσετε τους χαρακτήρες σας ώστε να αντιδρούν στο πάτημα των πλήκτρων του πληκτρολόγιου ή στα κλικ του ποντικιού! Αυτήν την δυνατότητα θα την ονομάσουμε «*αλληλεπίδραση του αντικειμένου με τον χρήστη*». Επιπλέον, πιο μεθοδικά, θα εξετάσουμε μια σειρά διαφορετικών αισθητήρων και ενεργειών μέσα από μια σειρά παραδειγμάτων.

5.1 Συμπεριφορές: όταν (γεγονός), κάνε (ενέργεια)

Όπως είδαμε στο Κεφάλαιο 3, ο προγραμματισμός των αντικειμένων έχει σχέση με το να αποδώσουμε συμπεριφορές στο αντικείμενο αυτό. Πάμε να θυμηθούμε τι σημαίνει συμπεριφορά ενός αντικειμένου και πως ορίζεται. Φέρτε στο νου σας το παιχνίδι Super Mario. Κάποιες από τις συμπεριφορές που έχει ο πρωταγωνιστής του παιχνιδιού, δηλαδή το αντικείμενο Super Mario, είναι:

- **ΟΤΑΝ** $\overbrace{\text{πέσει πάνω σε μία χελώνα}}^{\textbf{ΓΕΓΟΝΟΣ}}$, **ΤΟΤΕ** $\overbrace{\text{ο Super Mario ζαλίζεται πολύ}}^{\textbf{ΕΝΕΡΓΕΙΑ}}$
- **ΟΤΑΝ** $\overbrace{\text{έχει ένα αστέρι}}^{\textbf{ΓΕΓΟΝΟΣ}}$, **ΤΟΤΕ** $\overbrace{\text{ο Super Mario τρέχει γρήγορα}}^{\textbf{ΕΝΕΡΓΕΙΑ}}$
- **ΟΤΑΝ** $\overbrace{\text{κλωτσήσει ένα τούβλο}}^{\textbf{ΓΕΓΟΝΟΣ}}$, **ΤΟΤΕ** $\overbrace{\text{σπάει το τούβλο}}^{\textbf{ΕΝΕΡΓΕΙΑ}}$

Παρατηρούμε πως μια συμπεριφορά ορίζεται ως ο συνδυασμός ενός γεγονότος και μίας ενέργειας που εκτελείται όταν το αντικείμενο αντιληφθεί αυτό το γεγονός. Κατά αντίστοιχο τρόπο ορίζονται οι συμπεριφορές των αντικειμένων και στο MSKodu:

Τα γεγονότα γίνονται αντιληπτά από τους ***αισθητήρες*** του αντικείμενου μας. Για παράδειγμα στο MSKodu μπορούμε να προγραμματίσουμε ένα αντικείμενο ώστε να μπορεί να δει ένα άλλο αντικείμενο, να ακούσει ένα ήχο στον κόσμο, να καταλάβει ότι βρίσκεται πάνω από συγκεκριμένο τύπο εδάφους ή πάνω από νερό, ακόμη και να αντιληφθεί κάποια ενέργεια του χρήστη. Αυτοί είναι μερικοί από την πληθώρα αισθητήρων που μας παρέχει το MSKodu. Σημαντικό είναι να θυμηθούμε ότι όταν χρησιμοποιούμε έναν αισθητήρα, πρέπει να συμπληρώνουμε και κάποια πρόσθετα χαρακτηριστικά για τον αισθητήρα ώστε να περιγράψουμε πλήρως το γεγονός που θέλουμε να αντιληφθεί το αντικείμενο (π.χ. «όταν ακούς *το υποβρύχιο*» ή «όταν πέσεις πάνω *σε κόκκινο μήλο*»). Αυτά ονομάζονται ***προσδιοριστικά*** του αισθητήρα. Μη κατάλληλη χρήση προσδιοριστικών, κατά τον προγραμματισμό των αντικειμένων μας, μπορεί πολλές φορές να οδηγήσει σε μη αναμενόμενες συμπεριφορές!

Συνεπώς, όλα τα γεγονότα ορίζονται από τον προγραμματιστή με τον εξής τρόπο:

Πάμε να εντοπίσουμε τους ***αισθητήρες*** και τα ***προσδιοριστικά*** τους, στις συμπεριφορές του Super Mario που περιγράψαμε παραπάνω.

Στην πρώτη συμπεριφορά, ο αισθητήρας είναι ο «*πέφτω πάνω*», με τον οποίο δηλώνουμε πως θέλουμε ο Super Mario να αντιλαμβάνεται το γεγονός ότι έχει ακουμπήσει ένα αντικείμενο. Στη συμπεριφορά αυτή χρησιμοποιούμε το προσδιοριστικό *χελώνα* για να καθορίσουμε ποιο από τα αντικείμενα θα αντιλαμβάνεται ο αισθητήρας.

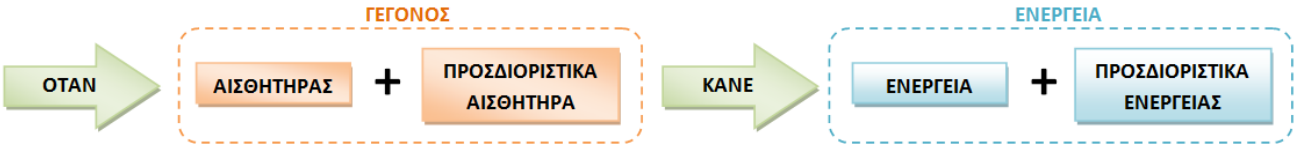
Στην επόμενη συμπεριφορά, ο αισθητήρας που χρησιμοποιείται είναι ο «*έχει*» με προσδιοριστικό το στοιχείο που *έχει-κουβαλάει* το αντικείμενο, δηλαδή το *αστέρι*.

Στην τελευταία συμπεριφορά ο αισθητήρας που χρησιμοποιείται είναι ο «*κλωτσάω*» με προσδιοριστικό το αντικείμενο *τούβλο*.

Και πώς μπορούμε να προσδιορίσουμε το πώς θα ενεργεί ένα αντικείμενο, όταν αντιλαμβάνεται ένα γεγονός;

Ο προγραμματιστής πρέπει να ορίσει πώς θα δράσει ένα αντικείμενο, όταν αντιληφθεί ένα γεγονός με τους αισθητήρες του. Στο MSKodu, τα αντικείμενα μπορούν να εκτελέσουν πολλές ενέργειες όπως π.χ. να περιστρέφονται, να πηδάνε, να τρώνε, να πυροβολούν ή ακόμη και να αλλάζουν χρώμα. Και οι ***ενέργειες*** συνοδεύονται από τα δικά τους ***προσδιοριστικά***. Για παράδειγμα, με την χρήση προσδιοριστικών ορίζουμε αν το αντικείμενό μας θα περιστρέφεται *δεξιά* ή *αριστερά*, αν θα πηδάει *ψηλά* ή *χαμηλά* αν θα βαφτεί *κόκκινο* ή *πράσινο*. Σημαντικό είναι να κατανοήσουμε πως ένα αντικείμενο μπορεί να κάνει ενέργειες που το αφορούν και όχι οποιεσδήποτε ενέργειες στον κόσμο (π.χ. ένα κανόνι μπορεί να κινείται αλλά δε μπορούμε να το προγραμματίσουμε να κινεί κάποιο άλλο αντικείμενο).

Στις συμπεριφορές του Super Mario που περιγράψαμε παραπάνω, οι ενέργειες που χρησιμοποιούνται είναι η «*ζαλίζω*» με προσδιοριστικό αυτής το *πολύ* (δηλαδή την ένταση που θα έχει η ενέργεια), η «*τρέχω*» με προσδιοριστικό το *γρήγορα* (δηλαδή την ταχύτητα με την οποία θα τρέχει ο Super Mario) και η «*σπάω*» με προσδιοριστικό το *τούβλο* (δηλαδή το αντικείμενο που θα σπάσει). Έτσι είναι καλύτερο να έχουμε στο μυαλό μας το παρακάτω ολοκληρωμένο σχήμα για το πώς μπορεί ο προγραμματιστής να ορίσει τις συμπεριφορές των αντικειμένων στο Kodu:



5.2 Οι αισθητήρες του Kodu: αλληλεπίδραση με το χρήστη

Τι νόημα θα είχαν τα παιχνίδια, χωρίς να χειρίζεται ο παίκτης τους ήρωες; Απολύτως καμία. Για αυτό και το MSKodu παρέχει αισθητήρες στα αντικείμενα μας ώστε να μπορούν να αντιλαμβάνονται το πάτημα των πλήκτρων του πληκτρολογίου και το κλικ του ποντικιού και να αντιδρούν σε αυτά. Στην ενότητα αυτή θα μελετήσουμε αναλυτικά τους αισθητήρες αυτούς και τα προσδιοριστικά τους.

5.2.1 Ο αισθητήρας: Πληκτρολόγιο (Keyboard)

Ο αισθητήρας ***Πληκτρολόγιο (Keyboard)*** επιτρέπει στα αντικείμενα να αντιδρούν στη χρήση του πληκτρολογίου από το χρήστη. Πιο συγκεκριμένα, αποτελεί έναν τρόπο επικοινωνίας ανάμεσα στο αντικείμενο και το χρήστη αφού τα αντικείμενα μας μπορούν να αντιδρούν όταν ο χρήστης πατήσει ένα πλήκτρο ή ένα συνδυασμό πλήκτρων του πληκτρολογίου. Θα μπορούσαμε λοιπόν να προγραμματίσουμε τον Kodu έτσι ώστε όταν ο χρήστης πατήσει το *πλήκτρο Α,* αυτός να πηδάει (π.χ. για να περάσει κάποιο εμπόδιο) ή/και όταν ο χρήστης πατήσει το *πλήκτρο Β*, αυτός να εκτοξεύει *Σφαιρίδια (Blips)* (π.χ. για να σκοτώσει κάποιον εχθρό). Ο αισθητήρας ***Πληκτρολόγιο (Keyboard)*** πάντα συνοδεύεται με προσδιοριστικό το πλήκτρο που σηματοδοτεί το γεγονός.

Ένα παράδειγμα εντολής με χρήση του αισθητήρα:

Τι είχε ο προγραμματιστής στο μυαλό του;

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βελάκι αριστερά] **ΤΟΤΕ** [στρίψε]

Είναι σημαντικό να τονίσουμε ότι το αντικείμενο το οποίο αντιδρά στο γεγονός «*πάτημα ενός πλήκτρου του πληκτρολογίου*» είναι αυτό στο οποίο προσθέσαμε την παραπάνω εντολή, δηλαδή ο Kodu. Τα άλλα αντικείμενα δεν θα αντιδράσουν, εκτός αν προγραμματίσουμε και αυτά να

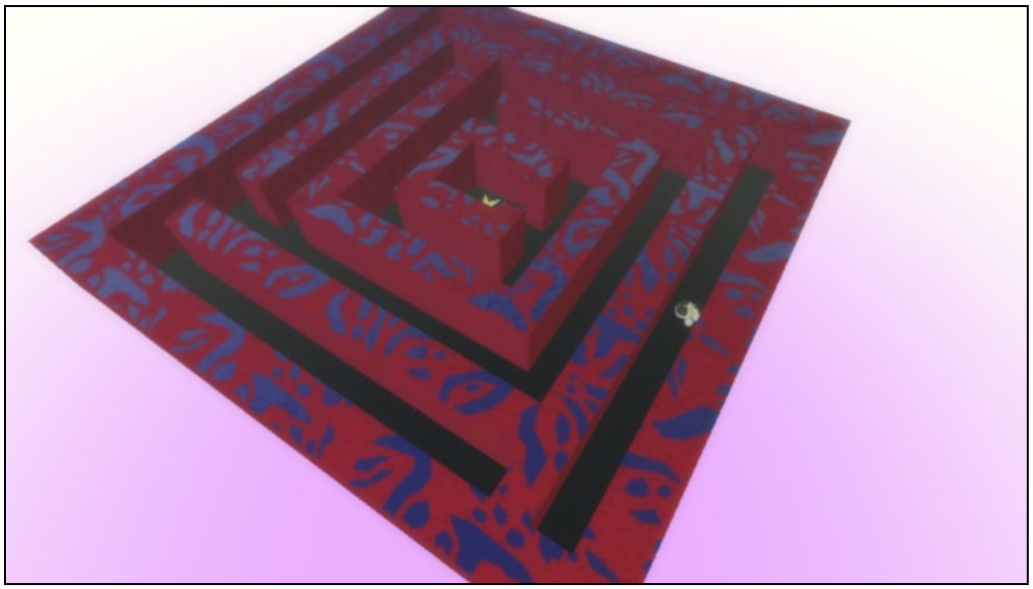
αντιλαμβάνονται το ίδιο γεγονός. Ας παρατηρήσουμε όμως λίγο καλύτερα την εντολή; Προς τα πού θα στρίψει ο Kodu; Μήπως ξεχάσαμε να προσθέσουμε κάτι; Στην ενέργεια ***Στρίβω (Turn)*** θα πρέπει να χρησιμοποιήσουμε και ένα προσδιοριστικό το οποίο θα δηλώνει την κατεύθυνση προς την οποία θέλουμε να στρίψει ο Kodu π.χ. ***Δεξιά (Right)*** ή ***Αριστερά (Left)***. Με βάση αυτό πρέπει να εμπλουτίσουμε την παραπάνω εντολή ως εξής:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το αριστερό βελάκι] **ΤΟΤΕ** [στρίψε][αριστερά]

Εκφράσεις που θα πρέπει να μας φέρνουν στο νου το συγκεκριμένο αισθητήρα:

{Πλήκτρα πληκτρολογίου, βελάκια πληκτρολογίου, χειρισμός αντικειμένου, μετακίνηση αντικειμένου, αλληλεπίδραση με το χρήστη}

Πάμε να δούμε την χρήση του αισθητήρα αυτού σε ένα παιχνίδι λαβύρινθο. Στο περιβάλλον του παιχνιδιού μας υπάρχει ο χαρακτήρας Kodu που είναι και αυτός που χειρίζεται ο χρήστης. Επιπλέον υπάρχει το αντικείμενο *Αστέρι (Star)*. Ο σκοπός του παιχνιδιού είναι ο χειριστής του χαρακτήρα Kodu να φτάσει στο τέλος του λαβυρίνθου εκεί όπου βρίσκεται το αστέρι. Ο κόσμος του παιχνιδιού φαίνεται στην ακόλουθη εικόνα:



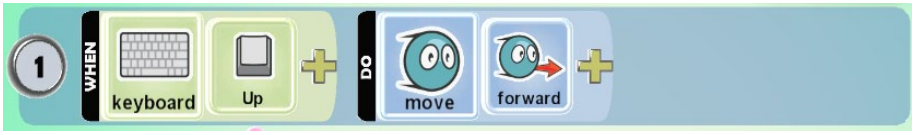
Δείτε το παράδειγμα *05_01.kodu*

Φορτώστε τον κόσμο **[05_01.kodu]** πατώντας διπλό κλικ στο αντίστοιχο αρχείο που βρίσκεται στο συνοδευτικό υλικό του βιβλίου. Αφού φορτώσετε το παιχνίδι πατήστε *Esc* για να εμφανιστεί η ***Παλέτα Εργαλείων (Tool Palette)*** και στην συνέχεια επιλέξτε το ***Εργαλείο Αντικειμένων (Object Tool)***. Κάντε δεξί κλικ πάνω στον Kodu και από τις επιλογές που θα εμφανιστούν πατήστε το ***Προγραμμάτισε (Program)*** έτσι ώστε να ξεκινήσουμε να προγραμματίζουμε τη συμπεριφορά του και να δημιουργήσουμε το παιχνίδι.

Θέλουμε ο Kodu να αξιοποιήσει τον αισθητήρα ***Πληκτρολόγιο (Keyboard)*** για να καταλάβει πότε ο χρήστης έχει πατήσει τα βέλη ***Πάνω (Up)***, ***Αριστερά (Left)*** και ***Δεξιά (Right)*** του πληκτρολογίου. Με αυτόν τον τρόπο ο χρήστης θα μπορεί να χειρίζεται τον Kodu ώστε να τον κινεί/κατευθύνει μέσα στον κόσμο-λαβύρινθο.

Ας δούμε βήμα-βήμα πώς θα το κάνουμε αυτό. Αρχικά θέλουμε να δώσουμε τη δυνατότητα στον Kodu να κινείται προς τα εμπρός όταν ο χρήστης πατάει το βέλος ***Πάνω (Up)***. Η ενέργεια η οποία επιτρέπει στα αντικείμενα να κινούνται είναι η ***Κινήσου (Move)***. Η ενέργεια δέχεται ως προσδιοριστικά την *κατεύθυνση (Direction)* της κίνησης και την *ταχύτητα* (γρήγορα ή αργά). Στην περίπτωση του λαβυρίνθου θα χρησιμοποιήσουμε την ενέργεια αυτή μαζί με ένα από τα προσδιοριστικά της, το ***Μπροστά (Forward)***. Συνεπώς μία συμπεριφορά που θα εμφανίζει ο Kodu είναι:

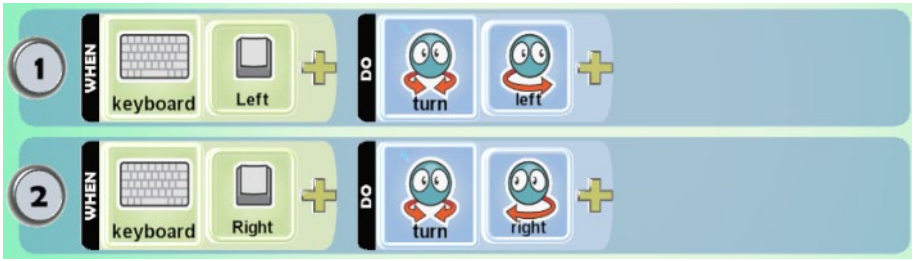
ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος up] **ΤΟΤΕ** [κινήσου][προς τα μπροστά]



Αρκεί αυτή η συμπεριφορά του Kodu, ώστε ο χρήστης να μπορεί να το μετακινήσει μέχρι το αστέρι; Όχι! Ο χρήστης θα μπορεί να κινεί τον Kodu μόνο προς τα εμπρός χωρίς να έχει την δυνατότητα να αλλάξει τον προσανατολισμό του και, συνεπώς, να μπορέσει να τον κατευθύνει μέσα στα δρομάκια του λαβυρίνθου. Για τον λόγο αυτό θα προγραμματίσουμε τον Kodu να *στρίβει* όταν ο χρήστης πατάει τα βέλη του πληκτρολογίου *Αριστερά (Left)* και *Δεξιά (Right)*. Άρα, χρειαζόμαστε πάλι τον αισθητήρα ***Πληκτρολόγιο (Keyboard)***. Όπως είδαμε και παραπάνω, η ενέργεια που επιτρέπει στα αντικείμενα να στρίβουν είναι η ***Στρίψε (Turn)***. Θυμηθείτε τα προσδιοριστικά αυτής ***Δεξιά (Right)*** και ***Αριστερά (Left)***. Σε αυτήν την περίπτωση θα μας φανούν πολύ χρήσιμα αφού θέλουμε να δώσουμε στον Kodu τη δυνατότητα να στρίβει *αριστερά* όταν ο χρήστης πατάει το βέλος *Αριστερά (Left)* και δεξιά όταν πατάει το βέλος *Δεξιά (Right)*. Συνεπώς θα προσθέσουμε στον Kodu δύο ακόμη συμπεριφορές:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Left] **ΤΟΤΕ** [στρίψε][αριστερά]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Right] **ΤΟΤΕ** [στρίψε][δεξιά]



Προσθέστε στον Kodu τις παραπάνω εντολές, βγείτε από το *πάνελ προγραμματισμού* και πατήστε ***Παίξε (Play)*** για να παίξετε το παιχνίδι.

Το MSKodu μας παρέχει την δυνατότητα να χρησιμοποιήσουμε οποιοδήποτε πλήκτρο του πληκτρολογίου ως προσδιοριστικό του αισθητήρα *Πληκτρολόγιο (Keyboard)*. Έτσι μπορούμε να προγραμματίσουμε τους χαρακτήρες μας να αντιδρούν στο πάτημα των πλήκτρων με τα *Γράμματα (Letters)*, με τους *Αριθμούς (Numbers)*, με τα *Σύμβολα (Symbols)* αλλά και τα *Διάφορα (Misc)* υπόλοιπα πλήκτρα του *Πληκτρολογίου (Keyboard)*. Υπάρχει όμως ένα ακόμη ενδιαφέρον προσδιοριστικό, το ***Όχι (Not)*** το οποίο μπορεί να προστεθεί επιπρόσθετα του γράμματος και το οποίο αντιστρέφει το νόημα της συμπεριφοράς μας. Δηλαδή αν είχαμε την έκφραση:

ΌΤΑΝ[στο πληκτρολόγιο][**όχι**][πατηθεί το βέλος Right] **ΤΟΤΕ** [στρίψε][δεξιά],

τότε ο Kodu θα έστριβε διαρκώς όσο δεν πατούσαμε το δεξί βέλος! Δοκιμάστε το!

5.2.2 Ο αισθητήρας Ποντίκι (Mouse)

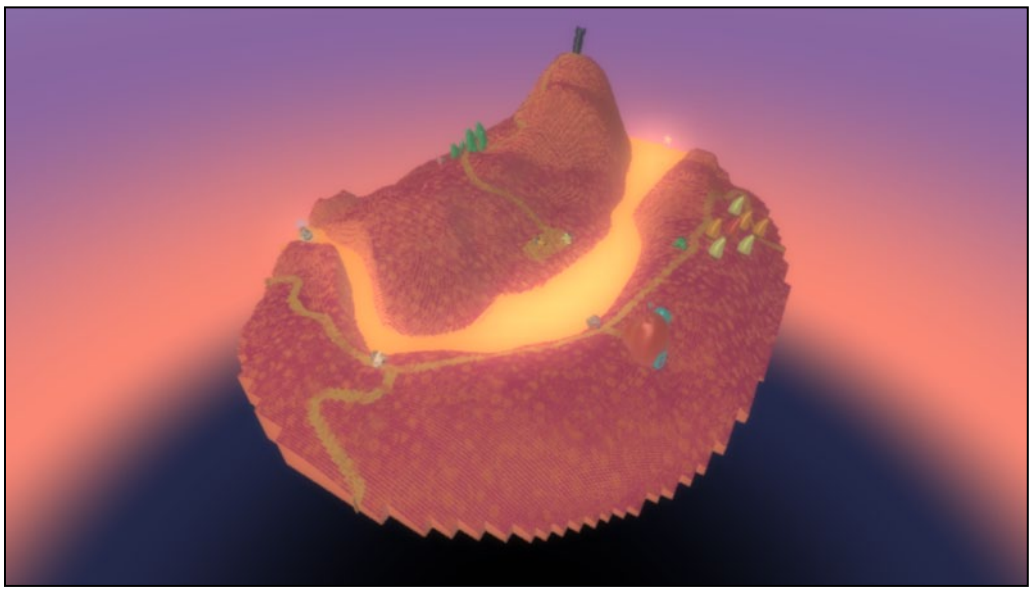
Ο αισθητήρας ***Ποντίκι (Mouse)*** επιτρέπει στα αντικείμενα να αντιδρούν στη χρήση του ποντικιού από τον χρήστη. Όπως ο αισθητήρας ***Πληκτρολόγιο (Keyboard)***, και ο αισθητήρας ***Ποντίκι (Mouse)*** αποτελεί έναν τρόπο επικοινωνίας ανάμεσα στο αντικείμενο και το χρήστη, αφού έτσι τα αντικείμενα μας μπορούν να αναλάβουν δράση όταν ο χρήστης πατήσει το αριστερό ή το δεξί κλικ του ποντικιού, ή όταν το περάσει πάνω από ένα αντικείμενο. Θα μπορούσαμε λοιπόν να προγραμματίσουμε τον Kodu έτσι ώστε όταν ο χρήστης πατήσει το αριστερό κλικ αυτός να αλλάξει χρώμα (π.χ. για να κάνει σινιάλο στο συνεργάτη του) ή να αλλάξει διάθεση (ναι, στο MSKodu οι χαρακτήρες μπορούν να εκφράζουν τα συναισθήματά τους αλλάζοντας τη φατσούλα τους!).

Εκφράσεις που θα πρέπει να μας φέρνουν στο νου το συγκεκριμένο αισθητήρα:

{Ποντίκι, έλεγχος από το ποντίκι, δεξί κλικ, αριστερό κλικ, δείχνω στην οθόνη, το ποντίκι πάνω από ένα αντικείμενο, χειρισμός αντικειμένου από το ποντίκι}

Πάμε να δούμε την χρήση του αισθητήρα αυτού σε ένα παιχνίδι. Στο περιβάλλον του παιχνιδιού μας, υπάρχει το αντικείμενο ***Πλοίο (Ship)*** το οποίο είναι και το αντικείμενο που χειρίζεται ο χρήστης (δείτε την παρακάτω εικόνα). Επιπλέον, στο περιβάλλον υπάρχουν μερικά ***Κανόνια (Cannon)*** τα οποία είναι προγραμματισμένα να κινούνται γύρω από το κίτρινο ποτάμι και να πυροβολούν όποιο αντικείμενο προσπαθεί να το διασχίσει. Τα υπόλοιπα αντικείμενα που

υπάρχουν είναι διακοσμητικά. Ο σκοπός του παιχνιδιού είναι ο χειριστής του αντικείμενου ***Πλοίο (Ship)*** να το οδηγήσει στο ***Αστέρι (Star)*** που βρίσκεται στην άλλη άκρη του ποταμού αντικρούοντας την επίθεση των ***Κανονιών (Cannon)***. Ο κόσμος του παιχνιδιού φαίνεται στην ακόλουθη εικόνα:



Δείτε το παράδειγμα *05_02.kodu*

Φορτώστε τον κόσμο **[05_02.kodu]** πατώντας διπλό κλικ στο αντίστοιχο αρχείο που βρίσκεται στο συνοδευτικό υλικό του βιβλίου μας. Αφού φορτώσετε το παιχνίδι πατήστε *Esc* για να εμφανιστεί η Παλέτα *Εργαλείων (Tool Palette)* και στη συνέχεια επιλέξτε το *Εργαλείο Αντικειμένων (Object Tool)*. Κάντε δεξί κλικ πάνω στον Kodu και από τις επιλογές που θα εμφανιστούν πατήστε το *Προγραμμάτισε (Program)* έτσι ώστε να ξεκινήσουμε να προγραμματίζουμε την συμπεριφορά του και να δημιουργήσουμε το παιχνίδι.

Θέλουμε ο χρήστης να μπορεί να χειρίζεται το *Πλοίο (Ship)* ώστε πρώτον να το κινεί/κατευθύνει μέσα στο ποτάμι και δεύτερον να μπορεί να πυροβολεί τα *Κανόνια (Cannon)* για να αντικρούει την επίθεση τους. Για το δεύτερο, θέλουμε ο Kodu να αξιοποιεί τον αισθητήρα ***Ποντίκι (Mouse)*** για να καταλάβει πότε ο χρήστης πατάει το ***Αριστερό Κλικ (Left)*** και τότε να πυροβολεί. Ας δούμε βήμα-βήμα πως θα κάνουμε τα παραπάνω.

Αρχικά για να μπορεί ο χρήστης να κατευθύνει τον Kodu, θα χρησιμοποιήσουμε τον αισθητήρα ***Πληκτρολόγιο (Keyboard)*** σε συνδυασμό με τις ενέργειες ***Κινούμαι (Move)*** και ***Στρίβω (Turn)***. Συγκεκριμένα, η συμπεριφορά που εμφανίζει ο Kodu θα είναι:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Up] **ΤΟΤΕ** [κινήσου][προς τα εμπρός]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Left] **ΤΟΤΕ** [στρίψε][αριστερά]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Right] **ΤΟΤΕ** [στρίψε][δεξιά]

Κατά αντιστοιχία με το παιχνίδι του λαβυρίνθου που είδαμε στην προηγούμενη υποενότητα, προσθέτουμε στον Kodu τις εξής εντολές:

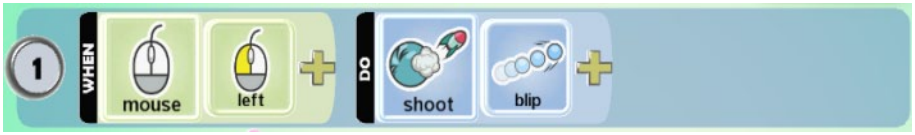
Επιπλέον, θέλουμε ο χρήστης να έχει την δυνατότητα να χειρίζεται το *Πλοίο (Ship)* με σκοπό να αντικρούει την επίθεση των *Κανονιών (Cannon)*. Για τον λόγο αυτό θα προγραμματίσουμε το *Πλοίο (Ship)* να πυροβολεί, όταν ο χρήστης πατάει το *Αριστερό Κλικ (Left)* του ποντικιού. Άρα θα χρησιμοποιήσουμε τον αισθητήρα ***Ποντίκι (Mouse)*** με προσδιοριστικό το ***Αριστερό Κλικ (Left)***. Η

ενέργεια που επιτρέπει στα αντικείμενα να πυροβολούν είναι η ***Πυροβολώ (Shoot)***. Η ενέργεια μπορεί να συνοδεύεται με τα προσδιοριστικά ***Σφαιρίδιο (Blip)*** και ***Πύραυλος (Missile)***. Με την χρήση τους μπορούμε να καθορίσουμε αντίστοιχα αν το αντικείμενο μας θα πυροβολεί με σφαιρίδια ή με πυραύλους.

Αν προγραμματίσουμε ένα αντικείμενο να πυροβολεί *Σφαιρίδια (Blip)*, τότε έχει την δυνατότητα να πυροβολεί συνεχόμενα με γρήγορο ρυθμό. Οι *Πύραυλοι (Missile)* είναι αρκετά πιο αργοί, αλλά έχουν μεγαλύτερη καταστροφική ικανότητα αφού, σε αντίθεση με τα *Σφαιρίδια (Blip)*, αρκεί ένας από αυτούς για να καταστρέψει το αντικείμενο που πετυχαίνει. Αυτή είναι και η βασική τους διαφορά. Το Kodu μας προσφέρει μία πληθώρα προσδιοριστικών που μπορούμε να χρησιμοποιήσουμε σε συνδυασμό με την ενέργεια *Πυροβολώ (Shoot)* όπως η κατεύθυνση της βολής και το είδος της ζημιάς που θα προκαλέσει σε ένα άλλο αντικείμενο όταν το χτυπήσει. Θα μάθουμε περισσότερα από αυτά στην επόμενη ενότητα.

Σε αυτό το παράδειγμα, θα χρησιμοποιήσουμε την ενέργεια ***Πυροβολώ (Shoot)*** με προσδιοριστικό το ***Σφαιρίδιο (Blip)***. Συνεπώς μία ακόμη συμπεριφορά που θα εμφανίζει ο Kodu είναι:

ΌΤΑΝ[στο ποντίκι][πατηθεί το αριστερό κλικ] **ΤΟΤΕ** [πυροβόλησε][με σφαιρίδιο]



Το ιδιαίτερο χαρακτηριστικό αυτού του συνδυασμού, δηλαδή του αισθητήρα ***Ποντίκι (Mouse)*** με την ενέργεια ***Πυροβολώ (Shoot)***, είναι ότι το αντικείμενο μας πυροβολεί προς το σημείο όπου κάνουμε κλικ το ποντίκι μας! Αυτό δεν θα συνέβαινε αν είχαμε χρησιμοποιήσει τον αισθητήρα ***Πληκτρολόγιο (Keyboard)*** αφού τότε το αντικείμενο μας θα πυροβολούσε προς τα μπροστά. Σε τέτοια περίπτωση, θα έπρεπε ο χρήστης να εκμεταλλευτεί τη δυνατότητα του αντικειμένου μας να *στρίβει* για να μπορέσει να πετύχει το στόχο του.

Για να κάνουμε το παιχνίδι μας πιο ενδιαφέρον μπορούμε επιπρόσθετα να προγραμματίσουμε το *Πλοίο (Ship)* να ***Πυροβολεί (Shoot) Πύραυλους (Missile)*** όταν ο χρήστης πατάει το δεξί κλικ του ποντικιού. Άρα, χρειαζόμαστε πάλι τον αισθητήρα ***Ποντίκι (Mouse)***. Πιο συγκεκριμένα, η συμπεριφορά που θέλουμε να εμφανίζει το *Πλοίο (Ship)* είναι:

ΌΤΑΝ[στο ποντίκι][πατηθεί το δεξί κλικ] **ΤΟΤΕ** [πυροβόλησε] [πυραύλους]

Προσθέστε στον Kodu τις παραπάνω εντολές, βγείτε από το *πάνελ προγραμματισμού* και πατήστε *Παίξε (Play)* για να παίξετε το παιχνίδι.

Στον αισθητήρα *Ποντίκι (Mouse)*, υπάρχει ένα ακόμη προσδιοριστικό το ***Πάνω Από (Over)***. Μπορούμε να το χρησιμοποιήσουμε αν επιθυμούμε ο χαρακτήρας μας να ενεργεί όταν ο χρήστης μετακινήσει το ποντίκι πάνω από το ίδιο ή κάποιο άλλο αντικείμενο. Συχνά χρησιμοποιείται μαζί με το προσδιοριστικό ***Όχι (Not)*** το οποίο αντιστρέφει το νόημα της συμπεριφοράς μας, ακριβώς όπως και κατά την χρήση του με τον αισθητήρα ***Πληκτρολόγιο (Keyboard)***. Δηλαδή αν στον Kodu είχαμε προσθέσει την συμπεριφορά:

ΌΤΑΝ[το ποντίκι][όχι][είναι πάνω από][τον Kodu] **ΤΟΤΕ** [πυροβόλησε] [με σφαιρίδιο],

τότε αυτός θα πυροβολούσε διαρκώς και θα σταματούσε μόνο όταν πηγαίναμε το ποντίκι μας πάνω του!

Σημείωση: Για τον αισθητήρα *Ποντίκι (Mouse)* υπάρχει και το προσδιοριστικό *Κινώ (Move)*, το οποίο θα αναμέναμε να υλοποιεί το γεγονός «όταν το ποντίκι κινηθεί». Όμως με την χρήση αυτού του προσδιοριστικού, ο χαρακτήρας εμφανίζει απρόβλεπτη συμπεριφορά. Χμ, οι προγραμματιστές της Microsoft έχουν κάποια δουλειά ακόμη….!

5.2.3. Ο αισθητήρας Χειριστήριο (Gamepad)

Ο αισθητήρας ***Χειριστήριο (Gamepad)*** χρησιμοποιείται κατά τον σχεδιασμό παιχνιδιών τα οποία πρόκειται να παιχτούν μέσω της κονσόλας του X-BOX ή εφόσον έχετε συνδεδεμένο στον υπολογιστή σας ένα χειριστήριο παιχνιδιών. Ναι, με το Kodu έχετε την δυνατότητα να δημιουργείται παιχνίδια για την κονσόλα παιχνιδιών X-BOX! Σε αυτήν την περίπτωση, ο τρόπος επικοινωνίας ανάμεσα στο αντικείμενο και τον χρήστη είναι ο αισθητήρας ***Χειριστήριο (Gamepad)***, αφού τα αντικείμενα μας μπορούν να αναλάβουν δράση όταν ο χρήστης πατήσει ένα πλήκτρο ή ένα συνδυασμό πλήκτρων του χειριστηρίου. Ο αισθητήρας ***Χειριστήριο (Gamepad)***, κατά αντιστοιχία με τον αισθητήρα ***Πληκτρολόγιο (Keyboard)***, πάντα συνοδεύεται με προσδιοριστικό το πλήκτρο που θέλουμε να αντιλαμβάνεται το αντικείμενό μας. Στην επόμενη εικόνα παρουσιάζετε το απλό χειριστήριο της κονσόλας X-BOX, όπου μπορείτε να δείτε τα διαθέσιμα πλήκτρα του (π.χ. τα πλήκτρα *X*, *Y*, *A* και *B*, την *αριστερή σκανδάλη* και *τη δεξιά σκανδάλη (Right Trig)*):

Ένα παράδειγμα εντολής με χρήση του αισθητήρα αυτού:

Τι είχε ο προγραμματιστής στο μυαλό του;

ΌΤΑΝ[στο gamepad][πατηθεί το πλήκτρο Left Trig] **ΤΟΤΕ** [πυροβόλησε]

Παρατηρείστε πως το σκεπτικό της εντολής είναι ανάλογο με αυτό της χρήσης του αισθητήρα *Πληκτρολόγιο (Keyboard)*. Θυμηθείτε πως αν θέλαμε να δώσουμε την αντίστοιχη συμπεριφορά στο αντικείμενό μας χρησιμοποιώντας τον αισθητήρα *Πληκτρολόγιο (Keyboard)* με προσδιοριστικό ένα πλήκτρο (π.χ. το Space), το σκεπτικό μας θα ήταν το εξής:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο Space] **ΤΟΤΕ** [πυροβόλησε]

Και η αντίστοιχη εντολή:

Γενικά, αν έχετε κατανοήσει την χρήση του αισθητήρα ***Πληκτρολόγιο (Keyboard)***, μπορείτε πολύ εύκολα, με λίγη εξάσκηση, να κατανοήσετε και την χρήση του αισθητήρα ***Χειριστήριο (Gamepad)***.

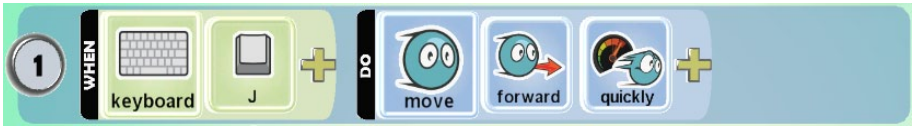
5.3 Η συμπεριφορά του Kodu: ενέργειες

Ας εξετάσουμε λίγο πιο προσεκτικά τις ενέργειες κίνησης και πυροβολισμών που έχουμε στη διάθεσή μας ως προγραμματιστές!

5.3.1 Ενέργεια Κινούμαι (Move)

Όπως αναφέραμε και παραπάνω, η ενέργεια ***Κινούμαι (Move)*** δίνει την δυνατότητα στα αντικείμενα να κινούνται μέσα στο κόσμο. Στην πλειοψηφία των παιχνιδιών, ο ήρωας έχει αυτή την δυνατότητα. Μπορούμε να καθορίσουμε τον τρόπο και την ταχύτητα με τον οποίο θα κινείται ένα αντικείμενο καθώς και την κατεύθυνση προς τη οποία θα κινείται, χρησιμοποιώντας τα διάφορα προσδιοριστικά που μας παρέχει το MSKodu. Θα μπορούσαμε επίσης να προγραμματίσουμε το χαρακτήρα μας να κινείται *προς* ένα αντικείμενο (π.χ. για να μπορέσει στην

συνέχεια να το συλλέξει) ή να *αποφεύγει* ένα αντικείμενο (π.χ. για να μην σκοτωθεί αν αυτό είναι ένα αντικείμενο *Κανόνι (Cannon)* που τον πυροβολεί). Ένα παράδειγμα εντολής με χρήση αυτής της ενέργειας είναι:



Τι σκεφτόταν ο προγραμματιστής;

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο J] **ΤΟΤΕ** [κινήσου προς τα μπροστά][γρήγορα]

Παρατηρείστε πως σε αυτή την εντολή χρησιμοποιήσαμε δύο από τα προσδιοριστικά της ενέργειας ***Κινούμαι (Move)***. Πολλές φορές θα χρειαστεί να χρησιμοποιήσετε παραπάνω από ένα προσδιοριστικά μιας ενέργειας έτσι ώστε να καθορίσετε πλήρως την συμπεριφορά που θέλετε να έχει το αντικείμενο που προγραμματίζετε. Για παράδειγμα θα μπορούσατε να εμπλουτίσετε την παραπάνω εντολή προσθέτοντας άλλη μία φορά το προσδιοριστικό ***Γρήγορα (Quickly)*** αν θέλατε να προγραμματίσετε το αντικείμενό σας να κινείται *ακόμη πιο γρήγορα*.

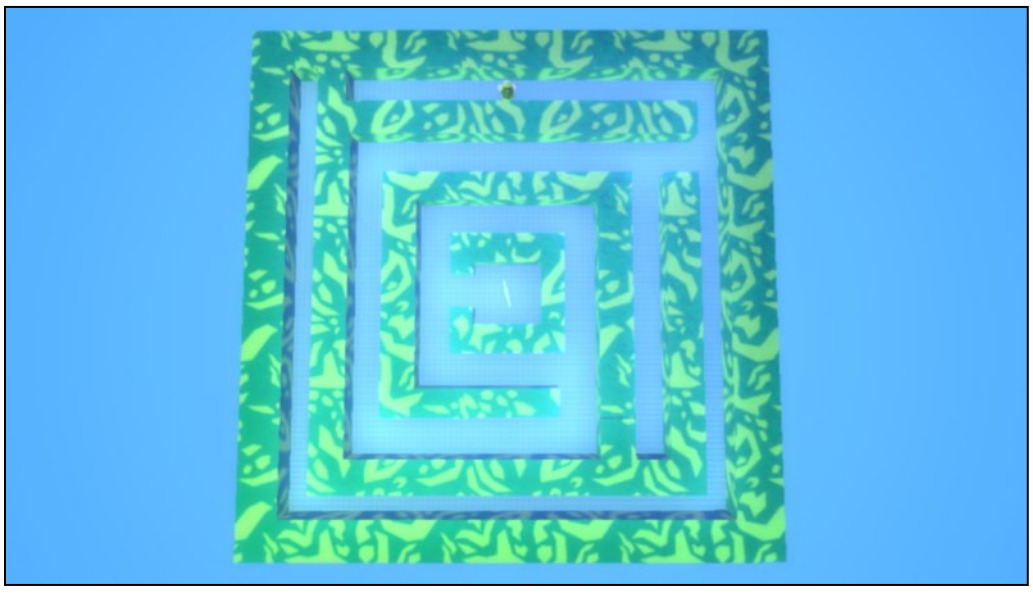
Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Κινούμαι, ακολουθώ, κατευθύνω, τρέχω, κυνηγώ, περπατάω, προσπερνάω, οδηγώ, περιπλανιέμαι, διασχίζω, αποφεύγω, απομακρύνομαι}

Ας δούμε τώρα πως μπορούμε να χρησιμοποιήσουμε την ενέργεια ***Κινούμαι (Move)*** και τα προσδιοριστικά της ***Κατεύθυνσης (Direction)*** με βάση τα σημεία του ορίζοντα, ώστε να προγραμματίσουμε την κίνηση των αντικειμένων μας με έναν εναλλακτικό τρόπο. Και γιατί χρειάζεται να το μάθουμε αυτό; Σε πολλά παιχνίδια, όπως στο παιχνίδι του λαβυρίνθου που είδαμε παραπάνω, ο χρήστης βλέπει τον κόσμο από την θέση του αντικειμένου το οποίο χειρίζεται. Υπάρχουν όμως και άλλα παιχνίδια, όπως το Pacman όπου ο χρήστης βλέπει το παιχνίδι από μία σταθερή θέση από όπου έχει πλήρη επαφή με όλο τον κόσμο του παιχνιδιού. Σε τέτοιες περιπτώσεις είναι προτιμότερο να χρησιμοποιούμε τα σημεία του ορίζοντα για να προγραμματίσουμε την *Κίνηση* του αντικειμένου.

Δείτε το παράδειγμα *05_03.kodu*

Πάμε να δούμε πιο πρακτικά τη συγκεκριμένη περίσταση και να πειστούμε για την χρησιμότητα αυτής της προσέγγισης. Φορτώστε το παιχνίδι **[05_03.kodu]** το οποίο είναι το γνωστό μας παιχνίδι του λαβυρίνθου. Αυτή τη φορά, όμως, βλέπουμε το λαβύρινθο από μια σταθερή θέση όπου εμφανίζεται όλος ο κόσμος, κι όχι από την θέση του Kodu! Στόχος του παιχνιδιού είναι και πάλι ο Kodu να φτάσει στο *Αστέρι (Star)*. Ο κόσμος του παιχνιδιού φαίνεται στην ακόλουθη εικόνα:



Αρχικά προσθέστε στον Kodu την επόμενη συμπεριφορά:

Ποια συμπεριφορά υποδεικνύει αυτή η εντολή;

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί κάποιο από τα βέλη] **ΤΟΤΕ** [κινήσου προς αντίστοιχη κατεύθυνση]

Παρατηρείστε πως αυτή η εντολή αποτελεί μία «συντόμευση» των εντολών που χρησιμοποιήσαμε προηγουμένως για να αντιδρά το αντικείμενό μας σε κάθε βέλος του πληκτρολογίου ξεχωριστά. Αυτή η εντολή χρησιμοποιείται πολύ συχνά κατά τον προγραμματισμό των αντικειμένων στο MSKodu.

Επίσης αρκετά συχνά στην θέση του προσδιοριστικού *Βέλη (Arrows)*, χρησιμοποιούμε το προσδιοριστικό *WASD που αντιστοιχεί στα πλήκτρα W,A,S,D.* Παρατηρείστε πως τα πλήκτρα W, A, S και D, σχηματίζουν διάταξη ίδια με αυτή των *Βελών (Arrows)* όπου το W αντιστοιχεί στο *Πάνω (Up),* το S στο *Κάτω (Down)*, το A στο *Αριστερά (Left)* και το D στο *Δεξιά (Right)*. Μπορείτε να χρησιμοποιήσετε αυτήν την παραλλαγή της παραπάνω εντολής, αν για παράδειγμα θέλετε ο χρήστης να μπορεί να κινεί και έναν δεύτερο χαρακτήρα στο παιχνίδι, χρησιμοποιώντας όμως διαφορετικά πλήκτρα για τον χειρισμό του από τον πρώτο. Επίσης χρησιμοποιούμε το συνδυασμό *WASD* όταν θέλουμε ο χρήστης με το δεξί χέρι να εκτελεί ενέργειες με το ποντίκι.

Αποθηκεύστε τις αλλαγές και παίξτε το παιχνίδι.

- Τι συμβαίνει όταν πατάτε το *δεξί βέλος;*

Αυτό που αναμένουμε, σύμφωνα με την συμπεριφορά που προσθέσαμε στον Kodu, είναι να στρίβει *δεξιά* δηλαδή προς την αντίστοιχη κατεύθυνση που υποδηλώνει το δεξί *βέλος*. Όμως ο kodu κινείται προς την *αριστερή* κατεύθυνση (σε σχέση με το πώς κοιτάτε την οθόνη).

- Γιατί συμβαίνει αυτό;

Ο Kodu στρίβει προς την κατεύθυνση που είναι δεξιά σε σχέση με την θέση που είναι ο ίδιος. Όμως σε σχέση με τη δική μας θέση που είναι απέναντι από την οθόνη, φαίνεται να στρίβει αριστερά.

Σε τέτοιες περιπτώσεις μπορούμε να προγραμματίσουμε τα αντικείμενά μας να κινούνται σε σχέση με τα σημεία του ορίζοντα δηλαδή ***Βόρεια (North)***, ***Νότια (South)***, ***Δυτικά (West)*** και ***Ανατολικά (East)***. Για το σκοπό αυτό χρησιμοποιούμε τα αντίστοιχα προσδιοριστικά της ενέργειας ***Κινούμαι (Move)*** που βρίσκονται στην επιλογή ***Κατεύθυνση (Direction)***. Συνεπώς οι συμπεριφορές που θέλουμε να προσθέσουμε στον Kodu είναι:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος up] **ΤΟΤΕ** [κινήσου][προς τα βόρεια]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος down] **ΤΟΤΕ** [κινήσου][προς τα νότια]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος left] **ΤΟΤΕ** [κινήσου][προς τα δυτικά]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος right] **ΤΟΤΕ** [κινήσου][προς τα ανατολικά]

Διαγράψτε από τον Kodu την εντολή που προσθέσαμε προηγουμένως κάνοντας δεξί κλικ πάνω στον αριθμό της εντολής και επιλέγοντας ***Διαγραφή Γραμμής (Cut Row)***. Έπειτα προσθέστε τις παραπάνω αλλαγές και παίξτε το παιχνίδι. Παρατηρείστε πως τώρα είναι αρκετά πιο εύκολο να κατευθύνετε τον Kodu. Ο συνδυασμός των πλήκτρων και των προσδιοριστικών που αφορούν την κατεύθυνση που παρουσιάσαμε παραπάνω, δεν είναι ο μοναδικός. Ανάλογα με τις ανάγκες του

παιχνιδιού που σχεδιάζετε μπορεί να είναι πιο βολικό να χρησιμοποιήσετε κάποιο άλλο συνδυασμό πλήκτρων-κατεύθυνσης.

Στα παιχνίδια στα οποία χρησιμοποιείτε αυτή την μέθοδο κίνησης, μπορείτε να συμβουλεύεστε την πυξίδα η οποία βρίσκεται στο δεξί κάτω μέρος της οθόνης, ώστε να καθοδηγείτε πιο εύκολα τον χαρακτήρα σας

Το Kodu μας παρέχει κι άλλα προσδιοριστικά που μπορούμε να χρησιμοποιήσουμε σε συνδυασμό με την ενέργεια ***Κινούμαι (Move)***. Μπορούμε να ορίσουμε την κίνηση του αντικείμενου μας να είναι τυχαία μέσα στον κόσμο, χρησιμοποιώντας το προσδιοριστικό ***Περιπλάνηση (Wander)*** (π.χ. αν θέλουμε να προσθέσουμε ένα διακοσμητικό αντικείμενο το οποίο να μην έχει συγκεκριμένη διαδρομή ή στόχο κίνησης). Επιπλέον, υπάρχουν και κάποια πολύ χρήσιμα προσδιοριστικά όπως το ***Κατά Πάνω (Toward), Αποφεύγω (Avoid)*** και ***Απομακρύνομαι (Away)***. Αυτά εμφανίζονται όταν χρησιμοποιήσουμε την ενέργεια ***Κινούμαι (Move)*** σε συνδυασμό με αισθητήρες που σχετίζονται με άλλα αντικείμενα, όπως οι αισθητήρες ***Βλέπω (See), Ακούω (Hear), Πέφτω Πάνω (Bump)***. Θα εξετάσουμε αναλυτικά αυτούς τους αισθητήρες στο επόμενο Κεφάλαιο.

Πριν κλείσουμε αυτήν την υποενότητα καλό είναι να επισημάνουμε πως δεν έχουν *όλα* τα αντικείμενα τη δυνατότητα να εκτελούν *όλες* τις ενέργειες που υπάρχουν στο MSKodu!

Ας πάρουμε για παράδειγμα το αντικείμενο ***Βράχος (Rock).*** Αυτό το αντικείμενο δεν έχετε την δυνατότητα να το προγραμματίσετε να *κινείται.* Αν προσπαθήσετε να το προγραμματίσετε θα ανακαλύψετε ότι η ενέργεια ***Κινούμαι (Move)*** δεν εμφανίζεται στις πίτες ενεργειών. Ο λόγος είναι ότι το αντικείμενο αυτό δεν έχει δημιουργηθεί για να χρησιμοποιείται ως χαρακτήρας που κινείται στα παιχνίδια, κάτι που είναι και διαισθητικά κατανοητό. Υπάρχουν και άλλα αντικείμενα που δεν έχουν την δυνατότητα να κινούνται από μόνα τους:

Η Μπάλα (Ball) 

, το Κέρμα (Coin) 

, το Κάστρο (Castle) 

, το Αστέρι (Star) 

, η Καρδιά (Heart) 

, το Εργοστάσιο (Factory) 

,η Καλύβα (Hut) 

, τα Πυρομαχικά (Ammo) 

και όλα τα ήδη Δέντρων (Tree) 

.

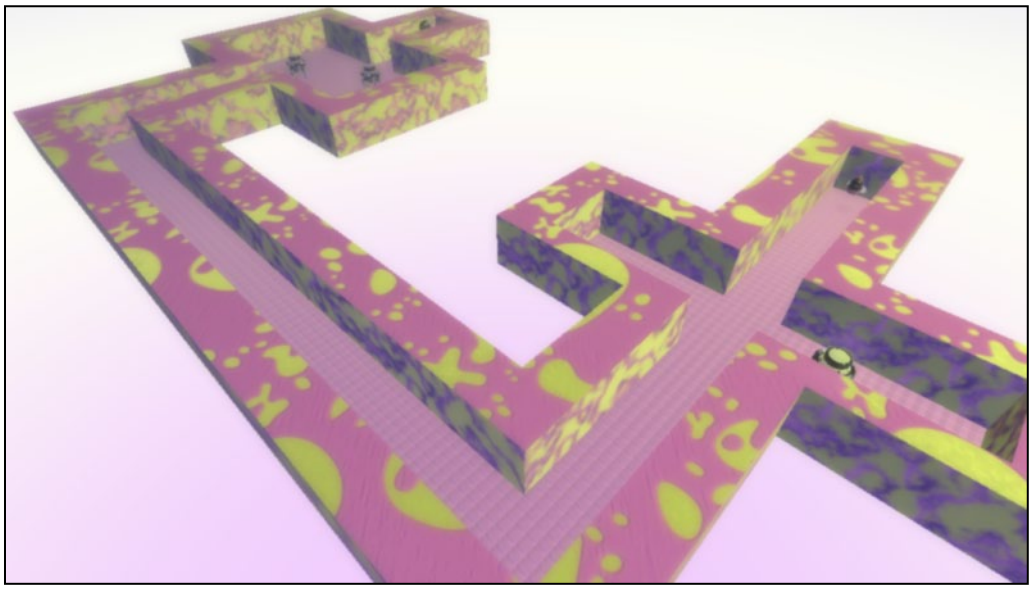
Επίσης, δεν έχουμε τη δυνατότητα να προγραμματίσουμε όλα τα παραπάνω αντικείμενα για να εκτελούν καμία άλλη ενέργεια που σχετίζεται με την κίνηση όπως οι ***Στρίβω (Turn)*** και ***Πηδάω (Jump)***.

5.3.2 Δημιουργώντας Μονοπάτια (Path) συγκεκριμένης κίνησης

Το MSKodu μας παρέχει τη δυνατότητα να δημιουργούμε συγκεκριμένα ***Μονοπάτια (Path)*** κίνησης μέσα στον κόσμο μας. Τα μονοπάτια είναι διαδρομές που σχεδιάζουμε μέσα στον κόσμο, πάνω στις οποίες μπορούμε να προγραμματίσουμε τα αντικείμενα μας να κινούνται αυτόματα! Κάτι τέτοιο είναι χρήσιμο σε πολλά παιχνίδια, όπως για παράδειγμα σε ένα παιχνίδι αγώνα ταχύτητας όπου θέλουμε να προγραμματίσουμε τους αντιπάλους του βασικού παίχτη να κινούνται αυτόματα πάνω στο δρόμο της πίστας. Ή ακόμη σε περιπτώσεις που θέλουμε οι αντίπαλοι να κινούνται σε συγκεκριμένες σύνθετες διαδρομές ώστε να δυσκολεύουν τη ζωή του χρήστη όταν προχωράει σε στενά περάσματα. Για να προγραμματίσουμε ένα αντικείμενο να κινείται πάνω σε ένα συγκεκριμένο ***Μονοπάτι (Path)***, πρέπει να χρησιμοποιήσουμε την ενέργεια ***Κινούμαι (Move)*** σε συνδυασμό με το προσδιοριστικό ***Πάνω στο μονοπάτι (On path)***.

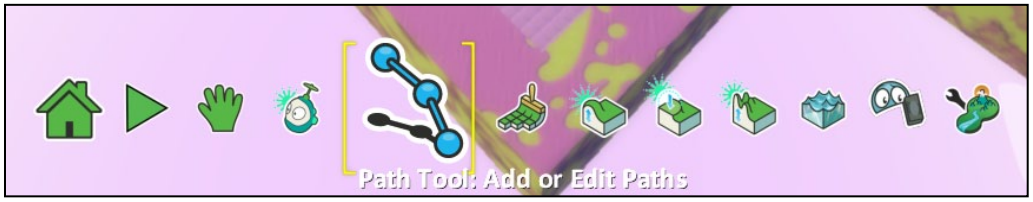
Δείτε το παράδειγμα 05_04.kodu

Ας δούμε βήμα-βήμα πως το κάνουμε αυτό. Ανοίξτε τον κόσμο **[05_04.kodu]**. Στο περιβάλλον του παιχνιδιού υπάρχουν δύο Kodu, ένας μπλε και ένας ροζ που είναι η φίλη του! Επιπλέον υπάρχουν τρία ***Κουμπιά Πίεσης (Pushpad)***. Η ροζ Kodu είναι αυτή που χειρίζεται ο παίχτης. Στόχος του παιχνιδιού είναι η ροζ Kodu να φτάσει εκεί που βρίσκεται ο μπλε Kodu, αποφεύγοντας τα *Κουμπιά Πίεσης (Pushpad)* πάνω στα οποία αν πέσει θα τελειώσει το παιχνίδι. Ο κόσμος του παιχνιδιού φαίνεται στην ακόλουθη εικόνα:

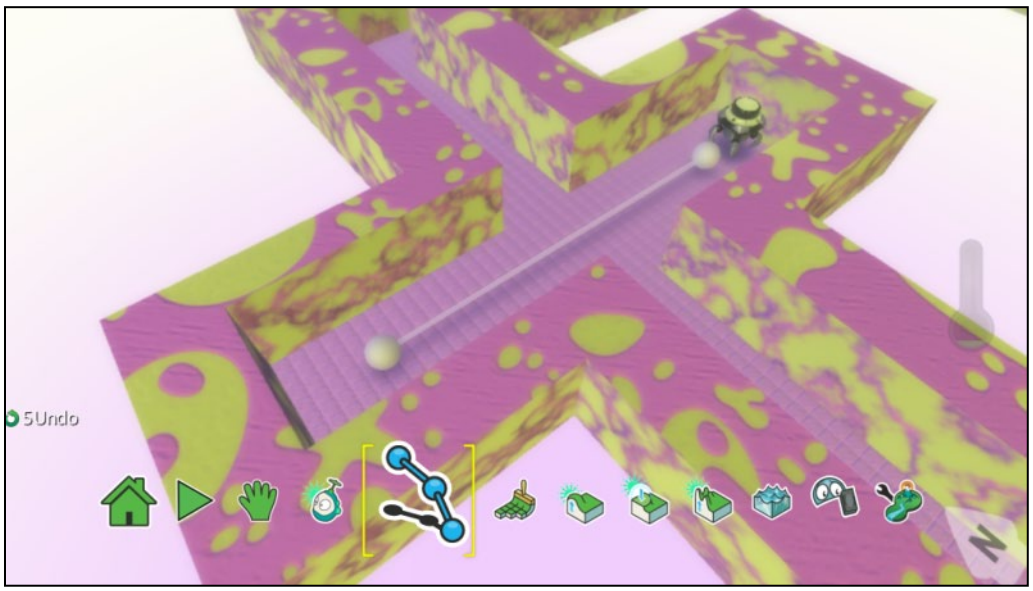


Αν τρέξετε το παιχνίδι, θα παρατηρήσετε ότι τα ***Κουμπιά Πίεσης (Pushpad)*** είναι ακίνητα. Έτσι όμως το παιχνίδι δεν θα είχε νόημα, αφού ο παίχτης με χρήση των βελών του πληκτρολογίου θα οδηγούσε τη ροζ Kodu χωρίς καμία δυσκολία, κατευθείαν στο φίλο της μπλε Kodu. Θέλουμε να προγραμματίσουμε τα ***Κουμπιά Πίεσης (Pushpad)*** ώστε να κινούνται αυτόματα σε κάποια συγκεκριμένη διαδρομή, έτσι ώστε να αποτελούν εμπόδια στην διαδρομή της ροζ Kodu. Για το σκοπό αυτό θα χρησιμοποιήσουμε τα μονοπάτια.

Αρχικά, πάμε να δούμε πως μπορούμε να δημιουργήσουμε ένα μονοπάτι. Αυτό γίνεται με την χρήση του ***Εργαλείου Μονοπατιού (Path Tool)*** που βρίσκεται στο ***Κεντρικό Μενού (Main Menu)***.



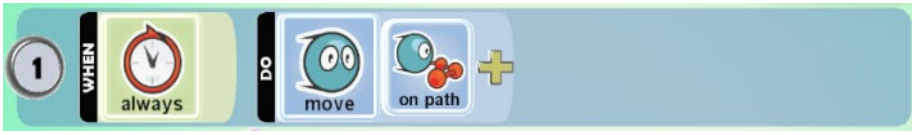
Αφού το επιλέξετε, πηγαίνετε με το ποντίκι σας στην μία άκρη της διασταύρωσης όπου βρίσκεται το κίτρινο ***Κουμπί Πίεσης (Pushpad)*** και πατήστε αριστερό κλικ για να προσθέσετε έναν ***Κόμβο (Node)***. Ξεκινώντας από αυτόν τον κόμβο, μπορείτε να σύρετε το ποντίκι σας ώστε να σχεδιάσετε ένα ευθύγραμμο μονοπάτι κατά μήκος του δρόμου. Όταν φτάσετε στην απέναντι άκρη του δρόμου, πατήστε αριστερό κλικ ώστε να προσθέσετε έναν ακόμη κόμβο και να ολοκληρώσετε την σχεδίαση του μονοπατιού. Το μονοπάτι που σχεδιάσατε θα πρέπει να είναι κάπως έτσι:



Οι *Κόμβοι (Node)* χρησιμοποιούνται ως σημεία σύνδεσης ευθύγραμμων μονοπατιών. Με τον τρόπο αυτό, με λίγα κλικ, μπορείτε να δημιουργήσετε μεγάλα και σύνθετα μονοπάτια.

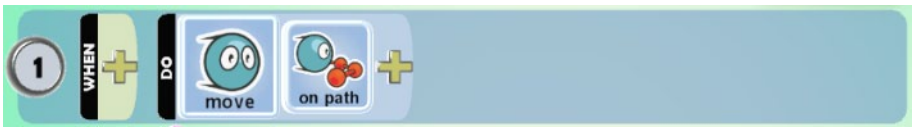
Τώρα θέλουμε να προγραμματίσουμε το κίτρινο *Κουμπί Πίεσης (Pushpad)* ώστε να κινείται αυτόματα και καθ' όλη την διάρκεια του παιχνιδιού πάνω σε αυτό το μονοπάτι. Όπως αναφέραμε παραπάνω, για να πετύχουμε την κίνηση πάνω στο μονοπάτι θα χρησιμοποιήσουμε την ενέργεια ***Κινούμαι (Move)*** με προσδιοριστικό ***Πάνω στο μονοπάτι (On path)***. Όμως, ποιον αισθητήρα θα χρησιμοποιήσουμε; Δε θέλουμε ο χειρισμός της κίνησης να γίνεται από τον χρήστη, συνεπώς σίγουρα δεν θα πρέπει να χρησιμοποιήσουμε κάποιον από τους αισθητήρες, *Πληκτρολόγιο (Keyboard) ή Ποντίκι (Mouse)*. Ουσιαστικά αυτό σημαίνει πως δεν χρειάζεται να χρησιμοποιήσουμε κάποιον αισθητήρα που να προκαλέσει την κίνηση! Για να δηλώσουμε τέτοιες συμπεριφορές, χρησιμοποιούμε την επιλογή ***Πάντα (Always)*** στη θέση του γεγονότος. Οπότε, η συμπεριφορά που θα προσθέσουμε στα κίτρινο *Κουμπί Πίεσης (Pushpad)* θα είναι η εξής:

ΌΤΑΝ[πάντα] **ΤΟΤΕ** [κινήσου][πάνω στο μονοπάτι]

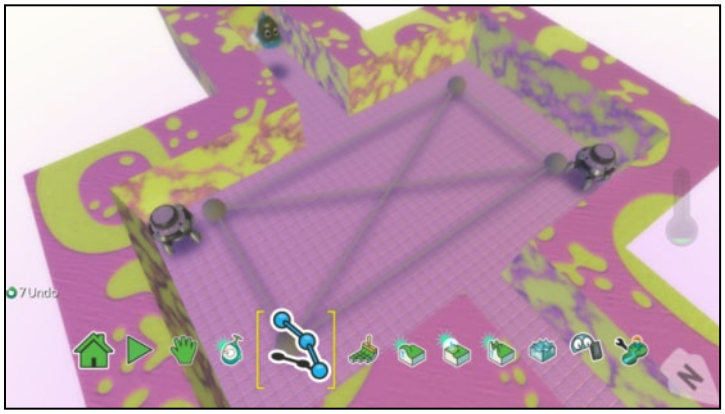


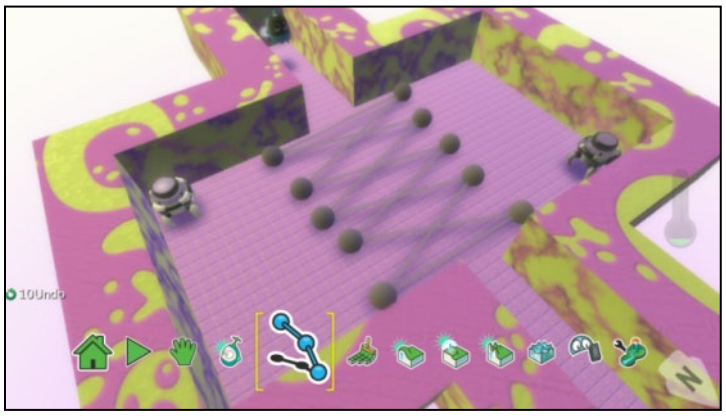
Λογικά θα αναρωτηθήκατε: Αφού δεν χρησιμοποιούμε κάποιον αισθητήρα, μπορούμε απλά να μη συμπληρώσουμε τίποτα στο πεδίο του γεγονότος; Ναι! Η επιλογή ***Πάντα (Always)*** χρησιμοποιείται ώστε να κάνουμε την εντολή μας πιο κατανοητή! Αν θέλουμε, μπορούμε να την παραλείψουμε. Έτσι, θα μπορούσαμε ισοδύναμα με την παραπάνω εντολή να γράψουμε:

ΌΤΑΝ[] **ΤΟΤΕ** [κινήσου][πάνω στο μονοπάτι]



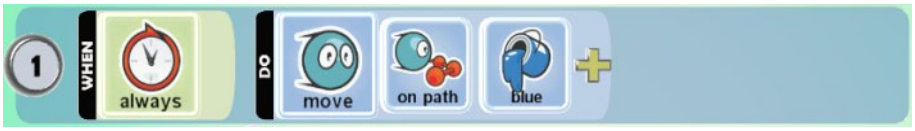
Τώρα, προσπαθήστε να σχεδιάσετε ένα πιο σύνθετο μονοπάτι στην περιοχή όπου βρίσκονται τα δύο μωβ *Κουμπιά Πίεσης (Pushpad)*. Μπορείτε, για παράδειγμα, να σχεδιάσετε κάποιο μονοπάτι που να μοιάζει με τα παρακάτω:





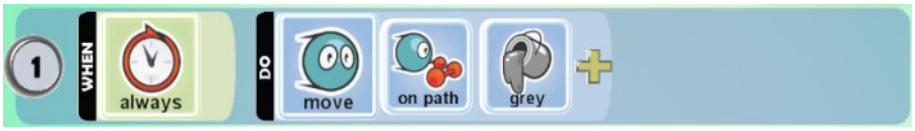
Παρατηρείστε ότι μπορείτε να ***Αλλάξετε το Χρώμα (Change Color)*** του μονοπατιού χρησιμοποιώντας το αριστερό και δεξί βέλος του πληκτρολογίου. Αλλάξτε το χρώμα του μονοπατιού που σχεδιάσατε σε μπλε. Για να εφαρμοστεί το χρώμα σε όλα τα επιμέρους μονοπάτια κρατείστε πατημένο το *πλήκτρο Tab*. Θα χρησιμοποιήσουμε το χρώμα του μονοπατιού ώστε να προσδιορίσουμε σε ποιο από τα δύο μονοπάτια που έχει τώρα ο κόσμος μας, θέλουμε κάθε ένα από τα *Κουμπιά Πίεσης (Pushpad)* να κινείται. Οπότε, η συμπεριφορά που θα προσθέσουμε στα μωβ *Κουμπιά Πίεσης (Pushpad)* θα είναι η εξής:

ΌΤΑΝ[πάντα] **ΤΟΤΕ** [κινήσου][πάνω στο μονοπάτι][με μπλε χρώμα]



Επίσης, θα πρέπει τώρα να προσδιορίσουμε και το χρώμα του μονοπατιού που θα κινείται το Κίτρινο *Κουμπί Πίεσης (Pushpad).* Δεδομένου ότι το πρώτο μονοπάτι που σχεδιάσαμε έχει το προκαθορισμένο χρώμα γκρι, αφού εμείς δεν το αλλάξαμε, θα εμπλουτίσουμε την συμπεριφορά του ως εξής:

ΌΤΑΝ[πάντα] **ΤΟΤΕ** [κινήσου][πάνω στο μονοπάτι][με γκρι χρώμα]



Αποθηκεύστε τις αλλαγές και παίξτε το παιχνίδι.

Το Kodu μας παρέχει την δυνατότητα να έχουμε και άλλους τύπους μονοπατιού τους οποίους μπορούμε να χρησιμοποιήσουμε ώστε με λίγη φαντασία να δημιουργήσουμε εντυπωσιακά σκηνικά. Μπορείτε να ***Αλλάξετε τον Τύπο του Μονοπατιού (Change Path Style)*** χρησιμοποιώντας το πάνω και κάτω βέλος του πληκτρολογίου. Επιπλέον, μπορείτε να κάνετε δεξί κλικ με το ποντίκι σας πάνω σε ένα μονοπάτι ώστε να δείτε μερικές ακόμη επιλογές. Μία από αυτές είναι η ***Αλλαγή του Ύψους (Change Height)*** του μονοπατιού (π.χ. για να δημιουργήσετε ένα μονοπάτι πάνω στον ουρανό στο οποίο θα κινούνται κάποια αερόπλοια ή μέσα στο νερό στο οποίο θα κινούνται μερικά χελιδονόψαρα).

Σε περιπτώσεις που έχουμε προγραμματίσει το αντικείμενό μας να κινείται ***Πάντα (Always)*** πάνω σε ένα μονοπάτι, μπορούμε να χρησιμοποιούμε και κάποια επιπλέον προσδιοριστικά της ενέργειας ***Κινούμαι (Move)***. Αυτά εμφανίζονται στην επιλογή ***Όριο (Limit)*** της πρώτης πίτας προσδιοριστικών και έχουν να κάνουν με τον περιορισμό της κίνησης του αντικειμένου. Έτσι μπορούμε να πούμε σε ένα αντικείμενο να κινείται ***μόνο Βόρεια και Νότια (N/S)*** ή ***μόνο Ανατολικά και Δυτικά (E/W)*** (π.χ. αν θέλουμε να κινείται διαρκώς κάθετα ή οριζόντια στον κόσμο) ή ακόμη και να ***Ακινητοποιείται (Freeze)*** (π.χ. όταν ο χρήστης πατάει κάποιο πλήκτρο του πληκτρολογίου, διακόπτοντας έτσι την διαρκή αυτόματη κίνησή του).

Θυμηθείτε το προσδιοριστικό ***Περιπλάνηση (Wander)*** της ενέργειας *Κινούμαι (Move)*. Σκεφτείτε κάποια παραδείγματα παιχνιδιών που θα ήταν χρήσιμο να το χρησιμοποιήσετε σε συνδυασμό με την έννοια *Πάντα (Always)*. Θα είχε κάποιο πρακτικό νόημα αν το χρησιμοποιούσατε σε συνδυασμό με τον αισθητήρα *Πληκτρολόγιο (Keyboard)* ή *Ποντίκι (Mouse)*;

5.3.3 Ενέργεια Πηδάω (Jump)

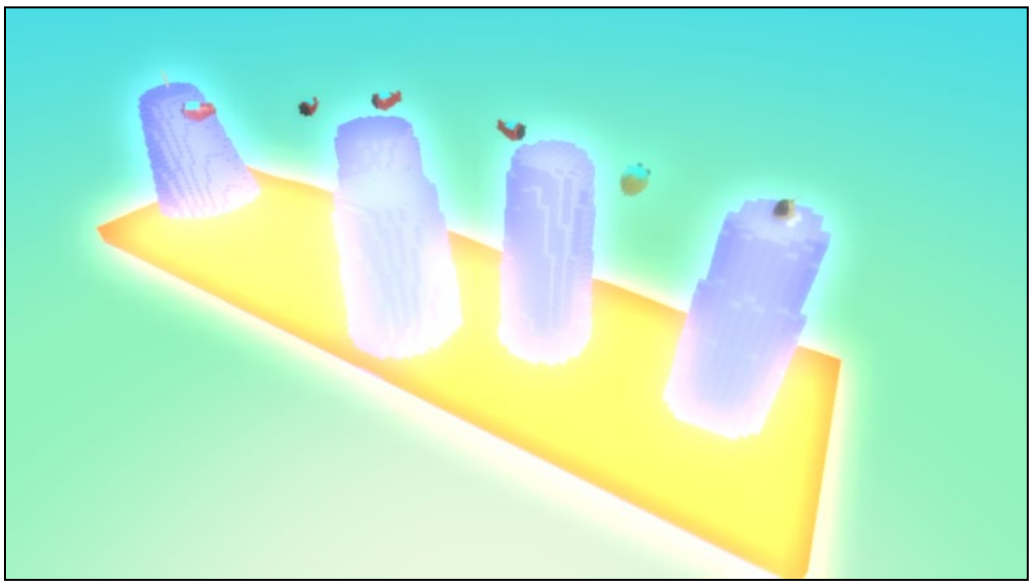
Άλλη μια ενέργεια που μπορούν να κάνουν τα αντικείμενα στο MSKodu είναι να πηδούν στον αέρα είτε για να αποφεύγουν εμπόδια είτε απλά για να χορεύουν! Στα περισσότερα παιχνίδια ο ήρωας έχει αντίστοιχες δυνατότητες και στο MSKodu μας δίνεται αυτή η δυνατότητα με την ενέργεια ***Πηδάω (Jump)***.

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Πηδάω, περνάω, σκαρφαλώνω, ανεβαίνω}

Ας δούμε πως μπορούμε να χρησιμοποιήσουμε την ενέργεια στο επόμενο παιχνίδι. Στο παιχνίδι μας, υπάρχει ο χαρακτήρας Kodu που είναι και αυτός που χειρίζεται ο χρήστης. Επιπλέον, υπάρχουν ένα αντικείμενο ***Αστέρι (Star)*** και πέντε ***Αερόπλοια (Blimps)*** τα οποία είναι προγραμματισμένα να κινούνται στον αέρα ανάμεσα στις κολώνες που βλέπετε στην παρακάτω

εικόνα. Σκοπός του παιχνιδιού είναι ο χειριστής του χαρακτήρα Kodu να μπορέσει να φτάσει στο ***Αστέρι (Star)*** το οποίο βρίσκεται στην τελευταία κολώνα, χωρίς ο Kodu να πέσει σ τη λάβα ή να συγκρουστεί με κάποιο από τα αερόπλοια. Ο κόσμος του παιχνιδιού φαίνεται στην εικόνα που ακολουθεί:



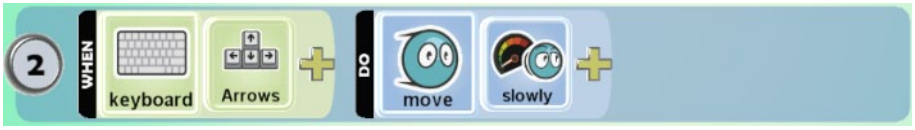
Δείτε το παράδειγμα *05_05.kodu*

Φορτώστε το παιχνίδι **[05_05.kodu]** για να ξεκινήσουμε τον προγραμματισμό του Kodu. Θέλουμε ο Kodu να έχει την δυνατότητα να *πηδάει* έτσι ώστε να μπορέσει να φτάσει στην τελευταία κολώνα, πηδώντας από κολώνα σε κολώνα. Για το λόγο αυτό θα χρησιμοποιήσουμε την ενέργεια ***Πηδάω (Jump)***. Επιπλέον θέλουμε να προγραμματίσουμε τον Kodu να *πηδάει* όταν ο χρήστης πατήσει ένα πλήκτρο του πληκτρολογίου (π.χ. το *Space)*. Προφανώς, για το σκοπό αυτό θα χρησιμοποιήσουμε τον αισθητήρα ***Πληκτρολόγιο (Keyboard)***. Ο Kodu λοιπόν θα έχει την εξής συμπεριφορά:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο space] **ΤΟΤΕ** [πήδα]

Αρκεί αυτή η συμπεριφορά του Kodu ώστε να μπορεί να πηδάει από την μία κολώνα στην επόμενη; Προς τα πού πηδάει ο Kodu; Μήπως ο παίχτης χρειάζεται να κινεί τον Kodu πάνω σε αυτήν την επιφάνεια ώστε να πάρει μία πιο βολική θέση προτού πηδήξει στην απέναντι κολώνα; Για το λόγο αυτό προσθέτουμε στον Kodu ακόμη μία συμπεριφορά:

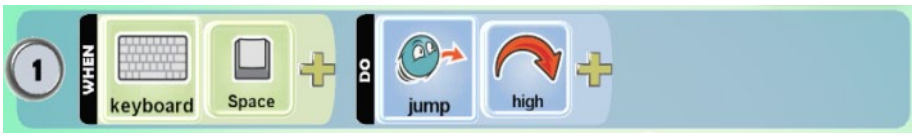
ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί κάποιο από τα βέλη] **ΤΟΤΕ** [κινήσου προς την αντίστοιχη κατεύθυνση][αργά]



Παρατηρείστε πως χρησιμοποιήσαμε το προσδιοριστικό ***Αργά (Slowly)*** στην ενέργεια ***Κινούμαι (Move)*** γιατί θέλουμε ο Kodu να κινείται αργά πάνω στην μικρή επιφάνεια της κάθε κολώνας ώστε να μην πέφτει πολύ εύκολα από κάτω. Αποθηκεύστε τις αλλαγές και παίξτε το παιχνίδι.

Παρατηρείστε ότι ο Kodu δυσκολεύεται να περάσει πάνω από το τελευταίο μεγάλο κενό ώστε να φτάσει στην κολώνα που υπάρχει το αστέρι. Πως θα μπορούσαμε να ενισχύσουμε την ενέργεια ***Πηδάω (Jump)*** ώστε ο Kodu να ***πηδάει ψηλά/μακριά;*** Ουσιαστικά θέλουμε να προσθέσουμε ένα προσδιοριστικό στην ενέργεια ***Πηδάω (Jump)***. Το MSKodu μας παρέχει για την ενέργεια ***Πηδάω (Jump)*** τα προσδιοριστικά ***Ψηλά (High)*** και ***Χαμηλά (Low)*** τα οποία επιτρέπουν να προσδιορίσουμε αν τα άλματα που θα κάνει ο χαρακτήρας είναι ψηλά ή χαμηλά, αντίστοιχα. Εμείς χρειαζόμαστε το προσδιοριστικό ***Ψηλά (High)*** στο συγκεκριμένο παράδειγμα. Εμπλουτίζουμε την συμπεριφορά του Kodu ως εξής:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο space] **ΤΟΤΕ** [πήδα][ψηλά]



Τώρα ο Kodu θα μπορεί να πηδάει πιο εύκολα το τελευταίο μεγάλο κενό προς την κολώνα που βρίσκεται το *Αστέρι (Star)*. Δοκιμάστε το!

5.3.4 Ενέργεια Στρίβω (Turn)

Όπως είδαμε σε πολλά παραδείγματα, η ενέργεια ***Στρίβω (Turn)*** δίνει τη δυνατότητα στα αντικείμενα να στρίβουν, δηλαδή να αλλάζουν τον προσανατολισμό τους, είτε για να προχωρήσουν προς τη νέα κατεύθυνση, είτε για να πετύχουν πιο εύκολα κάποιο στόχο τους, είτε απλά για να κοιτάξουν προς μια κατεύθυνση.

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Στρίβω, γυρνάω, περιστρέφομαι, αλλάζω κατεύθυνση, στριφογυρνάω, στροβιλίζομαι}

Πάμε να δούμε πως μπορεί αυτή η ενέργεια να φανεί χρήσιμη στο παράδειγμα με τα άλματα που μόλις είδαμε. Φορτώστε το παιχνίδι του προηγούμενου παραδείγματος **05_05.kodu**. Θυμηθείτε πως είχαμε προγραμματίσει τον Kodu να κινείται όταν ο παίχτης πατάει τα βέλη του πληκτρολογίου ώστε να μπορεί πάρει μία πιο βολική θέση προτού πηδήξει στην απέναντι κολώνα:

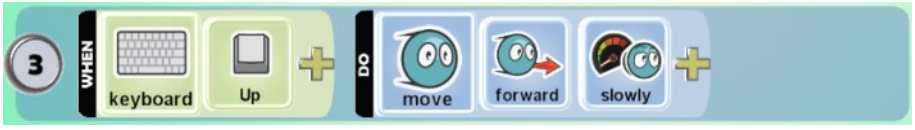
Θα διαπιστώσατε, όταν παίξατε το παιχνίδι, πως με αυτόν τον τρόπο κίνησης είναι πολύ εύκολο ο Kodu να πέσει κάτω από την κολώνα καθώς ο χρήστης προσπαθεί να διορθώσει την θέση του. Συνεπώς, θα ήταν καλύτερο ο χρήστης να μπορεί απλά να αλλάζει τον προσανατολισμό του Kodu χωρίς να κινείται ταυτόχρονα προς την αντίστοιχη κατεύθυνση. Για το λόγο αυτό θα χρησιμοποιήσουμε την ενέργεια ***Στρίβω (Turn)*** σε συνδυασμό με τα προσδιοριστικά αυτής ***Αριστερά (Left)*** και ***Δεξιά (Right)***. Οπότε θα προσθέσουμε στον Kodu τις εξής συμπεριφορές:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Left] **ΤΟΤΕ** [στρίψε][αριστερά]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος Right] **ΤΟΤΕ** [στρίψε][δεξιά]

Καλό είναι να προσθέσουμε στον Kodu και την συμπεριφορά:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το βέλος up] **ΤΟΤΕ** [κινήσου] [προς τα εμπρός]

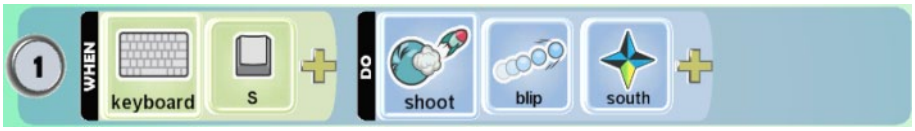


Έτσι, ο χρήστης θα έχει την δυνατότητα να τον κινήσει προς τα εμπρός αφού πρώτα έχει διορθώσει την θέση του.

Η ενέργεια ***Στρίβω (Turn)*** δέχεται ως προσδιοριστικά την ***Κατεύθυνση (Direction)***, με βάση τα σημεία του ορίζοντα και την ταχύτητα (γρήγορα ή αργά). Θυμηθείτε πως αυτά τα προσδιοριστικά τα συναντήσαμε και στην ενέργεια *Κινούμαι (Move)*. Η χρησιμότητά τους είναι η ίδια και σε αυτήν την περίπτωση.

5.3.5 Ενέργεια Πυροβολώ (Shoot)

Η ενέργεια ***Πυροβολώ (Shoot)*** δίνει την δυνατότητα στα αντικείμενα να πυροβολούν για να εξολοθρεύσουν ή απλά να ζαλίσουν κάποιον άλλο χαρακτήρα (συνήθως κάποιον αντίπαλο). Έχουμε ήδη δει πως μπορούμε να καθορίσουμε το είδος της βολής (με ***Σφαιρίδιο (Blip)*** ή με ***Πύραυλο (Missile)***) χρησιμοποιώντας τα αντίστοιχα προσδιοριστικά που μας παρέχει το MSKodu για την ενέργεια αυτή. Ανατρέξτε στην ενότητα 5.2.2 για να θυμηθείτε τη διαφορά τους. Επιπλέον, μπορούμε να καθορίσουμε την ***Κατεύθυνση (Direction)*** της βολής καθώς και το ***Είδος της Ζημιάς (Combat)*** που θα προκαλέσει στο αντικείμενο που θα πυροβοληθεί. Θα μπορούσαμε για παράδειγμα να προγραμματίσουμε το χαρακτήρα μας, ώστε όταν ο χρήστης πατάει ένα πλήκτρο του *Πληκτρολογίου (Keyboard)* τότε να πυροβολεί προς τα πάνω (π.χ. για να πετύχει κάποια αντικείμενα που πετάνε από πάνω του) ή όταν δει ένα αντικείμενο-αντίπαλο να τον πυροβολεί με *Σφαιρίδια (Blip)* που θα έχουν την ιδιότητα να ζαλίζουν τον στόχο (π.χ. με σκοπό να τον καθυστερήσει). Ένα παράδειγμα εντολής με χρήση αυτής της ενέργειας είναι:



Τι σκεφτόταν ο προγραμματιστής;

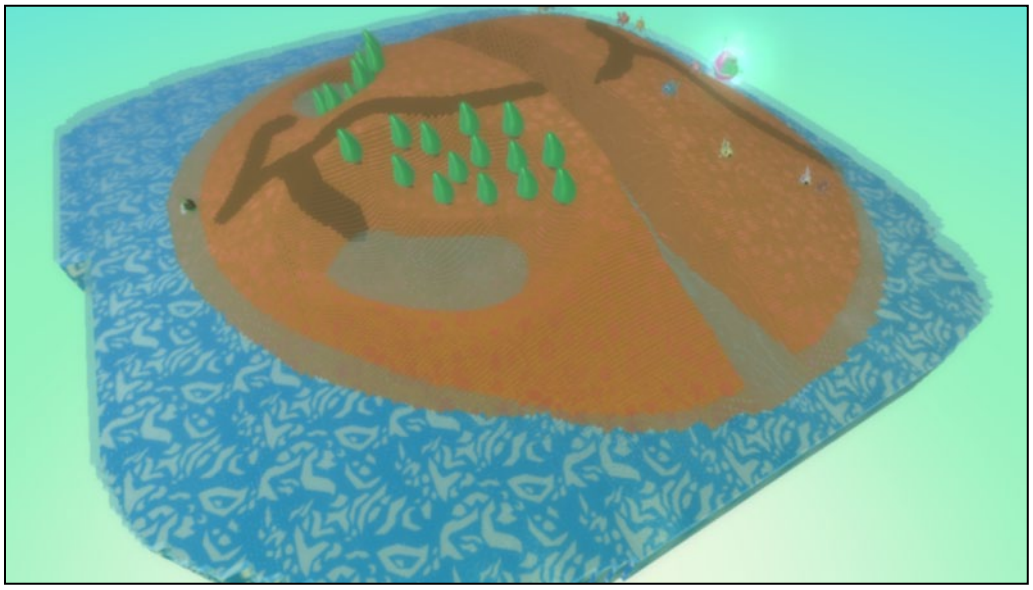
ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο S] **ΤΟΤΕ** [πυροβόλησε][με σφαιρίδιο][προς τα νότια]

Αν δεν καθορίσουμε κανένα προσδιοριστικό για αυτή την ενέργεια, τότε το αντικείμενό μας πυροβολεί *Σφαιρίδια (Blip)* με κατεύθυνση προς τα εμπρός.

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Πυροβολώ, καταστρέφω, εξολοθρεύω, σκοτώνω, ζαλίζω, εξαφανίζω, εξουθενώνω, διαλύω, εκτοξεύω}

Πάμε να δούμε την χρήση αυτής της ενέργειας σε ένα παιχνίδι. Στο παιχνίδι μας υπάρχει ο χαρακτήρας Kodu που χειρίζεται ο παίχτης/χρήστης. Επιπλέον, υπάρχουν μερικά ***Χελιδονόψαρα (Fly Fish)*** τα οποία έχουν προγραμματιστεί έτσι ώστε όταν τα ακουμπήσει ο Kodu, το παιχνίδι να τελειώνει και ο παίχτης να χάνει. Σκοπός λοιπόν του παιχνιδιού είναι ο παίχτης να οδηγήσει τον Kodu στην απέναντι όχθη και συγκεκριμένα εκεί όπου βρίσκεται το ***Πλοίο (Ship)*** του, χωρίς να ακουμπήσει τα χελιδονόψαρα αλλά και χωρίς να πέσει μέσα στον γκρεμό που υπάρχει στον κόσμο. Ο κόσμος του παιχνιδιού φαίνεται στην ακόλουθη εικόνα:



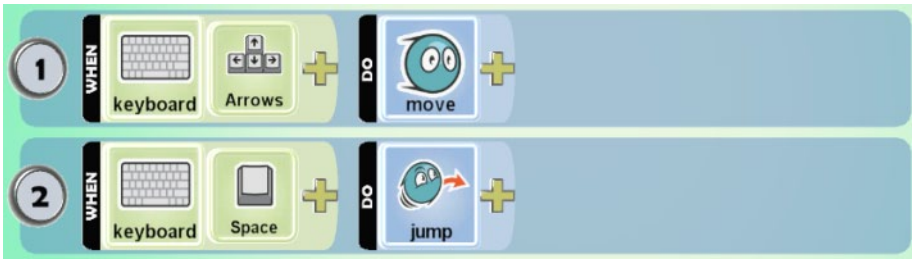
Δείτε το παράδειγμα *05_06.kodu*

Φορτώστε το παιχνίδι **[05_06.kodu]** για να ξεκινήσουμε τον προγραμματισμό του Kodu. Όπως και στα προηγούμενα παραδείγματα για την κίνηση του πρωταγωνιστή μας, θα χρησιμοποιήσουμε την ενέργεια *Κινούμαι (Move)*, σε συνδυασμό με τον αισθητήρα *Πληκτρολόγιο (Keyboard)* και τα βέλη του πληκτρολογίου. Επιπλέον, για να μπορέσει ο Kodu να περάσει τον γκρεμό, θα τον

προγραμματίσουμε ώστε να ***Πηδάει (Jump)*** όταν ο χρήστης πατάει το *πλήκτρο Z*. Συνεπώς δύο συμπεριφορές που θα εμφανίζει ο Kodu, είναι:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί κάποιο από τα βέλη] **ΤΟΤΕ** [κινήσου προς αντίστοιχη κατεύθυνση]

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο Z] **ΤΟΤΕ** [πήδα]



Στην συνέχεια, παρατηρούμε πως όλα τα *Χελιδονόψαρα (Fly Fish)* κινούνται κοντά στην περιοχή όπου είναι το *Πλοίο (Ship)* στο οποίο πρέπει να φτάσει ο Kodu. Έτσι θα είναι πολύ δύσκολο ο παίχτης να καταφέρει να περάσει τον Kodu ανάμεσα τους χωρίς να τον ακουμπήσουν. Τι μπορούμε να κάνουμε για να μειώσουμε την δυσκολία του παιχνιδιού; Μία καλή ιδέα (τρόπος του λέγειν!) είναι να προγραμματίσουμε τον Kodu να μπορεί να *πυροβολεί* ώστε να σκοτώνει τα *Χελιδονόψαρα (Fly Fish)* πριν προλάβουν να τον ακουμπήσουν. Για τον λόγο αυτό θα χρησιμοποιήσουμε την ενέργεια ***Πυροβολώ (Shoot)***. Αυτήν την φορά θα προτιμήσουμε να χρησιμοποιήσουμε το προσδιοριστικό ***Πύραυλος (Missile)***. Επιπλέον θα χρησιμοποιήσουμε και ένα ακόμη προσδιοριστικό της ενέργειας ***Πυροβολώ (Shoot)***, το ***Εξαφανίζω (Vanish)*** ώστε τα *Χελιδονόψαρα (Fly Fish)* να εξαφανίζονται αμέσως μόλις τα πετύχει ο *Πύραυλος (Missile)*. Ο χειρισμός και αυτής της συμπεριφοράς του Kodu θέλουμε να γίνεται από τον χρήστη οπότε θα χρησιμοποιήσουμε και πάλι τον αισθητήρα *Πληκτρολόγιο (Keyboard)*. Θα προσθέσουμε στον Kodu την εξής συμπεριφορά:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο s] **ΤΟΤΕ** [πυροβόλησε][με πύραυλο][που εξαφανίζει τον στόχο]

Αποθηκεύστε τις αλλαγές και παίξτε το παιχνίδι.

Όταν στην ενέργεια *Πυροβολώ (Shoot)* χρησιμοποιούμε το προσδιοριστικό *Πύραυλος (Missile)*, τότε εμφανίζονται δύο νέα προσδιοριστικά, το *Κρουζ (Cruise)* και το *Στο ίδιο Επίπεδο (Level)*. Τα προσδιοριστικά αυτά χρησιμοποιούνται για να καθορίσουν την κίνηση του πυραύλου. Το πρώτο καθορίζει ότι ο πύραυλος θα κινηθεί ακολουθώντας την καμπυλότητα του εδάφους (!) ενώ το δεύτερο καθορίζει ότι ο πύραυλος ότι θα κινηθεί ευθεία στο ίδιο επίπεδο που βρίσκεται ο χαρακτήρας που πυροβολεί.

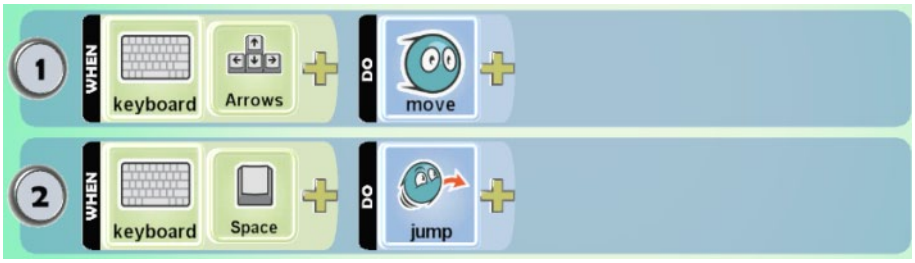
Ας δούμε πως μπορεί το προσδιοριστικό ***Κρουζ (Cruise)*** να μας φανεί χρήσιμο στο παραπάνω παιχνίδι. Παρατηρείστε πως τα *Χελιδονόψαρα (Fly Fish)* δεν περνάνε πάνω από τον γκρεμό. Μία καλή σκέψη, λοιπόν, είναι ο χειριστής του Kodu να μπορεί να τα εξαφανίσει πυροβολώντας τα με πυραύλους, πριν περάσει τον Kodu πάνω από τον γκρεμό. Έτσι είναι σίγουρο ότι δεν θα ακουμπήσουν τον Kodu. Όμως δεν αρκεί η τρέχουσα συμπεριφορά του Kodu για να μπορέσει να το κάνει αυτό, αφού οι πύραυλοι κινούνται ευθεία μπροστά του, ενώ τα περισσότερα *Χελιδονόψαρα (Fly Fish)* βρίσκονται πίσω από τον λόφο. Για το λόγο αυτό θα χρησιμοποιήσει επιπλέον στην ενέργεια ***Πυροβολώ (Shoot)*** το προσδιοριστικό ***Κρουζ (Cruise)***. Η τελευταία του συμπεριφορά θα γίνει πλέον:

ΌΤΑΝ[στο πληκτρολόγιο][πατηθεί το πλήκτρο space] **ΤΟΤΕ** [πυροβόλησε][με πύραυλο][που εξαφανίζει τον στόχο][και κινείται ακολουθώντας την καμπυλότητα του κόσμου]

Αποθηκεύστε τις αλλαγές και παίξτε το παιχνίδι. Ελπίζουμε να έχετε πειστεί ότι στο MSKodu μπορείτε να κάνετε θαύματα!

5.4 Όλα Παράλληλα!

Ίσως κάποιοι από εσάς αναρωτηθήκατε με ποια σειρά ελέγχονται τα γεγονότα των συμπεριφορών όταν προγραμματίζουμε ένα αντικείμενο στο MSKodu. Σκεφτείτε για παράδειγμα κάποιο παιχνίδι στο οποίο είχαμε προγραμματίσει τον Kodu να κινείται μπροστά όταν ο χρήστης πατάει το *πάνω Βέλος (Up)* του πληκτρολογίου και να πυροβολεί με *Σφαιρίδιο (Blip)* όταν ο χρήστης πατάει το *πλήκτρο space*. Κάθε ενέργεια θεωρητικά εκτελείται όταν το αντικείμενό μας αντιλαμβάνεται το αντίστοιχο γεγονός. Δεν είχε σημασία με ποια σειρά θα συμβεί το κάθε γεγονός, ούτε με ποια σειρά είχαμε γράψει τις εντολές στο αντικείμενο μας! Αυτό σημαίνει πως όλες οι εντολές εκτελούνται παράλληλα! Για να το καταλάβουμε καλύτερα αυτό, ανοίξτε έναν *Άδειο Κόσμο (Empty World)* και προσθέστε έναν Kodu. Στην συνέχεια προσθέστε σε αυτόν τις παρακάτω συμπεριφορές:

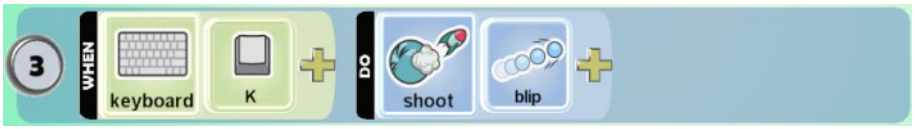


Αποθηκεύστε τις αλλαγές και δοκιμάστε να πατήσετε μαζί ένα από τα *Βέλη (Arrows)* και το *πλήκτρο Space*. Ο Kodu κινείται και πηδάει ταυτόχρονα, δηλαδή την ίδια στιγμή. Ουσιαστικά όλοι οι αισθητήρες λειτουργούν παράλληλα! Κάθε φορά που ενεργοποιούνται ένας ή περισσότεροι αισθητήρες, εκτελούνται και οι αντίστοιχες ενέργειες. Μπορείτε να ανατρέξτε σε κάποια από τα παιχνίδια που συναντήσαμε σε αυτό το Κεφάλαιο για να το επιβεβαιώσετε.

Τι γίνεται όμως αν ο προγραμματιστής ορίσει δύο ή παραπάνω διαφορετικές ενέργειες να συμβαίνουν με τον ίδιο αισθητήρα; Για να το δούμε αυτό αλλάξτε την παραπάνω συμπεριφορά του Kodu προσθέτοντάς του τις εντολές:

Αποθηκεύστε τις αλλαγές και πατήστε το *πλήκτρο K*. Παρατηρείστε πως ο Kodu πάλι κινείται μπροστά και πηδάει ταυτόχρονα!

Γενικά στο MSKodu μπορείτε να προγραμματίσετε ένα αντικείμενο να αντιλαμβάνεται το ίδιο γεγονός σε όσες εντολές θέλετε, προσαρτώντας του κάθε φορά διαφορετική ενέργεια. Για να πειστείτε περισσότερο μπορείτε να προσθέσετε στον Kodu επιπλέον και την εντολή:

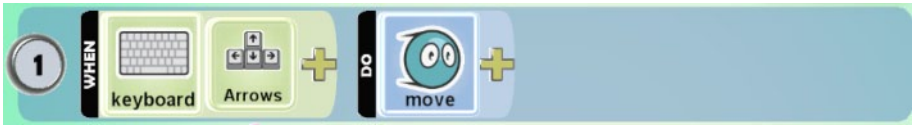


Σύμφωνα με τα παραπάνω, αναμένετε πως όταν πατήσετε το *πλήκτρο K*, ο Kodu θα κινηθεί μπροστά, θα πηδήξει και θα πυροβολήσει ταυτόχρονα. Δοκιμάστε το! Κάνει αυτό που αναμένατε!

Αν θέλαμε να εκφράσουμε προφορικά αυτή τη λογική τότε θα λέγαμε ότι στο MSKodu επιτρέπονται εντολές σαν την επόμενη (προσέξτε τη λέξη κλειδί ΚΑΙ μετά την ΤΟΤΕ):

ΟΤΑΝ [στο πληκτρολόγιο][πατηθεί το πλήκτρο Κ] **ΤΟΤΕ** [κινήσου][προς τα εμπρός] **και** [πήδα] **και** [πυροβόλησε][με σφαιρίδιο]]

Καλό είναι να επισημάνουμε πως παράλληλα ελέγχονται και τα γεγονότα των εντολών κάθε άλλου αντικειμένου. Για να το δείτε αυτό πρακτικά προσθέστε ένα *Κανόνι (Cannon)* στον προηγούμενο κόσμο. Έπειτα προσθέστε στον Kodu και στο κανόνι την παρακάτω συμπεριφορά:



Αποθηκεύστε τις αλλαγές, παίξτε το παιχνίδι και δοκιμάστε να πατήσετε τα βέλη του πληκτρολογίου σας. Παρατηρείστε πως οι δύο χαρακτήρες κινούνται μαζί! Το γεγονός ότι προγραμματίσαμε τους χαρακτήρες μας να αντιλαμβάνονται το ίδιο γεγονός και να αντιδρούν με τον ίδιο τρόπο, δεν επηρέασε καθόλου τη συμπεριφορά τους καθενός. Για να πειστείτε περισσότερο, μπορείτε να δοκιμάσετε να προσθέσετε ακόμη ένα αντικείμενο της επιλογής σας. Προγραμματίστε και τα τρία αντικείμενα να έχουν την ίδια συμπεριφορά, χρησιμοποιώντας τον ίδιο αισθητήρα. Το αποτέλεσμα είναι το ίδιο. Μπορείτε να δημιουργήσετε ένα χορευτικό με διάφορους χαρακτήρες του MSKodu; Δεν είναι ιδιαίτερα δύσκολο!

Θα εμβαθύνουμε περισσότερο στην αλληλεπίδραση μεταξύ των αντικειμένων στο επόμενο κεφάλαιο όπου και θα κατανοήσετε καλύτερα τις παραπάνω έννοιες.

Περίληψη

Στο κεφάλαιο αυτό αρχίσαμε να μελετούμε πιο συστηματικά τους αισθητήρες και τις ενέργειες του Kodu. Πιο συγκεκριμένα, εξετάσαμε αισθητήρες που επιτρέπουν στα αντικείμενά μας να αντιδρούν σε ενέργειες του χρήστη, όπως το *Πληκτρολόγιο (Keyboard)*, *το Ποντίκι (Mouse)* και το *Χειριστήριο (Gamepad)*. Φυσικά οι αισθητήρες δεν θα είχαν νόημα από μόνοι τους! Χρειαζόμαστε ενέργειες όπως *η Κινήσου (Move)* για να κινούνται οι πρωταγωνιστές μας, η *Πηδάω (Jump)* για να πηδούν, η *Στρίβω (Turn)* για να στρίβουν καθώς και η *Πυροβολώ (Shoot)* για να μπορούν να πυροβολούν! Μην ξεχνάτε ότι παρότι είναι εύκολο να αντιληφθούμε ποιον αισθητήρα και ποια ενέργεια χρειαζόμαστε, πρέπει να είμαστε προσεκτικοί στην επιλογή των κατάλληλων προσδιοριστικών ώστε να μη συμβαίνουν απρόοπτα. Πάμε να προγραμματίσουμε τους ήρωες μας να αλληλεπιδρούν μεταξύ τους;

Ερωτήσεις

1. Ποια συμπεριφορά υποδεικνύει η παρακάτω εντολή;

Αλλάξτε την εντολή ώστε να αναγνωρίζει την είσοδο του χρήστη από το αριστερό κλικ του ποντικιού. Στην συνέχεια, επιλέξτε τα κατάλληλα προσδιοριστικά στην ενέργεια *Πυροβολώ (Shoot)* ώστε να δημιουργήσετε την ενέργεια «πυροβολώ προς τα μπρος με σφαιρίδιο που ζαλίζει τον στόχο».

2. Στο παιχνίδι **[05_02.kodu]**, ποια εντολή πιστεύετε ότι χρησιμοποιήθηκε ώστε τα κανόνια να κινούνται διαρκώς γύρω από το ποτάμι; Ποια προεργασία είναι απαραίτητη ώστε να μπορέσετε να χρησιμοποιήσετε αυτήν την εντολή;

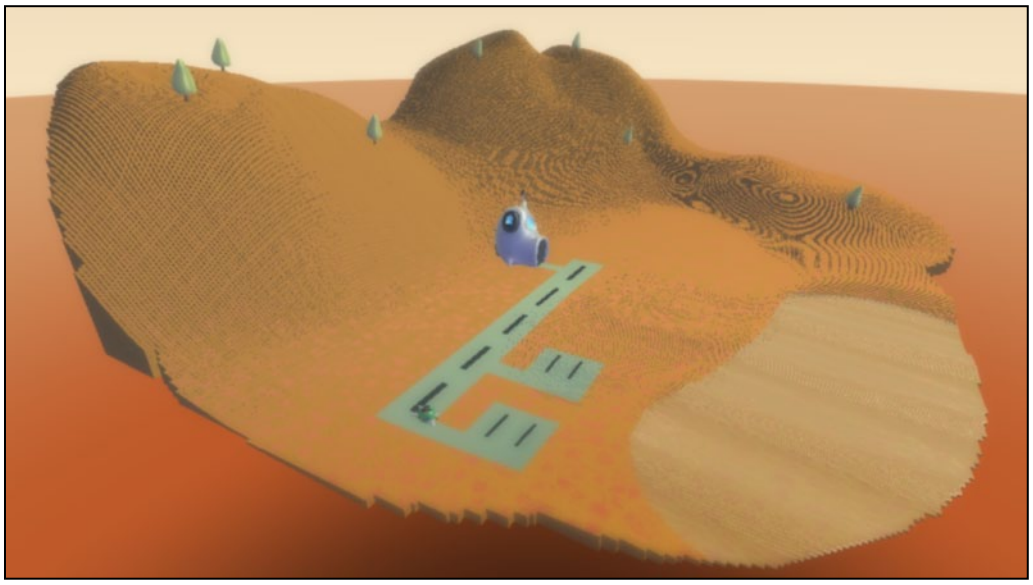
3. Τι θα συμβεί αν στην ενέργεια *Κινούμαι (Move)* εισάγω το προσδιοριστικό Up; Σκεφτείτε 2 παραδείγματα παιχνιδιών που θα ήταν χρήσιμο το συγκεκριμένο προσδιοριστικό.

4. Φορτώστε το παιχνίδι **[05_06.kodu]**. Θα μπορούσατε να χρησιμοποιήσετε τον αισθητήρα *Ποντίκι (Mouse)* ώστε ο Kodu να πετυχαίνει πιο εύκολα τα *Χελιδονόψαρα (Fly Fish);* Δικαιολογήστε το γιατί.

5. Μπορούμε να προγραμματίσουμε ένα αντικείμενο ταυτόχρονα να πηδάει ψηλά και να προχωράει μπροστά όταν ο χρήστης πατάει το πλήκτρο «Η» του πληκτρολογίου; Πως θα το κάνουμε αυτό;

Δραστηριότητες

1. Έστω ότι θέλουμε να φτιάξουμε ένα παιχνίδι προσομοιωτή πτήσης ενός αεροπλάνου *Τζετ (Jet)*. Φανταστείτε τον κόσμο να περιέχει έναν αεροδιάδρομο όπως για παράδειγμα φαίνεται στην παρακάτω εικόνα:



Το περιβάλλον περιέχει το *Τζετ (Jet)* το οποίο είναι και το αντικείμενο που χειρίζεται ο χρήστης. Προγραμματίστε το *Τζετ (Jet)* ώστε να κινείται προς τα πάνω και προς τα κάτω, και να στρίβει αριστερά και δεξιά, όταν ο χρήστης πατάει τα αντίστοιχα βελάκια του πληκτρολογίου. Προσοχή! Με τις έννοιες πάνω και κάτω, εννοούμε ότι ο χρήστης θα αλλάζει το υψόμετρο στο οποίο βρίσκεται το *Τζετ (Jet)*, όμως αυτό δεν θα κινείται μπροστά. Για το λόγο αυτό, προγραμματίστε το *Τζετ (Jet)* ώστε να κινείται *πάντα* προς τα μπροστά. Τέλος, θέλουμε ο χρήστης να μπορεί να ακινητοποιήσει το αεροπλάνο ώστε να μην πέσει στον γκρεμό που υπάρχει στο τέλος του αεροδιαδρόμου, κατά την προσγείωση. Αξιοποιήστε τον έτοιμο κόσμο **[05_07.kodu]**

2. Αρχικά φτιάξτε έναν κόσμο με ανώμαλο δρόμο κατάλληλο για έναν αγώνα ράλι. Χρησιμοποιείστε το *Εργαλείο Μονοπατιού (Path Tool)* σε συνδυασμό με τους διάφορους *Τύπους Μονοπατιού* (*Path Type)* για να δημιουργήσετε μονοπάτια που αντιστοιχούν στη διαδρομή της πίστας. Αλλάξτε τα ύψη των κόμβων του μονοπατιού ώστε να σχεδιάσετε ανηφόρες και κατηφόρες στο δρόμο σας. Αφού τελειώσετε με την κατασκευή του κόσμου, προσθέστε μερικά αντικείμενα και προγραμματίστε τα κατάλληλα ώστε να κινούνται αυτόματα επάνω στα μονοπάτια. Τέλος, προσθέστε και έναν χαρακτήρα, ο χειρισμός του οποίου να γίνεται από τον χρήστη με σκοπό να τρέξει στον αντίστοιχο αγώνα.

Κεφάλαιο 6ο: Αλληλεπιδρώ με τα αντικείμενα!

Ενώ στο προηγούμενο κεφάλαιο προγραμματίσαμε το αντικείμενό μας να αντιδρά ανάλογα με τη χρήση των συσκευών εισόδου, στο συγκεκριμένο κεφάλαιο θα μελετήσουμε πώς μπορούμε να το προγραμματίσουμε έτσι ώστε αλληλεπιδρά με άλλα αντικείμενα του κόσμου μας. Θα εξετάσουμε, δηλαδή, αισθητήρες αντίληψης άλλων αντικειμένων και θα δούμε μια σειρά νέων ενεργειών που μπορούν να κάνουν τα αντικείμενα. Ακόμη, θα εμβαθύνουμε περισσότερο στην έννοια του αντικειμένου, μαθαίνοντας πώς μπορούμε να επιφέρουμε αλλαγές σε αυτό χρησιμοποιώντας το μενού *Αλλαγή Ιδιοτήτων (Change Settings).*

6.1 Αντικείμενα

Οτιδήποτε εισάγουμε στον κόσμο ενός παιχνιδιού μέσω του *Εργαλείου Αντικειμένων (Object Tool)* , μπορούμε και να το προγραμματίσουμε. Αυτό μπορούμε να το επιβεβαιώσουμε, αν κάνουμε δεξί κλικ πάνω σε κάποιο αντικείμενο και δούμε την επιλογή ***Προγραμμάτισε (Program)***. Ήρθε λοιπόν η ώρα να γνωρίσουμε την οικογένεια των αντικειμένων του MSKodu που είναι πιο πλούσια από όσο φαντάζεστε!

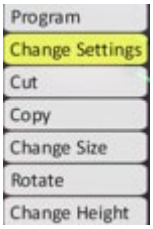
	*Kodu*: Ο βασικός ήρωας, για αυτό και το περιβάλλον δημιουργίας παιχνιδιών έχει πάρει το όνομά του! Μπορεί να χρησιμοποιηθεί για οποιοδήποτε παιχνίδι και αντιμετωπίζει όλες τις καταστάσεις.		*Μηχανάκιας (Cycle)*: Είναι πολύ γρήγορος, μπορεί να σκαρφαλώνει εύκολα σε απόκρημνους λόφους και να κάνει άλματα! Ότι πρέπει δηλαδή για αγώνες δρόμου και παιχνίδια καταδίωξης.
	*Τζετ (Jet)*: Σε φυσιολογικές συνθήκες κινείται κοντά στο έδαφος, όμως μπορεί να προγραμματιστεί να πηγαίνει πάνω και κάτω! Ιδανικό για περιπέτειες στον αέρα αλλά και αερομαχίες!		*Χελιδονόψαρο (Fly Fish)*: Μπορεί να αιωρείται και να κάνει γρήγορες κινήσεις! Ιδανικό όταν ο χρόνος είναι σημαντικός για το παιχνίδι σας ή όταν θέλετε να αποφύγετε τον εχθρό!
	*Πιατάκι (Saucer)*: Το πιο γρήγορο αντικείμενο! Μπορεί επίσης να αλλάξει κατεύθυνση αμέσως! Πολύ καλό για αγώνες ταχύτητας!		*Πλοίο (Ship)*: Μπορεί να επιπλέει πάνω στο νερό, αλλά όχι να κινείται στη στεριά. Μπορεί να είναι πρωταγωνιστής στη μεγαλύτερη ναυμαχία που έγινε ποτέ!
	*Υποβρύχιο (Sub)*: Όπως και τα υποβρύχια του πραγματικού κόσμου, έτσι και αυτά λειτουργούν καλύτερα κάτω από το νερό! Αντάξιος αντίπαλος αλλά και ύπουλος εχθρός των πλοίων.		*Κανόνι (Cannon)*: Είναι μεγάλο και αργό αλλά πολύ δυνατό! Ότι πρέπει για να κατατροπώσετε τον εχθρό με ένα τόσο δυνατό όπλο! Ή για να κατατροπώσετε το χρήστη!
	*Αερόπλοιο (Blimp)*: Πετάει τριγύρω με αργούς ρυθμούς. Σημαντικό για αερομαχίες!		*Λάμψη (Wisp)*: Κινείται γρήγορα αφήνοντας ίχνη λάμψης στο πέρασμά της! Για εντυπωσιασμό του αντιπάλου και του χρήστη!
	*Μπαλόνι (Balloon)*: Το Μπαλόνι πετάει αργά στον αέρα και παρατηρεί τα πάντα από ψηλά. Να, ο καινούργιος σας διαιτητής!		*Ντραμ (Drum)*: Όταν κάποιος χαρακτήρας πηδήξει πάνω στο ντραμ, εκτοξεύεται στον αέρα! Κάτι σαν τραμπολίνο!

	*Φως (Light)*: Κινείται γρήγορα και μπορεί να φωτίσει τον κόσμο. Ιδανικός σύντροφος του ήρωα σε μια σκοτεινή σπηλιά!		*Ψάρι*: Λειτουργεί καλύτερα στο νερό. Στη στεριά απλώς είναι ακινητοποιημένο! Απαραίτητο, όπως και το υποβρύχιο, σε έναν υποθαλάσσιο κόσμο.
	*Χελώνα (Turtle)*: Μπορεί να πετάει και να μπαινοβγαίνει στο καβούκι της! Σκεφτείτε όμως ότι όταν μπαίνει στο καβούκι της γίνεται αόρατη από τους άλλους!		*Σύννεφο (Cloud)*: Χαρακτηριστικό αντικείμενο του περιβάλλοντος όταν ο καιρός δεν είναι καλός!
	*Κάστρο (Castle)*: Ένα επιβλητικό κάστρο που δεν κινείται όμως! Μπορεί να συνδυαστεί με ένα κανόνι για την προστασία του από τους εχθρούς του.		*Πυρομαχικά (Ammo)*: Ιδανικά για παιχνίδια με μάχες! Πολύ ισχυρό για την κατατρόπωση του εχθρού!
	*Μήλο (Apple)*: Το γνωστό φρούτο, νόστιμο, δίνει πολλές φορές ενέργεια στο χρήστη αλλά ταυτόχρονα μπορεί να εκτοξεύεται!		*Νάρκη (Sine)*: Περιέχει καρφιά που μπορούν να είναι ενεργοποιημένα ή απενεργοποιημένα. Μπορεί να προκαλέσει έκρηξη!
	*Δίσκος Χόκεϋ (Cursor)*: Είναι ιδανικό για τα γρήγορα παιχνίδια! Πετάει χωρίς τριβή και αναπηδά χωρίς να χάνει ταχύτητα!		*Κουμπί Πίεσης (Pushpad)*: Ένα μεγάλο και δυνατό κουμπί!
	*Δορυφόρος (Sputnik)*: Κάνει πολύ καλή παρέα με το πιατάκι!		*Περισκόπιο (Stick)*: Όταν είναι απενεργοποιημένο, είναι αόρατο! Όταν ενεργοποιείται, εμφανίζεται! Ότι πρέπει για παιχνίδια κατασκοπίας!
	*Εργοστάσιο (Factory)*: Μπορεί να αποτελέσει ορόσημο στον κόσμο που θα δημιουργήσετε, μιας και είναι επιβλητικό οικοδόμημα! Δημιουργήστε αντιπάλους που βγαίνουν από το εργοστάσιο		*Κέρμα (Coin)*: Σημαντικό στον κόσμο του MSKodu όπως και στον πραγματικό κόσμο! Ειδικά για arcade παιχνίδια!
	*Καλύβα (Hut)*: Για τη διακόσμηση του κόσμου σας και όχι μόνο!		*Καρδιά (Heart)*: Σημαντική για να αυξήσετε ίσως την ενέργειά σας. Μπορείτε όμως και να την δώσετε σε κάποιο άλλο αντικείμενο ως ένδειξη αγάπης!
	*Δέντρο Tree)*: Μπορεί να χρησιμοποιηθεί για τη διακόσμηση του κόσμου σας, αλλά σε ορισμένες περιπτώσεις μπορεί να εκτελεί και ενέργειες. Προσοχή όμως, γιατί δεν έχει ικανότητα κίνησης!		*Βράχος (Rock)*: Και ναι, στο MSKodu ακόμα και οι βράχοι μπορούν να προγραμματιστούν!
	*Αστέρι (Star)*: Ακόμα και αυτό προγραμματίζεται! Σκεφτείτε μια νύχτα πεφταστεριών!		*Μπάλα (Ball)*: Μπορείτε απλώς να την πετάξετε ή και να την προγραμματίσετε. Στο Kodu αρέσει και το ποδόσφαιρο!

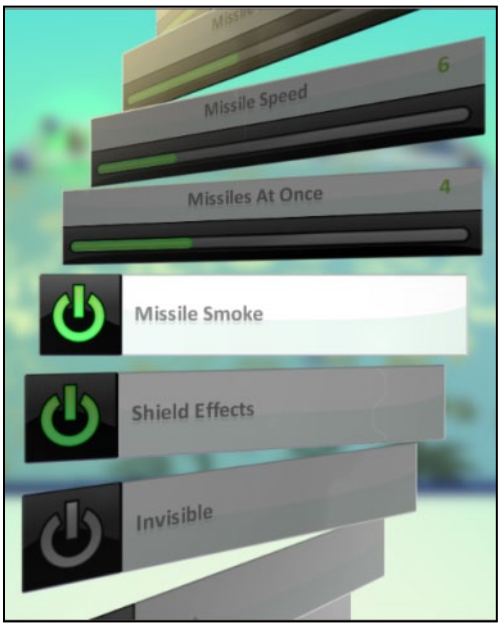
6.2 Ιδιότητες αντικειμένων

Όπως και τα αντικείμενα του πραγματικού κόσμου, έτσι και τα αντικείμενα του MSKodu έχουν ένα σύνολο από ιδιότητες, που καθορίζουν ως ένα βαθμό τη συμπεριφορά τους. Ένα αυτοκίνητο έχει ως ιδιότητα τη μέγιστη ταχύτητά του, και αντίστοιχα ένας ήρωας έχει ως ιδιότητα την ταχύτητά του ή τη συχνότητα εκτόξευσης πυραύλων (πόσο γρήγορα δηλαδή εκτοξεύει πυραύλους). Το ενδιαφέρον είναι ότι το MSKodu μας δίνει τη δυνατότητα να αλλάξουμε τις ιδιότητες αυτές! Δηλαδή μπορούμε να κάνουμε τον Kodu να κινείται πιο γρήγορα ή να εκτοξεύει 1 πύραυλο κάθε 1 δευτερόλεπτο! Τις ιδιότητες των αντικείμενων δεν μπορούμε να τις ελέγξουμε προγραμματιστικά αλλά πρέπει να τις ρυθμίσουμε πριν την έναρξη του παιχνιδιού.

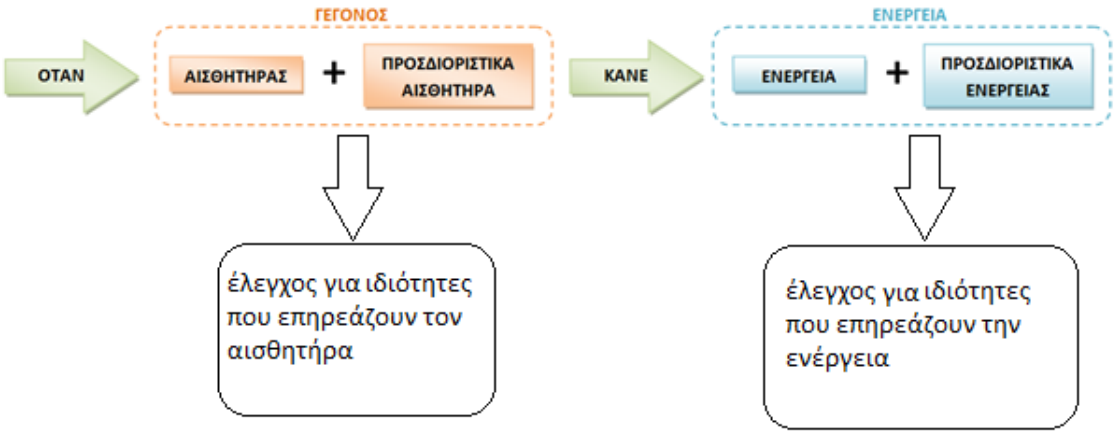
Το MSKodu παρέχει ένα μεγάλο πλήθος ιδιοτήτων, τις οποίες μπορούμε να ρυθμίσουμε με δεξί κλικ του ποντικιού πάνω στο αντικείμενο που μας ενδιαφέρει και επιλέγοντας ***Αλλαγή Ρυθμίσεων (Change Settings)***.



Ξαφνικά εμφανίζεται μπροστά σας μία σκάλα με όλες τις διαθέσιμες ιδιότητες του αντικειμένου! Χρησιμοποιείστε τη ροδέλα του ποντικιού για να περιηγηθείτε σε αυτές!



Θα πρέπει να γνωρίζουμε ότι για τους περισσότερους αισθητήρες και τις περισσότερες ενέργειες υπάρχουν ιδιότητες που επηρεάζουν τον τρόπο λειτουργίας τους.



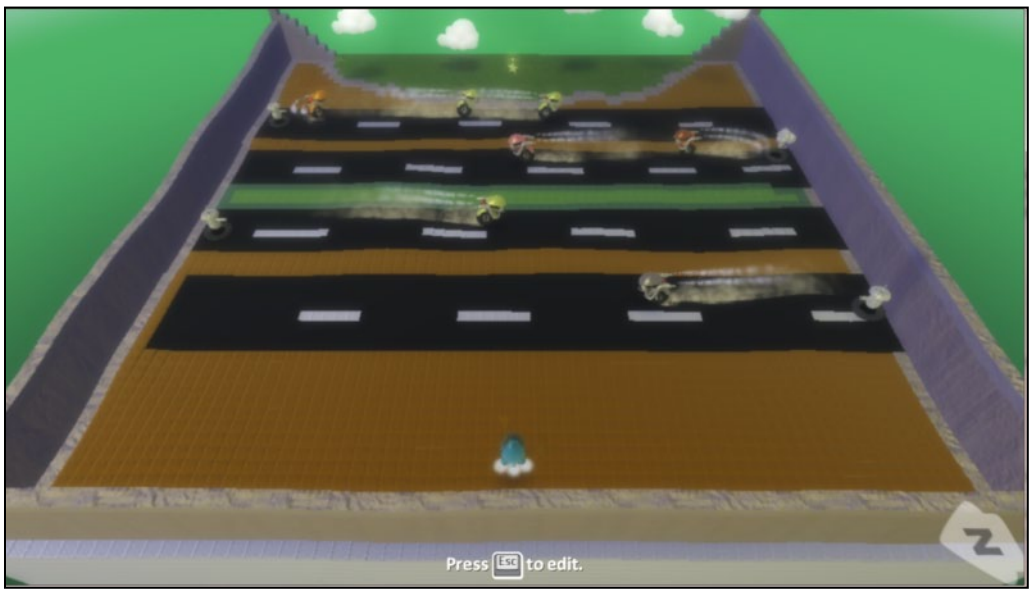
Θα αναρωτιέστε βέβαια αν κάθε φορά που δημιουργούμε ένα παιχνίδι θα πρέπει να ρυθμίζουμε όλες αυτές τις ιδιότητες. Η απάντηση είναι πως όχι! Δεν χρειάζεται! Όλες οι ιδιότητες έχουν ρυθμιστεί εκ των προτέρων, δηλαδή έχουν μία καθορισμένη αρχική τιμή. Μόνο στην περίπτωση που θέλετε να τις αλλάξετε πρέπει να επισκέπτεστε τη συγκεκριμένη καρτέλα. Ας ρίξουμε μια πιο αναλυτική ματιά στις διαθέσιμες ιδιότητες.

6.2.1 Ιδιότητες Κίνησης Αντικειμένων

Ο *Kodu*, όπως και κάθε άλλο αντικείμενο που μπορεί να κινηθεί, έχει μία σειρά από ιδιότητες που αφορούν την κίνησή του, όπως η ταχύτητα που αναπτύσσει, η επιτάχυνση κτλ.

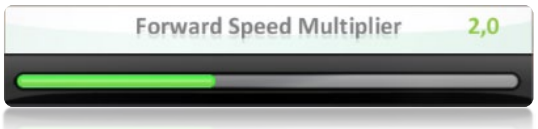
Δείτε το παράδειγμα *06_01.kodu*

Φορτώστε το παιχνίδι **[06_01.kodu]**. Έχει παρουσιαστεί στο δεύτερο κεφάλαιο και μπορεί να χαρακτηριστεί ως παιχνίδι επιβίωσης. Υπενθυμίζουμε ότι ο *Kodu* θα πρέπει να διασχίσει τέσσερις δρόμους για να περάσει στην απέναντι πλευρά και να ακουμπήσει ένα *Αστέρι (Star)* προκειμένου ο παίκτης να κερδίσει. Οι δρόμοι όμως έχουν κυκλοφοριακή συμφόρηση από *Μηχανάκια (Cycles)* τα οποία, αν ακουμπήσουν τον *Kodu*, το σκοτώνουν.



Στόχος μας είναι να ρυθμίσουμε τις ιδιότητες που αφορούν την κίνηση του *Kodu* έτσι ώστε να κάνουμε εύκολο το χειρισμό του *Kodu* για τον παίκτη, αλλά και ταυτόχρονα να αυξήσουμε λίγο τη δυσκολία του παιχνιδιού. Ας ρίξουμε λοιπόν μια ματιά στις ιδιότητες που μπορούμε να ρυθμίσουμε.

Ταχύτητα Κίνησης



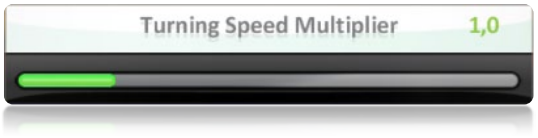
Η ιδιότητα αυτή ρυθμίζει την ταχύτητα με την οποία κινείται ένα αντικείμενο. Παίρνει τιμές από 0.1 έως 5.0 . Αν αυξήσουμε την ταχύτητα του *Kodu*, τότε φυσικά θα κινείται πιο γρήγορα. Αυτό με

μία πρώτη ματιά μπορεί να φαίνεται ότι διευκολύνει τον παίκτη, καθώς θα μπορεί να περάσει πιο γρήγορα από τον ένα δρόμο στον άλλο. Όμως, όπως συμβαίνει και στην πραγματικότητα, όταν ένα αντικείμενο αναπτύσσει μεγάλη ταχύτητα, τότε σταματάει και πιο δύσκολα!! Έτσι λοιπόν, μπορεί ο παίκτης να προλάβει να περάσει από τον ένα δρόμο στον άλλο, όμως υπάρχει περίπτωση να μην προλάβει να σταματήσει έγκαιρα και να μπει στον επόμενο δρόμο, αυξάνοντας έτσι την πιθανότητα να χτυπήσει με κάποιον *Μηχανάκια (Cycle)*. Καλό θα ήταν λοιπόν να δώσετε στην ιδιότητα αυτή την τιμή 3. Σε ένα παιχνίδι ράλι αυτή η τιμή είναι προτιμότερο να έχει τη μέγιστη δυνατή τιμή.

Επιτάχυνση Κίνησης

Όπως γνωρίζετε, όταν ένα αυτοκίνητο είναι σταματημένο (αρχική κατάσταση) και ξεκινάει να κινείται, χρειάζεται κάποια ώρα για να φτάσει στην τελική του ταχύτητα. Έτσι λοιπόν, η ιδιότητα αυτή ρυθμίζει το πόσο γρήγορα αυξάνει η ταχύτητα κίνησης του *Kodu*. Οι τιμές που μπορεί να πάρει η ιδιότητα κυμαίνονται από 0,1 έως 5.0 και είναι εμφανές ότι η ιδιότητα αυτή αποτελεί σημαντικό συμπλήρωμα της προηγούμενης. Όσο πιο γρήγορα αυξάνεται η ταχύτητα του *Kodu*, τόσο πιο σύντομα θα φτάσει στην τελική της τιμή που έχει καθοριστεί από την προηγούμενη ιδιότητα. Για άμεσες αντιδράσεις, θα μπορούσατε να ρυθμίσετε την ιδιότητα αυτή στην τιμή 5.

Ταχύτητα Περιστροφής



Οποιαδήποτε αλλαγή στην ιδιότητα αυτή θα αλλάξει και την ταχύτητα με την οποία ο *Kodu* στρίβει για να αλλάξει κατεύθυνση. Και αυτή η ιδιότητα παίρνει τιμές από 0,1 έως 5,0. Επειδή στο παιχνίδι βολεύει περισσότερο ο παίκτης να κινείται ευθύγραμμα για να εξοικονομεί χρόνο, η ιδιότητα αυτή δεν έχει μεγάλη σημασία. Μπορούμε να την αφήσουμε ως έχει. Αν όμως ρυθμίζαμε την ιδιότητα αυτή σε ένα παιχνίδι ράλι ή κυνηγητού, θα μπορούσαμε να της δώσουμε μεγαλύτερη τιμή.

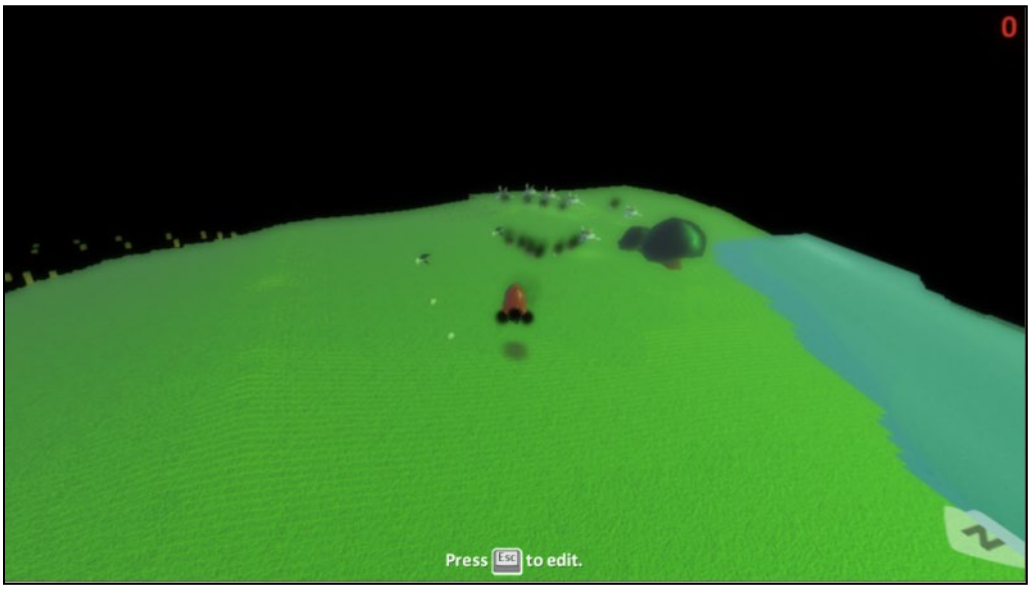
Επιτάχυνση Περιστροφής

Είναι αντίστοιχη ιδιότητα με αυτήν της επιτάχυνσης κίνησης και εξίσου σημαντική για την περιστροφή του *Kodu*. Προσδιορίζει πόσο γρήγορα θα φτάσει ο *Kodu* τη μέγιστη τιμή της ταχύτητας περιστροφής του. Παίρνει τιμές από 0,1 έως 5,0. Μπορούμε να την αφήσουμε ως έχει.

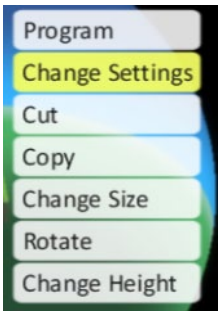
6.2.2 Ιδιότητες Μάχης

Δείτε το παράδειγμα *06_02.kodu*

Οι ιδιότητες μάχης αφορούν κυρίως παιχνίδια δράσης και μάχης. Φορτώστε το επόμενο παιχνίδι **[06_02.kodu]**.

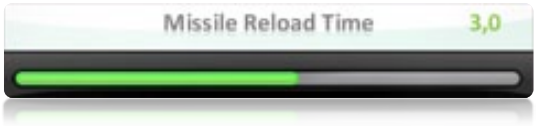


Ο κόσμος του παιχνιδιού αποτελείται από δέκα *Χελώνες (Turtles)*, δύο *Νομίσματα (Coins)*, *Ψάρια (Fish)* και φυσικά τον ήρωα του παίκτη, τον *Kodu*. Σκοπός του παιχνιδιού είναι ο *Kodu* να καταστρέψει όλες τις *Χελώνες (Turtles)* εκτοξεύοντας *Πυραύλους (Missiles)* και στη συνέχεια να φάει τα *Ψάρια (Fish)* που υπάρχουν ανενόχλητος. Ο *Kodu* είναι νικητής μόνο όταν δεν υπάρχει καμία *Χελώνα (Turtle)* και κανένα *Ψάρι (Fish) στην πίστα*. Τα πράγματα όμως δεν είναι τόσο εύκολα όσο νομίζετε! Οι *Χελώνες (Turtles)*, τρεις εκ των οποίων είναι τεράστιες, εκτοξεύουν συνεχώς *Πυραύλους (Missiles)*! Ο παίκτης μπορεί να κινεί τον ήρωα του με τα βέλη του πληκτρολογίου και μπορεί επίσης να εκτοξεύει πυραύλους πατώντας το πλήκτρο *Κενό (Space)*. Οι συμπεριφορές των αντικειμένων είναι ήδη προγραμματισμένες από κάποιον άλλον προγραμματιστή. Ωστόσο, ως προγραμματιστές καλό θα ήταν να σκεφτούμε ποιες ιδιότητες των συγκεκριμένων αντικειμένων πρέπει να ρυθμίσουμε έτσι ώστε να ολοκληρώσουμε το παιχνίδι δράσης. Στο μεγαλύτερο κομμάτι του παιχνιδιού, ο παίκτης ανταλλάσσει πυροβολισμούς με τις *Χελώνες (Turtles)*. Συνεπώς θα πρέπει να ρυθμίσουμε τις ιδιότητες που αφορούν την εκτόξευση πυραύλων τόσο του *Kodu* όσο και των *Χελωνών (Turtles)*.



Πατήστε το κουμπί *Escape* στο πληκτρολόγιό σας και στη συνέχεια επιλέξτε την ***Αλλαγή Ιδιοτήτων (Change Settings)*** για τον *Kodu*. Μην ξεχνάμε ότι οι ιδιότητες που επιθυμούμε να αλλάξουμε αφορούν τους *Πυραύλους*. Χρησιμοποιώντας τη ροδέλα του ποντικιού μπορούμε να διακρίνουμε τις εξής ιδιότητες που αφορούν την εκτόξευση *Πυραύλων* :

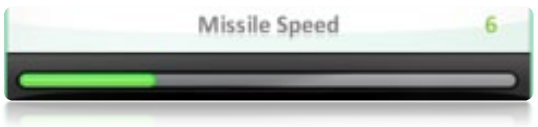
Χρονικό Διάστημα Μεταξύ Εκτοξεύσεων Πυραύλων



Με την ιδιότητα αυτή ρυθμίζουμε μετά από πόση ώρα μπορεί ο *Kodu* να ξαναρίξει έναν πύραυλο. Ο χρόνος μετριέται σε δευτερόλεπτα και μπορεί να πάρει τιμές από 0,5 έως 5 δευτερόλεπτα. Αν ο *Kodu* εκτοξεύει πολύ αργά πυραύλους (έχουμε δώσει μεγάλη τιμή στην ιδιότητα) τότε δεν θα προλάβει να καταστρέψει όλες τις *Χελώνες (Turtles)* χωρίς να τον χτυπήσουν.

Από την άλλη, αν δώσουμε πολύ χαμηλή τιμή στην ιδιότητα αυτή, ο *Kodu* θα εκτοξεύει πολύ γρήγορα πυραύλους και το παιχνίδι θα είναι πολύ εύκολο. Ας την ρυθμίσουμε λοιπόν στα 1,5 δευτερόλεπτα.

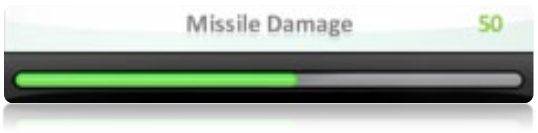
Ταχύτητα Πυραύλου



Με την *Ταχύτητα Πυραύλου (Missile Speed)* μπορούμε να ρυθμίσουμε την ταχύτητα του πυραύλου που εκτοξεύεται. Η ιδιότητα αυτή παίρνει τιμές από 1 έως 20. Όσο μεγαλύτερη η τιμή,

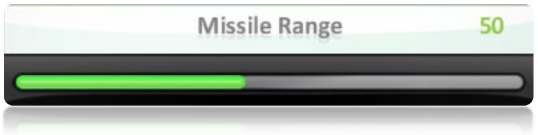
τόσο γρηγορότερα ταξιδεύει ένας πύραυλος και τόσο συντομότερα χτυπάει μία *Χελώνα (Turtle)*. Στο παιχνίδι αυτό, επειδή οι *Χελώνες (Turtles)* είναι πάρα πολλές και εκτοξεύουν συνεχώς πυραύλους εναντίον του *Kodu*, θα πρέπει να δώσουμε στην ιδιότητα αυτή μία μεγάλη τιμή. Με το ίδιο σκεπτικό θα πρέπει να ρυθμίσουμε την ιδιότητα αυτή και για τις *Χελώνες (Turtles)*. Όμως, προκειμένου να δημιουργήσουμε ένα δίκαιο παιχνίδι, δεν θα πρέπει να δώσουμε πολύ μεγάλη ταχύτητα στους πυραύλους των *Χελωνών (Turtles)*. Σκεφτείτε ότι η ιδιότητα αυτή είναι πολύ σημαντική σε περίπτωση που ο αντίπαλός του παίκτη κινείται. Για να έχει μεγαλύτερες πιθανότητες ο παίκτης να πετύχει τον αντίπαλό του, θα πρέπει να δώσουμε στην ιδιότητα αυτή όσο το δυνατόν μεγαλύτερη τιμή.

Ζημιά που προκαλεί ένας πύραυλος



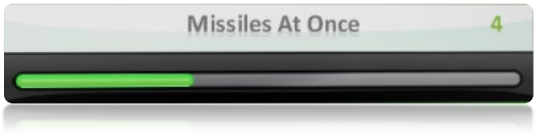
Η ιδιότητα αυτή καθορίζει πόση απώλεια ενέργειας προκαλεί ένας πύραυλος όταν χτυπήσει ένα αντικείμενο. Παίρνει τιμές από το -500 έως το 500. Αποτελεί σημαντικό παράγοντα για την εξέλιξη του παιχνιδιού. Ανάλογα με το πόσο επιθυμούμε να δυσκολέψουμε τους παίκτες, ρυθμίζουμε την τιμή της ιδιότητας αυτής. Όσο χαμηλότερη είναι η τιμή της, τόσο πιο δύσκολα καταστρέφεται το αντίπαλο αντικείμενο!

Εύρος Κίνησης Πυραύλου



Καθορίζει την απόσταση που μπορεί να διανύσει ένας πύραυλος, πριν εξαφανιστεί και εφόσον δεν χτυπήσει κάποιο αντικείμενο. Οι τιμές που παίρνει είναι μεταξύ του 10 και του 100. Στο παιχνίδι αυτό ο παίκτης θα πρέπει να κινεί τον ήρωα του συνέχεια για να αποφύγει τους πυραύλους. Υπάρχει επομένως μεγάλη πιθανότητα να απομακρυνθεί ο *Kodu* πολύ από τις χελώνες. Εάν δώσουμε μικρή τιμή στην ιδιότητα για τους πυραύλους που εκτοξεύει ο *Kodu*, τότε υπάρχει πιθανότητα οι πύραυλοι σε κάποιες περιπτώσεις να μη φτάνουν τις χελώνες! Εμείς ως προγραμματιστές του παιχνιδιού θα δώσουμε μία μεγάλη τιμή ώστε να κάνουμε το παιχνίδι ευκολότερο.

Αριθμός Εκτοξευμένων Πυραύλων



Όπως έχετε παρατηρήσει, ο πύραυλος που εκτοξεύει ο *Kodu*, ταξιδεύει στον αέρα αν δεν έχει χτυπήσει κάποιο αντικείμενο και αν δεν έχει ξεπεράσει το *Εύρος Κίνησής (Missile Range)* του. Με την ιδιότητα αυτή, μπορούμε να ρυθμίσουμε πόσοι πύραυλοι που έχουν εκτοξευτεί από το ίδιο αντικείμενό μπορούν να ταξιδεύουν ταυτόχρονα στον αέρα. Εμμέσως, δηλαδή, προσδιορίζουμε το πότε ο Kodu να μπορεί να πυροβολήσει ξανά. Η ιδιότητα παίρνει τιμές από το 1 έως και το 10. Αν η τιμή είναι 5, ο Kodu μπορεί να εκτοξεύσει μέχρι και 5 πυραύλους που βρίσκονται ταυτόχρονα στον αέρα. Αν πετύχει το στόχο του ή εξαφανιστεί έστω και ένας από αυτούς τους πυραύλους, τότε και μόνο τότε ο Kodu θα μπορέσει να εκτοξεύσει πάλι πυραύλους μέχρι να φτάσει πάλι το μέγιστο των 5 πυραύλων στον αέρα.

Η ιδιότητα αυτή εξαρτάται από την *Ταχύτητα Πυραύλου (Missile Speed)*, το *Χρονικό Διάστημα Μεταξύ Εκτοξεύσεων (Missile Reload Time)* καθώς και το *Εύρος Κίνησης (Missile Range)* του πυραύλου. Αν ο ήρωας του παίκτη εκτοξεύει πολύ συχνά πυραύλους που κινούνται πολύ γρήγορα και οι εχθροί βρίσκονται πολύ μακριά τότε καλό θα ήταν να δώσουμε στην ιδιότητα αυτή μία μεγάλη τιμή

Υπάρχουν αντίστοιχες ιδιότητες για την περίπτωση στην οποία ο Kodu εκτοξεύει *Σφαιρίδιο (Blip)* και ρυθμίζονται όπως και οι παραπάνω

Ας τρέξουμε τώρα το παιχνίδι! Καλό θα ήταν να δοκιμάσουμε τις αλλαγές που έχουμε κάνει στις ιδιότητες των αντικειμένων για να διαπιστώσουμε αν έχουμε δώσει στο παιχνίδι τον κατάλληλο βαθμό δυσκολίας. Αποθηκεύστε λοιπόν το παιχνίδι και δοκιμάστε το.

6.2.3 Ιδιότητες αλληλεπίδρασης με το περιβάλλον

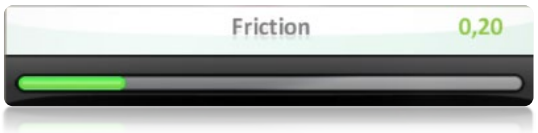
Ένα από τα βασικά στοιχεία σε όλα τα παιχνίδια είναι η αλληλεπίδραση του Kodu, τόσο με την πίστα του παιχνιδιού όσο και με τα υπόλοιπα αντικείμενα.

Δείτε το παράδειγμα *06_03.kodu*

Φορτώστε το παιχνίδι **[06_03.kodu]**. Είναι ένα παιχνίδι αγώνα δρόμου. Ο παίκτης, χρησιμοποιώντας το πληκτρολόγιο, κινεί τον μπλε *Μηχανάκια (Cycle)* προκειμένου να τον οδηγήσει στη γραμμή τερματισμού.

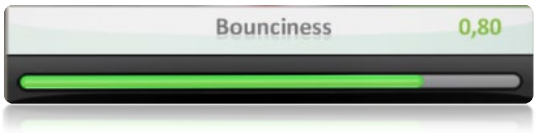
Στόχος μας είναι να κάνουμε το παιχνίδι όσο πιο ρεαλιστικό γίνεται, ρυθμίζοντας κάποιες από τις ιδιότητες των αντικειμένων που αφορούν την αλληλεπίδρασή τους με το περιβάλλον. Για να δούμε λοιπόν δύο χαρακτηριστικές ιδιότητες που αφορούν την κίνηση του *Μηχανάκια (Cycle)*.

Τριβή



Ώρα για ένα γρήγορο μάθημα Φυσικής! Όπως μπορεί να γνωρίζετε ήδη, όταν ένα αντικείμενο κινείται πάνω σε οποιαδήποτε επιφάνεια, αναπτύσσεται τριβή μεταξύ τους. Η τριβή είναι μία δύναμη η οποία εμποδίζει την κίνηση του αντικειμένου, με αποτέλεσμα το αντικείμενο να αναπτύσσει ταχύτητα με πιο αργό ρυθμό. Ταυτόχρονα, όμως η τριβή βοηθάει το αντικείμενο να σταματά και πιο γρήγορα. Στο MSKodu η τριβή παίρνει τιμές από 0.00 έως 1.0, όπου το μηδέν σημαίνει ότι δεν υπάρχει καθόλου τριβή και 1 είναι η μέγιστη τιμή τριβής που μπορεί να υπάρξει. Δοκιμάστε το παιχνίδι μας για διαφορετικές τιμές της τριβής.

Αναπήδηση



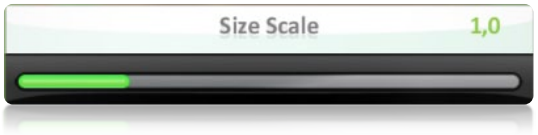
Στο παιχνίδι που εξετάζουμε υπάρχει περίπτωση ο μπλε *Μηχανάκιας (Cycle)* του παίκτη να συγκρουστεί με κάποιον άλλο *Μηχανάκιας (Cycle)*. Η ιδιότητα αυτή καθορίζει κατά πόσο ένα αντικείμενο αναπηδά όταν συγκρουστεί με κάποιο άλλο αντικείμενο, με έναν τοίχο ή μετά την προσγείωση από ένα άλμα. Θυμηθείτε πραγματικούς αγώνες ράλι που μπορεί να έχετε παρακολουθήσει. Δεν είναι πολύ τρομακτικό αυτό που συμβαίνει όταν συγκρουστούν κάποια αμάξια μεταξύ τους; Όπως και η *Τριβή (Friction),* έτσι και η *Αναπήδηση (Bounciness)* παίρνει τιμές

από το 0.00 έως το 1.00. Αν ρυθμίσουμε την ταχύτητα από τα *Μηχανάκια (Cycles)* να έχει τη μεγαλύτερη τιμή, τότε η Φυσική επιβάλλει να ρυθμίσουμε την αναπήδηση στη μέγιστη τιμή της.

6.2.4 Καλαισθητικές ιδιότητες

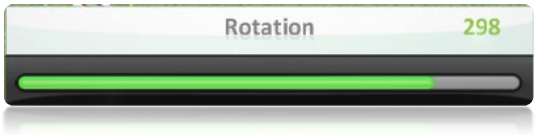
Οι ιδιότητες αυτές επηρεάζουν την εμφάνιση των αντικειμένων σας και δεν επηρεάζουν άμεσα την εξέλιξη του παιχνιδιού. Σχετίζονται μόνο με την ευχαρίστηση του παίκτη όταν βλέπει στην οθόνη τα αντικείμενα ενός παιχνιδιού. Ας εξετάσουμε μερικές από αυτές και τη λειτουργία τους.

Μέγεθος αντικειμένου



Δεν χρειάζονται ιδιαίτερες συστάσεις για αυτή την ιδιότητα. Αφορά το μέγεθος ενός αντικειμένου. Φανταστείτε ότι έχετε εισάγει ένα αντικείμενο, για παράδειγμα έναν μαύρο *Kodu* ο οποίος θα είναι ο τελικός εχθρός μίας πίστας. Προκειμένου να τον κάνετε να φαίνεται πιο απειλητικός θα μπορούσατε να αυξήσετε το μέγεθός του. Η ιδιότητα αυτή παίρνει τιμές από 0.2 έως 4.0.

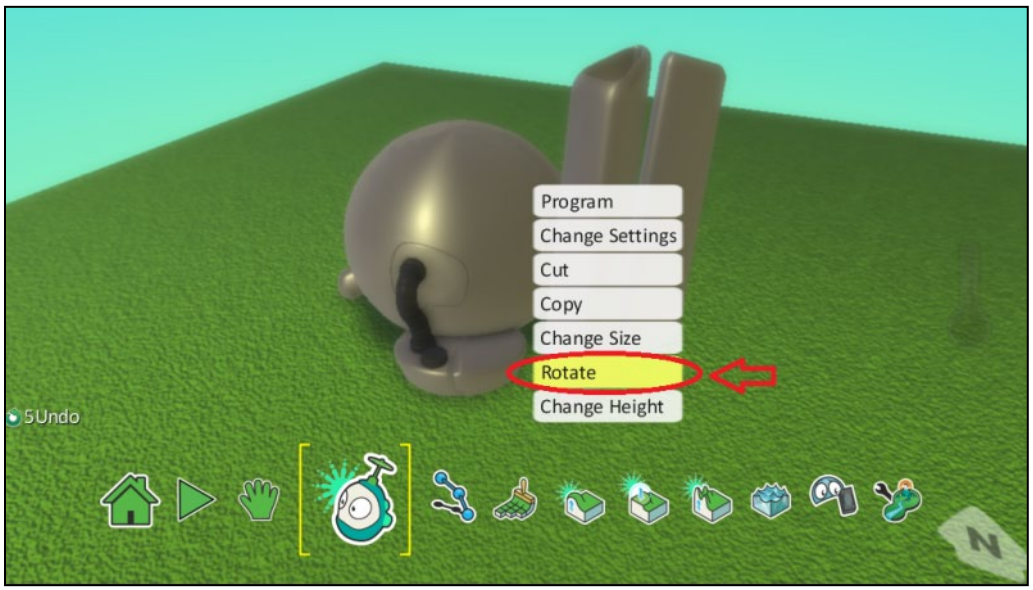
Περιστροφή αντικειμένου



Ρυθμίζοντας αυτή την ιδιότητα καθορίζουμε ποιος θα είναι ο αρχικός προσανατολισμός ενός αντικειμένου. Η ιδιότητα αυτή παίρνει τιμές από το 0 έως και το 360. Για να περιστρέψουμε ένα αντικείμενο ακολουθούμε την εξής διαδικασία.

1. Επιλέγουμε το αντικείμενο που θέλουμε να περιστρέψουμε και πατάμε δεξί κλικ με το ποντίκι.

2. Πατάμε πάνω στην επιλογή *Περιστρέφω (Rotate).*



3. Καθώς μεταβάλουμε την τιμή της ιδιότητας, παρατηρούμε ότι το αντικείμενο περιστρέφεται. Σταματάμε να περιστρέφουμε το αντικείμενο όταν αυτό έχει αποκτήσει τον προσανατολισμό που επιθυμούμε.

Για τον ήρωα του παίκτη η ιδιότητα αυτή έχει σημασία μόνο κατά την εκκίνηση του παιχνιδιού, διότι έτσι κι αλλιώς στη συνέχεια ο παίκτης θα τον κινήσει και ο αρχικός προσανατολισμός θα χαθεί. Έτσι, για παράδειγμα, μπορούμε να ρυθμίσουμε τον ήρωα του παίκτη να κοιτάζει αρχικά προς το μέρος του για να αυξήσουμε τον ενθουσιασμό του. Η ιδιότητα αυτή όμως έχει πολύ μεγάλη σημασία για τα αντικείμενα που δεν κινούνται. Φανταστείτε ότι εισάγετε σε έναν κόσμο ένα *Εργοστάσιο (Factory)* που παράγει σατανικά *Kodu* με τέτοιο τρόπο ώστε ο παίκτης να μην μπορεί να δει πότε βγαίνουν από την πόρτα του οι εχθροί! Αυτό θα δυσκόλευε την ζωή του παίκτη αλλά θα ήταν και αντιαισθητικό για την πίστα.

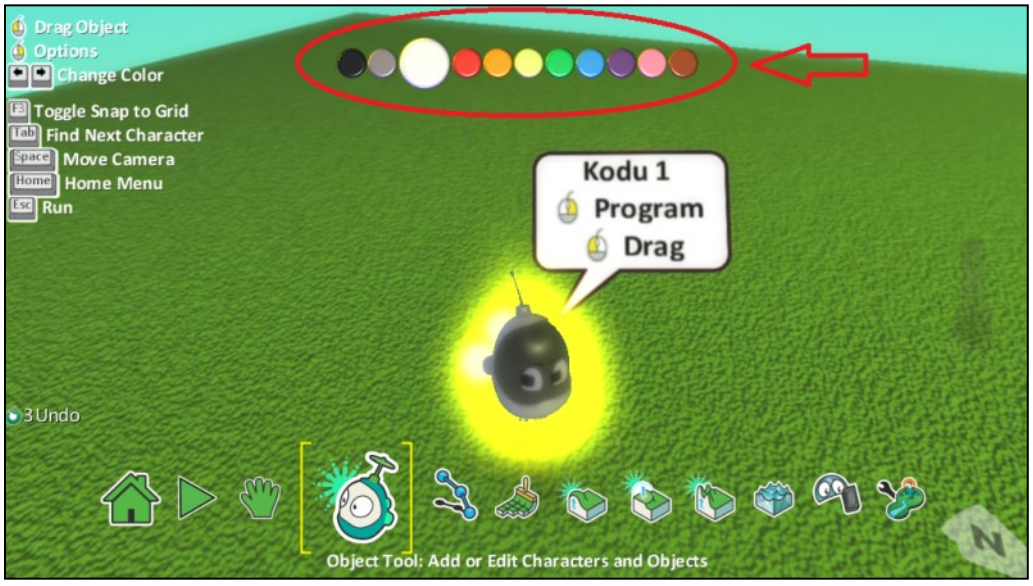
Χρώμα

Όπως όταν παίζουμε ένα επιτραπέζιο παιχνίδι διαλέγουμε το πιόνι με το χρώμα που μας αντιπροσωπεύει περισσότερο, έτσι και στο παιχνίδι που θα δημιουργήσουμε θέλουμε τα αντικείμενά μας να έχουν διαφορετικά χρώματα, ώστε να μπορούμε να τα ξεχωρίζουμε. Σκεφτείτε το παιχνίδι **[06_03.kodu]** που είδαμε παραπάνω. Τα *Μηχανάκια (Cycles)* έχουν διαφορετικά χρώματα μεταξύ τους. Ο χαρακτήρας του παίκτη είναι μπλε ώστε να μπορεί ο παίκτης να τον διακρίνει ευκολότερα!

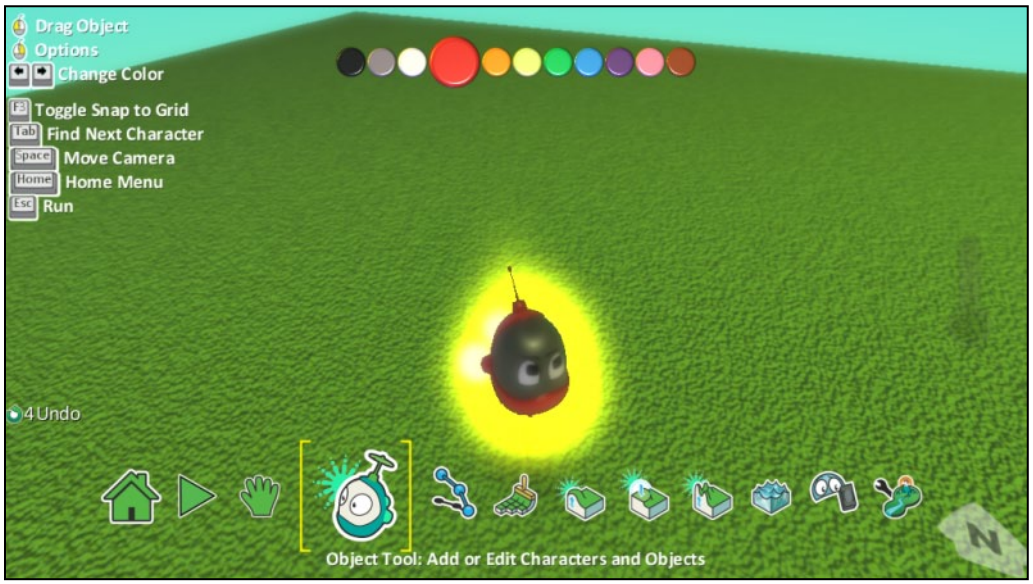
Για να μεταβάλλουμε το χρώμα ενός αντικειμένου, εκτελούμε την εξής διαδικασία:

1. Επιλέγουμε πρώτα το Εργαλείο Αντικειμένων (Object Tool).

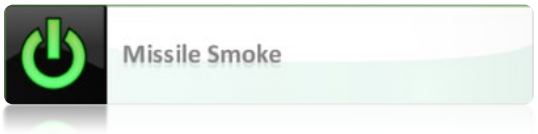
2. Μετακινούμε το δείκτη του ποντικιού πάνω στο αντικείμενο που επιθυμούμε να αλλάξουμε το χρώμα. Παρατηρούμε ότι στο πάνω μέρος της οθόνης μας εμφανίζεται μία παλέτα χρωμάτων.



3. Πατάμε τα βέλη του πληκτρολογίου αριστερά ή δεξιά για να επιλέξουμε το επιθυμητό χρώμα.



Εφέ Καπνού



Η ιδιότητα αυτή σχετίζεται με την αληθοφάνεια του παιχνιδιού. Αφορά το αν οι πύραυλοι αφήνουν ίχνος καπνού στο πέρασμά τους. Επιπλέον, όσοι από εσάς δεν έχουν καλή κάρτα γραφικών θα μπορούσατε να την απενεργοποιήσετε για να μην επιβαρύνετε επιπλέον το σύστημα σας. Παρατηρήστε ότι η εικόνα της ιδιότητας έχει το γνωστό σήμα του on/off . Όταν έχει έντονο πράσινο χρώμα τότε η ιδιότητα είναι ενεργοποιημένη.

Υπάρχουν αρκετές ακόμη ιδιότητες που δεν θα καλυφθούν στην παράγραφο αυτή. Σε κάθε περίπτωση, δίπλα από κάθε ιδιότητα υπάρχει η επιλογή βοήθειας, η οποία σας παρέχει μία περιγραφή της.

6.3 Οι αισθητήρες του MSKodu: κατανοώ τον κόσμο μου

Αν το καλοσκεφτούμε, οι αισθητήρες του MSKodu είναι κάτι αντίστοιχο με τις αισθήσεις που έχουν οι άνθρωποι. Οι άνθρωποι χρησιμοποιούν βασικές αισθήσεις, όπως η ακοή, η όραση κτλ. ώστε να αντιλαμβάνονται τα ερεθίσματα του περιβάλλοντός τους και να ενεργούν αναλόγως. Έτσι και τα αντικείμενα στο MSKodu! Χρησιμοποιούν αισθητήρες όπως το ***Ακούω (Hear)***, το ***Βλέπω (See)***, το ***Πέφτω Πάνω (Bump)***, ώστε να αντιλαμβάνονται με διάφορους τρόπους τα υπόλοιπα αντικείμενα.

6.3.1 Ο αισθητήρας Βλέπω (See)

Ο αισθητήρας ***Βλέπω (See)*** επιτρέπει στα αντικείμενα να αντιδρούν στην παρουσία άλλων αντικειμένων στο κόσμο των παιχνιδιών μας. Με απλά λόγια, αποτελεί τα μάτια των πρωταγωνιστών μας, που πλέον μπορούν να αναλάβουν δράση όταν *δουν* ένα φίλο τους, ένα *Νόμισμα (Coin)* ή κάποιον εχθρό. Θα μπορούσαμε λοιπόν να προγραμματίσουμε ένα *Υποβρύχιο (Sub)* έτσι ώστε, όταν *βλέπει* τον *Kodu*, να εκτοξεύει πυραύλους εναντίον του, ή να προγραμματίσουμε έναν *Μηχανάκια (Cycle)* ώστε, όταν *δει* τον *Kodu*, να προσπαθεί να τον πλησιάσει για να τον εξολοθρεύσει. Ο αισθητήρας ***Βλέπω (See)*** πάντα συνοδεύεται με προσδιοριστικό το αντικείμενο που θέλουμε να αναγνωρίσει ο χαρακτήρας μας. Ένα παράδειγμα συμπεριφοράς με χρήση του αισθητήρα:

Τι είχε ο προγραμματιστής στο μυαλό του;

ΌΤΑΝ[*Βλέπω*][τον *Kodu*] **ΤΟΤΕ** [*Κινούμαι*][*Προς* αυτόν]

Παρατηρήστε ότι το προσδιοριστικό ενέργειας ***Προς (Toward)*** εξαρτάται από το αντικείμενο που θα χρησιμοποιήσουμε ως προσδιοριστικό στον αισθητήρα της συγκεκριμένης συμπεριφοράς. Σε αυτήν την περίπτωση το προσδιοριστικό του αισθητήρα αναφέρεται στον Kodu.

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα:

{Βλέπω, εντοπίζω, αναγνωρίζω, διακρίνω, αντιλαμβάνομαι, συναντώ, ανιχνεύομαι}

Ας δούμε όμως τη χρήση του αισθητήρα ***Βλέπω (See)*** πρακτικά σε ένα παιχνίδι αερομαχίας. Στο περιβάλλον του παιχνιδιού μας, υπάρχουν ο ουρανός, *Σύννεφα (Clouds)* και τέσσερα *Τζετ (Jet)*, που τρία από αυτά είναι μαύρα και ένα είναι πράσινο. Το πράσινο *Τζετ (Jet)* είναι αυτό που χειρίζεται ο παίκτης. Ο σκοπός του παιχνιδιού είναι ο χειριστής του πράσινου *Τζετ (Jet)* να καταστρέψει τα μαύρα τζετ ενώ ταυτόχρονα προσπαθεί να αποφύγει τις βολές τους. Τα *Σύννεφα (Clouds)* υπάρχουν ως καλλωπιστικά στοιχεία του κόσμου και δεν παρουσιάζουν κάποια συμπεριφορά. Ο κόσμος λοιπόν του παιχνιδιού περιγράφεται με την εικόνα που ακολουθεί.

Δείτε το παράδειγμα *06_04.kodu*

Φορτώστε το παιχνίδι **[06_04.kodu]** και ας ξεκινήσουμε βήμα βήμα τον προγραμματισμό των αντικειμένων, προκειμένου να δημιουργήσουμε το παιχνίδι αερομαχίας.

Ας ξεκινήσουμε με τον προγραμματισμό του πράσινου *Τζετ (Jet)*, δηλαδή του αντικειμένου που θα χειρίζεται ο παίκτης. Το πράσινο *Τζετ (Jet)* θα πρέπει να αξιοποιήσει κάποιον αισθητήρα για να καταλαβαίνει το πότε ο χρήστης έχει πατήσει τα *Βέλη* ή το *Κενό Space* στο *Πληκτρολόγιο*. Ποιον αισθητήρα όμως; Προφανώς, θα χρησιμοποιήσουμε για μία φορά ακόμη τον αισθητήρα *Πληκτρολόγιο (Keyboard)* με προσδιοριστικό τα *Βέλη (Arrows)* για να μετακινεί ο χρήστης το *Τζετ (Jet)* και μια ακόμη φορά με προσδιοριστικό το πλήκτρο *Κενό (Space)* για να εκτοξεύει το πράσινο τζετ πυραύλους. Συνεπώς, το πράσινο *Τζετ (Jet)* εμφανίζει δύο συμπεριφορές.

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί κάποιο από τα Βέλη] **ΤΟΤΕ** [Κινήσου προς την αντίστοιχη κατεύθυνση][Γρήγορα]

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο Κενό] **ΤΟΤΕ** [Εκτόξευσε][Πύραυλο]

Ας προγραμματίσουμε τώρα τα μαύρα *Τζετ (Jet)*.

Κάθε φορά που ένα μαύρο Τζετ αναγνωρίζει ένα πράσινο Τζετ, τότε θα πρέπει να αρχίσει να το ακολουθεί και να εκτοξεύει πυραύλους εναντίον του. Πώς μπορούν να αντιληφθούν τα μαύρα Τζετ (Jet) το πράσινο; Αφού θα πρέπει πρώτα να το *δουν*, θα χρησιμοποιήσουμε τον αισθητήρα ***Βλέπω (See)***. Ο αισθητήρας αυτός θα πρέπει να έχει ως προσδιοριστικό το αντικείμενο *Τζετ*. Συνεπώς, για τη συμπεριφορά των μαύρων Τζετ, δημιουργούμε τις εξής εντολές:

ΌΤΑΝ[Δω][ένα Τζετ] **ΤΟΤΕ** [Κινούμαι][Προς αυτό]

ΌΤΑΝ[Δω][ένα Τζετ] **ΤΟΤΕ** [Εκτοξεύω][Πυραύλους]

Είναι όμως το παράδειγμά μας ολοκληρωμένο; Αν προγραμματίσουμε τον αισθητήρα ***Βλέπω (See)*** με προσδιοριστικό το αντικείμενο *Τζετ (Jet)*, τότε κάθε μαύρο Τζετ θα επιδεικνύει την ίδια συμπεριφορά όταν δει και όλα τα υπόλοιπα *Τζετ (Jet)*, είτε είναι μαύρα είτε πράσινα είτε κόκκινα! Θα αρχίσει να τα ακολουθεί ανεξαρτήτως χρώματος και θα πυροβολεί οποιοδήποτε είναι κοντά του! Μήπως θα έπρεπε να προσθέσουμε ένα επιπλέον προσδιοριστικό στον αισθητήρα ***Βλέπω (See)***, για παράδειγμα το χρώμα; Δεν επιθυμούμε τα μαύρα *Τζετ (Jet)* να αντιδρούν μόνο όταν βλέπουν το πράσινο τζετ; Βελτιώστε τις εντολές σας όπως φαίνεται στο παρακάτω σχήμα:

ΌΤΑΝ[Βλέπω][Πράσινο][Τζετ] **ΤΟΤΕ** [Κινούμαι][Προς αυτό]

ΌΤΑΝ[Βλέπω][Πράσινο][Τζετ] **ΤΟΤΕ** [Εκτοξεύω][Πυραύλους]

Αποθηκεύστε τις αλλαγές και τρέξτε το πρόγραμμα.

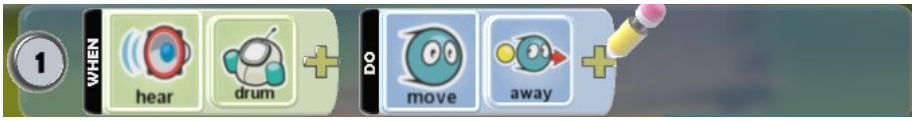
Το MSKodu μας προσφέρει μία πληθώρα προσδιοριστικών που μπορούμε να χρησιμοποιήσουμε σε συνδυασμό με τον αισθητήρα ***Βλέπω (See)***. Μπορούμε να ορίσουμε ως προσδιοριστικό οποιοδήποτε αντικείμενο του κόσμου μας, οποιοδήποτε χρώμα (π.χ. όταν το αντικείμενό μας βλέπει οτιδήποτε κόκκινο μπορεί να κινείται πιο αργά!), οποιαδήποτε έκφραση άλλου

αντικειμένου (όταν βλέπει π.χ. ένα θυμωμένο αντικείμενο μπορεί να το εξολοθρεύει!). Στις ***Επιλογές (Options)*** της πρώτης πίτας προσδιοριστικών, εμφανίζονται νέα προσδιοριστικά που έχουν να κάνουν με τη θέση του αντικειμένου σε σχέση με τα άλλα αντικείμενα του κόσμου. Έτσι αν επιθυμούμε ο *Kodu* να ενεργεί όταν βλέπει κάποια αντικείμενα που βρίσκονται σε μακρινή θέση, μπορούμε να χρησιμοποιήσουμε το προσδιοριστικό ***Μακρυά (Far away)***.

Το πόσο μακριά «βλέπει» το αντικείμενό σας, μπορείτε να το προσδιορίσετε από την ιδιότητα *Μακριά (Far away range)*.

6.3.2 Ο αισθητήρας Ακούω (Hear)

Ο αισθητήρας αυτός είναι στην ουσία τα αυτιά κάθε αντικειμένου. Μπορεί να σας φαίνεται όμως παράξενο που υπάρχει ο αισθητήρας αυτός. Τι μπορεί δηλαδή να ακούσει ένα αντικείμενο στον κόσμο του MSKodu; Μα φυσικά τα πάντα! Μπορούμε να προγραμματίσουμε τον *Kodu*, ώστε αν ακούσει κάποιο φίλο του, να τον ακολουθήσει! Ακόμη, μπορούμε να προγραμματίσουμε ένα *Μπαλόνι (Baloon)* να εκτοξεύει *Σφαιρίδια (Blips)* όταν ακούσει κοντά του κάποιο *Τζετ (Jet)*. Δεν πρέπει να ξεχνάμε ότι από τη στιγμή που θα εισάγουμε οποιοδήποτε αντικείμενο σε ένα παιχνίδι, αυτό παράγει αυτόματα ήχους. Ο αισθητήρας ***Ακούω (Hear)*** πάντα συνοδεύεται με προσδιοριστικό στοιχείο είτε το αντικείμενο που θέλουμε να ακούσει ο χαρακτήρας μας ή κάποιον ήχο από τους πολλούς που μας προσφέρει το περιβάλλον MSKodu. Ένα παράδειγμα εντολής με χρήση του αισθητήρα:



Ο προγραμματιστής που προγραμμάτισε την παραπάνω εντολή είχε στο μυαλό του:

ΌΤΑΝ[*Ακούσω*][κάποιο *Ντραμ*] **ΤΟΤΕ** [*Κινούμαι*][*Μακρυά* από αυτό]

Ρήματα συμπεριφορών που θα πρέπει να μας φέρουν στο μυαλό τον αισθητήρα ***Ακούω (Hear)*** είναι:

{Ακούω, αφουγκράζομαι, αντιλαμβάνομαι, ανιχνεύω, εντοπίζω}

Ας αξιοποιήσουμε λοιπόν τον αισθητήρα σε ένα παιχνίδι δράσης! Έστω ότι ο κόσμος του παιχνιδιού αποτελείται από ανισόπεδο έδαφος με ένα μονοπάτι που οδηγεί σε ένα *Κάστρο (Castle)*. Τα αντικείμενα που υπάρχουν στον κόσμο είναι ο *Kodu*, ο καλός του φίλος *Μηχανάκιας (Cycle)*, το *Κάστρο (Castle)* στο τέλος του μονοπατιού, ένα *Κανόνι (Canon)*, *Σύννεφα (Clouds)* και *Φώτα (Lights)*. Σκοπός του παιχνιδιού είναι ο *Kodu*, τον οποίο χειρίζεται ο παίκτης, να οδηγήσει το φίλο του, το *Μηχανάκια (Cycle)*, στο *Κάστρο (Castle)*. Το μονοπάτι όμως που ακολουθούν είναι δύσβατο καθώς το *Κανόνι (Canon)*, όταν *ακούει* τον *Μηχανάκια (Cycle)* να περνάει από μπροστά του, εκτοξεύει πυραύλους εναντίον του. Τα *Σύννεφα (Clouds)* και τα *Φώτα (Lights)* αποτελούν διακοσμητικά στοιχεία του περιβάλλοντος και δεν παρουσιάζουν κάποια συμπεριφορά. Ο κόσμος λοιπόν του παιχνιδιού περιγράφεται με την εικόνα που ακολουθεί:

Δείτε το παράδειγμα *06_05.kodu*

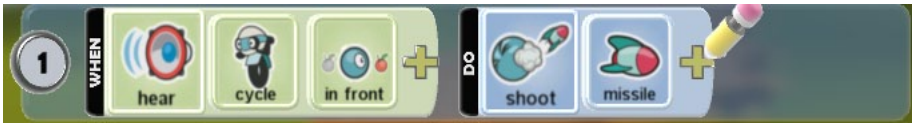
Ανοίξτε το παιχνίδι **[06_05.kodu]** και ας ξεκινήσουμε με τον προγραμματισμό του *Kodu*. Ο *Kodu* αντιλαμβάνεται μέσω του αισθητήρα *Πληκτρολόγιο (Keyboard)* πότε ο παίκτης έχει πατήσει κάποιο από τα βέλη ώστε να κινηθεί προς την αντίστοιχη κατεύθυνση γρήγορα. Η συμπεριφορά αυτή είναι γνωστή σε εσάς:

Ας προγραμματίσουμε τώρα τον *Μηχανάκια (Cycle).* Κάθε φορά που ο *Μηχανάκιας (Cycle)* ακούει τον *Kodu*, θα πρέπει να τον ακολουθεί. Ποιον αισθητήρα μπορούμε να χρησιμοποιήσουμε; Αφού ο *Μηχανάκιας (Cycle)* αντιλαμβάνεται τον *Kodu* από τον ήχο που κάνει, θα χρησιμοποιήσουμε τον αισθητήρα ***Ακούω (Hear)***, έτσι ώστε, ο *Μηχανάκιας (Cycle)* να ακολουθεί τον *Kodu*, μέσω των ήχων που παράγει. Το προσδιοριστικό που θα συνοδεύει τον προγραμματισμό του αισθητήρα είναι ο *Kodu*. Επομένως για τη συμπεριφορά του *Μηχανάκια*, δημιουργούμε τις παρακάτω εντολές:

ΌΤΑΝ[*Ακούω*][τον *Kodu*] **ΤΟΤΕ** [*Κινούμαι*][*Προς* αυτόν]

Δεν τελειώσαμε όμως με τον προγραμματισμό όλων των αντικειμένων! Μας μένει ο προγραμματισμός του *Κανονιού (Cannon).* Η περιγραφή του παιχνιδιού μας οδηγεί και πάλι στη χρήση του αισθητήρα ***Ακούω (Hear)***. Αυτή τη φορά όμως οι εντολές που θα χρησιμοποιήσουμε είναι διαφορετικές. Τώρα επιθυμούμε το *Κανόνι* να ακούει, όχι τον *Kodu*, αλλά τον *Μηχανάκια (Cycle)*, επομένως το προσδιοριστικό του αισθητήρα είναι διαφορετικό. Η περιγραφή όμως του προβλήματος κρύβει και ένα άλλο προσδιοριστικό. Μπορείτε να το φανταστείτε; Έχουμε πει ότι το *Κανόνι (Cannon)* αντιλαμβάνεται τον *Μηχανάκια (Cycle)* όταν περνάει από μπροστά του και τότε εκτοξεύει πυραύλους εναντίων του. Συνεπώς, χρειαζόμαστε ένα ακόμη προσδιοριστικό για τον αισθητήρα, το οποίο έχει να κάνει με τη θέση του *Μηχανάκια (Cycle)*. Αυτό είναι το προσδιοριστικό ***Μπροστά (In Front)*** που βρίσκεται στο μενού ***Επιλογές (Options)***. Άρα, η συμπεριφορά που θα εμφανίζει το *Κανόνι (Cannon)* είναι:

ΌΤΑΝ[Ακούω][τον Μηχανάκια][Μπροστά μου] **ΤΟΤΕ** [Εκτοξεύω][Πυραύλους]



Αποθηκεύστε τις αλλαγές που έχετε κάνει και τρέξτε το παιχνίδι.

Ποια άλλα προσδιοριστικά μπορούμε να χρησιμοποιήσουμε σε συνδυασμό με τον αισθητήρα ***Ακούω (Hear)***; Μπορούμε να ορίσουμε ως προσδιοριστικό οποιοδήποτε αντικείμενο του κόσμου μας, οποιοδήποτε χρώμα (μπορεί να σας φαίνεται παράξενο, αλλά, για παράδειγμα, μπορεί ο *Kodu*, όταν ακούει έναν μαύρο *Μηχανάκια (Cycle)*, να τον κλωτσάει), ακριβώς όπως κάναμε και για τον αισθητήρα ***Βλέπω (See)***. Όταν όμως ένα αντικείμενο διαθέτει τον αισθητήρα αυτό, εμφανίζεται μια επιπλέον κατηγορία προσδιοριστικών με το όνομα ***Ήχοι (Sounds)***. Μπορούμε λοιπόν να χρησιμοποιήσουμε ως προσδιοριστικό, όχι μόνο ένα αντικείμενο, αλλά και κάποιον συγκεκριμένο ήχο (όταν ο *Kodu* ακούει π.χ. έναν ήχο δράματος, τότε να εκφράζει λύπη). Είναι σημαντικό να τονίσουμε ότι μπορούμε να ρυθμίσουμε το πόσο μακριά ακούει το αντικείμενο μας ρυθμίζοντας την ιδιότητα ***Ακοή (Hearing)***.

Υπάρχει μια ιδιότητα η οποία, όταν είναι ενεργοποιημένη, τότε ένα αντικείμενο δεν μπορεί να γίνει αντιληπτό από τον αισθητήρα *Ακούω (Hear)*. Μπορείτε να βρείτε ποια είναι αυτή;

6.3.3 Ο αισθητήρας Πέφτω Πάνω (Bump)

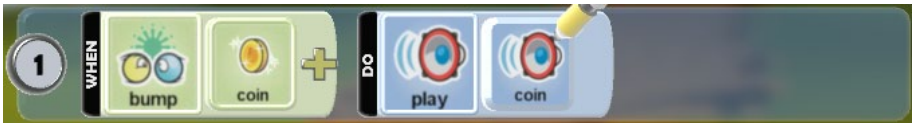
Ο αισθητήρας ***Πέφτω Πάνω (Bump)*** επιτρέπει στα αντικείμενα να αντιλαμβάνονται το γεγονός ότι έχουν ακουμπήσει άλλα αντικείμενα. Θα μπορούσαμε δηλαδή να πούμε ότι ο συγκεκριμένος αισθητήρας αποτελεί την αίσθηση της αφής για τα αντικείμενα. Έτσι λοιπόν μπορούμε να προγραμματίσουμε τον *Kodu* όταν *Πέφτει Πάνω* σε ένα *Μήλο* να το τρώει ή όταν *Πέφτει Πάνω* σε κάποιον εχθρό να χάνει ενέργεια.

Κάποια ρήματα συμπεριφορών που σχετίζονται με τη χρήση του συγκεκριμένου αισθητήρα είναι:

{Πέφτω πάνω, ακουμπώ, αγγίζω, έρχομαι σε επαφή, συγκρούομαι}

Στα περισσότερα παιχνίδια που έχετε παίξει, ο ήρωας μαζεύει κάποια αντικείμενα για να πετύχει το στόχο του. Για παράδειγμα αν ακουμπήσει ένα νόμισμα, τότε αυξάνονται τα χρήματα που διαθέτει. Όταν πέσει πάνω σε μια καρδιά, προστίθεται μια ζωή στις υπάρχουσες.

Ρίξτε μια ματιά στην παρακάτω συμπεριφορά. Μπορείτε να καταλάβετε τι είχε ο προγραμματιστής στο μυαλό του όταν προγραμμάτιζε το συγκεκριμένο αντικείμενο;



Ας πάρουμε τα πράγματα με τη σειρά. Προγραμματίσαμε το αντικείμενο να διαθέτει έναν αισθητήρα που θα του επιτρέπει να αντιλαμβάνεται πότε έχει ακουμπήσει ένα νόμισμα. Ο αισθητήρας που θα επιτρέπει στο αντικείμενο να αντιληφθεί το γεγονός αυτό, είναι ο αισθητήρας ***Πέφτω Πάνω (Bump)***. Όπως γνωρίζουμε όμως, για να λειτουργήσει σωστά ένας αισθητήρας, θα πρέπει να το συνδυάσουμε με τα κατάλληλα προσδιοριστικά. Στο συγκεκριμένο παράδειγμα μας ενδιαφέρει η επαφή του αντικειμένου με ένα *Νόμισμα (Coin)*, άρα το *Νόμισμα (Coin)* θα αποτελεί και το προσδιοριστικό του. Αυτό θα γίνει με την παρακάτω εντολή :

ΌΤΑΝ[Πέσω Πάνω][σε ένα Νόμισμα] **ΤΟΤΕ** [να Παίξω][τον Ήχο Νομίσματος]

Όπως και κάθε άλλος αισθητήρας, έτσι κι αυτός διαθέτει μία μεγάλη ποικιλία προσδιοριστικών που μπορούμε να χρησιμοποιήσουμε. Σε ένα παιχνίδι ράλι για παράδειγμα θα μπορούσαμε να προγραμματίσουμε τον *Kodu*, όταν πέφτει πάνω σε ***Οτιδήποτε (Anything)***, αντί για κάποιο συγκεκριμένο αντικείμενο, να χάνει ενέργεια. Ακόμη, θα μπορούσαμε να χρησιμοποιήσουμε κάποια από τα προσδιοριστικά που δηλώνουν θέση (π.χ. ο *Kodu* να αντιλαμβάνεται όταν πέφτει πάνω του μια *Χελώνα (Turtle)* ***από τα αριστερά*** τότε να καταστρέφεται).

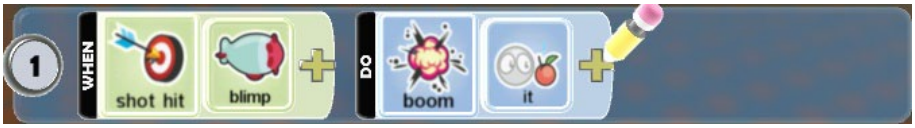
Μία από τις ιδιότητες που μπορούμε να δώσουμε στα αντικείμενά μας είναι η ιδιότητα *Φάντασμα (Ghost)*. Όταν κάποιο αντικείμενο διαθέτει αυτή την ιδιότητα τότε ο αισθητήρας *Πέφτω Πάνω (Bump)* δεν λειτουργεί για αυτό το αντικείμενο!

6.2.4 Ο αισθητήρας Πετυχαίνω (Shot Hit)

Ο αισθητήρας ***Πετυχαίνω (Shot Hit)*** βοηθάει τον *Kodu* να αντιληφθεί πότε ένας πύραυλος ή ένα σφαιρίδιο που έχει εκτοξεύσει, έχει χτυπήσει κάποιο άλλο αντικείμενο. Όπως μπορείτε να φανταστείτε, εάν προγραμματίσουμε ένα αντικείμενο να εκτελεί την ενέργεια ***Πυροβολώ (Shoot)***, τότε είναι πολύ πιθανό ότι θα χρειαστούμε να προγραμματίσουμε και τον αισθητήρα ***Πετυχαίνω (Shot Hit)***. Έτσι, θα μπορούσαμε να δημιουργήσουμε μια συμπεριφορά στην οποία όταν ο *Kodu*

πετύχει ένα *Κανόνι (Cannon)*, να το καταστρέφει ή όταν πετύχει τον *Μηχανάκια (Cycle)*, να τον βάφει πράσινο! Ένα ακόμη παράδειγμα εντολής με χρήση του αισθητήρα:

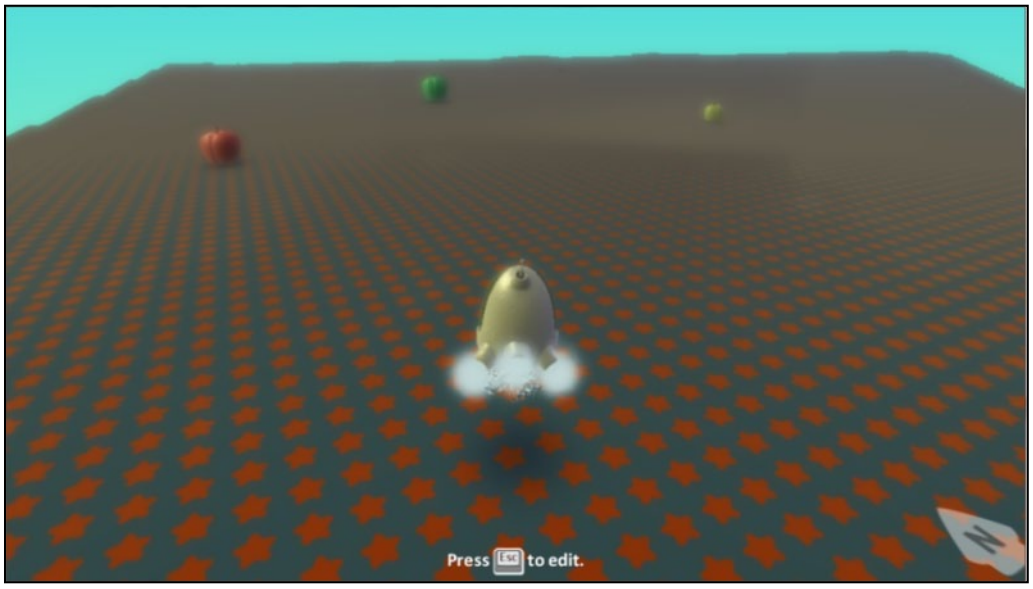
ΌΤΑΝ[Πετυχαίνω][Αερόπλοιο] **ΤΟΤΕ** [Ανατινάζω][το Αερόπλοιο]



Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα ***Πετυχαίνω (Shot Hit)*** είναι:

{Ευστοχώ, βρίσκω στόχο, πετυχαίνω, χτυπάω διάνα}

Ας δούμε πώς μπορούμε να χρησιμοποιήσουμε τον αισθητήρα αυτό μέσα από ένα παιχνίδι σκοποβολής. Στο περιβάλλον του παιχνιδιού μας υπάρχει ο *Kodu* και τρία *Μήλα (Appls)*. Ένα κόκκινο, ένα πράσινο και ένα κίτρινο. Ο σκοπός του παιχνιδιού είναι ο χειριστής του *Kodu* να πετύχει τα τρία *Μήλα (Apples)* με όσο το δυνατόν λιγότερους πυραύλους. Μην ξεχνάτε άλλωστε ότι πρόκειται απλά για εξάσκηση στη σκοποβολή! Κάθε φορά όμως που ο *Kodu* πετυχαίνει κάποιο *Μήλο (Apple)*, θα πρέπει να χρωματίζεται με το αντίστοιχο χρώμα. Ο κόσμος που περιγράφεται μπορεί να μοιάζει όπως στην παρακάτω εικόνα.



Το μοναδικό αντικείμενο που θα παρουσιάζει συμπεριφορά στο συγκεκριμένο παιχνίδι είναι ο *Kodu*. Ο χρήστης θα πρέπει με τη βοήθεια του πληκτρολογίου να μπορεί να στρίβει αριστερά και δεξιά τον Kodu και να εκτοξεύει πυραύλους. Ας υποθέσουμε ότι το αντικείμενο αντιλαμβάνεται τα κουμπιά του πληκτρολογίου για την περιστροφή και το πάτημα του αριστερού κουμπιού του ποντικιού για την εκτόξευση πυραύλων. Ο προγραμματισμός των παραπάνω συμπεριφορών δεν θα πρέπει να σας δυσκολεύει ιδιαίτερα. Επιθυμούμε:

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο A] **ΤΟΤΕ** [Στρίψε][Αριστερά]

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο D, **ΤΟΤΕ** [Στρίψε][Δεξιά]

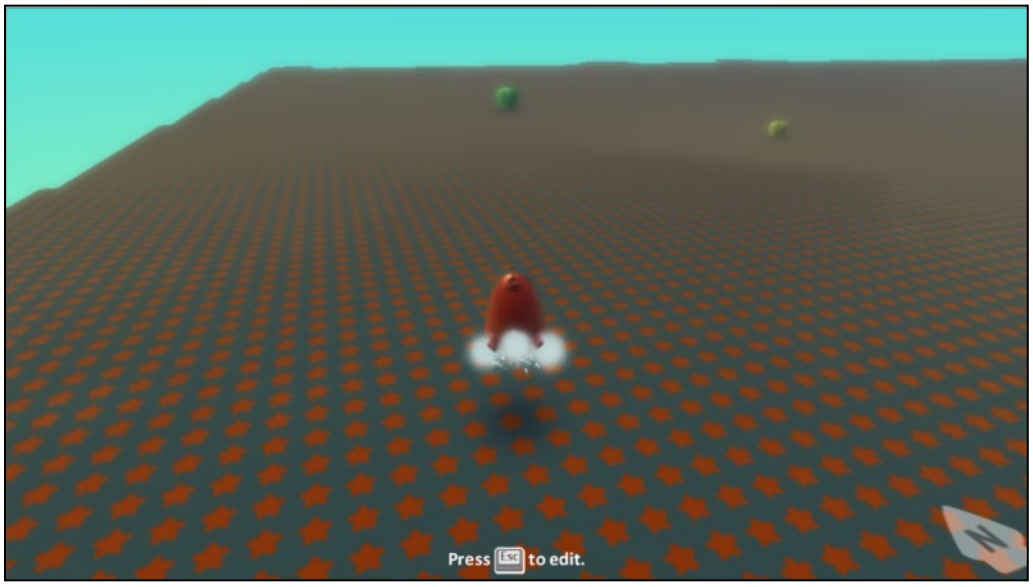
ΌΤΑΝ[στο Ποντίκι][πατηθεί το Αριστερό Κλικ] **ΤΟΤΕ** [Εκτόξευσε][Πύραυλο]

Πώς θα είναι όμως ο *Kodu* ικανός να αντιληφθεί ότι ο πύραυλος που έχει εκτοξεύσει έχει χτυπήσει κάποιο *Μήλο (Apple)*; Η χρήση του αισθητήρα ***Πέφτω Πάνω (Bump)*** δεν θα ήταν σωστή, αφού αφορά μόνο την περίσταση στην οποία ο *Kodu* έχει ακουμπήσει κάποιο άλλο αντικείμενο. Εμείς επιθυμούμε ο αισθητήρας που θα χρησιμοποιήσουμε, να αντιλαμβάνεται αν ο πύραυλος που έχει εκτοξεύσει ο Kodu, έχει χτυπήσει κάποιο αντικείμενο. Ο *Kodu* μπορεί να αντιληφθεί το γεγονός αυτό αν τον προγραμματίσουμε με τον αισθητήρα ***Πετυχαίνω (Shot Hit)*** και προσδιοριστικό στοιχείο το *Μήλο (Apple)*. Άρα χρειαζόμαστε μια εντολής της μορφής:

ΌΤΑΝ[Πετυχαίνω][Μήλο] **ΤΟΤΕ** [Χρωμάτισε][Εμένα]

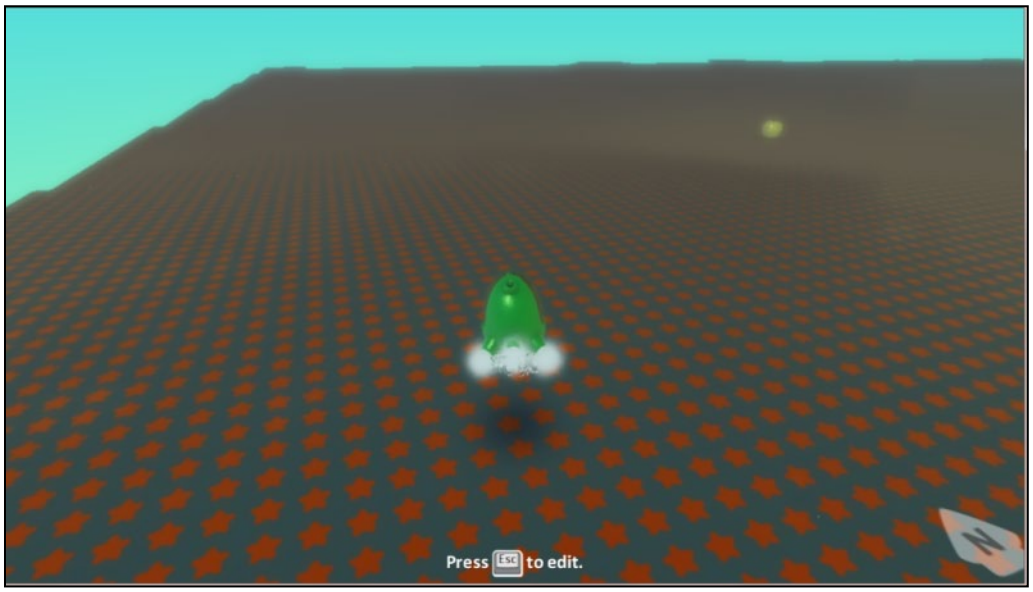
Μήπως όμως λείπει κάτι από αυτήν την εντολή; Μήπως πρέπει να προσθέσουμε κάποιο επιπλέον προσδιοριστικό στον αισθητήρα; Προφανώς, καθώς θα πρέπει να προσδιορίσουμε και το χρώμα του *Μήλου (Apple)* που ο *Kodu* πετυχαίνει κάθε φορά. Το ίδιο προσδιοριστικό χρώματος θα πρέπει να προσθέσουμε και στην ενέργεια χρωματισμού. Συγκεκριμένα όταν ο *Kodu* πετυχαίνει το κόκκινο *Μήλο*:

ΟΤΑΝ[Πετυχαίνω][Μήλο][Κόκκινο] **ΤΟΤΕ** [Χρωμάτισε][Εμένα][Κόκκινο]



Παρατηρείστε ότι στην παραπάνω εικόνα λείπει το κόκκινο *Μήλο (Apple)*. Άρα ο *Kodu* το πέτυχε και έτσι χρωμάτισε τον εαυτό του κόκκινο. Όταν ο *Kodu* πετυχαίνει το πράσινο *Μήλο*:

ΟΤΑΝ[Πετυχαίνω][Μήλο][Πράσινο] **ΤΟΤΕ** [Χρωμάτισε][Εμένα][Πράσινο]



Κάνουμε κάτι αντίστοιχο και για το κίτρινο μήλο.

Μπορούμε να χρησιμοποιήσουμε πολλά επιπλέον προσδιοριστικά στον αισθητήρα ***Πετυχαίνω (Shot Hit)***. Για παράδειγμα, θα μπορούσαμε να χρησιμοποιήσουμε το ***Δεν (Not)*** στο παραπάνω παράδειγμα έτσι ώστε, όταν ο *Kodu* δεν πετυχαίνει κάποιο *Μήλο (Apple)*, να χρωματίζει τον εαυτό του μαύρο. Ακόμη, φανταστείτε σε ένα παιχνίδι μάχης τον *Kodu* να εκτοξεύει πυραύλους εναντίων ακίνητων *Κανονιών (Cannons)* και κινούμενων *Χελωνών (Turtles)*. Δεν θα ήταν δίκαιο να επιβραβεύουμε με περισσότερους πόντους τον παίκτη κάθε φορά που χτυπάει ένα κινούμενο αντικείμενο από ότι όταν χτυπάει ένα ακίνητο; Για να το πετύχουμε αυτό θα πρέπει να χρησιμοποιήσουμε το προσδιοριστικό ***Κινούμενο (Moving)*** μετά το αντικείμενο που επιθυμούμε να αντιλαμβάνεται ο αισθητήρας.

6.2.5 Ο αισθητήρας Έχω (Got)

Ο αισθητήρας ***Έχω (Got)*** αναγνωρίζει το γεγονός ότι ο *Kodu* κρατάει κάποιο άλλο αντικείμενο. Μα πώς είναι αυτό δυνατόν, αφού δεν έχει χέρια! Και όμως, στο MSKodu κάποιο αντικείμενο είναι ικανό να κρατήσει κάποιο άλλο αντικείμενο. Θα μπορούσαμε λοιπόν να προγραμματίσουμε τον *Kodu* έτσι ώστε, όταν έχει μία *Καρδιά (Heart)*, να εκπέμπει κόκκινο φως ή όταν κρατάει τη φίλη του *Χελώνα (Turtle)*, να *Πηδάει (Jump)* ώστε να τη βοηθήσει να διασχίσει ένα ποτάμι! Ένα αντιπροσωπευτικό παράδειγμα με χρήση του αισθητήρα είναι:

Ο μάλλον ερωτοχτυπημένος προγραμματιστής είχε στο μυαλό του:

ΌΤΑΝ[Έχω][μια Καρδιά] **ΤΟΤΕ** [Λάμπω][Εγώ][με Κόκκινο Χρώμα]

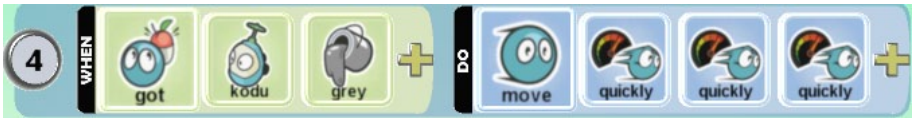
Δείτε το παράδειγμα *06_05.kodu*

Πάμε λοιπόν να δούμε πώς μπορούμε να προγραμματίζουμε τον αισθητήρα αυτό, δημιουργώντας ένα παιχνίδι επιβίωσης. Φορτώστε το παιχνίδι **[06_05.kodu]**. Η πίστα αυτού του παιχνιδιού αποτελείται από ένα μακρύ διάδρομο από γρασίδι. Στην αρχή του διαδρόμου υπάρχει ο άσπρος *Kodu*, που είναι ο χαρακτήρας του παίκτη, όπως και αρκετά γκρι *Kodu* που περιφέρονται. Στο τέλος του διαδρόμου υπάρχουν δύο *Κάστρα (Castles)* και μία *Καλύβα (Hut)*, ενώ στον ουρανό περιφέρονται εχθρικά *Τζετ (Jet)* τα οποία ρίχνουν πυραύλους προς τα κάτω. Ο παίκτης θα πρέπει να πάρει ένα προς ένα όλα τα γκρι *Kodu* και να τα μεταφέρει με ασφάλεια στην άλλη πλευρά. Τα γκρι *Kodu* είναι ασφαλή μόνο όταν ακουμπήσουν την *Καλύβα (Hut)*. Ο παίκτης είναι νικητής όταν δεν υπάρχει κανένα γκρι *Kodu* στην πίστα. Τα *Κάστρα* αποτελούν απλά διακοσμητικό στοιχείο της πίστας.

Ας μελετήσουμε τη συμπεριφορά καθενός από τα αντικείμενα που υπάρχουν στο παιχνίδι. Αρχικά ο *Kodu* αλληλεπιδρά με τον χρήστη μέσω του αισθητήρα *Πληκτρολόγιο (Keyboard)* και συγκεκριμένα όταν ο παίκτης πατάει κάποιο από τα πλήκτρα *WASD*. Όταν συμβεί κάτι τέτοιο τότε ο *Kodu* κινείται προς την αντίστοιχη κατεύθυνση. Επιπλέον, ο παίκτης ελέγχει τον *Kodu* μέσω του αισθητήρα *Ποντίκι (Mouse)*. Όταν ο *Kodu* αντιληφθεί μέσω του αισθητήρα *Ποντίκι (Mouse)* ότι έχει πατηθεί το αριστερό κουμπί του ποντικιού τότε ***Αρπάζει (Grab)*** ό,τι βρει μπροστά του. Η συμπεριφορά αυτή επιτρέπει στον *Kodu* να παίρνει ένα-ένα τα γκρι *Kodu* και να τα μεταφέρει όπου επιθυμεί. Όταν πλέον ο *Kodu* φτάσει στο τέλος του διαδρόμου θα πρέπει να αφήσει το γκρι *Kodu* που κρατάει. Για την υλοποίηση της συμπεριφοράς αυτής μας βοηθάει το δεύτερο γεγονός που αντιλαμβάνεται ο αισθητήρας *Ποντίκι (Mouse)*. Όταν ο *Kodu* αναληφθεί ότι έχει πατηθεί το δεξί κλικ του ποντικιού τότε ***Αφήνει (Drop)*** το γκρι *Kodu*:

Ας εξετάσουμε λοιπόν την τελευταία συμπεριφορά με την οποία έχει προγραμματιστεί ο *Kodu*. Από τη στιγμή που ο *Kodu* εκτελεί την ενέργεια ***Αρπάζω (Grab)*** προκειμένου να κρατήσει ένα γκρι *Kodu* θα πρέπει να διαθέτει και έναν αισθητήρα για να αντιληφθεί το γεγονός ότι το κουβαλάει. Ποιος άραγε μπορεί να είναι ο αισθητήρας αυτός; Μα φυσικά ο αισθητήρας ***Έχω (Got)***! Μάλιστα θα πρέπει να συνοδεύεται με δύο προσδιοριστικά. Το πρώτο προσδιοριστικό αφορά το αντικείμενο που έχει ο *Kodu* και το δεύτερο είναι το χρώμα του, δηλαδή το γκρι. Όταν συμβεί κάτι τέτοιο τότε ο *Kodu* κινείται πάρα πολύ γρήγορα προκειμένου να φτάσει το συντομότερο δυνατό στο *Κάστρο*.

ΌΤΑΝ[Έχω][ένα Γκρι][Kodu] **ΤΟΤΕ** [να Κινηθώ][Πολύ Γρήγορα]



Συνεχίζουμε με τη μελέτη συμπεριφοράς των γκρι *Kodu*. Εφόσον τα γκρι *Kodu* είναι πανικοβλημένα από την ξαφνική επίθεση των *Τζετ*, το μόνο που κάνουν είναι να περιφέρονται. Συνεπώς, για τον προγραμματισμό της συμπεριφοράς τους δεν χρειάζεται κάποιος αισθητήρας:

Υπενθυμίζουμε ότι όταν θέλουμε ένα αντικείμενο να εκτελεί μία ενέργεια για πάντα, τότε είτε μπορούμε να χρησιμοποιήσουμε τον αισθητήρα *Πάντα (Always)* είτε να μην εισάγουμε κανέναν αισθητήρα.

Ούτε τα *Τζετ* από την πλευρά τους χρειάζεται να διαθέτουν κάποιον αισθητήρα. Είναι όμως προγραμματισμένα να εκτελούν ταυτόχρονα δύο ενέργειες. Η πρώτη ενέργεια είναι να περιφέρονται και η δεύτερη να εκτοξεύουν πυραύλους προς τα κάτω.

Τέλος, η *Καλύβα (Hut)* είναι προγραμματισμένη να εκδηλώνει δύο συμπεριφορές. Αρχικά θα πρέπει να είναι εφοδιασμένη με τον αισθητήρα *Πέφτω Πάνω (Bump)* για να αντιλαμβάνεται πότε ένα γκρι *Kodu* την έχει ακουμπήσει. Όταν συμβεί κάτι τέτοιο, τότε το εξαφανίζει. Η *Καλύβα* όμως είναι και υπεύθυνη για να ανακηρύξει τον *Kodu* νικητή. Θυμηθείτε ότι ο *Kodu* είναι νικητής μόνο όταν δεν υπάρχει κάποιο γκρι *Kodu* στην πίστα. Όπως είδαμε στο κεφάλαιο αυτό, μπορούμε να χρησιμοποιήσουμε τον αισθητήρα *Βλέπω (See)* προκειμένου η καλύβα να αντιληφθεί την απουσία των γκρι *Kodu* από την πίστα.

Τρέξτε τώρα το παιχνίδι και διασκεδάστε!

Ποια άλλα προσδιοριστικά μπορούμε να χρησιμοποιήσουμε με τον αισθητήρα ***Έχω (Got)***; Όπως θα έχετε καταλάβει, μπορούμε να χρησιμοποιήσουμε οποιοδήποτε αντικείμενο και χρώμα. Μπορούμε ακόμα, όπως και με τους άλλους αισθητήρες, να χρησιμοποιήσουμε το προσδιοριστικό ***Δεν (Not)***. Για παράδειγμα, θα μπορούσαμε να εμπλουτίσουμε το παραπάνω

παράδειγμα προγραμματίζοντας μία καινούρια συμπεριφορά του *Kodu*. Θα μπορούσαμε να τον προγραμματίσουμε, όταν δεν κρατάει ***Τίποτα (Anything)***, τότε να εκφράζει λύπη.

6.2.6 Ο αισθητήρας Κουβαλιέμαι (Held By)

Ο αισθητήρας ***Κουβαλιέμαι (Held By)*** επιτρέπει στα αντικείμενα να αντιδρούν στο γεγονός ότι κάποιο άλλο αντικείμενο τα κρατάει. Ουσιαστικά είναι ο αντίστροφος αισθητήρας από τον ***Έχω (Got)***. Ο πρώτος αφορά κάποιον που κουβαλιέται από κάποιον άλλο, ενώ ο δεύτερος αναφέρεται σε αυτόν που κουβαλάει! Με τη βοήθεια του αισθητήρα ***Κουβαλιέμαι (Held By)***, θα μπορούσαμε να προγραμματίσουμε τη φίλη του *Kodu* να ζητάει βοήθεια όταν την κρατάει ένας διαβολικός *Μηχανάκιας (Cycle).* Επίσης θα μπορούσαμε να πούμε σε ένα *Αστέρι* να λάμπει *(Star)* όταν το κουβαλάει ο *Kodu*.

Ένα παράδειγμα εντολής με χρήση του αισθητήρα είναι:

ΌΤΑΝ[Με Κρατάει][ένας Μηχανάκιας] **ΤΟΤΕ** [να Εκφράζομαι][με Κακές Κουβέντες]

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα:

{Κουβαλιέμαι, με κουβαλάνε, με μεταφέρουν, κρατιέμαι, μεταφέρομαι, είμαι αιχμάλωτος}

Δείτε το παράδειγμα *06_06.kodu*

Προκειμένου να κατανοήσουμε τη χρήση του αισθητήρα αυτού θα προσθέσουμε μία επιπλέον συμπεριφορά στα γκρι *Kodu* του παιχνιδιού **[06_06.kodu]** της προηγούμενης υποενότητας. Συγκεκριμένα, στόχος μας είναι να τα προγραμματίσουμε έτσι ώστε, όταν αντιλαμβάνονται ότι τα κρατάει ο *Kodu*, τότε αυτά να εκπέμπουν μία πράσινη λάμψη. Πώς λοιπόν μπορούν να αντιληφθούν τα γκρι *Kodu* ότι τα μεταφέρει ο *Kodu*; Θα χρησιμοποιήσουμε τον αισθητήρα ***Κουβαλιέμαι (Held By)*** με προσδιοριστικό το αντικείμενο *Kodu*. Από τη στιγμή που κανένα άλλο αντικείμενο δεν μπορεί να κρατήσει κάποιο από τα γκρι *Kodu*, δεν θα χρειαστεί να ορίσουμε ως προσδιοριστικό το χρώμα του Kodu-κουβαλητή. Συνεπώς για την καινούρια συμπεριφορά των πληγωμένων *Kodu*, δημιουργούμε την εξής εντολή:

ΌΤΑΝ[Κουβαλιέμαι][από τον Kodu] **ΤΟΤΕ** [να Λάμπω][Εγώ][με Πράσινο Χρώμα]

Φορτώστε λοιπόν το παιχνίδι, προσθέστε την παραπάνω συμπεριφορά στα γκρι *Kodu* αποθηκεύστε και τρέξτε το.

Όπως και οι υπόλοιποι αισθητήρες, έτσι κι ο αισθητήρας ***Κουβαλιέμαι (Held By)*** μπορεί να συνδυαστεί με αρκετά προσδιοριστικά. Μπορούμε να ορίσουμε ως προσδιοριστικό οποιοδήποτε αντικείμενο του κόσμου μας, οποιοδήποτε χρώμα (π.χ. όταν ο *Kodu* κουβαλιέται από έναν μωβ

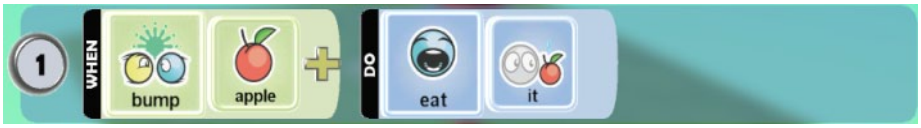
Μηχανάκια (Cycle)), οποιαδήποτε έκφραση άλλου αντικειμένου (π.χ. όταν ο *Kodu* κρατιέται από μία στενοχωρημένη *Χελώνα (Turtle)*!).

6.4 Η συμπεριφορά του Kodu: ενέργειες

Στην παράγραφο αυτή θα γνωρίσουμε ενέργειες που αφορούν την αλληλεπίδραση μεταξύ αντικειμένων.

6.4.1 Ενέργεια Τρώω (Eat)

Η ενέργεια ***Τρώω (Eat)*** ικανοποιεί την όρεξη του πεινασμένου *Kodu*. Αστειευόμαστε! Είτε πεινάει είτε όχι, από τη στιγμή που ο *Kodu* είναι προγραμματισμένος να εκτελεί την ενέργεια αυτή, μπορεί να φάει οποιοδήποτε αντικείμενο βάλουμε ως προσδιοριστικό της! Θα μπορούσαμε δηλαδή να προγραμματίσουμε τον *Kodu*, όταν βλέπει ένα *Μήλο (Apple)*, να το *Τρώει (Eat)* ή όταν πέσει πάνω σε ένα *Ψάρι (Fish)*, να το φάει. Ένα παράδειγμα συμπεριφοράς με χρήση της συγκεκριμένης ενέργειας:



Τι είχε ο προγραμματιστής στο μυαλό του:

ΌΤΑΝ[πέσεις πάνω][στο μήλο] **ΤΟΤΕ** [Καταβρόχθισέ][Το]

Ρήματα που μπορούν να σας φέρνουν στο μυαλό την ενέργεια αυτή είναι:

Τρώω, καταβροχθίζω, εξαφανίζω, καταπίνω.

Ας μάθουμε όμως περισσότερα για την ενέργεια με ένα παράδειγμα. Ανοίξτε έναν *Άδειο Κόσμο (Empty World)* και προσθέστε τον *Kodu*, δύο *Μήλα (Apples)*, καθώς και ένα διακοσμητικό *Δέντρο (Tree)*. Χρωματίστε το ένα μήλο ως κόκκινο και το άλλο ως μωβ (μη ξεχνάτε ότι για να χρωματίσετε ένα αντικείμενο, απλά το επιλέγετε και χρησιμοποιείτε τα βελάκια του πληκτρολογίου). Ο κόσμος μας θα μπορούσε να είναι όπως φαίνεται στην παρακάτω εικόνα:

Σκοπός μας είναι να προγραμματίσουμε τον *Kodu*, όταν ακουμπά ένα κόκκινο *Μήλο (Apple)*, τότε να το καταπίνει. Τυπική η διαδικασία για την κίνηση του *Kodu* από το χρήστη:

Αποθηκεύστε το παιχνίδι και τρέξτε το. Κατευθυνθείτε πάνω στο μήλο. Πρέπει να παρατηρήσατε ότι ο *Kodu* το μόνο που κάνει όταν έρχεται σε επαφή με το μήλο, είναι να το σπρώχνει προς οποιαδήποτε κατεύθυνση κινείται. Πώς άραγε θα μπορούσαμε να προγραμματίσουμε τον *Kodu*

να καταπίνει το κόκκινο *Μήλο* όταν το ακουμπά; Σίγουρα θα χρειαστούμε τον αισθητήρα *Πέφτω Πάνω (Bump)* και την ενέργεια ***Τρώω (Eat)***.

Προσθέτουμε λοιπόν στις συμπεριφορές του *Kodu* την εντολή:

ΌΤΑΝ[πέσεις πάνω σε] [Κόκκινο][Μήλο] **ΤΟΤΕ** [Φάε][Το]

Αποθηκεύστε τις αλλαγές, τρέξτε το παιχνίδι και κατευθύνετε τον *Kodu* προς το κόκκινο *Μήλο (Apple)*. Ο *Kodu* το τρώει μόλις το ακουμπήσει! Παρατηρήσατε ότι ακούστηκε και ο χαρακτηριστικός ήχος που κάνει το μήλο όταν τρώγεται;

Η ενέργεια *Τρώω (Eat)* μπορεί να πάρει δύο προσδιοριστικά, το ***Το (It)*** και το ***Μόνο Μία Φορά (Once)***. Το πρώτο προσδιοριστικό χρησιμοποιείτε όταν επιθυμούμε ο *Kodu* να τρώει μόνο το αντικείμενο που προσδιορίζεται από τον αισθητήρα της αντίστοιχης συμπεριφοράς. Το δεύτερο προσδιοριστικό καθορίζει το αν η ενέργεια ***Τρώω (Eat)*** θα ενεργοποιηθεί για μόνο μια φορά ανεξάρτητα από την επαναλαμβανόμενη ενεργοποίηση του αισθητήρα.

Είναι σημαντικό να σημειώσουμε ότι τα διαθέσιμα προσδιοριστικά των ενεργειών αλλάζουν ανάλογα με τον αισθητήρα που χρησιμοποιούμε στην εκάστοτε συμπεριφορά. Έτσι, αν για παράδειγμα χρησιμοποιήσουμε τον αισθητήρα ***Πέφτω Πάνω (Bump)***, θα παρατηρήσουμε ότι δεν είναι διαθέσιμο μετά την ενέργεια ***Τρώω (Eat)*** το προσδιοριστικό ***Μόνο Μία Φορά (Once)***. Μπορείτε να εξηγήσετε γιατί;

6.4.2 Ενέργεια Εξαφανίζω (Vanish)

Η ενέργεια ***Εξαφανίζω (Vanish)*** είναι διαθέσιμη στο μενού *Μάχη (Combat)*. Ας σκεφτούμε λίγο την προηγούμενη ενέργεια, δηλαδή την ενέργεια *Τρώω (Eat)*. Στην περίπτωση, που στη θέση του *Μήλου (Apple)* είχαμε ένα *Νόμισμα (Coin)* ή μια *Καρδιά (Heart)*, θα μπορούσαμε και πάλι να χρησιμοποιήσουμε την ενέργεια *Τρώω (Eat)*. Ωστόσο, στις περιπτώσεις αυτές, ουσιαστικά επιθυμούμε να εξαφανίζουμε τα αντικείμενα αυτά από τον κόσμο, και όχι να φάει ο ήρωάς μας! Άλλωστε δεν θα έπρεπε να τρώγονται! Για αυτό το λόγο μπορούμε να αξιοποιήσουμε την ενέργεια ***Εξαφανίζω (Vanish)***. Η ενέργεια ***Εξαφανίζω (Vanish)*** συνδυάζεται με ένα από τα προσδιοριστικά ***Εμένα (Me)*** και ***Το (It)***. Το πρώτο προσδιοριστικό θα έχει ως αποτέλεσμα να εξαφανιστεί το ίδιο το αντικείμενο που περιέχει τη συγκεκριμένη συμπεριφορά, ενώ το δεύτερο προσδιοριστικό θα προκαλέσει την εξαφάνιση του αντικειμένου που προσδιορίζεται στον αντίστοιχο αισθητήρα. Αν δεν βάλουμε κάποιο προσδιοριστικό στην ενέργεια υπονοείται το ***Το (It)***. Ένα παράδειγμα εντολής με χρήση της ενέργειας *Εξαφανίζω (Vanish)*:

Τι είχε ο προγραμματιστής στο μυαλό του:

ΌΤΑΝ[Πετυχαίνω][Οτιδήποτε] **ΤΟΤΕ** [Εξαφάνισε].....[Το]

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Εξαφανίζω, καταστρέφω, κάνω αόρατο, εξολοθρεύω, αφανίζω}

Ας υποθέσουμε λοιπόν ότι σε ένα κόσμο υπάρχει ο *Μηχανάκιας (Cycle)* και μία *Χελώνα (Turtle)* όπως φαίνεται στην παρακάτω εικόνα.

Ας υποθέσουμε ότι επιθυμούμε να προγραμματίσουμε τον *Μηχανάκια (Cycle)* έτσι ώστε να μπορεί να εξαφανίσει τη χελώνα αν πέσει πάνω της στο τέλος ενός άλματος! Για ακόμη μια φορά θα χρειαστεί να προγραμματίσουμε την αλληλεπίδρασή του Μηχανάκια με τον παίκτη μέσω του πληκτρολογίου, έτσι ώστε να κινείται και να εκτελεί άλματα. Επιπλέον ο μηχανάκιας θα πρέπει να διαθέτει κάποιον αισθητήρα που να αντιλαμβάνεται πότε έχει ακουμπήσει τη χελώνα. Θα χρησιμοποιήσουμε τον αισθητήρα *Πέφτω Πάνω (Bump)* με προσδιοριστικό αντικείμενο τη *Χελώνα (Turtle)* και θα χρειαστούμε και το προσδιοριστικό που υποδηλώνει κατεύθυνση. Εφόσον ο *Μηχανάκιας (Cycle)* θα πέφτει πάνω στη χελώνα από τον αέρα, η *Χελώνα (Turtle)* θα βρίσκεται από κάτω του κατά την προσγείωσή του. Άρα, θα πρέπει να χρησιμοποιήσουμε και το προσδιοριστικό ***Κάτω (Below)***. Αυτό είναι το γεγονός που θα τον οδηγεί στην εκτέλεση της ενέργειάς του, η οποία θα είναι να εξαφανίσει τη *Χελώνα (Turtle)*. Θα χρησιμοποιήσουμε συνεπώς την ενέργεια ***Εξαφανίζω (Vanish)***, σε συνδυασμό με το προσδιοριστικό ***Αυτό (It)***, έτσι ώστε να εξαφανίζεται η χελώνα.

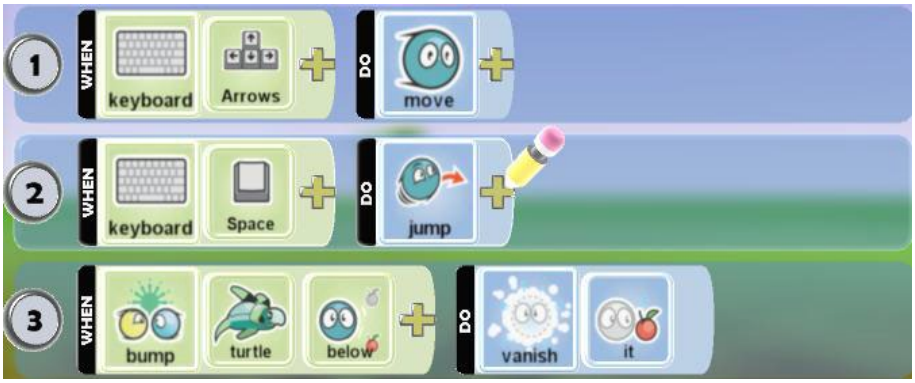
Οι συμπεριφορές που περιγράψαμε παραπάνω:

ΌΤΑΝ[στο *Πληκτρολόγιο*][πατηθεί κάποιο από τα *Βέλη*] **ΤΟΤΕ** [*Κινήσου*]

ΌΤΑΝ[στο *Πληκτρολόγιο*][πατηθεί το πλήκτρο *Κενό*] **ΤΟΤΕ** [*Πήδα*]

ΌΤΑΝ[Ακουμπήσεις][Χελώνα][Από Κάτω σου] **ΤΟΤΕ** [Εξαφάνισε][Την]

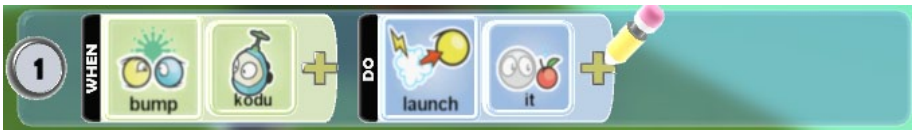
Και σε μορφή συμπεριφορών στο MSKodu είναι:



Αν στο συγκεκριμένο παράδειγμα χρησιμοποιούσαμε το προσδιοριστικό ***Εμένα (Me)*** τότε θα εξαφανίζονταν ο *Μηχανάκιας (Cycle)* σε μια αντίστοιχη σύγκρουση.

6.4.3 Ενέργεια Εκτινάζω (Launch)

Η ενέργεια ***Εκτινάζω (Launch)*** δίνει την ικανότητα στον *Kodu* να εκτινάζει άλλα αντικείμενα, δηλαδή να τα κλωτσάει! Θα μπορούσαμε για παράδειγμα να προγραμματίσουμε τον *Kodu* σε ένα παιχνίδι μάχης, όταν πέφτει πάνω σε έναν *Μηχανάκια (Cycle),* να τον πετάει μακριά. Ωστόσο, πρόκειται για μία ιδιαίτερη ενέργεια, καθώς ανάλογα με τα προσδιοριστικά που θα χρησιμοποιήσουμε αλλάζει τελείως τον τρόπο με τον οποίο ο *Kodu* την εκτελεί. Κάντε λίγη υπομονή και θα τα ξεκαθαρίσουμε στη συνέχεια. Ένα παράδειγμα εντολών περιγραφής της ενέργειας είναι:



Τι είχε ο προγραμματιστής στο μυαλό του;

ΌΤΑΝ[πέσεις πάνω][στον Kodu] **ΤΟΤΕ** [Εκτίναξε][Τον]

Ρήματα που μπορεί να μας φέρουν στο μυαλό την ενέργεια αυτή:

{Εκτινάζω, πετάω, κλωτσάω}

Ας δούμε πρακτικά όμως την ενέργεια αυτή σε ένα παιχνίδι εκτέλεσης πέναλτι! Έστω ότι ο κόσμος του παιχνιδιού αποτελείται από ένα γήπεδο ποδοσφαίρου. Στο βάθος ακούγεται πλήθος να ζητωκραυγάζει, ενώ στο κέντρο του υπάρχει ο αφενώς ο *Kodu* τον οποίο χειρίζεται ο παίκτης με το πληκτρολόγιο και αφετέρου μία *Μπάλα (Ball)*, όπως φαίνεται και στην παρακάτω εικόνα. Σκοπός του παιχνιδιού είναι ο *Kodu* να στείλει τη μπάλα στην εστία που βρίσκεται στα αριστερά του.

Δείτε το παράδειγμα *06_03.kodu*

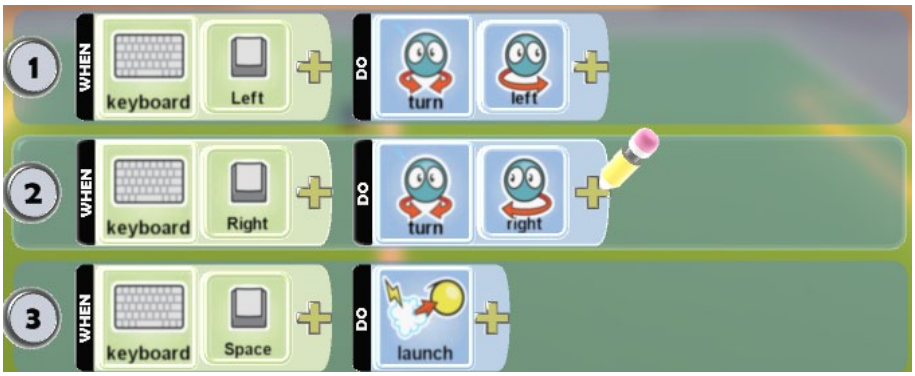
Φορτώστε το παιχνίδι **[06_07.kodu]** κι ας ξεκινήσουμε τον προγραμματισμό του *Kodu*. Ο *Kodu* θα πρέπει να αξιοποιήσει τον αισθητήρα *Πληκτρολόγιο (Keyboard)* για να καταλαβαίνει πότε ο χρήστης έχει πατήσει τα πλήκτρα αριστερό και δεξί βέλος έτσι ώστε να *Γυρνά (Turn)* καθώς και το *Κενό (Space)* για να εκτελεί το σουτ. Για την κίνηση του *Kodu* λοιπόν πρέπει να δημιουργήσουμε τις εξής συμπεριφορές:

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο Αριστερά] **ΤΟΤΕ** [Στρίψε][Αριστερά]

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο Δεξιά] **ΤΟΤΕ** [Στρίψε][Δεξιά]

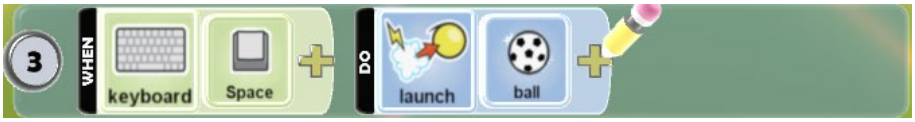
Πώς όμως μπορούμε να προγραμματίσουμε τον *Kodu* έτσι ώστε να χτυπάει τη μπάλα και αυτή να εκτοξεύεται στην εστία προκειμένου να βάλει γκολ; Θα χρησιμοποιήσουμε την ενέργεια ***Εκτινάζω (Launch)***. Μάλιστα, δεν θα πρέπει να τη συνδυάσουμε με κανένα προσδιοριστικό. Έτσι λοιπόν, μπορούμε να προσθέσουμε στον *Kodu* την εξής συμπεριφορά:

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το Κενό] **ΤΟΤΕ** [Εκτίναξε]

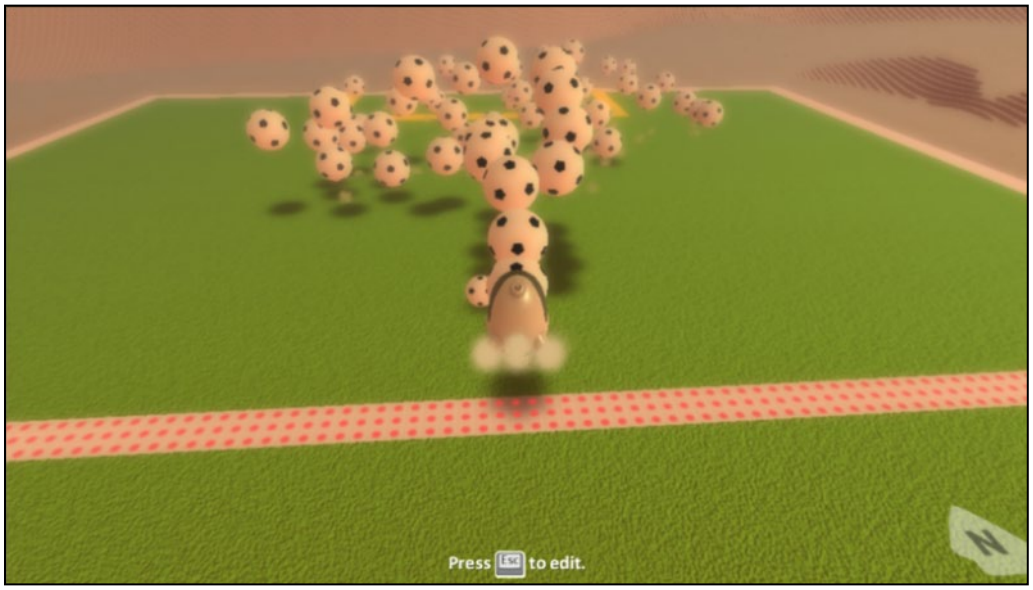


Αποθηκεύστε το παιχνίδι και τρέξτε το.

Μήπως σας φαίνεται παράξενο που χρησιμοποιούμε την ***Εκτινάζω (Launch)*** χωρίς προσδιοριστικό; Για να πάμε να δούμε τι θα συμβεί άμα προσθέσουμε και το προσδιοριστικό *Μπάλα (Ball),* αφού στην ουσία αυτό είναι το αντικείμενο που επιθυμούμε να ***Εκτινάξει (Lanch)*** ο *Kodu*. Ας ανοίξουμε λοιπόν τον επεξεργαστή εντολών και ας συμπληρώσουμε το προσδιοριστικό αυτό στην ενέργεια του *Kodu*.



Αποθηκεύστε τις αλλαγές και τρέξτε πάλι το πρόγραμμα. Αυτή τη φορά κρατείστε πατημένο το πλήκτρο *Κενό (Space).* Όσο κρατάτε πατημένο το πλήκτρο, ο κόσμος θα πρέπει να μοιάζει όπως φαίνεται στην εικόνα που ακολουθεί:



Γέμισε ο τόπος μπάλες! Επιπλέον παρατηρείστε ότι το αρχικό αντικείμενο *Μπάλα (Ball)* που έχουμε εισάγει στον κόσμο του παιχνιδιού δεν έχει κινηθεί καθόλου. Αυτό συμβαίνει διότι στην περίπτωση που προσθέσουμε ως προσδιοριστικό κάποιο από τα αντικείμενα που μας προσφέρει το MSKodu στην ενέργεια ***Εκτινάζω (Launch)***, ο *Kodu* αναπαράγει το αντικείμενο αυτό και το εκτοξεύει ασχέτως με ό,τι βρίσκεται μπροστά του.

Θα συμφωνείτε τώρα ότι η ενέργεια αυτή είναι αρκετά παράξενη! Η ενέργεια αυτή συνδυάζεται με προσδιοριστικά που αφορούν τη *δύναμη* και το *ύψος* που εκτοξεύει ο *Kodu* το αντικείμενο. Για παράδειγμα αν επιθυμούσαμε ο *Kodu* να ρίξει ψηλοκρεμαστό σουτ τότε θα προσθέταμε στην ενέργεια ***Εκτινάζω (Lanch)*** τα προσδιοριστικά ***Δυνατά (Strong)*** και ***Ψηλά (High)***, ενώ αν θέλαμε

να εκτινάξει ο Kodu ένα *Μήλο (Apple)* προς ένα αερόπλοιο, τότε θα επιλέγαμε το προσδιοριστικό ***Πάνω (Up)***. Προσπαθήστε να απαντήσετε μόνοι σας: Ο *Kodu* μπορεί να δημιουργήσει και να εκτινάξει οποιοδήποτε αντικείμενο; Σημαντικό να τονίσουμε ότι η Δύναμη Κλωτσιάς και ο Ρυθμός Κλωτσιάς είναι δύο ιδιότητες που επηρεάζουν τη λειτουργία της ενέργειας *Εκτινάζω (Launch)*.

Μόνο Μια Φορά (Once)

Κάντε ένα πείραμα. Δημιουργήστε τη συμπεριφορά:

ΌΤΑΝ[στο Πληκτρολόγιο][πατηθεί το πλήκτρο Κενό] **ΤΟΤΕ** [Εκτόξευε][Σφαιρίδια]

Παρατηρήστε ότι για όσο χρονικό διάστημα πατάμε το πλήκτρο *Κενό*, ο *Kodu* θα εκτοξεύει σφαιρίδια. Στην περίπτωση που προσθέσουμε το προσδιοριστικό *Μόνο Μία Φορά (Once)*, τότε για όσο χρονικό διάστημα κι αν κρατήσουμε πατημένο το πλήκτρο ο *Kodu* θα εκτοξεύσει μόνο ένα σφαιρίδιο. Προσοχή όμως! Αυτό δεν σημαίνει ότι θα είναι και η μοναδική φορά. Αν ξαναπατήσουμε το πλήκτρο *Κενό (Space)* τότε ο *Kodu* θα εκτοξεύσει και πάλι ένα σφαιρίδιο.

6.4.4 Ενέργεια Αρπάζω (Grab)

Η ενέργεια ***Αρπάζω (Grab)*** επιτρέπει στα αντικείμενα να κρατήσουν ένα αντικείμενο. Η ενέργεια αυτή ανήκει στην οικογένεια ενεργειών *Κρατάω (Holding)* και προφανώς αποτελεί προϋπόθεση για να έχει αξία ο αισθητήρας ***Κουβαλιέμαι (Held by)***. Έτσι λοιπόν, ένας κακός *Μηχανάκιας (Cycle)* μπορεί να αρπάξει τη φίλη του *Kodu* όταν την ακουμπήσει ή ο Kodu να αρπάζει νομίσματα για να κερδίσει το παιχνίδι. Ένα παράδειγμα εντολής με χρήση της ενέργειας:

Τι είχε ο προγραμματιστής στο μυαλό του:

ΌΤΑΝ[Ακούω][τον Kodu] **ΤΟΤΕ** [Άρπαξέ][Τον]

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Αρπάζω, αιχμαλωτίζω, κρατάω, πιάνω}

Όπως σας υποσχεθήκαμε, στο τέλος αυτής της παραγράφου θα έχουμε ένα ολοκληρωμένο παιχνίδι με πέναλτι! Ας συμπληρώσουμε το προηγούμενο παιχνίδι προσθέτοντας στο στάδιο έναν *Μηχανάκια (Cycle),* ο οποίος θα είναι ο τερματοφύλακας. Συνεπώς, ο κόσμος του παιχνιδιού τώρα θα αποτελείται από τον *Kodu*, την *Μπάλα (Ball)* και τον *Μηχανάκια (Cycle)*. Σκοπός του παιχνιδιού είναι ο χειριστής του *Kodu* να βάλει γκολ στην εστία του *Μηχανάκια (Cycle)*. Ας ρίξουμε μία ματιά στον κόσμο έτσι όπως έχει διαμορφωθεί τώρα.

Η συμπεριφορά του *Kodu* δεν χρειάζεται να αλλάξει. Συνεπώς δεν χρειάζεται να προγραμματίσουμε τον *Kodu* εκ νέου. Θα πρέπει όμως να προγραμματίσουμε τον *Μηχανάκια (Cycle)* έτσι ώστε να συμπεριφέρεται σαν πραγματικός τερματοφύλακας. Ο *Μηχανάκιας (Cycle)* θα πρέπει με κάποιον τρόπο να αντιλαμβάνεται την *Μπάλα (Ball)* που εκτινάζει ο *Kodu* προς την

εστία έτσι ώστε να κατευθυνθεί προς αυτήν και να την πιάσει. Άρα, με όσα έχουμε πει μέχρι τώρα, πρέπει να τον προγραμματίσουμε είτε με τον αισθητήρα *Βλέπω (See)* είτε με τον αισθητήρα *Ακούω (Hear)*. Εμείς θα επιλέξουμε τον πρώτο με προσδιοριστικό την *Μπάλα (Ball)*. Θα πρέπει, ωστόσο, να προσθέσουμε κάποια επιπλέον προσδιοριστικά στον αισθητήρα. Αν προγραμματίσουμε τον *Μηχανάκια (Cycle)* μόνο να αντιλαμβάνεται την *Μπάλα (Ball)*, τότε, όταν τρέξουμε το παιχνίδι, ο *Μηχανάκιας (Cycle)* θα κατευθυνθεί προς αυτήν οπουδήποτε κι αν βρίσκεται. Η συμπεριφορά αυτή δεν είναι σωστή. Θα πρέπει να προγραμματίσετε τον αισθητήρα έτσι ώστε ο *Μηχανάκιας (Cycle)* να την αντιλαμβάνεται μόνο όταν κινείται και είναι κοντά σε αυτόν. Όταν πλέον ο *Μηχανάκιας (Cycle)* θα έχει φτάσει στην *Μπάλα (Ball),* θα πρέπει με κάποιον τρόπο να την αποκρούσει. Επομένως, θα πρέπει πρώτα να αισθανθεί ότι έχει έρθει σε επαφή μαζί της και στη συνέχεια να την κρατήσει προκειμένου να μην μπει γκολ.

Για να δούμε λοιπόν πώς μπορούμε να προγραμματίσουμε την παραπάνω συμπεριφορά. Το γεγονός ότι ο *Μηχανάκιας (Cycle)* έχει αγγίξει την *Μπάλα (Ball)* μπορεί να το αντιληφθεί μέσω του αισθητήρα *Πέφτω Πάνω (Bump)* με προσδιοριστικό την *Μπάλα (Ball)*. Για να την κρατήσει, θα πρέπει να χρησιμοποιήσουμε την ενέργεια ***Αρπάζω (Grab)*** με προσδιοριστικό το ***Αυτό (It)***. Συνεπώς, για τη συμπεριφορά του *Μηχανάκια* δημιουργούμε τις εξής εντολές:

ΌΤΑΝ[Δω][Μπάλα][που Κινείται][Κοντά μου] **ΤΟΤΕ** [Κινούμαι][Προς αυτή]

ΌΤΑΝ[Πέσω Πάνω][σε Μπάλα] **ΤΟΤΕ** [Αρπάζω][την Μπάλα]

Αποθηκεύστε τις αλλαγές και τρέξτε το παιχνίδι.

Η ενέργεια ***Αρπάζω (Grab)*** μπορεί να συνδυαστεί μόνο με ένα προσδιοριστικό, το *Αυτό (It)*, πράγμα που σημαίνει ότι το αντικείμενο αρπάζει αυτό που έχει «αντιληφθεί» με κάποιον αισθητήρα. Μπορείτε να αλλάξετε το παιχνίδι ώστε ο χρήστης να μπορεί να εκτελεί επαναλαμβανόμενα πέναλτι;

6.5 Ο Επεξεργαστής Εντολών (Editor)

Η συνεχής ενασχόληση με τον προγραμματισμό μας οδηγεί στην ανάπτυξη πιο σύνθετων προγραμμάτων. Αυτό σημαίνει ότι θα ξοδεύουμε περισσότερη ώρα στον *επεξεργαστή εντολών (editor)* προκειμένου να τα υλοποιήσουμε. Έτσι, οι σχεδιαστές του MSKodu προκειμένου να κάνουν την προγραμματιστική μας εμπειρία πιο εύκολη, έχουν προσθέσει στον *επεξεργαστή εντολών (editor)* κάποιες λειτουργίες. Αξίζει λοιπόν να του ρίξουμε μια πιο προσεκτική ματιά. Φορτώστε το παιχνίδι **[06_08.kodu]**. Ο κόσμος του παιχνιδιού αποτελείται από τον *Kodu* και έξι *Χελιδονόψαρα (Flyfish)* διαφορετικού χρώματος το καθένα. Σκοπός του παιχνιδιού είναι ο *Kodu*,

Δείτε το παράδειγμα *06_08.kodu*

τον οποίο χειρίζεται ο παίκτης από το πληκτρολόγιο, να σκοτώσει όλα τα *Χελιδονόψαρα (Flyfish)* που υπάρχουν, εκτοξεύοντας πυραύλους. Ο κόσμος του παιχνιδιού όπως περιγράφηκε φαίνεται στην ακόλουθη εικόνα:

Τα αντικείμενα του παιχνιδιού είναι ήδη προγραμματισμένα έτσι ώστε να εμφανίζουν τις εξής συμπεριφορές:

Kodu:

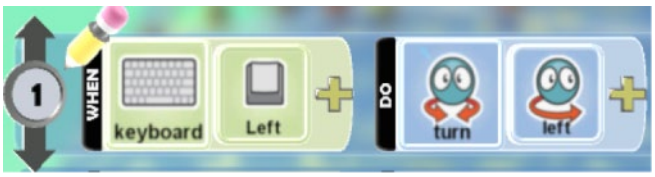
Χελιδονόψαρα:

Ας δούμε λοιπόν ποιες λειτουργίες μας προσφέρει ο *επεξεργαστής εντολών (editor)*.

Αντιγραφή Γραμμής (Copy Row) και Επικόλληση Γραμμής (Paste Row)

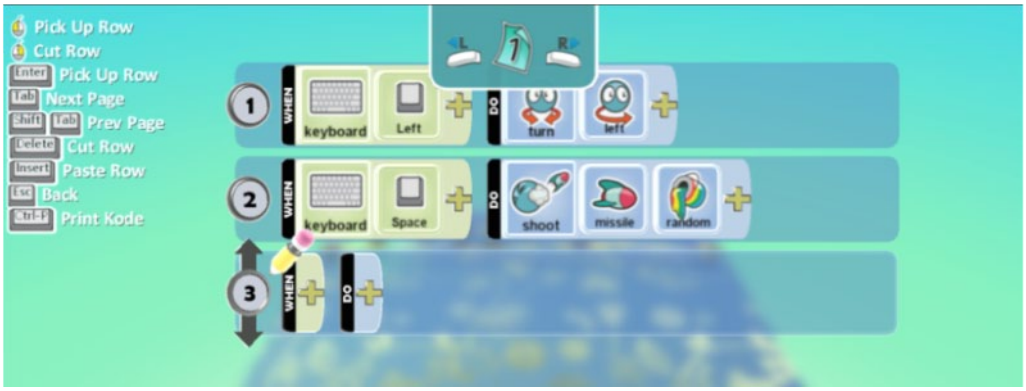
Η λειτουργία αυτή μας επιτρέπει να αντιγράψουμε γραμμές εντολών και να τις προσθέσουμε τόσο στο αντικείμενο από το οποίο τις πήραμε όσο και σε κάποιο άλλο αντικείμενο. Όπως μπορεί να έχετε ήδη παρατηρήσει, ο *Kodu* μπορεί να στρίψει μόνο αριστερά. Ας τον προγραμματίσουμε να μπορεί να στρίψει και δεξιά. Δε χρειάζεται να συντάξουμε από την αρχή τη συγκεκριμένη εντολή. Απλά θα αντιγράψουμε την εντολή που αφορά το να στρίβει αριστερά και θα αλλάξουμε τα προσδιοριστικά της:

1. Επιλέγουμε τη γραμμή εντολής που επιθυμούμε να αντιγράψουμε, πατώντας το αριστερό κουμπί του ποντικιού μας στον αριθμό που βρίσκεται στην αρχή της. Καταλαβαίνουμε ότι έχουμε επιλέξει την εκάστοτε γραμμή από τα βέλη με έντονο γκρι που εμφανίζονται μπροστά από την εντολή.

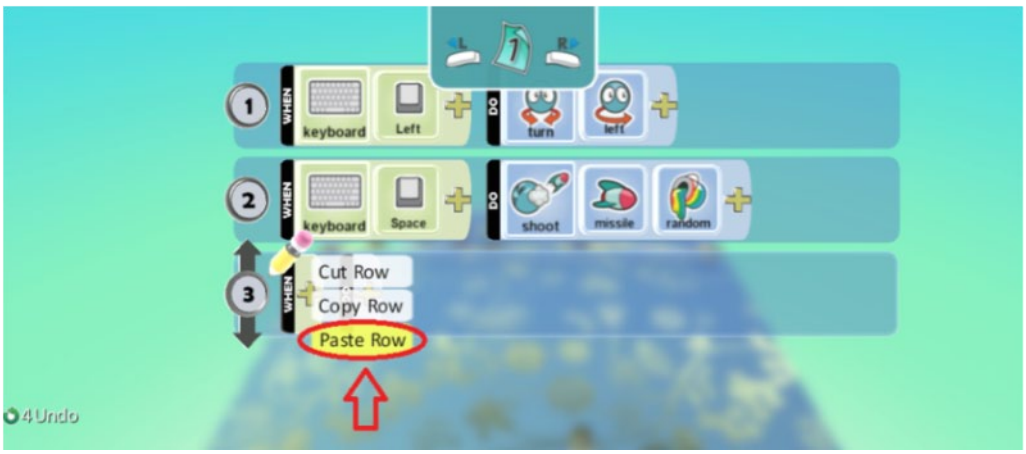


2. Πατάμε το δεξί κουμπί του ποντικιού και επιλέγουμε *Αντιγραφή Γραμμής (Copy Row)*

3. Πατάμε το αριστερό κουμπί του ποντικιού στον αριθμό μιας κενής γραμμής προκειμένου να την επιλέξουμε.



4. Πατάμε το δεξί πλήκτρο του ποντικιού και επιλέγουμε την *Επικόλληση Γραμμής (Paste Row)*



5. Στον επεξεργαστή εντολών εμφανίζεται ένα αντίγραφο της γραμμής.

6. Το μόνο που μένει τώρα είναι να αλλάξουμε τα προσδιοριστικά του αισθητήρα (από αριστερό βελάκι σε δεξί βελάκι) και της ενέργειας (από *Αριστερά (Left)* σε *Δεξιά (Right)*). Πατάμε πάνω στα αντίστοιχα προσδιοριστικά και επιλέγουμε τις νέες τιμές τους.

Αποκοπή Γραμμής (Cut Row) και Επικόλληση γραμμής (Paste Row)

Η λειτουργία της *Αποκοπής Γραμμής (Cut Row)* μπορεί να μας φανεί πολύ χρήσιμη εάν θέλουμε να μετακινήσουμε μία εντολή από ένα αντικείμενο σε ένα άλλο (π.χ. αν έχουμε μπερδευτεί και προγραμματίζαμε άλλο αντικείμενο από αυτό που θέλαμε). Φανταστείτε να είχαμε μπερδευτεί και να είχαμε προσθέσει την εξής συμπεριφορά στον *Kodu* και όχι στο *Χελιδονόψαρο (Flyfish)*:

Με την *Αποκοπή Γραμμής (Cut Row)* θα μπορούσαμε να μεταφέρουμε τη συμπεριφορά από τον *Kodu* στο *Χελιδονόψαρο (Flyfish)* με την εξής απλή διαδικασία:

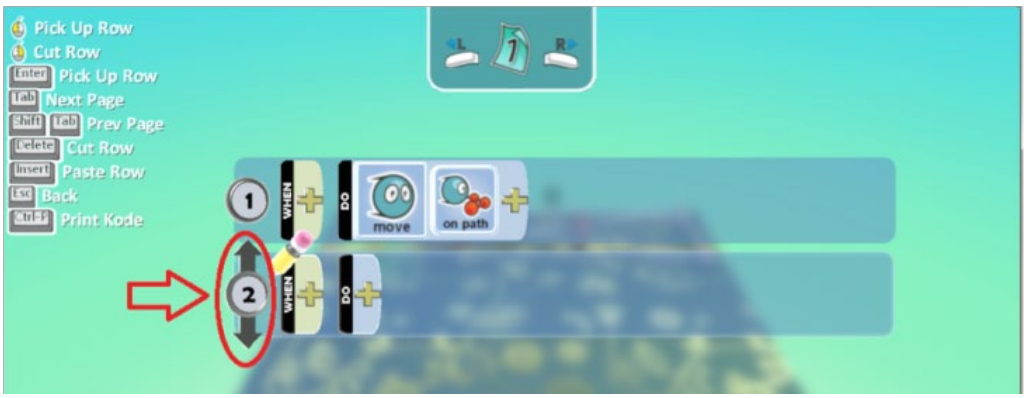
1. Επιλέγουμε τη γραμμή εντολής που επιθυμούμε να αποκόψουμε, πατώντας το αριστερό κουμπί του ποντικιού μας στον αριθμό που βρίσκεται στην αρχή της. Καταλαβαίνουμε ότι έχουμε επιλέξει τη γραμμή από τα βέλη με έντονο γκρι που εμφανίζονται.

2. Πατάμε το δεξί κουμπί του ποντικιού μας και επιλέγουμε την *Αποκοπή Γραμμής (Cut Row)*.

3. Παρατηρούμε ότι η γραμμή εξαφανίζεται.

4. Βγαίνουμε από τον επεξεργαστή εντολών του *Kodu* πατώντας το πλήκτρο Escape και ανοίγουμε τον επεξεργαστή εντολών του *Χελιδονόψαρου (Flyfish)*.

5. Πατάμε το αριστερό κουμπί του ποντικιού στον αριθμό μιας κενής γραμμής προκειμένου να την επιλέξουμε.



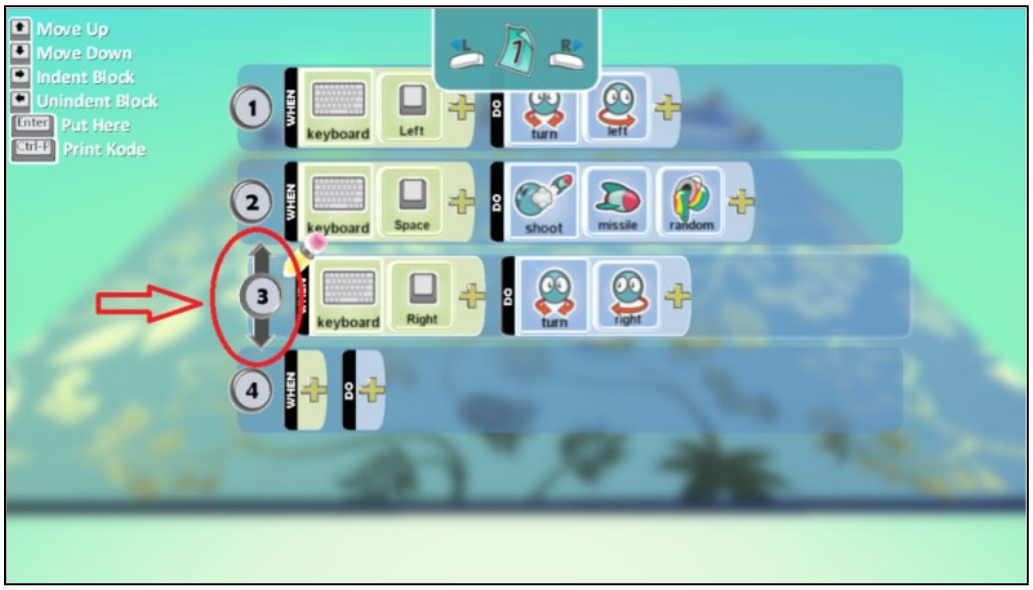
6. Πατάμε το δεξί κουμπί του ποντικιού και επιλέγουμε την *Επικόλληση Γραμμής (Paste Row)*

7. Η γραμμής εντολής που αποκόψαμε επικολλήθηκε με επιτυχία.

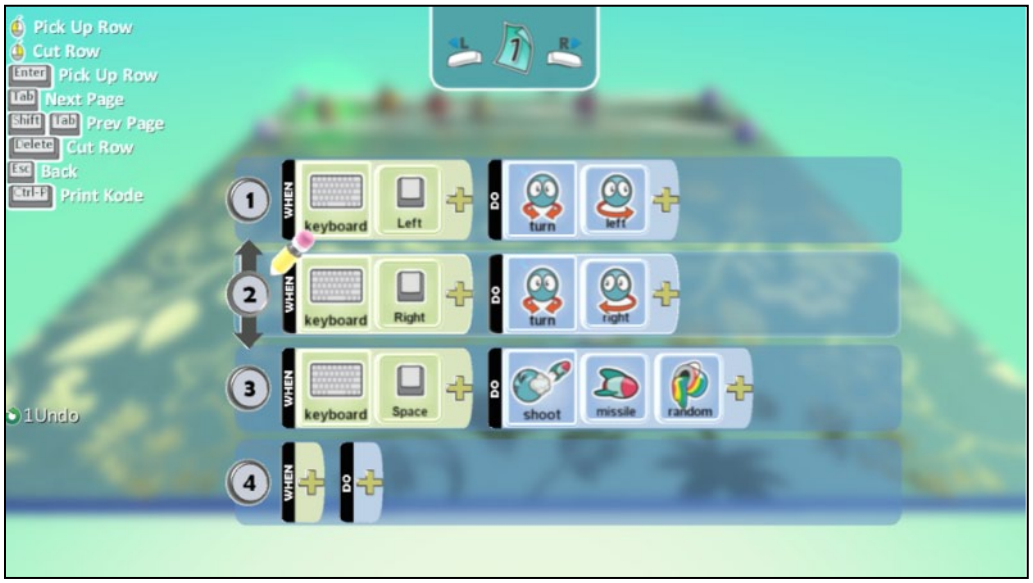
Μετακίνηση Γραμμών Εντολών

Χρησιμοποιούμε τη μετακίνηση γραμμών για να αλλάξουμε τη σειρά των εντολών είτε για την καλύτερη αναγνωρισιμότητα του προγράμματός μας (π.χ. για να συγκεντρώσουμε όλες τις συμπεριφορές μετακίνησης σε ένα σημείο και όλες τις συμπεριφορές πυροβολισμών σε ένα άλλο) είτε για να αλλάξουμε την προτεραιότητα των συμπεριφορών που έχουμε προγραμματίσει. Ας μετακινήσουμε λοιπόν την τρίτη γραμμή εντολής του *Kodu* και ας την τοποθετήσουμε κάτω από την πρώτη.

1. Επιλέγουμε τη γραμμή που επιθυμούμε να μετακινήσουμε είτε πατώντας το πλήκτρο Enter είτε κρατώντας πατημένο το αριστερό κουμπί του ποντικιού. Παρατηρούμε ότι στον αριθμό μπροστά από τη γραμμή εμφανίζονται τα χαρακτηριστικά γκρι βέλη, όπως και το γεγονός ότι η γραμμή βρίσκεται πλέον πιο δεξιά από ότι συνήθως.



2. Εάν έχουμε επιλέξει τη γραμμή πατώντας το πλήκτρο Enter, αρκεί να χρησιμοποιήσουμε τα βέλη πάνω ή κάτω του πληκτρολογίου (πάνω για να αυξήσουμε την προτεραιότητα και κάτω για να την ελαττώσουμε). Στην περίπτωση που έχουμε χρησιμοποιήσει το ποντίκι, αρκεί να σύρουμε τη γραμμή είτε προς τα πάνω (αύξηση προτεραιότητας) είτε προς τα κάτω (μείωση προτεραιότητας).

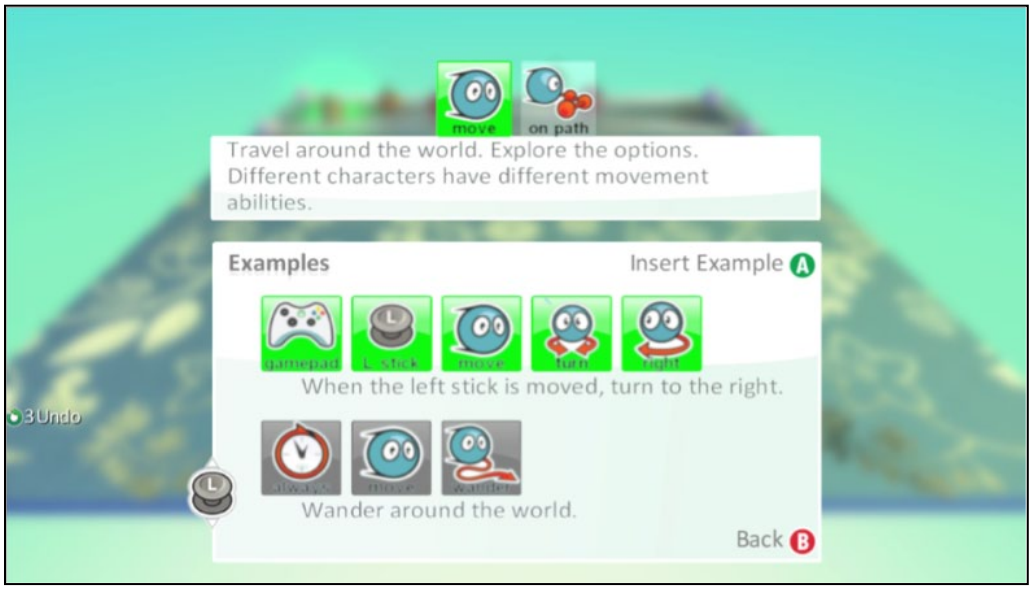


Παρατηρείστε τους αριθμούς που υπάρχουν δίπλα από κάθε γραμμή. Οι αριθμοί δεν υπάρχουν μόνο για να βλέπετε πόσες γραμμές εντολών έχετε εισάγει. Δηλώνουν και την προτεραιότητα των ενεργειών. Μπορεί οι ενέργειες να εκτελούνται παράλληλα εφόσον συμβούν τα αντίστοιχα γεγονότα, ωστόσο κάθε μία έχει διαφορετική προτεραιότητα ανάλογα με τη σειρά που την έχουμε εισάγει.

Βοήθεια

Ο επεξεργαστής εντολών είναι σχεδιασμένος έτσι ώστε να μας βοηθά να συντάξουμε σωστά τις συμπεριφορές των αντικειμένων. Ωστόσο, μερικές φορές κατά τη διάρκεια που προγραμματίζουμε ένα αντικείμενο, μπορεί να ξεχάσουμε τη σημασία μιας εντολής. Σίγουρα αν ανατρέξουμε σε αυτό το βιβλίο θα ήταν ένας τρόπος για να λύσουμε την απορία μας, αλλά δεν θα ήταν και ο πιο γρήγορος. Αρκεί να αφήσουμε τον κέρσορα του ποντικιού μας πάνω στην εντολή που επιθυμούμε να χρησιμοποιήσουμε και θα εμφανιστεί μία γκρι φούσκα που θα περιγράφει τη σημασία της εντολής.

Αν πάλι δεν είμαστε σίγουροι ότι έχουμε καταλάβει πώς ακριβώς συντάσσεται η εντολή και ποια είναι η λειτουργία της, αρκεί να πατήσουμε το πλήκτρο Y στο πληκτρολόγιο. Αμέσως θα ξεπροβάλλει μπροστά μας μια σειρά από παραδείγματα με την εντολή που έχουμε επιλέξει.



Περίληψη

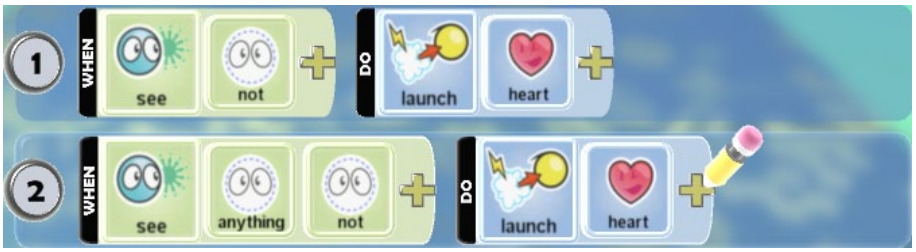
Στο κεφάλαιο αυτό είδαμε τα αντικείμενα που μας προσφέρει το MSKodu και εξετάσαμε πως αλληλεπιδρούν μεταξύ τους. Πιο συγκεκριμένα, μάθαμε να αλλάζουμε τις ιδιότητες των αντικειμένων από το μενού *Αλλαγή Ιδιοτήτων (Change Settings)*, ώστε τα αντικείμενά μας να έχουν ιδιότητες που ταιριάζουν καλύτερα στα παιχνίδια μας (π.χ. ταχύτητα, επιτάχυνση,ταχύτητα πυραύλου, τριβή, μέγεθος αντικειμένου κτλ.). Μελετήσαμε επίσης εκτενέστερα τους αισθητήρες με τους οποίους τα αντικείμενα μπορούν να αντιλαμβάνονται τα υπόλοιπα αντικείμενα ενός παιχνιδιού και να αντιδρούν σε αυτά (*Βλέπω (See), Πέφτω Πάνω (Bump), Πετυχαίνω (Shot), Έχω (Got), Κουβαλιέμαι(Held By)*). Οι αισθητήρες αυτοί θυμίζουν κατά πολύ τις αισθήσεις των ανθρώπων! Εξετάσαμε ταυτόχρονα πιο προσεκτικά μια σειρά από χρήσιμες ενέργειες των αντικειμένων (*Τρώω(Eat), Εξαφανίζω (Vanish), Εκτινάζω (Launch), Αρπάζω(Grab)*). Τέλος, διερευνήσαμε τον *επεξεργαστή εντολών (editor)* ώστε να γράφουμε τις συμπεριφορές μας πιο εύκολα και γρήγορα και να φροντίζουμε για την αναγνωσιμότητα των προγραμμάτων μας.

Ερωτήσεις

1. Στο παιχνίδι **[06_07.kodu]** που δημιουργήσατε στην παράγραφο 6.4 μπορείτε να εντοπίσετε ποιες ιδιότητες του *Μηχανάκια* – τερματοφύλακα επηρεάζουν τη συμπεριφορά του;

2. Μπορείτε να αναφέρετε τους λόγους για τους οποίους οι παρακάτω συμπεριφορές είναι ή δεν είναι ίδιες;

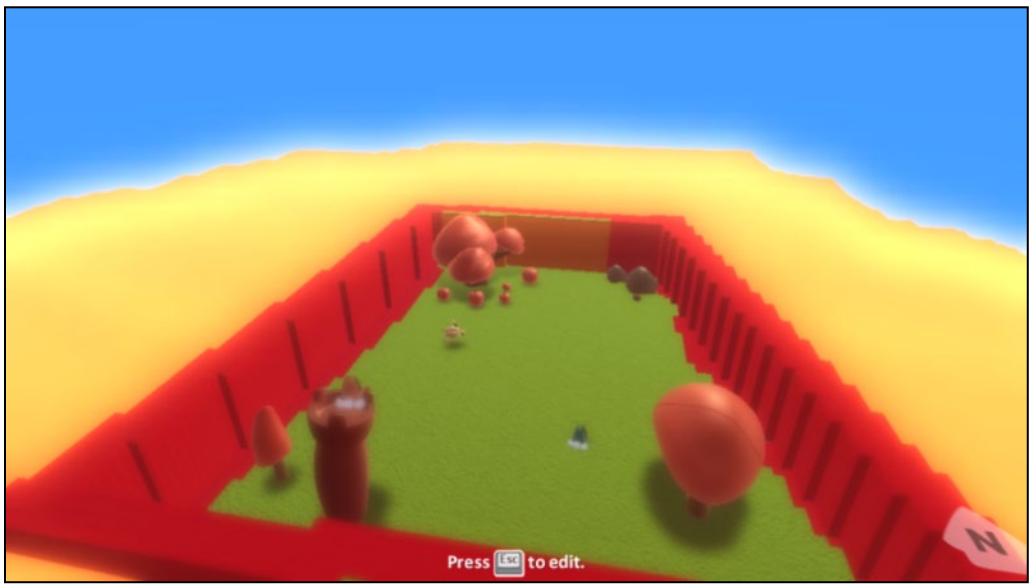
3. Μπορείτε να καταλάβετε αν οι παρακάτω συμπεριφορές είναι ίδιες;



4. Στον κόσμο ενός παιχνιδιού υπάρχουν ο *Kodu* και ένας *Μηχανάκιας*. Ποια από τις δύο συμπεριφορές θα εκτελέσει ο *Kodu*;

Δραστηριότητες

1. Δημιουργείστε έναν κόσμο όπως φαίνεται στην παρακάτω εικόνα.



Εισάγεται στον κόσμο τα εξής αντικείμενα:

- τον μπλε *Kodu* τον οποίο χειρίζεται ο παίκτης
- ένα ροζ *Kodu,* που είναι η φίλη του *Kodu*
- μία άσπρη *Χελώνα (Turtle)*
- ένα κόκκινο *Δέντρο (Tree)*
- ένα *Κάστρο (Castle)* πάνω στο οποίο βρίσκεται ο ροζ *Kodu*
- πέντε κόκκινα *Μήλα (Apples)* διαφορετικού μεγέθους
- τρεις *Βράχους (Rocks)* διαφορετικού μεγέθους

Σκοπός του παιχνιδιού είναι ο *Kodu* να καταστρέψει τη *Χελώνα (Turtle),* εκτοξεύοντας πυραύλους, προκειμένου να σώσει τη φίλη του. Η *Χελώνα (Turtle),* όταν αντιλαμβάνεται τον ήχο που κάνει ο *Kodu*, τον ακολουθεί πολύ αργά και, όταν τον ακουμπήσει, τον εκτινάσσει μακριά. Όταν η *Χελώνα (Turtle)* σκοτωθεί, τότε η ροζ *Kodu* πετάει μια *Καρδιά (Heart)* στον *Kodu*, την οποία και πρέπει να ακουμπήσει ο *Kodu* για να τερματιστεί με επιτυχία το παιχνίδι. Προκειμένου να κάνετε το παιχνίδι να φαίνεται ρεαλιστικό, προγραμματίστε τα *Μήλα (Apples)* και τους *Βράχους (Rocks)* έτσι ώστε, όταν χτυπηθούν από κάποιο πύραυλο, να εξαφανιστούν από τον κόσμο. Το *Δέντρο (Tree)* και το *Κάστρο (Castle)* αποτελούν στοιχεία του περιβάλλοντος και δεν θα παρουσιάζουν κάποια συμπεριφορά. Ωστόσο, δεν θα πρέπει να καταστρέφονται από τους πυραύλους του *Kodu* σε περίπτωση που χτυπηθούν.

2. Δημιουργείστε έναν υποθαλάσσιο κόσμο όπως τον φαντάζεστε. Στόχος σας είναι να δημιουργήσετε ένα παιχνίδι ναυμαχίας. Τα αντικείμενα του παιχνιδιού είναι *Υποβρύχια (Subs)* και *Πλοία (Ships)*. Ο παίκτης θα χειρίζεται ένα μπλε *Υποβρύχιο (Sub)* το οποίο θα το ακολουθεί μία ομάδα από άλλα δύο μπλε *Υποβρύχια (Subs)* τα οποία και θα αποτελούν το συμμαχικό στόλο του παίκτη. Στόχος του παίκτη θα είναι να καταστρέψει τον εχθρικό στόλο ο οποίος αποτελείται από πέντε κόκκινα *Υποβρύχια (Subs)* και τέσσερα κόκκινα *Πλοία (Ships)*.

Προγραμματίστε το υποβρύχιο του παίκτη έτσι ώστε να μπορεί να τον κινεί από το πληκτρολόγιο. Επιπλέον ο παίκτης, όταν πατάει το αριστερό κουμπί του ποντικιού, το *Υποβρύχιό (Sub)* του θα εκτοξεύει πυραύλους προς την κατεύθυνσή του ενώ όταν πατάει το δεξί κουμπί του ποντικιού θα εκτοξεύει σφαιρίδια προς τα επάνω. Επιπλέον προγραμματίστε το έτσι ώστε, όταν αντιλαμβάνεται ηχητικά ότι από πάνω του βρίσκεται κάποιο *Πλοίο (Ship),* τότε να λάμπει με κόκκινο χρώμα.

Τα φιλικά προς τον παίκτη μπλε *Υποβρύχια (Subs)* θα πρέπει να τον ακολουθούν, ενώ, όταν αντιλαμβάνονται παρουσία κόκκινων *Υποβρυχίων (Subs),* θα πρέπει να εκτοξεύουν πυραύλους εναντίων τους. Τα κόκκινα *Υποβρύχια (Subs)* θα πρέπει να περιπλανιούνται στο βυθό και, όταν αντιλαμβάνονται την παρουσία κάποιου μπλε *Υποβρυχίου (Sub),* θα πρέπει να εκτοξεύουν κόκκινους πυραύλους εναντίων του. Τέλος, τα κόκκινα *Πλοία (Ships)* θα πρέπει συνέχεια να εκτοξεύουν μαύρους πυραύλους προς τα κάτω.

Κεφάλαιο 7ο: Αλληλεπιδρώ με τον κόσμο!

Φαντάζεστε ένα παιχνίδι αγώνων ταχύτητας, όπου το αυτοκίνητο που οδηγείτε μπορεί να πηγαίνει με την ίδια ταχύτητα μέσα στα όρια της πίστας αλλά και εκτός αυτής (όταν πατάει πάνω σε γρασίδι ή σε χώμα, ή όταν συγκρούεται σε φράχτες); Κάτι τέτοιο δε θα άλλαζε εντελώς τον χαρακτήρα του παιχνιδιού; Πλέον δεν θα έπρεπε να οδηγείτε το αυτοκίνητο μέσα στο δρόμο αποφεύγοντας τα εμπόδια, αλλά απλά θα σημαδεύατε την γραμμή τερματισμού και θα πατούσατε γκάζι (πράγμα που δεν μας κινούσε και τόσο το ενδιαφέρον).

Στο κεφάλαιο 4 μελετήσαμε τον τρόπο δημιουργίας του κόσμου των παιχνιδιών μας. Σε αυτό το κεφάλαιο θα μιλήσουμε για το πως οι χαρακτήρες των παιχνιδιών μας μπορούν να αλληλεπιδράσουν με τον κόσμο αυτό. Δηλαδή θα προσπαθήσουμε να εξηγήσουμε τον τρόπο με τον οποίο η διαμόρφωση και τα χαρακτηριστικά του κόσμου (τα οποία καθορίζουμε εμείς) επηρεάζουν την κίνηση και την συμπεριφορά των αντικειμένων καθώς και πως μπορούμε να αξιοποιήσουμε τα χαρακτηριστικά αυτά στον προγραμματισμό τους. Ας μην ξεχνάμε πως η διαμόρφωση του κόσμου γίνεται με σκοπό ο Kodu και υπόλοιπα αντικείμενα να αλληλεπιδρούν με αυτόν.

7.1 Ο κόσμος, οι ιδιότητες του κόσμου

Παρόλο που την σημερινή εποχή τα γραφικά παιχνιδιών είναι εκπληκτικά, πολλά παιχνίδια απορρίπτονται από τους χρήστες. Γιατί συμβαίνει άραγε αυτό; Όπως θα δούμε στην συνέχεια του κεφαλαίου, ένας από τους παράγοντες αποτελεί η διαμόρφωση του κόσμου στο οποίο διαδραματίζεται το παιχνίδι μας. Προτού ξεκινήσει ο προγραμματιστής να ενσωματώνει χαρακτήρες στον κόσμο, πρέπει να αποφασίσει ποια είναι τα κατάλληλα χαρακτηριστικά τα οποία θα αποδώσει σε αυτόν. Στόχοι του συνήθως είναι να δημιουργήσει την κατάλληλη αισθητική στο παιχνίδι και να το κάνει όσο πιο ρεαλιστικό γίνεται ως προς τις αισθήσεις του χρήστη.

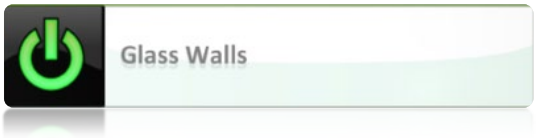
Το MSKodu προσφέρει μια γκάμα από επιλογές που μας επιτρέπουν να διαμορφώσουμε τον κόσμο του παιχνιδιού που θέλουμε να δημιουργήσουμε. Επιλέγοντας ***Ιδιότητες Κόσμου (Change World Settings)*** όπως φαίνεται στην παρακάτω εικόνα, προκύπτει το μενού επιλογών ***Change World Settings Menu*** το οποίο απαρτίζεται από σειρά ιδιοτήτων τις οποίες θα εξερευνήσουμε μία μία μέσα από παραδείγματα.

Θα ομαδοποιήσουμε τις διάφορες επιλογές ώστε να μπορείτε να τις ανακαλείτε ευκολότερα όταν τις χρειαστείτε.

7.1.1 Ιδιότητες Διαμόρφωσης Φυσικού περιβάλλοντος

Όταν αναφερόμαστε στη Φύση, αυτό που έρχεται στο μυαλό μας είναι η γη, το νερό, ο ουρανός και άλλα αντίστοιχα στοιχεία τα οποία θα πρέπει να μην θεωρούμε ως δεδομένα στα παιχνίδια μας στο MSKodu, καθώς μπορούμε να τα αλλάξουμε δραματικά!

Γυάλινος Τοίχος



Θεωρητικά κατά την κατασκευή ενός κόσμου μπορούμε να επεκτείνουμε το έδαφος όσο θέλουμε, αλλά αυτό αναγκαστικά κάπου θα τελειώνει. Με μία πρώτη ματιά αυτό δεν φαίνεται να μας επηρεάζει. Τι γίνετε όμως όταν ο πρωταγωνιστής μας φτάσει στο όριο, στην άκρη του εδάφους; Όπως και στην πραγματικότητα, εάν ο Kodu ξεπεράσει το όριο του εδάφους, θα πέσει στο κενό.

Ενεργοποιώντας την επιλογή *Γυάλινος Τοίχος* (γίνεται πράσινο το αντίστοιχο εικονίδιο), τοποθετούμε στον περίγυρο (στα όρια) του κόσμου μας, μια διάφανη *Γυάλινη Περίφραξη*. Αυτή εμποδίζει τον Kodu να πέσει στο κενό όταν φτάσει στα άκρα του κόσμου.

Στην πρώτη εικόνα, έχοντας την επιλογή Γυάλινος Τοίχος (Glass Walls) απενεργοποιημένη, παρατηρούμε τον Kodu να πέφτει στο κενό αφού έχει ξεπεράσει τα όρια του εδάφους. Αντίθετα, στη δεύτερη εικόνα όπου η επιλογή είναι ενεργοποιημένη ο Kodu συγκρούεται με τον Γυάλινο τοίχο (γι' αυτό και φαίνεται μια μικρή λάμψη γύρω του) αποτρέποντας έτσι την πτώση του στο κενό.

Γυάλινο Τείχος: Απενεργοποιημένο (Glass Wall: Off)

Γυάλινο Τείχος: Ενεργοποιημένο (Glass Wall : On)

Ουρανός

Η σημασία επιλογής *Ουρανού* στο παιχνίδι μας δεν είναι κάτι το οποίο μπορούμε να αγνοήσουμε! Ναι, μπορούμε να αλλάξουμε και τον ουρανό του κόσμου μας! Ένα παράδειγμα στο οποίο μπορούμε να αξιοποιήσουμε τη συγκεκριμένη ιδιότητα, είναι οι αγώνες αυτοκινήτων που διεξάγονται σε διάφορες χώρες του κόσμου. Οι καιρικές συνθήκες σε κάθε χώρα αλλάζουν, οπότε και η όψη του ουρανού διαφέρει σε κάθε μία από αυτές. Έτσι, ο προγραμματιστής, για κάθε πίστα που θα δημιουργήσει, θα πρέπει να κάνει την κατάλληλη επιλογή ουρανού, η οποία να αντιπροσωπεύει με τον καλύτερο δυνατό τρόπο τις πραγματικές συνθήκες που επικρατούν. Για

παράδειγμα, εάν οι αγώνες πραγματοποιούνται στην Ελλάδα, μία από τις επιλογές θα ήταν ο ουρανός να είναι ηλιόλουστος.

Αντίθετα, εάν ο αγώνας διαδραματίζεται στην έρημο, σωστό θα ήταν να επιλέξετε έναν ουρανό που να ταιριάζει με τις συνθήκες που επικρατούν σε αυτή, δηλαδή ένα πιο σκονισμένο και θερμό κλίμα.

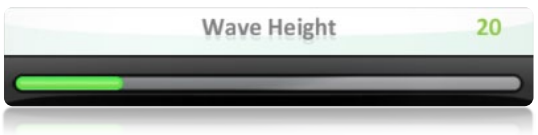
Φωτισμός

Τι θα γινόταν αν θέλαμε να δημιουργήσουμε την ίδια πίστα και ο αγώνας να διεξαγόταν βράδυ; Υπάρχουν αρκετές επιλογές για να ρυθμίσουμε το *Φωτισμό* στο παιχνίδι μας! Ας δούμε το πιο κάτω παράδειγμα. Συγκριτικά με την πιο πάνω εικόνα, επιλέξαμε τον Φωτισμό Lighting : Dark (σκοτάδι). Μπορούμε να πούμε πως προσομοιώνει αρκετά καλά ένα αγώνα ο οποίος διεξάγεται βράδυ.

Αυτό που προβληματίζει λίγο είναι πως ο ουρανός, παρόλο που έγινε πιο σκούρος, το περιβάλλον δεν πείθει πώς μιλάμε για βράδυ. Ο προγραμματιστής θα μπορούσε να χρησιμοποιήσει τον κατάλληλο συνδυασμό Ουρανού και Φωτισμού ώστε να προσομοιώσει καλύτερα το βράδυ στον κόσμο.

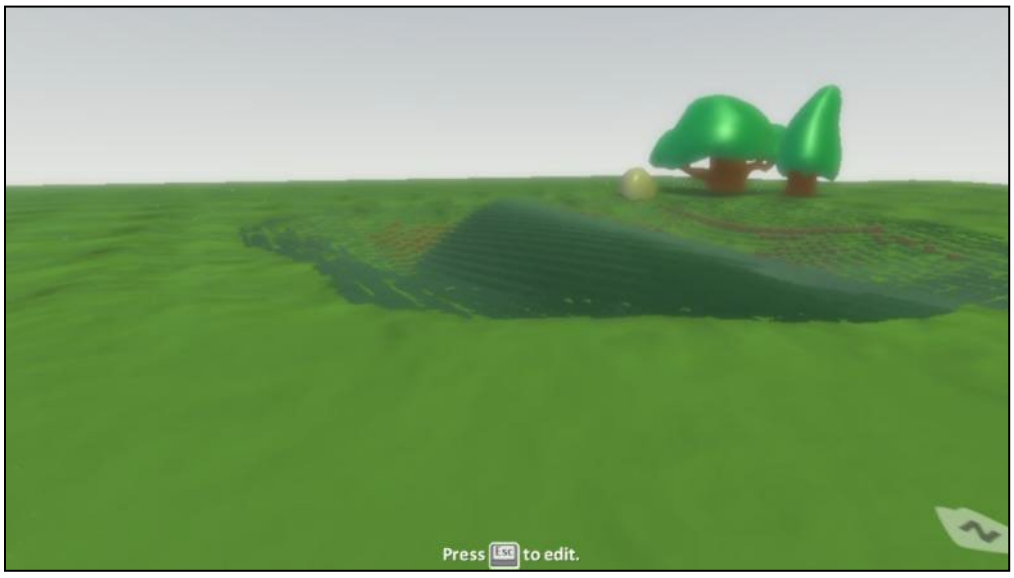
Συνδυάζοντας Ουρανό και Φωτισμό μπορείτε να δημιουργήσετε κατάλληλη ατμόσφαιρα στο παιχνίδι σας!

Ύψος Κύματος



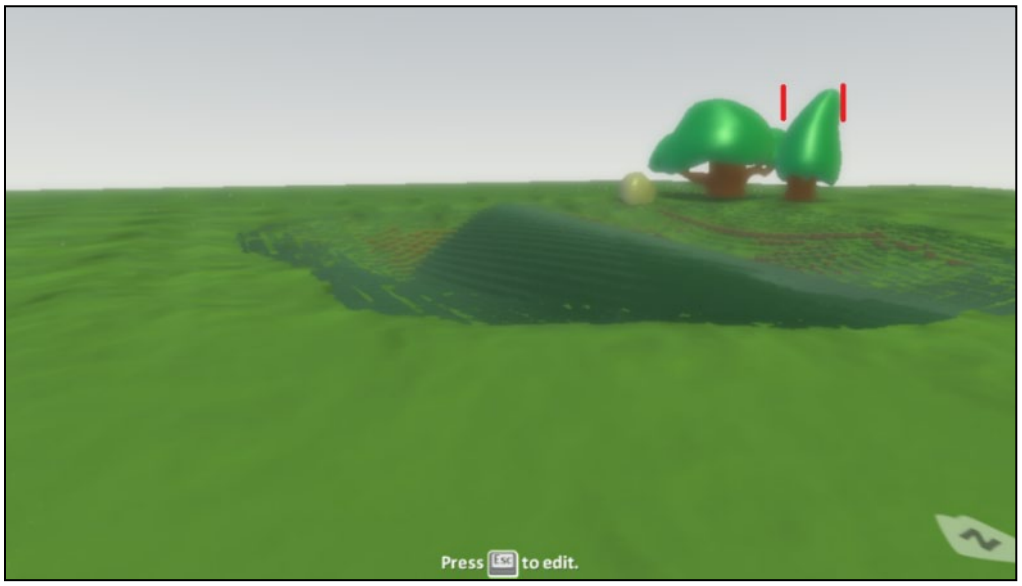
Μπορείτε να δημιουργήσετε ήρεμα αλλά και άγρια νερά για μια λιμνούλα, ένα ποτάμι ή μια θάλασσα που έχετε προσθέσει στον κόσμο σας. Δημιουργώντας μεγάλα κύματα μπορείτε να αναπαραστήσετε έως ένα βαθμό τις άσχημες καιρικές συνθήκες που θέλετε να επικρατούν στο παιχνίδι σας.

Στο πιο κάτω παράδειγμα βλέπουμε δύο εικόνες, όπου στην πρώτη δεν υπάρχει κύμα ενώ στην δεύτερη υπάρχει κύμα που συναρτήσει με τον μουντό καιρό (διαφορετικό ουρανό) δημιουργεί ένα αίσθημα κακοκαιρίας.



Μέγιστο Αεράκι και Ελάχιστο Αεράκι

Στηριζόμενοι στο προηγούμενο παράδειγμα, και υποθέτοντας ότι θέλουμε το παιχνίδι μας να εξελιχθεί σε άσχημες καιρικές συνθήκες, εκτός από το κύμα που δημιουργείται στην λίμνη, θα έπρεπε να υπάρχει και ανάλογη κίνηση στα δέντρα λόγω των ανέμων που φυσούν! Κύμα χωρίς αέρα δεν γίνεται! Επειδή ο άνεμος δεν πρέπει να είναι σταθερός, το MSKodu μας επιτρέπει να ορίσουμε δύο ιδιότητες που προσδιορίζουν το άνω και το κάτω όριο της έντασης του ανέμου. Για να αντιληφθούμε οπτικά τι συμβαίνει, τα κλαδιά των δέντρων κινούνται δεξιά και αριστερά τόσο, όσο ορίσουμε εμείς την ένταση. Στην πιο κάτω εικόνα έχουμε σημειώσει με δύο κόκκινες γραμμές την περιοχή μεταξύ της οποίας κινείται το πάνω μέρος του δέντρου που βρίσκεται στα δεξιά. Παρόμοια κίνηση γίνεται για όλα τα δέντρα που υπάρχουν στο περιβάλλον.



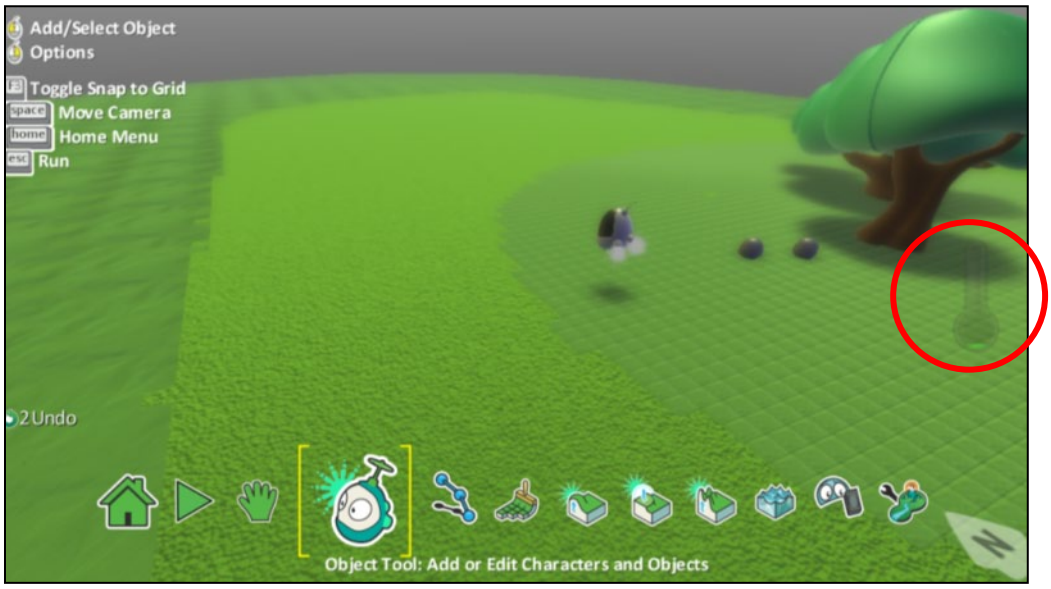
7.1.3 Ιδιότητες για τους υπολογιστικούς πόρους που απαιτεί το παιχνίδι

Ένα βασικό πρόβλημα, που μπορεί να προκύψει κατά την δημιουργία ενός παιχνιδιού, είναι η υπερφόρτωση του κόσμου μας με χαρακτήρες, αντικείμενα και έδαφος. Κάθε στοιχείο απαιτεί επεξεργαστική ισχύ και, κατά συνέπεια, μπορεί να φτάσουμε σε ένα σημείο όπου ο υπολογιστής μας δεν θα μπορεί να ανταποκριθεί στις απαιτήσεις του παιχνιδιού μας. Αυτό βέβαια δεν προκαλεί αλλοιώσεις στο προγραμματιστικό/σχεδιαστικό κομμάτι του παιχνιδιού μας αλλά στην αισθητική του. Το παιχνίδι μας θα συνεχίσει να λειτουργεί «σωστά», αλλά λόγο υπερφόρτωσης θα είναι πολύ αργό! Οι κινήσεις και, γενικά, οι συμπεριφορές των αντικειμένων θα εκτελούνται πιο αργά απ' ότι θα θέλαμε. Αυτό είναι κουραστικό για το χρήστη που αλληλεπιδρά με το παιχνίδι. Το MSKodu μας προσφέρει δύο επιλογές οι οποίες μας βοηθούν να υπολογίζουμε τον επεξεργαστικό φόρτο του παιχνιδιού μας ώστε να διατηρούμε τα επίπεδα σε πλαίσια που επιτρέπουν στη δράση να κυλά ομαλά.

Ένδειξη Μετρητή Πόρων

Με αυτή την επιλογή, το περιβάλλον μας ενημερώνει για το ποσοστό των πόρων (έδαφος, χαρακτήρες, αντικείμενα) που έχουμε χρησιμοποιήσει στο παιχνίδι μας, σε σχέση με τη μέγιστη δυνατότητα που έχουμε. Η ένδειξη αυτή παρουσιάζεται σαν ένα *θερμόμετρο* στα δεξιά της οθόνης μας. Όσο πιο πολλούς πόρους χρησιμοποιούμε, τόσο ανεβαίνει και η αντίστοιχη «θερμοκρασία». Εάν απενεργοποιήσουμε την επιλογή αυτή, το θερμόμετρο παύει πλέον να φαίνεται στην οθόνη μας.

Παιχνίδι που χρησιμοποιεί λίγους πόρους (ελάχιστη θερμοκρασία)



Παιχνίδι που χρησιμοποιεί πολλούς πόρους (θερμοκρασία στο πορτοκαλί)

Ενεργοποίηση Περιορισμού Πόρων

Εάν φτάσουμε στα όρια των υπολογιστικών πόρων που έχουμε διαθέσιμους, τότε το MSKodu μας περιορίζει και δεν μας επιτρέπει να προσθέσουμε οτιδήποτε άλλο στον κόσμο μας. Εάν όμως επιθυμούμε για κάποιο λόγο (έστω σχεδιαστικό) να προσθέσουμε κι άλλους πόρους, μπορούμε να **απενεργοποιήσουμε** την επιλογή *Περιορισμός Πόρων* (γκρίζο χρώμα). Σε αυτήν την περίπτωση ο προγραμματιστής δεν θα έχει πλέον στη διάθεση του την Ένδειξη του *Μετρητή Πόρων (Θερμόμετρο),* με αποτέλεσμα αν υπερβεί αρκετά το όριο να ωθήσει το παιχνίδι σε κατάρρευση. Αυτό πρακτικά σημαίνει πώς το παιχνίδι δεν θα λειτουργεί.

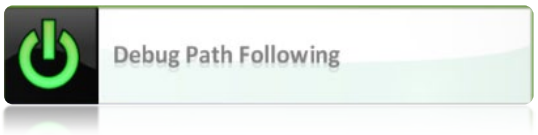
7.1.4 Ιδιότητες για την Αποσφαλμάτωση

Καθώς ο προγραμματιστής ξεκινά να δημιουργεί το παιχνίδι του, υπάρχει σημαντική πιθανότητα να κάνει λάθη στο σχεδιασμό και στην ανάπτυξη του προγράμματός του. Στο MSKodu, όταν αναφερόμαστε σε λάθη, εννοούμε τις μη αναμενόμενες συμπεριφορές χαρακτήρων στο παιχνίδι μας. Παρ' ότι ο προγραμματιστής μπορεί να θέλει να προσδιορίσει σε ένα χαρακτήρα κάποιες συγκεκριμένες κινήσεις ή κάποιες συγκεκριμένες αντιδράσεις (π.χ. να αλλάξει κατεύθυνση, να ξεκινήσει να μιλά) σε κάποια ερεθίσματα (π.χ. όταν ακούσει ένα ήχο, όταν δει κάποιον άλλο

χαρακτήρα), ο χαρακτήρας μπορεί τελικά να ενεργεί διαφορετικά απ' ότι σχεδίασε ο προγραμματιστής. Για τον λόγο αυτό, ο προγραμματιστής συχνά πρέπει να κάνει αποσφαλμάτωση, πρέπει δηλαδή να ελέγχει αν οι νέες λειτουργίες/συμπεριφορές που έχει προσθέσει, όντως εκτελούνται όπως ανέμενε αρχικά.

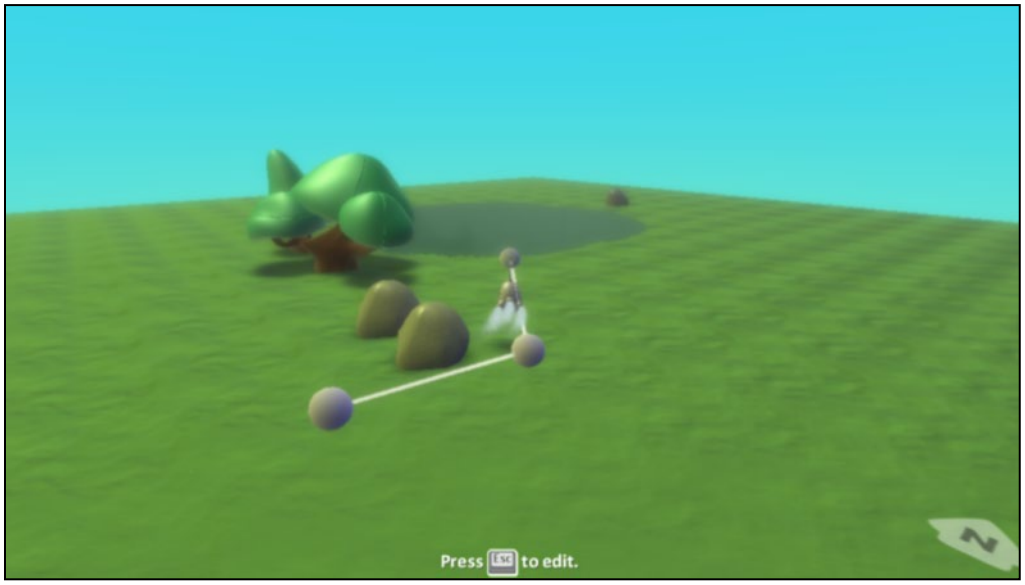
Οι παρακάτω επιλογές βοηθούν τον προγραμματιστή να κάνει ευκολότερα την αποσφαλμάτωση του παιχνιδιού του.

Αποσφαλμάτωση Μονοπατιού

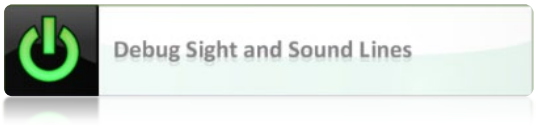


Όταν δημιουργείτε ένα μονοπάτι το οποίο θα ακολουθεί ένας ή περισσότεροι χαρακτήρες, αυτό δεν είναι ορατό όταν παίζουμε το παιχνίδι. Μπορεί από λάθος να έχουμε προγραμματίσει ασυναίσθητα κάποιον χαρακτήρα να ακολουθεί κάποιο άλλο μονοπάτι από αυτό που θα έπρεπε, χωρίς να πέσει στην αντίληψη μας. Ενεργοποιώντας την επιλογή *Αποσφαλμάτωση Μονοπατιού*, το μονοπάτι θα είναι ορατό και κατά τη διάρκεια του παιχνιδιού και θα μπορείτε να κατανοήσετε τη μορφή των μονοπατιών σας και ποια αντικείμενα κινούνται πάνω σε αυτά. Μη ξεχνάτε να απενεργοποιείτε τη συγκεκριμένη ιδιότητα όταν έχετε ολοκληρώσει την ανάπτυξη του παιχνιδιού σας.

Αποσφαλμάτωση Ακολουθίας Μονοπατιού: ενεργοποιημένη



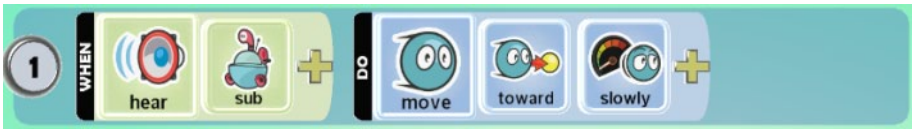
Αποσφαλμάτωση Οπτικής και Ηχητικής Επαφής



Εξίσου σημαντική είναι η αποσφαλμάτωση, όταν έχουμε εισάγει τους αισθητήρες *όρασης (see)* ή *ακοής (hear)* σε κάποια αντικείμενα του παιχνιδιού μας. Όταν υπάρχουν πολλά αντικείμενα και σύνθετοι κόσμοι μπορεί να μην είμαστε σίγουροι ποιος θα δει ποιον και ποιος θα ακούσει ποιον. Όταν η επιλογή αυτή είναι ενεργοποιημένη, μια φωτεινή γραμμή μας δείχνει τους χαρακτήρες που επικοινωνούν!

Στην παρακάτω εικόνα, εκτός από τον Kodu, ξεκινώντας από αριστερά βλέπουμε τον *Μηχανάκια (Cycle)*, το *Τζετ (Jet)* και το *Υποβρύχιο (Sub)*.

Στο παράδειγμα αυτό έχουμε προγραμματίσει τον Kodu, όταν ακούει το *Υποβρύχιο (Sub)*, να κινείται προς αυτόν.

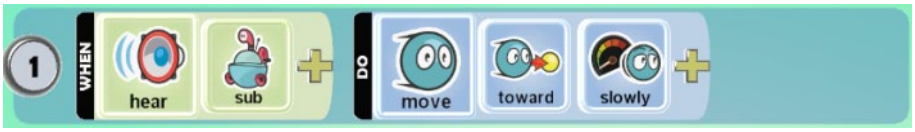


Καθώς ο προγραμματιστής δοκιμάζει αν το παιχνίδι του όντως λειτουργεί όπως θα ήθελε (αποσφαλμάτωση), δεν μπορεί να ξεχωρίσει από που προέρχεται αυτός ο ήχος και πότε προκαλείται. Ενεργοποιώντας την επιλογή ***Debug Sight and Sound Lines*** και εκτελώντας το παιχνίδι, εμφανίζεται μια γραμμή (άσπρη-πράσινη) μεταξύ Kodu και Sub που υποδεικνύει στον προγραμματιστή ότι μεταξύ Kodu και Sub δημιουργήθηκε μια σχέση (ο ένας παράγει ήχο και ο άλλος τον ακούει).

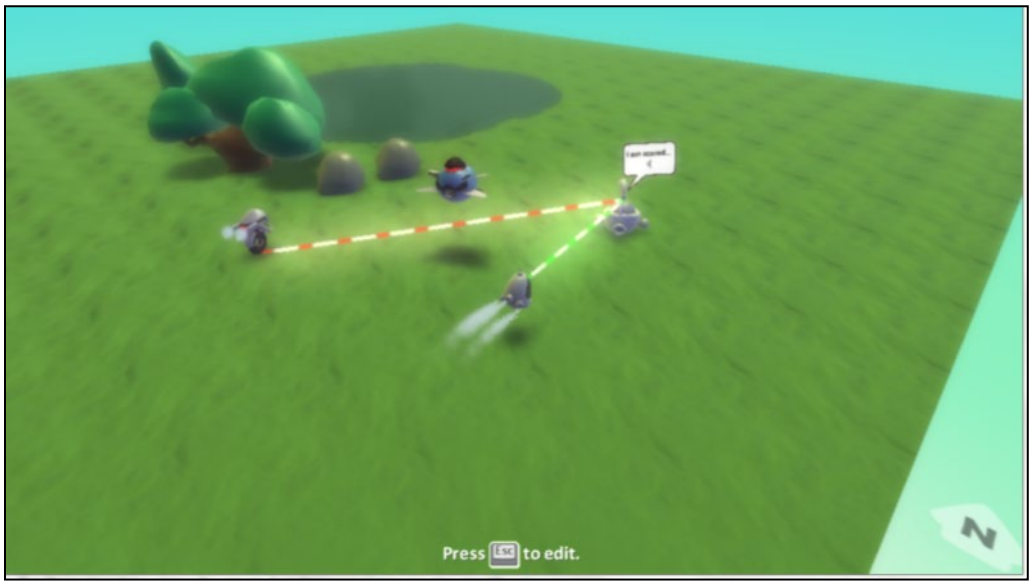
Η ίδια ιδιότητα ενεργοποιεί και τις σχέσεις «*Όρασης*» μεταξύ των αντικειμένων. Ας υποθέσουμε πως θέλουμε στο παιχνίδι μας ο Sub να έχει μαλώσει με τον Cycle και **μόλις τον δει** να τον φοβάται και να **ζητά από** τον Kodu (παράγει ήχο) να πάει δίπλα του (για να νιώθει ασφαλής). Περιμένετε να εμφανιστεί κι άλλη γραμμή; Μεταξύ ποιών χαρακτήρων;

Προγραμματισμός Sub:

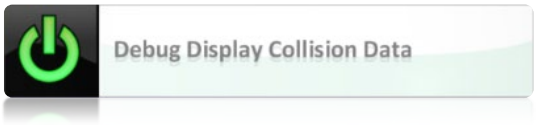
Προγραμματισμός Kodu:



Βλέποντας την παρακάτω εικόνα, διαπιστώνουμε πως όταν παίζουμε το παιχνίδι παρουσιάζεται ακόμη μια γραμμή (κόκκινη-άσπρη) μεταξύ του Sub και του Cycle. Αυτή σηματοδοτεί την έκφραση "**μόλις τον δει"** που προαναφέραμε.



Αποσφαλμάτωση Δεδομένων Σύγκρουσης



Η επιλογή *Αποσφαλμάτωση Δεδομένων Σύγκρουσης* εμφανίζει με μωβ χρώμα τα αντικείμενα που είναι σταθερά στο έδαφος και παραμένουν σταθερά εάν συγκρουστούν με κάποιο αντικείμενο. Στην παρακάτω εικόνα βλέπουμε ένα Κάστρο, δύο Δέντρα και δύο μεγάλους Βράχους.

Αποσφαλμάτωση Δεδομένων Σύγκρουσης: απενεργοποιημένη

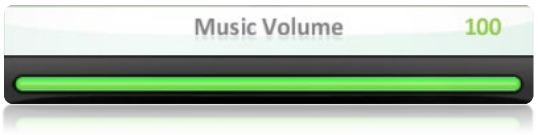
Ενεργοποιώντας την επιλογή αυτή, θα περιμέναμε να χρωματιστούν με μωβ χρώμα όλα τα αντικείμενα.

Αποσφαλμάτωση Δεδομένων Σύγκρουσης: ενεργοποιημένη

Και όμως αυτό δεν ισχύει. Όπως φαίνεται και από την παρακάτω εικόνα, όταν οι βράχοι συγκρουστούν με τον Kodu θα μετακινηθούν και για αυτό δε θεωρούνται σταθερά αντικείμενα στο έδαφος και συνεπώς δε χρωματίζονται.

7.1.5 Ρύθμιση Ήχου

Ένταση Ήχου



Τα αντικείμενά μας με την ενέργεια ***Παίξε (Play)*** έχουν τη δυνατότητα να αναπαράγουν μουσική και ήχους. Η ιδιότητα του κόσμου *Ένταση Ήχου* μας επιτρέπει να καθορίσουμε πόσο δυνατά θα ακούγεται η μουσική.

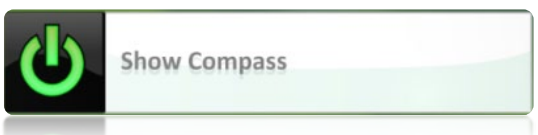
Ένταση Ηχητικών Εφέ

Όπως θα έχετε λογικά ήδη παρατηρήσει (αν έχετε τα ηχεία σας ανοικτά όταν φορτώνετε το MSKodu), οι διάφοροι χαρακτήρες που ζουν στο παιχνίδι μας παράγουν ήχους είτε από μόνοι τους είτε όταν πραγματοποιούν κάποιες ενέργειες που προκαλούν επιπρόσθετους ήχους (π.χ. εκρήξεις, συγκρούσεις). Για να ελέγξουμε την ένταση με την οποία θα ακούγονται οι ήχοι αυτοί θα πρέπει να αλλάξουμε την τιμή της ιδιότητας *Ένταση Ηχητικών Εφέ*.

Σημαντικό είναι να αναφέρουμε πως οι ήχοι που επέρχονται από ενέργειες των χαρακτήρων του παιχνιδιού μπορούν να διαδραματίσουν καθοριστικό ρόλο στην ψυχολογία του χρήστη. Όσο πιο δυνατή θα είναι η έκρηξη, τόσο πιο έντονα θα επηρεαστεί ο χρήστης.

7.1.6 Ρυθμίσεις Παιχνιδιού

Πυξίδα

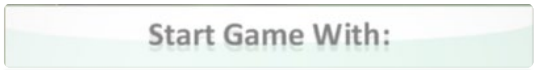


Έχοντας ενεργοποιήσει αυτή την επιλογή, η πυξίδα που βρίσκεται στο κάτω δεξί σημείο της οθόνης είναι ορατή, όχι μόνο κατά την διάρκεια δημιουργίας του παιχνιδιού αλλά και κατά τη διάρκεια που ο χρήστης παίζει με το παιχνίδι. Έτσι μπορεί να την συμβουλευτεί για να φέρει εις πέρας την αποστολή του. Για παράδειγμα, ας υποθέσουμε ότι ο Kodu έχει ως αποστολή να

περάσει μέσα από ένα άγνωστο δάσος, να φτάσει σε ένα χωριό και να παραδώσει ένα πολύ σημαντικό μήνυμα στους κατοίκους. Το δάσος όμως το βράδυ εγκυμονεί πολλούς κινδύνους για αυτό συμβουλεύεται τον μόνο κάτοικο του δάσους για το ποια κατεύθυνση θα πρέπει να ακολουθήσει. Το *Υποβρύχιο (Sub)* (κάτοικος δάσους) το μόνο που κάνει είναι να ενημερώσει τον Kodu να κατευθυνθεί Ανατολικά. Πώς γνωρίζει ο Kodu που βρίσκεται η Ανατολή;

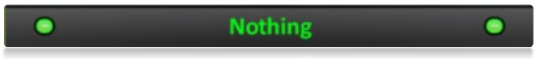
Ενεργοποιώντας την πυξίδα, βλέπουμε πως αυτή δείχνει το Βορρά προς τα αριστερά. Αφού το *Υποβρύχιο* λέει στον Kodu να κινηθεί Ανατολικά τότε αυτός πρέπει να προχωρήσει προς τα πάνω (δηλαδή δεξιά του Βορρά).

Έναρξη Παιχνιδιού Με:



Τι θέλουμε να εμφανίζεται όταν κάποιος τρέξει το παιχνίδι μας; Το περιβάλλον MSKodu μας δίνει 5 εναλλακτικές και μόνο μια από αυτές μπορεί να επιλεγεί για κάθε παιχνίδι.

Έναρξη Παιχνιδιού Με: Τίποτα



Το παιχνίδι ξεκινάει κανονικά, χωρίς να εμφανίζεται κάτι στην οθόνη.

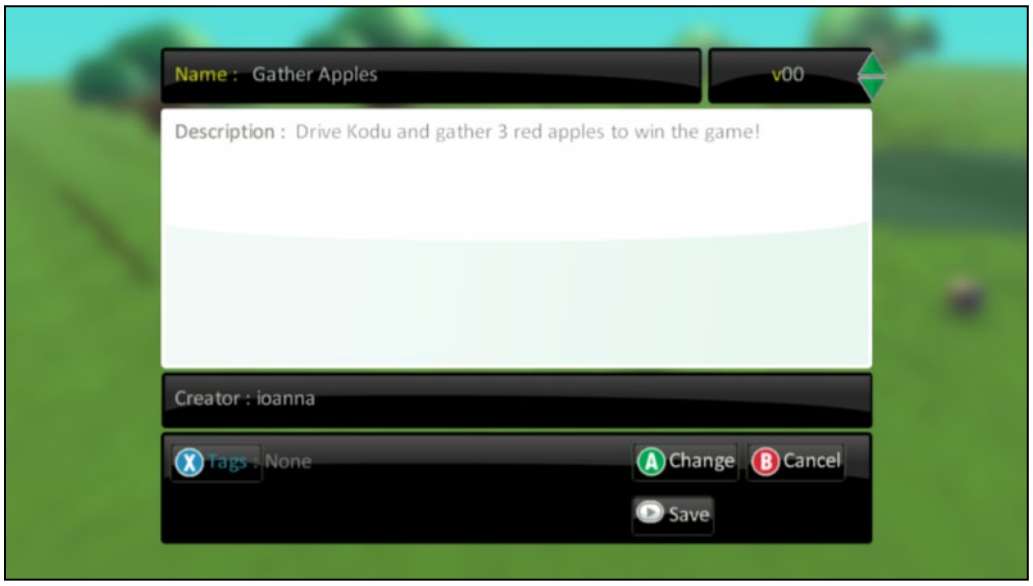
Έναρξη Παιχνιδιού Με: Όνομα Κόσμου

Η συγκεκριμένη επιλογή εμφανίζει το Όνομα του Κόσμου για 3 δευτερόλεπτα και το παιχνίδι ξεκινάει αμέσως μετά.

Έναρξη Παιχνιδιού Με: Περιγραφή Κόσμου

Μαζί με τον τίτλο, μπορούμε να δώσουμε οδηγίες στο χρήστη για τις ενέργειες που πρέπει να κάνει ώστε να πετύχει το σκοπό του παιχνιδιού μας.

Ο τίτλος και η περιγραφή μπορούν να προστεθούν όταν αποθηκεύουμε το παιχνίδι μας. Επιλέγοντας ***Αποθήκευσε τον κόσμο μου (Save my world)*** από το βασικό μενού εντολών, εμφανίζεται το πιο κάτω παράθυρο.



Σε αυτήν την περίπτωση το παιχνίδι ξεκινά εφόσον ο χρήστης πατήσει Enter.

Έναρξη Κόσμου Με: Αντίστροφη Μέτρηση

Σε συγκεκριμένα είδη παιχνιδιών, είναι απαραίτητο να δώσουμε κάποια δευτερόλεπτα στο χρήστη ώστε να προετοιμαστεί για αυτό που πρόκειται να αντιμετωπίσει. Πάρτε για παράδειγμα ένα παιχνίδι αγώνων δρόμου όπου ο χρήστης πρέπει να αντιδράσει ακαριαία με την έναρξη του παιχνιδιού για να μην τον προσπεράσουν οι συναγωνιστές του.

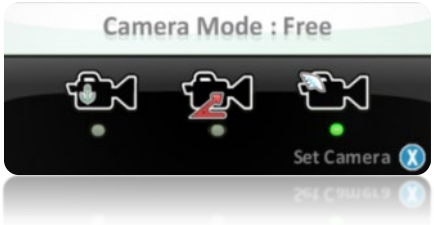
Η αντίστροφη μέτρηση ξεκινάει από το 3 και μετά το πρώτο δευτερόλεπτο, το παιχνίδι ξεκινάει.

Έναρξη Κόσμου Με: Περιγραφή Κόσμου και Αντίστροφη Μέτρηση

Η τελευταία επιλογή συνθέτει τις δύο προηγούμενες μαζί. Προτού ξεκινήσει το παιχνίδι μας, παρουσιάζεται αρχικά ο Τίτλος και η Περιγραφή του παιχνιδιού και μόλις ο χρήστης πατήσει Enter ξεκινά η αντίστροφη μέτρηση.

7.1.7 Ρυθμίσεις Κάμερας

Λειτουργία Κάμερας

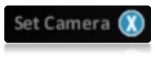


Η ιδιότητα αυτή επιτρέπει στον προγραμματιστή να επιλέξει τη θέση της κάμερας κατά τη διάρκεια του παιχνιδιού (μην μπερδεύετε την επιλογή αυτή με τη χρήση της κάμερας κατά τη σχεδίαση). Μας δίνονται τρεις επιλογές:

Α) *η κάμερα σε ελεύθερη θέση (Camera Mode: Free)*. Η κάμερα ακολουθεί το χρήστη από πίσω αλλά δε βρίσκεται σε ένα συγκεκριμένο σημείο. Ο χρήστης του παιχνιδιού μας μπορεί να την ελέγξει όπως την ελέγχουμε και εμείς κατά την κατασκευή του παιχνιδιού. Έτσι π.χ. πατώντας το δεξί πλήκτρο του ποντικιού και σέρνοντας το ποντίκι, μπορεί να αλλάζει τη θέση της κάμερας στις τρεις διαστάσεις ενώ χρησιμοποιώντας τη ροδέλα του ποντικιού μπορεί να κάνει ζουμ σε ένα συγκεκριμένο σημείο ή το αντίστροφο. Καθώς μετακινείται ο πρωταγωνιστής μας, μετακινείται και η κάμερα.

Β) *η κάμερα σε συγκεκριμένη θέση (Camera Mode: Fixed Position)*. Ενεργοποιώντας αυτή την επιλογή, η κάμερα παραμένει σε μια σταθερή θέση καθ' όλη τη διάρκεια του παιχνιδιού. Ο

παίκτης θα βλέπει από το ίδιο σημείο την πίστα, όπου και αν εξελίσσεται η δράση. Θυμηθείτε το παιχνίδι στο οποίο ο Kodu έπρεπε να περάσει τέσσερεις δρόμους για να πάρει το νόμισμα της νίκης! Ενώ ο Kodu μετακινούνταν, εμείς βλέπαμε την πίστα μας από σταθερή θέση. Πως μπορούμε να προσδιορίσουμε όμως ποια θα είναι η συγκεκριμένη θέση της κάμερας; Πατήστε πάνω στην επιλογή *Καθόρισε την κάμερα (Set Camera)*:



Στη συνέχεια αλλάξτε τη θέση της κάμερας ώστε να βρείτε την κατάλληλη για το παιχνίδι σας. Τέλος, πατήστε Enter. Θα διαπιστώσετε ότι κάθε φορά που παίζετε το παιχνίδι, η κάμερα βρίσκεται σε μια και μοναδική θέση. Αυτό, βεβαίως, δεν ισχύει κατά το σχεδιασμό του παιχνιδιού. Στην παρακάτω εικόνα, ο προγραμματιστής έχει πατήσει στην επιλογή *Fixed Camera*, στη συνέχεια *Set Camera* και αφού τοποθέτησε την κάμερα είναι έτοιμος να πατήσει *Enter* (διαβάστε τις οδηγίες στο κάτω μέρος της οθόνης).

Ο παίκτης του παιχνιδιού θα βλέπει μόνο από τη συγκεκριμένη θέση τον κόσμο του παιχνιδιού.

Από τη στιγμή που η κάμερά σας είναι σταθερή, υπάρχει πιθανότητα ο Kodu ή ο εκάστοτε πρωταγωνιστής του παιχνιδιού να βρεθεί εκτός οπτικού πεδίου της κάμερας. Αυτό σημαίνει ότι ο παίκτης δε θα βλέπει τον ήρωά του! Για το λόγο αυτό πρέπει να είστε προσεκτικοί στην τοποθέτηση της κάμερας όταν χρησιμοποιείτε τη συγκεκριμένη επιλογή.

Γ) *η κάμερα σε συγκεκριμένη θέση σχετικά με τον πρωταγωνιστή μας (Camera Mode: Fixed Offset)*. Στην τρίτη επιλογή, η κάμερα παραμένει σταθερή σχετικά με τη θέση του πρωταγωνιστή του παιχνιδιού, δηλαδή τον ακολουθεί όπου και αν πάει από μια συγκεκριμένη γωνία και απόσταση θέασης. Για τη ρύθμιση της σχετικής θέσης, πρέπει πάλι να πατήσουμε πάνω στην επιλογή *Set Camera X*, να επιλέξουμε τη θέση της κάμερας σε σχέση με τον πρωταγωνιστή μας και στη συνέχεια να πατήσουμε Enter. Ενώ λοιπόν η επιλογή *Σταθερή Θέση* κρατάει την κάμερα σε **σταθερή θέση σε όλη τη διάρκεια του παιχνιδιού**, η επιλογή *Σχετική Θέση* κρατάει την κάμερα σε **σταθερή θέση ως προς τον πρωταγωνιστή** που χειρίζεται ο χρήστης.

Ελαστικότητα Κάμερας

Η ιδιότητα *Ελαστικότητα Κάμερας* έχει νόημα όταν έχουμε επιλέξει την κάμερα να βρίσκεται σε ελεύθερη θέση. Όπως αναφέραμε σε αυτή την περίπτωση η κάμερα ακολουθεί την πλάτη του πρωταγωνιστή μας. Όταν όμως ο πρωταγωνιστής μας μετακινείται και στρίβει, πόσο γρήγορα θέλουμε να αντιδρά η κάμερά μας και να έρχεται ακριβώς πίσω από τον Kodu; Αν στην ιδιότητα αυτή προσδιορίσουμε τιμή 1, τότε η κάμερα πραγματοποιεί τη λήψη της διαρκώς πίσω από τον Kodu, όπως ακριβώς συμβαίνει μέχρι τώρα στα παιχνίδια μας. Αν στην ιδιότητα αυτή όμως προσδιορίσουμε τιμή μικρότερη, τότε η κάμερα αντιδρά πιο ομαλά-ήρεμα και φτάνει στην πλάτη του Kodu σε μεγαλύτερο χρόνο, χωρίς βίαιες μετακινήσεις. Έτσι, για παράδειγμα, καθώς ο Kodu

περιστρέφεται αριστερόστροφα, όταν έχουμε επιλέξει ένταση 0.3, η κάμερα θα καθυστερήσει να κάνει και αυτή την αριστερόστροφη κίνηση συγκριτικά με την ένταση 0.8. Δοκιμάστε μια μικρή τιμή για να δείτε πόσο πιο ομαλά κινείται η κάμερα σε αυτήν την περίπτωση.

7.2 Οι αισθητήρες του MSKodu: Ο kodu στη γη και στη θάλασσα

Ελπίζουμε ότι έχετε πειστεί για την αξία όλων των προηγούμενων ιδιοτήτων για τη διαμόρφωση του κόσμου των παιχνιδιών σας. Ήρθε η ώρα όμως να επιστρέψουμε στον προγραμματισμό και πιο συγκεκριμένα, να μελετήσουμε αισθητήρες που αφορούν την κατανόηση που έχουν τα αντικείμενα για τα χαρακτηριστικά του κόσμου τους.

7.2.1 Ο αισθητήρας Είμαι σε έδαφος (on land):

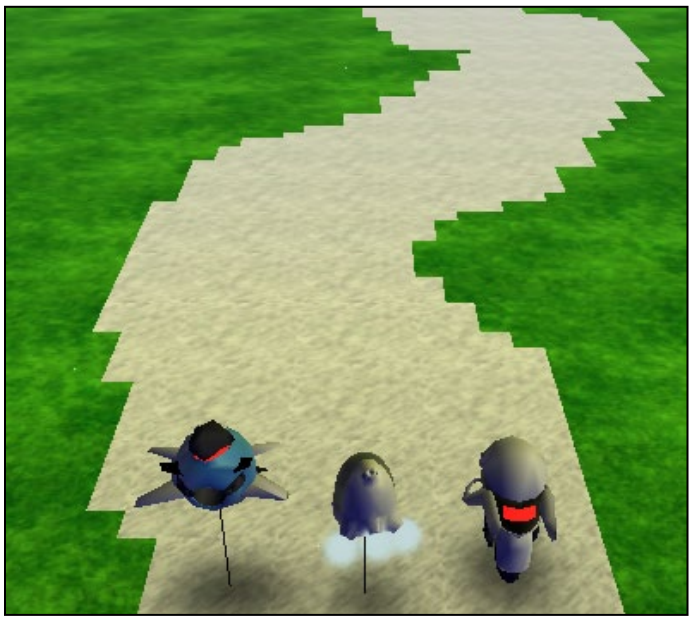
Το MSKodu, εκτός όλων των άλλων, μας δίνει την δυνατότητα να προσδιορίζουμε την συμπεριφορά των αντικειμένων ανάλογα με το είδος του εδάφους πάνω στο οποίο βρίσκονται. Για παράδειγμα, αν θέλουμε να δημιουργήσουμε ένα παιχνίδι αγώνων ταχύτητας, θα θέλαμε ο χαρακτήρας μας σε περίπτωση που βγει εκτός δρόμου και πατήσει γρασίδι να επιβραδύνεται μέχρι να ξαναμπεί στον κεντρικό δρόμο. Αυτό μπορούμε να το πετύχουμε αξιοποιώντας τον αισθητήρα ***Είμαι σε έδαφος (on land)*** που βρίσκεται στην κατηγορία αισθητήρων *more*. Ο αισθητήρας αυτός επιτρέπει στο αντικείμενο να κάνει ενέργειες ανάλογα με τον τύπο του εδάφους πάνω στο οποίο βρίσκεται. Η γενική χρήση του αισθητήρα έχει την εξής μορφή:

ΌΤΑΝ[Είμαι σε έδαφος][τύπος εδάφους] **ΤΟΤΕ** [Ενέργεια][Προσδιοριστικά ενέργειας]

Εκφράσεις συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα ***Είμαι σε έδαφος (on land)***:

{Πατάω, βρίσκομαι πάνω, ακουμπάω έδαφος, περπατάω σε, εκτός δρόμου}

Ας δούμε τον αισθητήρα ***Είμαι σε έδαφος (on land)*** στα πλαίσια ενός παραδείγματος. Έστω ότι θέλουμε να φτιάξουμε ένα παιχνίδι στο οποίο ο Kodu θα τρέχει σε ένα αγώνα ταχύτητας μαζί με άλλα αντικείμενα με σκοπό να τερματίσει πρώτος. Αφού δημιουργήσουμε το περιβάλλον και προγραμματίσουμε τα υπόλοιπα αντικείμενα να κινούνται πάνω σε συγκεκριμένα μονοπάτια, πρέπει να προγραμματίσουμε τον Kodu. Στόχος μας είναι ο Kodu να κινείται καθώς ο χρήστης χρησιμοποιεί τα βελάκια του πληκτρολογίου και όταν βγαίνει εκτός δρόμου να μειώνεται η ταχύτητά του.



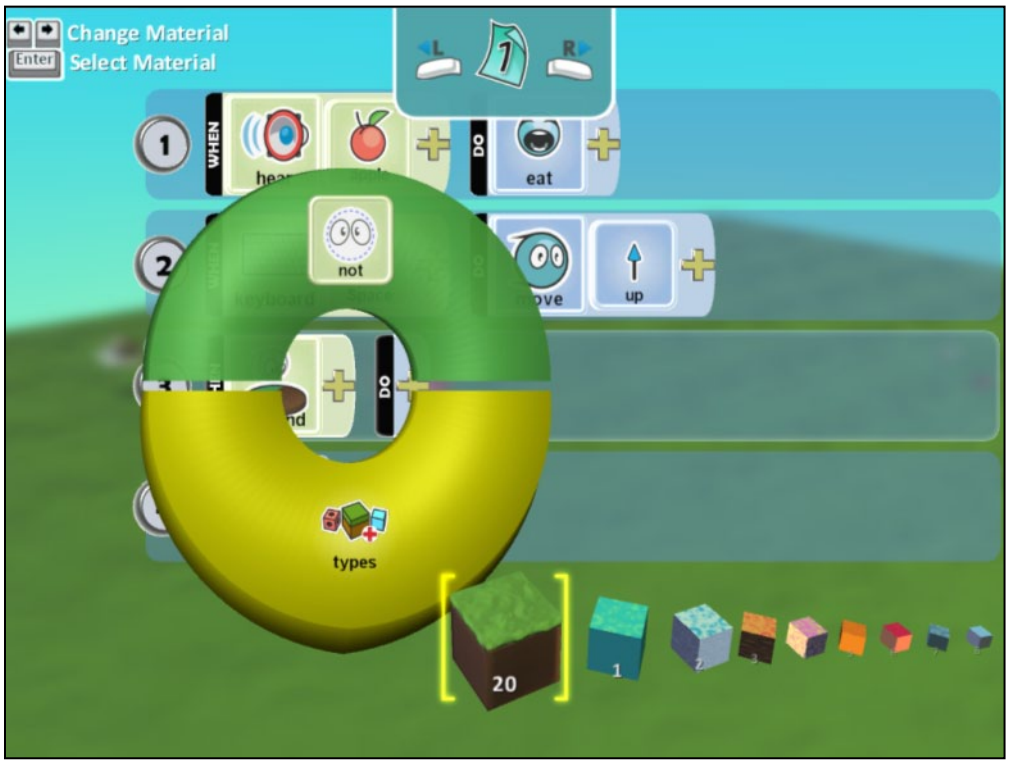
Και πως θα μπορούσε ο Kodu να αντιληφθεί πότε κινείται πάνω σε γρασίδι και πότε σε χώμα; Με τον αισθητήρα ***Είμαι σε έδαφος (on land)*** και προσδιοριστικά τους αντίστοιχους τύπους εδάφους. Στο συγκεκριμένο παράδειγμα θέλουμε να μειώνεται η ταχύτητα του Kodu όταν πατάει πάνω στο γρασίδι, οπότε χρειαζόμαστε μια μόνο συμπεριφορά που θα επιβραδύνει τον Kodu όταν

βρίσκεται πάνω στο γρασίδι (τύπος εδάφους 20). Επομένως, συνολικά η συμπεριφορά του Kodu θα είναι:

ΌΤΑΝ[Από το πληκτρολόγιο][πατηθούν τα βελάκια] **ΤΟΤΕ** [Κινήσου]

ΌΤΑΝ[Είσαι σε έδαφος][τύπου 20] **ΤΟΤΕ** [Κινήσου][Αργά]

Όταν προσθέτουμε προσδιοριστικό στον αισθητήρα ***Είμαι σε έδαφος (on land)*** ,εμφανίζεται η παρακάτω πίτα για την επιλογή τύπου εδάφους:



Δείτε το παράδειγμα *07_01.kodu*

Τρέξτε το παράδειγμα **[07_01.kodu]**. Γιατί δε λειτουργεί όπως θα περιμέναμε; Μήπως έχει σχέση με τη σειρά που έχουμε καταγράψει τις συμπεριφορές; Αλλάξτε τη σειρά τους. Τι παρατηρείτε;

Είναι πλέον εύκολο να κατανοήσετε πως μπορείτε να προσθέτετε διαφορετικούς τύπους εδάφους στον κόσμο σας και τα αντικείμενά σας να συμπεριφέρονται διαφορετικά σε κάθε έναν από αυτούς!

7.2.2 Ο αισθητήρας Είμαι σε νερό (on water):

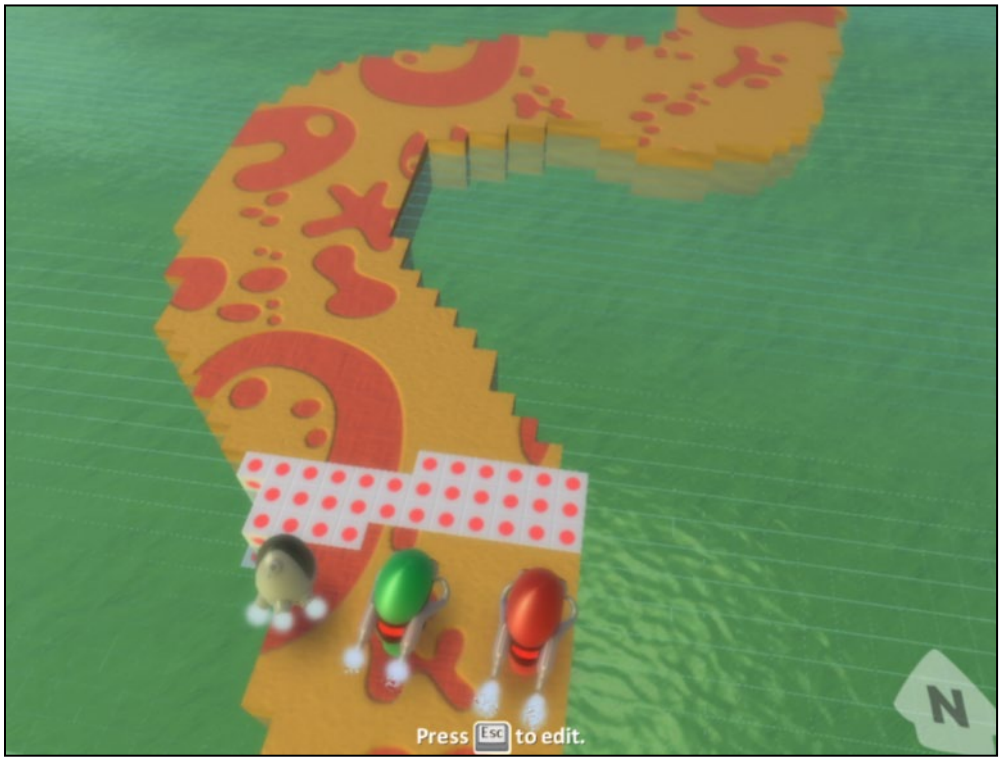
Στον κόσμο μας μπορούμε να βάλουμε νερό, και μάλιστα διάφορα "είδη" νερού. Έτσι, εκτός από το να μπορούμε να προγραμματίζουμε τη συμπεριφορά του Kodu ανάλογα με τον τύπο εδάφους, μπορούμε να προγραμματίζουμε τη συμπεριφορά του και ανάλογα με τον τύπο νερού πάνω ή μέσα στο οποίο βρίσκεται. Ας πούμε πως σε κάποιο παιχνίδι, ο κόσμος αποτελείται από μία λίμνη με λάβα (κόκκινο νερό) και ένα μονοπάτι στο οποίο ο χαρακτήρας μας πρέπει να κινηθεί με σκοπό να φτάσει στον τερματισμό. Σε περίπτωση που πέσει πάνω στην λάβα τότε θα πρέπει να μειώνονται οι πόντοι του. Για να υλοποιήσουμε αυτό το παιχνίδι θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα ***Είμαι σε νερό (on water)*** που βρίσκεται στην κατηγορία αισθητήρων *more* και επιτρέπει σε ένα αντικείμενο να συμπεριφέρεται ανάλογα με τον τύπο του νερού πάνω (ή μέσα) στο οποίο βρίσκεται. Και ο αισθητήρας αυτός δέχεται, ως προσδιοριστικό, τον τύπου του νερού που θέλουμε να αναγνωρίσουμε. Η γενική χρήση του αισθητήρα έχει την εξής μορφή:

ΌΤΑΝ[Είμαι σε νερό][τύπος νερού] **ΤΟΤΕ** [Ενέργεια][Προσδιοριστικό ενέργειας]

Εκφράσεις που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα ***Είμαι σε νερό (on water)***:

{Είμαι σε, βρίσκομαι πάνω, περνάω, ακουμπάω, επιπλέω σε, βουλιάζω σε, κολυμπάω σε}

Ας δούμε τη χρήση του αισθητήρα ***Είμαι σε νερό (on water)*** σε ένα παραπλήσιο παράδειγμα. Έστω ότι θέλουμε να δημιουργήσουμε ένα παιχνίδι στο οποίο ο Kodu θα τρέχει σε ένα δρόμο μαζί με άλλα αντικείμενα με σκοπό να τερματίσει πρώτος, όπως και πριν. Τώρα, όμως, αριστερά και δεξιά της πίστας υπάρχουν λίμνες με νερό. Αν ο Kodu πέσει μέσα σε αυτές, το παιχνίδι τελειώνει.



Εφόσον θέλουμε το παιχνίδι να τελειώνει όταν πατάει ο Kodu πάνω στο νερό, θα πρέπει να δημιουργήσουμε μια συμπεριφορά που να βασίζεται πάνω στον αισθητήρα ***Είμαι σε νερό (on water)***. Η συμπεριφορά του Kodu θα είναι:

ΌΤΑΝ[Από το πληκτρολόγιο][πατηθούν τα βελάκια] **ΤΟΤΕ** [Κινήσου]

ΌΤΑΝ[Είσαι σε νερό][πράσινου τύπου] **ΤΟΤΕ** [Το παιχνίδι τελειώνει]

Δείτε το παράδειγμα *07_02.kodu*

[07_02.kodu] Τα προσδιοριστικά του αισθητήρα ***Είμαι σε νερό (on water)*** είναι προφανώς οι διαθέσιμοι τύποι νερού που μπορούμε να χρησιμοποιήσουμε κατά την κατασκευή του κόσμου μας.

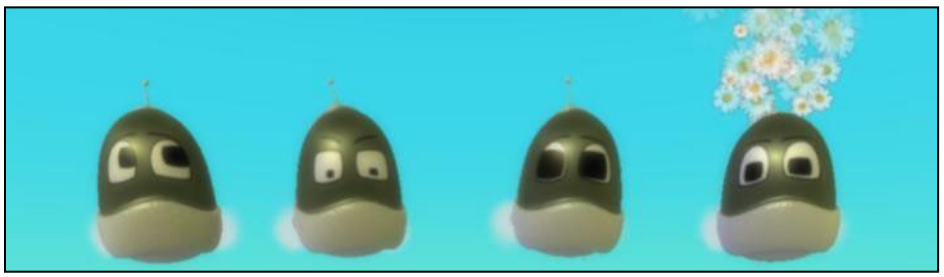
7.3 Η συμπεριφορά του Kodu: ενέργειες

Σε συνέχεια των ενεργειών που είδαμε στα προηγούμενα κεφάλαια, θα εξετάσουμε κάποιες πιο διασκεδαστικές εναλλακτικές ενέργειες για τους χαρακτήρες μας!

7.3.1 Η ενέργεια Εξέφρασε(Express):

Η έκφραση συναισθημάτων από τους χαρακτήρες είναι κάτι που δεν έχουμε συνηθίσει να βλέπουμε στα παιχνίδια μέχρι τώρα. Παρόλα αυτά το MSKodu μας δίνει την δυνατότητα να προγραμματίσουμε τα αντικείμενά μας ώστε να εκφράζονται ανάλογα με τις περιστάσεις. Ο Kodu λοιπόν μπορεί να εκφράσει διάφορα συναισθήματα όπως λύπη, χαρά, θυμό και τρέλα καθώς και συνδυασμούς τους μέσω της χρήσης της ενέργειας ***Εξέφρασε (Express)***. Η ενέργεια αυτή συνήθως

εξυπηρετεί δύο σκοπούς. Ο πρώτος είναι να μπορεί ο χρήστης να αντιλαμβάνεται την "ψυχολογική" διάθεση του χαρακτήρα (!) έτσι ώστε να προβαίνει στις κατάλληλες ενέργειες. Ο δεύτερος είναι να αλληλεπιδρά ο χαρακτήρας με τα άλλα αντικείμενα μέσα στο παιχνίδι. Στην επόμενη εικόνα φαίνονται τέσσερεις διαφορετικές εκφράσεις του Kodu!



Η γενική δομή συμπεριφορών που χρησιμοποιούν τον αισθητήρα ***Εξέφρασε (Express)*** είναι η εξής:

ΌΤΑΝ[Αισθητήρας][Προσδιοριστικά αισθητήρα] **ΤΟΤΕ** [Εξέφρασε][Προσδιοριστικά έκφρασης]

Εκφράσεις που θα πρέπει να μας φέρνουν στο μυαλό τη συγκεκριμένη ενέργεια:

{δείξε, εκφράσου, χαμογέλασε, νευρίασε, θύμωσε, ερωτεύσου, δείξε οργή}

Έστω ότι θέλουμε να δημιουργήσουμε ένα παιχνίδι στο οποίο ο Kodu όταν βλέπει ένα αερόστατο, τότε θα πρέπει να εκφράζει συναισθήματα χαράς. Τι θα έπρεπε να κάνουμε;

ΌΤΑΝ[Δεις][Αερόστατο] **ΤΟΤΕ** [Εξέφρασε][Χαρά]

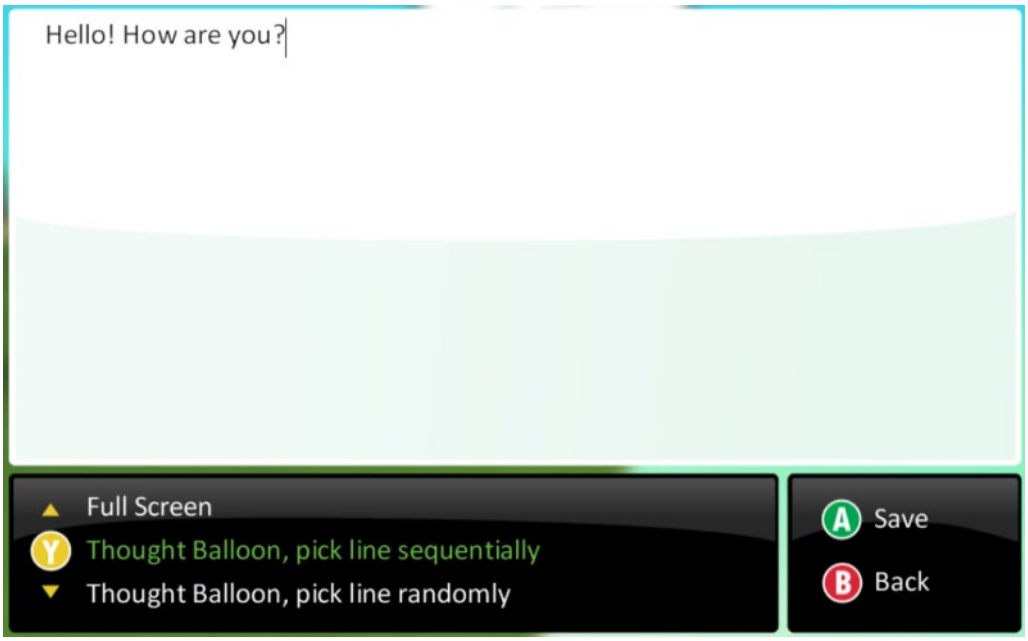
Διερευνήστε και δοκιμάστε από μόνοι σας την ποικιλία συναισθημάτων που μπορεί να εκφράσει ο Kodu.

Είναι σημαντικό να τονίσουμε ότι η ενέργεια ***Εξέφρασε (Express)*** μπορεί να συνδυαστεί με τον αισθητήρα ***Βλέπω (See)*** ώστε διαφορετικά αντικείμενα να είναι σε θέση να επικοινωνήσουν μεταξύ τους μέσω των συναισθημάτων τους! Παρατηρήστε στην επόμενη εικόνα ότι στα προσδιοριστικά του αισθητήρα ***Βλέπω (See)***, στην κατηγορία *Express* υπάρχουν όλες οι δυνατές εκφράσεις των αντικειμένων:

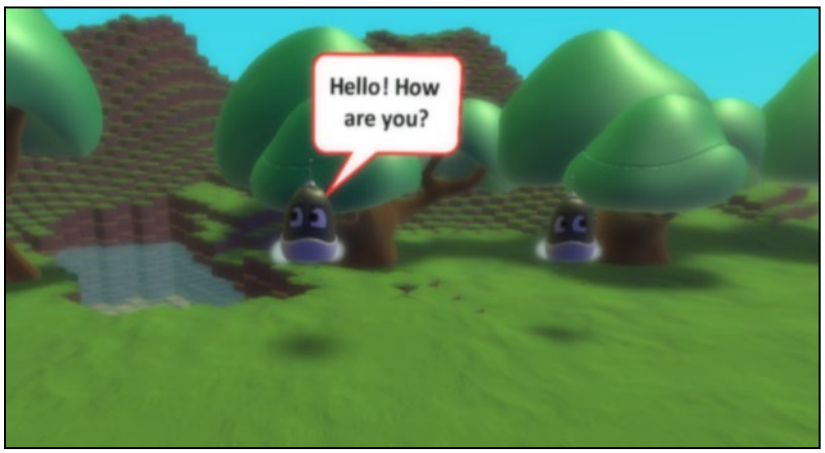
Συνεπώς, ένα αντικείμενο μπορεί να αναγνωρίσει τις εκφράσεις ενός άλλου αντικειμένου με τον αισθητήρα ***Βλέπω (See)*** και να αντιδράσει σε αυτές.

7.3.2 Η ενέργεια Πες (say):

Στο MSKodu, τα αντικείμενά μας όχι μόνο μπορούν να εκφράσουν τα συναισθήματά τους αλλά μπορούν να εκφράσουν και τις σκέψεις τους, μπορούν να μιλήσουν! Όχι προφέροντας λόγο αλλά με την εμφάνιση κειμένου μέσα σε συννεφάκια, όπως ακριβώς και στα κόμικ! Η ενέργεια που επιτρέπει στους χαρακτήρες μας να συζητήσουν είναι η ***Πες (Say)***. Αν την επιλέξουμε, θα εμφανιστεί αυτομάτως ένα παράθυρο μέσα στο οποίο θα πρέπει να γράψουμε τι θέλουμε να "πει" ο χαρακτήρας μας.



Το αποτέλεσμα θα είναι της μορφής:



Η γενική μορφή συμπεριφορών που χρησιμοποιούν τη συγκεκριμένη ενέργεια:

ΌΤΑΝ[Αισθητήρας][Προσδιοριστικά αισθητήρα] **ΤΟΤΕ** [Πες][Προσδιοριστικά λόγου]

Ρήματα συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τον αισθητήρα:

{πες, μίλα, εξήγησε, ανέλυσε, σχολίασε, συζήτησε, πρόσθεσε, αρνήσου, σημείωσε κτλ.}

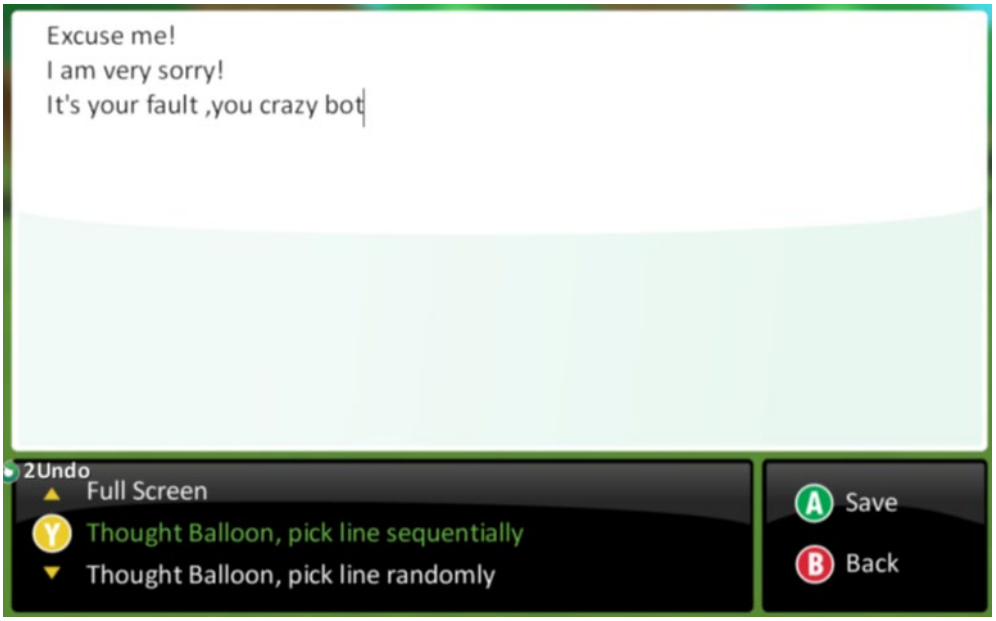
Έστω ότι θέλουμε να δημιουργήσουμε ένα παιχνίδι στο οποίο ο Kodu θα είναι ιδιαίτερα ευγενικός και θα λέει "excuse me", συγγνώμη δηλαδή, σε όποιο αντικείμενο πέσει πάνω. Θα πρέπει συνεπώς να χρησιμοποιήσουμε τον αισθητήρα ***Πέφτω πάνω (Bump)*** σε συνδυασμό με την ενέργεια ***Πες (Say)***. Ο προγραμματιστής μας θα σκεφτόταν:

ΌΤΑΝ[Πέσω επάνω][σε οποιοδήποτε αντικείμενο] **ΤΟΤΕ** [Πες][Excuse me]

Προσέξτε ότι στην παραπάνω εντολή έχουμε προσθέσει ένα προσδιοριστικό στην ενέργεια ***Πες (Say)***, το χρώμα του πλαισίου μέσα στο οποίο θα εμφανιστεί το κείμενό μας. Δηλαδή μπορούμε να επιλέγουμε αν θέλουμε οι σκέψεις του χαρακτήρα μας να εμφανίζονται σε ένα κόκκινο ή ένα πράσινο πλαίσιο! Επιπλέον, παρατηρήστε ότι όταν τρέξετε το παιχνίδι, η εμφάνιση του πλαισίου συνοδεύεται από αντίστοιχο ήχο που κάνει πιο αληθοφανή την ομιλία του χαρακτήρα μας. Δυστυχώς όμως προς το παρόν, τα αντικείμενα του MSKodu δεν μπορούν να μιλήσουν ελληνικά. Θα περιοριστούμε στους αγγλικούς χαρακτήρες.

Το MSKodu μας δίνει τρεις ακόμη επιλογές για την εμφάνιση κειμένου από τους χαρακτήρες μας:

Α) Πολλές γραμμές σειριακά: Προσθέστε στο πλαίσιο του μηνύματος της ενέργειας ***Πες (Say)*** τρεις γραμμές κειμένου που θα διαχωρίσετε πατώντας Enter (π.χ. Excuse Me – I am very sorry – It's your fault, you crazy bot).



Παίξτε ξανά με το παιχνίδι σας. Παρατηρήστε ότι το κείμενο δεν θα εμφανιστεί ενιαίο σε ένα πλαίσιο αλλά κάθε πρόταση ξεχωριστά. Οι προτάσεις παρουσιάζονται σειριακά, η μια μετά την άλλη.

Β) Πολλές γραμμές τυχαία: Πατήστε τώρα την επιλογή *Διάλεξε Τυχαία Γραμμή (Thought Balloon, pick line randomly)* που βρίσκεται στο κάτω μέρος του ίδιου παραθύρου. Παίξτε ξανά με το παράδειγμα. Τι παρατηρείτε; Ο χαρακτήρας σας επιλέγει με τυχαίο τρόπο το κείμενο το οποίο θα πει από τις διαφορετικές γραμμές που έχετε εισάγει!

Γ) *Σε όλη την οθόνη*: Υπάρχουν φορές που θέλουμε ένας χαρακτήρας να εμφανίζει μήνυμα σε ολόκληρη την οθόνη του παιχνιδιού. Για να το καταφέρουμε αυτό, αρκεί στο ίδιο παράθυρο να επιλέξουμε *Πλήρης Οθόνης (Full Screen)*. Σε αυτήν την περίπτωση θα εμφανιστούν όλες οι προτάσεις που συμπληρώσαμε στο παράθυρο μηνύματος της ενέργειας ***Πες (Say)***, όπως φαίνονται στην παρακάτω εικόνα:

Μη δοκιμάσετε να γράψετε «περίεργες» σκέψεις για τους χαρακτήρες σας, στα αγγλικά. Το MSKodu θα τις εμφανίσει με αστεράκια!

7.3.3 Η ενέργεια Χρωμάτισε (color):

Όπως γνωρίζετε, μπορούμε να αλλάξουμε το χρώμα ενός αντικειμένου κατά την κατασκευή ενός αντικείμενου, απλά επιλέγοντάς το με το εργαλείο Αντικειμένων και στη συνέχεια χρησιμοποιώντας τα βελάκια του πληκτρολογίου. Το MSKodu όμως μας δίνει τη δυνατότητα να αλλάζουμε το χρώμα των αντικειμένων και κατά τη διάρκεια των παιχνιδιών με τη δημιουργία αντίστοιχων συμπεριφορών! Η ενέργεια ***Χρωμάτισε (Color)*** είναι αυτή με την οποία ο δημιουργός ενός παιχνιδιού μπορεί να προγραμματίσει τον Kodu να αλλάξει χρώμα στον εαυτό του ή σε κάποιο άλλο αντικείμενο. Η γενική μορφή συμπεριφορών που χρησιμοποιούν την ενέργεια ***Χρωμάτισε (Color)***:

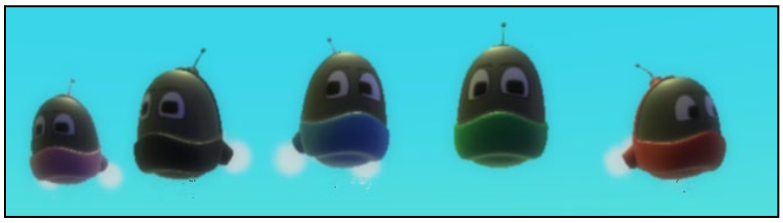
ΌΤΑΝ[Αισθητήρας][Προσδιοριστικά αισθητήρα] **ΤΟΤΕ** [Χρωμάτισε][Χρώμα]

Εκφράσεις συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια ***Χρωμάτισε (Color)***:

{Χρωμάτισε, βάψε, άλλαξα χρώμα, κοκκίνισε, πρασίνισε, κτλ.}

Πρέπει να σημειώσουμε πως υπάρχουν συγκεκριμένα χρώματα διαθέσιμα για το χρωματισμό των αντικειμένων όπως φαίνονται στην παρακάτω εικόνα:

Και πέντε Kodu με διαφορετικά χρώματα:



Έστω ότι θέλουμε να φτιάξουμε ένα παιχνίδι στο οποίο ο Kodu θα παρουσιάζει 2 συμπεριφορές:

α) Θα ντρέπεται και θα γίνεται κόκκινος όταν βλέπει κάποιον άλλο Kodu και

β) θα αλλάζει χρώμα σε οποιοδήποτε ρομπότ βλέπει (κατηγορία anybot)!

Άρα θα αλλάζει και το δικό του χρώμα αλλά και των άλλων αντικειμένων! Παρατηρείστε τα ρήματα στις συμπεριφορές. Ποιον αισθητήρα υποδηλώνουν; Προφανώς τον αισθητήρα ***Βλέπω (See)*** σε συνδυασμό με την ενέργεια ***Χρωμάτισε (Color)***. Πως όμως διαχωρίζουμε σε ποιο αντικείμενο θα εφαρμοστεί η ενέργεια ***Χρωμάτισε (Color)***; Η ενέργεια μπορεί να συνοδευτεί από δυο προσδιοριστικά, το ***Εμένα (Me)*** και το ***Αυτό (It)***. Το πρώτο εφαρμόζει το χρώμα στο αντικείμενο που περιέχει τη συμπεριφορά ενώ το δεύτερο εφαρμόζει το χρώμα στο αντικείμενο που προσδιορίζεται στον αισθητήρα. Συνεπώς, χρειαζόμαστε δυο συμπεριφορές **[07_05.kodu]**:

ΌΤΑΝ[Βλέπω][Kodu] **ΤΟΤΕ** [Χρωμάτισε][Εμένα][Κόκκινο]

ΌΤΑΝ[Βλέπω][Οποιοδήποτε Ρομπότ] **ΤΟΤΕ** [Χρωμάτισε][Αυτό][Τυχαίο χρώμα]

7.3.4 Η ενέργεια Παίξε (play):

Παιχνίδια χωρίς μουσική έχετε παίξει; Παιχνίδια χωρίς την αναπαραγωγή ήχων; Δύσκολα θα απαντήσετε αρνητικά. Όλα τα παιχνίδια χρησιμοποιούν τη μουσική για να διαμορφώσουν ατμόσφαιρα και να παρασύρουν το χρήστη στην πλοκή τους. Για αυτό ακριβώς το λόγο, το MSKodu παρέχει τη δυνατότητα στα αντικείμενα να αναπαράγουν μουσική με την ενέργεια ***Παίξε (Play)***. Η ενέργεια είναι ιδιαίτερα απλή και δέχεται ως προσδιοριστικό τον τύπο μουσικής που θέλουμε να ακούσουμε. Η γενική μορφή μιας συμπεριφοράς που περιλαμβάνει τη συγκεκριμένη ενέργεια:

ΌΤΑΝ [Αισθητήρας][Προσδιοριστικά αισθητήρα] **ΤΟΤΕ** [Παίξε][Τύπος ήχου]

Εκφράσεις συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό τη συγκεκριμένη ενέργεια:

{παίξε, αναπαράγω, τραγούδα, μουσική, ήχοι, ατμόσφαιρα}

Δημιουργήστε μια συμπεριφορά στην οποία προσθέστε ως ενέργεια την ***Παίξε (Play)*** και εξερευνήστε τους διαθέσιμους ήχους στο MSKodu.:

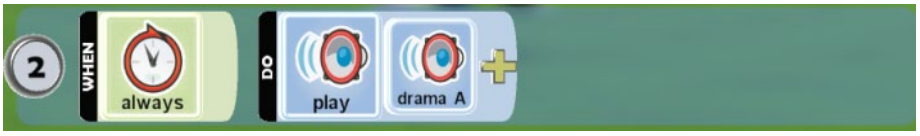
Παρατηρήστε ότι όταν μετακινήσετε το ποντίκι πάνω από ένα ήχο, τότε αυτός αρχίζει να αναπαράγεται! Γενικά, υπάρχουν τρεις κατηγορίες ήχων:

Α) *Ήχοι για γεγονότα*: ήχοι που αφορούν δηλαδή συμβάντα, όπως μια τυχαία συνάντηση, το χάσιμο μιας ζωής, την απώλεια ενέργειας, τα τελευταία δευτερόλεπτα ενός παιχνιδιού. Οι ήχοι αυτοί μάλλον δεν μπορούν να αναπαραχθούν για μεγάλο χρονικό διάστημα, καθώς, θα εκνευρίσουν το χρήστη! Υπάρχουν πολλές υποκατηγορίες όπως μουσικά όργανα και νότες, εκρήξεις, σειρήνες, κτλ.

Β) *Ήχοι για μουσική υπόκρουση*: μουσικά θέματα που μπορούν να αναπαράγονται στο υπόβαθρο του παιχνιδιού μας, για να δημιουργήσουν την κατάλληλη ατμόσφαιρα. Διακρίνονται σε ήχους δραματικούς, ήχους μυστηρίου και ήχους για την οδήγηση!

Γ) *Ήχοι του περιβάλλοντος*: ήχοι από τη φύση, ήχοι που προσθέτουν σημαντικά στην αληθοφάνεια του παιχνιδιού μας. Η κατηγορία αυτή περιέχει ήχους όπως ωκεανός, δάσος, πόλη, γήπεδο κτλ.

Έστω, λοιπόν, ότι θέλουμε να δημιουργήσουμε ένα παιχνίδι σε όλη τη διάρκειά του οποίου θα ακούγεται δραματική μουσική. Ποιον αισθητήρα θα χρησιμοποιήσουμε σε συνδυασμό με την ενέργεια ***Παίξε (Play)***; Μόνο ένας αισθητήρας μας δίνει αυτή τη δυνατότητα, ο αισθητήρας ***Για πάντα (Always)***. Είναι εύκολο να δημιουργήσετε την αντίστοιχη συμπεριφορά:



Η ενέργεια ***Παίξε (Play)*** μπορεί να αξιοποιηθεί σε συνδυασμό με τον αισθητήρα ***Ακούω (Hear)*** για να δημιουργηθούν πιο σύνθετες αλληλεπιδράσεις μεταξύ των αντικειμένων του παιχνιδιού μας. Παρατηρήστε ότι στα προσδιοριστικά του αισθητήρα ***Ακούω (Hear)***, υπάρχουν όλοι οι διαθέσιμοι ήχοι για την ενέργεια ***Παίξε (Play)***. Άρα, κάποιος μπορεί να τραγουδά και κάποιος άλλος να τον ακούει και να αντιδρά!

7.3.5 Η ενέργεια Σταμάτα (Quiet):

Η ενέργεια ***Σταμάτα (Quiet)*** λειτουργεί αντίστροφα από την εντολή ***Παίξε (Play)***, δηλαδή ενώ η δεύτερη αναπαράγει ήχους, η πρώτη σταματά τους ήχους που ήδη αναπαράγει ένα αντικείμενο! Συνεπώς, η ενέργεια ***Σταμάτα (Quiet)*** δεν έχει νόημα χωρίς να έχει προηγηθεί μια ενέργεια ***Παίξε (Play)***. Η ενέργεια ***Σταμάτα (Quiet)*** δέχεται ως προσδιοριστικό τον ήχο που θέλουμε να σταματήσει να αναπαράγεται. Γιατί όμως να θέλουμε να σταματήσει η αναπαραγωγή ενός ήχου; Για πολλούς λόγους, όπως γιατί πλησιάζει ένας επικίνδυνος εχθρός, γιατί το υποβρύχιό μας μπαίνει σε αθόρυβη λειτουργία για να μην εντοπιστεί, γιατί θέλουμε να αρχίσει να αναπαράγεται ένας άλλος ήχος, κτλ.

Η ενέργεια ***Σταμάτα (Quiet)*** σταματά τους ήχους του αντικειμένου που την περιέχει και όχι ήχους που παράγουν τα υπόλοιπα αντικείμενα. Η γενική μορφή της συμπεριφοράς που χρησιμοποιεί τη συγκεκριμένη ενέργεια:

ΌΤΑΝ[Αισθητήρας][Προσδιοριστικά αισθητήρα] **ΤΟΤΕ** [Σταμάτα][Προσδιοριστικά ήχου]

Εκφράσεις συμπεριφορών που θα πρέπει να μας φέρνουν στο μυαλό την ενέργεια:

{Σταμάτα, σώπασε, ησύχασε, σταμάτα, κλείσε τη μουσική}

Για παράδειγμα, αν θέλουμε όταν ο Kodu δεν βλέπει χαρούμενη την πριγκήπισσά του Kodula ☺ να παίζεται δραματική μουσική, τότε θα μπορούμε να δημιουργήσουμε μια συμπεριφορά σαν αυτή που ακολουθεί:

Είναι σημαντικό να τονίσουμε, ότι η ενέργεια ***Σταμάτα (Quiet)*** δεν παύει την αναπαραγωγή της μουσική για πάντα, αλλά για όσο ο αντίστοιχος αισθητήρας είναι ενεργοποιημένος. Στο

συγκεκριμένο παράδειγμα, αν η Kodula «ξαναθυμώσει», τότε η δραματική μουσική θα ξαναρχίσει.

7.3.6 Ενέργειες που αφορούν την Κάμερα

Η κάμερα μέχρι αυτό το σημείο του βιβλίου έχει χρησιμοποιηθεί με δυο τρόπους:

α) για την πλοήγησή μας στον κόσμο του παιχνιδιού κατά τη διάρκεια που το αναπτύσσουμε

β) για να προσδιορίσουμε από ποια οπτική γωνία ο παίκτης θα παρακολουθεί την πλοκή του παιχνιδιού (από τις ιδιότητες του κόσμου).

Το περιβάλλον του MSKodu όμως μας δίνει ακόμη μια δυνατότητα για τον έλεγχο της κάμερας κατά τη διάρκεια του παιχνιδιού: μέσω προγραμματισμού, δηλαδή μπορούμε να ελέγξουμε τη θέση της κάμερας μέσω αντίστοιχων ενεργειών. Οι τρεις ενέργειες που αφορούν την κάμερα βρίσκονται στο μενού View και είναι οι:

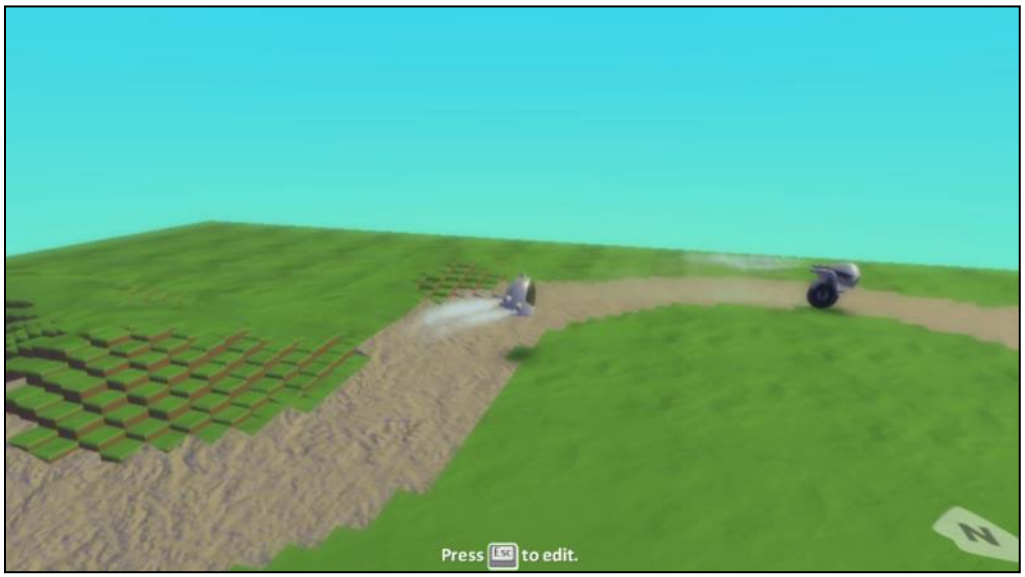
(α) ***Ακολούθησε (Follow)***: η ενέργεια αυτή αναγκάζει την κάμερα να έχει τέτοια θέση που ο πρωταγωνιστής μας να είναι διαρκώς στο κέντρο της οθόνης.

(β) ***Πρώτο πρόσωπο (First Person)***: η ενέργεια αυτή τοποθετεί την κάμερα στο κεφάλι του πρωταγωνιστή μας, με αποτέλεσμα ο παίκτης να βλέπει ότι βλέπει και ο ήρωάς του.

(γ) ***Αδιαφόρησε (Ignore)***: η ενέργεια αυτή, ρυθμίζει την κάμερα βάσει των ιδιοτήτων του κόσμου και αδιαφορεί για οποιαδήποτε προγραμματιστική ενέργεια ρύθμισης της κάμερας έχει εκτελεστεί προηγουμένως.

Ας δούμε πιο αναλυτικά τις διαφορετικές ενέργειες χειρισμού της κάμερας μέσω ενός παραδείγματος. Έστω πως ο προγραμματιστής αποφασίζει πως στο παιχνίδι αγώνων που έχει δημιουργήσει, θα ξεκινήσει με τύπο κάμερας ***Ακολούθησε (Follow)***, αλλά ταυτόχρονα θέλει να δώσει την ελευθερία στο χρήστη να επιλέγει από μόνος του κατά την διάρκεια του παιχνιδιού, το είδος της κάμερας που επιθυμεί.

Στην πρώτη εικόνα, βλέπουμε την περίπτωση όπου ο προγραμματιστής έχει επιλέξει από τις ιδιότητες του κόσμου την επιλογή *Κάμερα σε σταθερή σχετική θέση με το αντικείμενο (Fixed Offset Camera)*. Δεδομένου πως ο Kodu δεν έχει προγραμματιστεί διαφορετικά, η κάμερα αυτή θα διατηρήσει τον Kodu στο κέντρο της οθόνης, σε όλη την διάρκεια του παιχνιδιού.



Αξιοποιώντας τις ενέργειες που είδαμε προηγουμένως, μπορούμε να επιτρέψουμε στον χρήστη να αλλάζει τον τρόπο λειτουργίας της κάμερας, πατώντας διαφορετικά πλήκτρα του πληκτρολογίου. Μελετήστε τις παρακάτω συμπεριφορές:

Όταν ο χρήστης πατήσει το γράμμα C στο πληκτρολόγιο, τότε η κάμερα θα τεθεί σε ***Πρώτο Πρόσωπο (First Person)***. Αυτό σημαίνει, πώς η εικόνα που θα βλέπει ο χρήστης, θα είναι αυτή που βλέπει ο Kodu μέσα από το μάτια του.

Πατώντας το πλήκτρο B, θα εκτελεστεί η ενέργεια κάμερας ***Αδιαφόρησε (Ignore)***, η οποία θα ακυρώσει οποιαδήποτε επιλογή για την κάμερα έχει γίνει προγραμματιστικά και θα ενεργοποιήσει την αρχική ιδιότητα του κόσμου σε σχέση με την κάμερα. Συνεπώς, η λειτουργία της κάμερας θα επέστρεφε σε αυτήν της πρώτης εικόνας. **[07_03.kodu]**

Δείτε το παράδειγμα *07_03.kodu*

Έχοντας ενεργοποιήσει την επιλογή *η κάμερα σε συγκεκριμένη θέση σχετικά με τον πρωταγωνιστή μας (Fixed Offset)* στις ιδιότητες του κόσμου, εάν προσπαθήσουμε να εκτελέσουμε την ενέργεια κάμερας Ακολούθησε (Follow), τότε αυτή δεν θα εκτελεστεί! Ο συνδυασμός της ιδιότητας *Fixed Offset* και της ενέργειας ***Ακολούθησε (Follow)*** δεν είναι εφικτός.

Αν ο προγραμματιστής είχε επιλέξει στις ιδιότητες του κόσμου για την κάμερα *Σταθερή θέση (Fixed Position)*, τότε τα πράγματα θα άλλαζαν, καθώς θα μπορούσε να χρησιμοποιήσει και τις 3 ενέργειες στις συμπεριφορές των αντικειμένων.

Ενδιαφέρον είναι ότι η ενέργεια ***Πρώτο Πρόσωπο (First Person)*** δεν αντιστοιχεί σε κάποια ιδιότητα, άρα αυτή η κατάσταση της κάμερας μπορεί να προκληθεί μόνο προγραμματιστικά. Αντίθετα, η ενέργεια ***Ακολούθησε (Follow)*** αντιστοιχεί στην ιδιότητα *Fixed Offset*.

Είναι όμως η κάμερα χρήσιμη μόνο για να αλλάζουμε τη θέση της σε σχέση με έναν χαρακτήρα; Πως θα μπορούσατε να επιτρέψετε στο χρήστη κατά τη διάρκεια του παιχνιδιού να αλλάζει χαρακτήρες π.χ. ενώ εστιάζεται στον Kodu, κάποια στιγμή να εστιάζεται και να χειρίζεται έναν Μηχανάκια; Πολύ απλά, εισάγοντας την εντολή μεταφοράς της κάμερας στο αντίστοιχο αντικείμενο. Επομένως, στο MSKodu, ο χρήστης δεν είναι αναγκασμένος να χειρίζεται ένα και μόνο πρωταγωνιστή, αλλά με τη χρήση των ενεργειών που αφορούν την κάμερα, μπορούμε να του επιτρέψουμε να χειρίζεται πολλούς περισσότερους.

Σκεφτείτε ότι αυτό σας δίνει και τη δυνατότητα «τηλεμεταφοράς» του παίκτη σε διαφορετικά σημεία της πίστας! Αρκεί να υπάρχουν αντίστοιχα αντίγραφα του πρωταγωνιστή μας. Επομένως, αν θέλετε να δημιουργήσετε ένα παιχνίδι που αποτελείται από διαφορετικές δοκιμασίες και διαφορετικές πίστες, δεν έχετε παρά να δημιουργήσετε σε διαφορετικές θέσεις του κόσμου τις αντίστοιχες πίστες και να εισάγετε σε αυτές τα ίδια αντικείμενα, και την κατάλληλη στιγμή να

αλλάζετε τη θέση της κάμερας στον πρωταγωνιστή της αντίστοιχης δοκιμασίας! Έτσι, μπορείτε να δημιουργήσετε και παιχνίδια με διαφορετικά επίπεδα δυσκολίας.

Για να δοκιμάσετε τα προηγούμενα, απλά βάλετε δυο χαρακτήρες μέσα σε έναν απλό κόσμο και όταν συγκρουστούν, δώστε την κάμερα στο δεύτερο χαρακτήρα με την ενέργεια ***Πρώτο Πρόσωπο (First Person)***.

Περίληψη

Σε αυτό το κεφάλαιο περιγράψαμε τις γενικές ιδιότητες του κόσμου των παιχνιδιών μας, ιδιότητες για την διαμόρφωση του περιβάλλοντος (γυάλινο τείχος, ουρανός κτλ), ιδιότητες για τον έλεγχο της υπερφόρτωσης του προγράμματος, ιδιότητες που μας διευκολύνουν στην αποσφαλμάτωση των προγραμμάτων μας (ακολουθίας μονοπατιού, οπτικής επαφής κτλ) και ιδιότητες που αφορούσαν τις ρυθμίσεις ήχων και της κάμερας. Επιπρόσθετα, εξηγήσαμε τους αισθητήρες ***είμαι σε εδαφος (on land)*** και ***είμαι στο νερό (on water)*** που επιτρέπουν στα αντικείμενα να αλληλεπιδρούν με το περιβάλλον, να αναγνωρίζουν τον τύπο του εδάφους ή τον τύπο του νερού πάνω στον οποίο βρίσκονται. Ακόμη, αναλύσαμε ενέργειες που κάνουν περισσότερο διασκεδαστική την αλληλεπίδραση των αντικειμένων μεταξύ τους αλλά και με τους χρήστες. Πιο συγκεκριμένα μελετήσαμε τις ενέργειες ***Εξέφρασε (Express), Πες (Say), Χρωμάτισε (Color), Παίξε (Play), Σταμάτα (Quiet)*** και εξετάσαμε τα διαθέσιμα προσδιοριστικά τους. Στο τέλος του κεφαλαίου ασχοληθήκαμε με ενέργειες που αφορούν το χειρισμό της κάμερας κατά τη διάρκεια του παιχνιδιού και πιο συγκεκριμένα τις ***Ακολούθησε (Follow), Πρώτο πρόσωπο (First Person), Αδιαφόρησε (Ignore)*** ενώ σημειώσαμε ότι αυτές μπορούν να αλλάξουν δραματικά τους τύπους των παιχνιδιών που δημιουργούμε!

Ερωτήσεις

1. Τι θα συμβεί αν απενεργοποιήσουμε το γυάλινο τείχος;
2. Ποια είναι τα διαθέσιμα προσδιοριστικά του αισθητήρα *Στο έδαφος (on land)*;
3. Πως μπορούμε να βάλουμε αντίστροφη μέτρηση πριν την έναρξη του παιχνιδιού μας;
4. Θα μπορούσατε να κάνετε τον Kodu να δείχνει χαρούμενος με κάποια ενέργεια;
5. Ποιο είδος κάμερας αντιστοιχεί στο να βλέπει ο χρήστης ότι βλέπει και ο Kodu;
6. Πως μπορούμε να επιτρέψουμε στο χρήστη να «αλλάζει» ήρωες κατά τη διάρκεια του παιχνιδιού;

Δραστηριότητες

1. Δημιουργήστε ένα παιχνίδι στο οποίο ο Kodu τρέχει σε αγώνες δρόμου. Όταν βγαίνει εκτός δρόμου θα πρέπει να εκφράζει κάποιο συναίσθημα, να μειώνεται η ταχύτητά του και να μιλάει στο χρήστη του («Be careful!"). Δημιουργήστε μια γραμμή τερματισμού που όταν ο πρωταγωνιστής την ακουμπά, κηρύσσεται η λήξη του παιχνιδιού και μια διαφορετική γραμμή στη μέση της διαδρομής που μόλις την περνά θα γίνεται κόκκινος από την υπερπροσπάθεια!

2. Βασιστείτε στο προηγούμενο παιχνίδι και δημιουργήστε μια δεύτερη πίστα αγώνων στον ίδιο κόσμο, πιο δύσκολη. Ο χρήστης θα πρέπει μόλις ακουμπήσει τη γραμμή τερματισμού της πρώτης πίστας, να τηλε-μεταφέρεται στη δεύτερη πίστα για να τρέξει στο δεύτερο αγώνα. Εισάγετε στο παιχνίδι μουσική που να διαφέρει ανάλογα με την πίστα στην οποία αγωνίζεται ο πρωταγωνιστής!

Κεφάλαιο 8ο: Μεταβλητές

8.1 Εισαγωγή

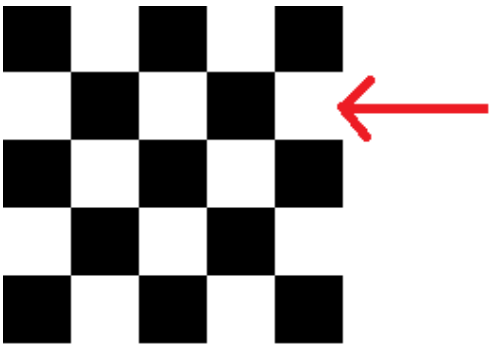
Τι θα λέγατε για ακόμη περισσότερη δράση στα παιχνίδια σας; Δεν θα ήταν πιο ενδιαφέρον να προσθέτατε σκορ, ενέργεια, χρόνο και ζωές; Έτσι, τα παιχνίδια σας θα έμοιαζαν πολύ περισσότερο με τα υπόλοιπα που γνωρίζετε! Θυμηθείτε! Σε πόσα από αυτά έχετε συγκεντρώσει πόντους π.χ. τρώγοντας μήλα, μαζεύοντας αστέρια, νομίσματα κλπ; Πόσες φορές έχει χρειαστεί να αντιμετωπίσετε το εκνευριστικό *Τέλος Παιχνιδιού (Game Over)* επειδή δεν έχετε άλλες ζωές, επειδή εξαντλήθηκε ο διαθέσιμος χρόνος ή επειδή η ενέργειά σας έχει φτάσει στο κόκκινο; Φανταστείτε λοιπόν τον *Kodu* να κερδίζει δέκα πόντους για κάθε *Χελιδονόψαρο* που σκοτώνει και να μειώνεται η αρχική ενέργεια του όταν συγκρουστεί με κάποιον από τους αντιπάλους του ή να μηδενίζεται όταν έρθει σε επαφή με το νερό που βρίσκεται δεξιά από το μονοπάτι που πρέπει να ακολουθήσει για να φτάσει στο τέρμα. Χμμμ... Και πως θα εισάγουμε τέτοια στοιχεία στα παιχνίδια μας; Η απάντηση βρίσκεται πίσω από μία ίσως καινούρια για εσάς έννοια...τη μεταβλητή. Μεταβλητή! Τι είναι όμως αυτό; Υπομονή, όλα θα ξεδιαλύνουν στη συνέχεια.

8.2 Η μεταβλητή και η σημασία της στα προγράμματα

Στον προγραμματισμό, η μεταβλητή είναι ένα συμβολικό όνομα που δίνεται σε κάποια γνωστή ή άγνωστη ποσότητα ή πληροφορία και χρησιμοποιείται για να αναπαραστήσει ένα στοιχείο δεδομένων. Στην ουσία δηλαδή, είναι μία θέση μνήμης του υπολογιστή με συγκεκριμένο όνομα, στην οποία εκχωρείται μία τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης ενός προγράμματος.

Μα τι κινέζικα είναι αυτά! Μην τρομάζετε, ας τα δούμε ένα ένα αναλυτικά.

Για αρχή, φανταστείτε τη μνήμη του υπολογιστή σαν ένα μεγάλο πίνακα, ο οποίος αποτελείται από πολλά μικρά κουτάκια, σαν να λέμε δηλαδή μία σκακιέρα.



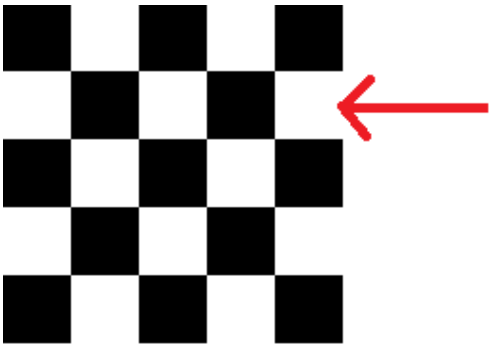
Στην πραγματικότητα, σε κάθε θέση μπορούμε να αποθηκεύσουμε μια τιμή, όπως το σκορ ή τον αριθμό των ζωών που μας απομένουν. Για να μπορούμε, όμως, να χρησιμοποιούμε αυτές τις τιμές στο πρόγραμμά μας, θα πρέπει να δώσουμε ένα όνομα σε κάθε αντίστοιχο κουτάκι. Το όνομα αυτό, είναι και το όνομα της αντίστοιχης *Μεταβλητής*. Η επιλογή της λέξης «*Μεταβλητή*» αντικατοπτίζει το γεγονός ότι τα περιεχόμενα της αντίστοιχης θέσης μνήμης μπορούν διαρκώς να μεταβάλλονται, μπορούν να αλλάζουν.

Αν λοιπόν το κουτάκι, που έχει επισημανθεί στην παραπάνω εικόνα, ανήκει για παράδειγμα σε μία μεταβλητή με όνομα *σκορ*, τότε το περιεχόμενό του θα αντιπροσωπεύει την τιμή που παίρνει το *σκορ* κατά τη διάρκεια εκτέλεσης του προγράμματος.

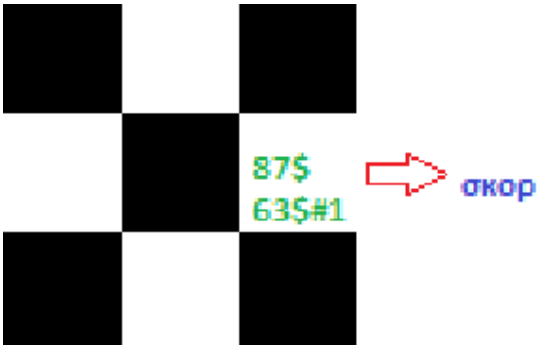
Φανταστείτε να παίζετε «φλίπερ» που στόχος του είναι να μαζέψετε όσο το δυνατόν περισσότερους πόντους.

Αν δεν υπήρχε το σκορ, τότε πώς θα γνωρίζατε τη βαθμολογία σας κατά τη διάρκεια του παιχνιδιού; Αντιμετωπίζοντας το «φλίπερ» καθαρά προγραμματιστικά, παρατηρήστε ότι το σκορ δεν είναι τίποτα άλλο παρά μία μεταβλητή που σας βοηθά να «κρατάτε» τους πόντους σας καθώς παίζετε, δηλαδή να τους αποθηκεύετε στο κουτάκι που τους αναλογεί στη σκακιέρα της μνήμης.

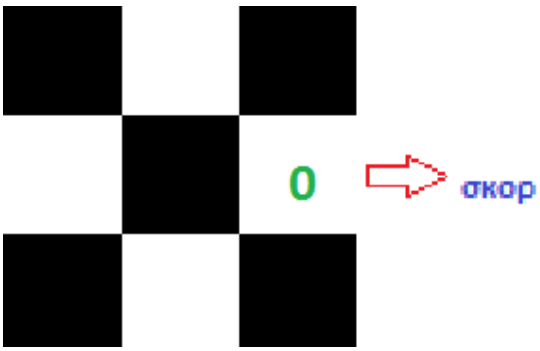
Πριν την έναρξη του παιχνιδιού η μεταβλητή «σκορ» απλά «δεσμεύει» ένα κουτάκι από τον παρακάτω πίνακα, δίνοντάς του ταυτόχρονα το όνομα «σκορ». Με τον όρο «δεσμεύει» εννοούμε ότι από τη στιγμή που το κουτάκι αυτό έχει επιλεχθεί για να αποθηκεύει την τιμή της μεταβλητής σκορ, δεν μπορεί κανένας άλλος να γράψει σε αυτό χωρίς να χρησιμοποιήσει τον όρο σκορ.



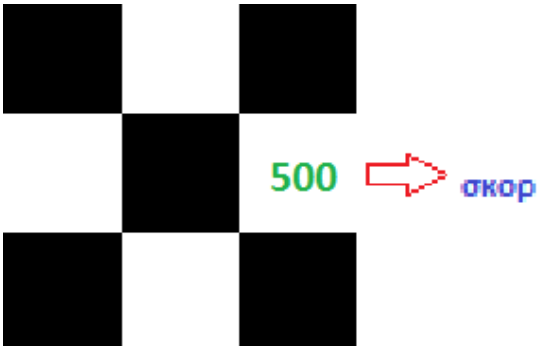
Το κουτάκι αυτό αρχικά δεν είναι άδειο, αλλά περιέχει έναν αριθμό ο οποίος δεν χρησιμεύει πουθενά και γι' αυτό πολλές φορές χαρακτηρίζουμε το περιεχόμενό του «*σκουπίδια*». Επομένως, μέχρι στιγμής η μνήμη του υπολογιστή, δηλαδή η σκακιέρα μας, είναι ως εξής:



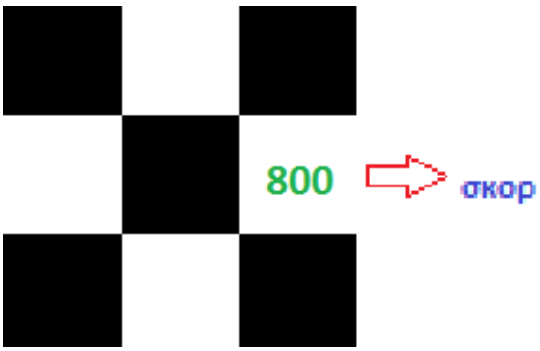
Τη στιγμή που θα αρχίσει το παιχνίδι και πριν σημειωθεί κάποιο σκορ, ο προγραμματιστής πρέπει να έχει φροντίσει να μηδενίσει την τιμή του σκορ, δηλαδή να έχει πάει στο συγκεκριμένο κελί και να έχει τοποθετήσει στη θέση των «σκουπιδιών» την τιμή μηδέν.



Αν τώρα κατά τη διάρκεια του παιχνιδιού θέλουμε να αυξηθεί το σκορ, έστω κατά πεντακόσιους πόντους, η τιμή της μεταβλητής σκορ θα πρέπει να «τροποποιηθεί». Ο προγραμματιστής δηλαδή, θα πρέπει να ανακαλέσει από τη μνήμη του υπολογιστή την παλιά τιμή (μηδέν), να προσθέσει σε αυτήν τους πεντακόσιους πόντους και, έπειτα, να αποθηκεύσει το αποτέλεσμα στο ίδιο κελί.



Αν τώρα το σκορ συνεχίσει να αυξάνεται, για παράδειγμα προστεθούν άλλοι τριακόσιοι πόντοι, η διαδικασία θα είναι αντίστοιχη. Όπως και πριν, θα πρέπει το πρόγραμμά μας να ανακαλέσει την παλιά τιμή της μεταβλητής σκορ, που βρίσκεται στο ανάλογο κελί, και να την αθροίσει με τους καινούριους πόντους, δηλαδή 500+300=800. Στη συνέχεια, θα πρέπει να αποθηκεύσει το αποτέλεσμα αυτό στο αντίστοιχο κελί. Η εικόνα επομένως της μνήμης θα είναι κάπως έτσι:



Υποστηρίζει το MSKodu μεταβλητές; Όπως όλα τα προγραμματιστικά περιβάλλοντα έχει μεταβλητές και μας παρέχει αισθητήρες που διαβάζουν τις τιμές τους και ενέργειες που μας επιτρέπουν να αλλάζουμε τις τιμές τους.

Οι μεταβλητές που προσφέρει είναι δύο ειδών: οι καθολικές και οι τοπικές. Οι καθολικές μεταβλητές είναι ορατές από όλους, δηλαδή οποιοδήποτε αντικείμενο μπορεί να ανακαλέσει μια τέτοια μεταβλητή και να γράψει πάνω σε αυτήν. Από την άλλη, έχουμε τις τοπικές μεταβλητές, που μοιάζουν περισσότερο με ιδιότητες ενός χαρακτήρα και παρόλο που μπορεί να επηρεάζονται από το γύρω περιβάλλον, ελέγχονται, δηλαδή διαβάζονται, γράφονται, αρχικοποιούνται, κλπ. μόνο από τον ιδιοκτήτη τους, δηλαδή το αντίστοιχο αντικείμενο.

Ένα χαρακτηριστικό παράδειγμα καθολικής μεταβλητής στο MSKodu είναι το σκορ. Από τη στιγμή που θα εμφανιστεί στο παιχνίδι, ο προγραμματιστής μπορεί να αλλάξει τη τιμή του σκορ μέσα από οποιοδήποτε αντικείμενο του παιχνιδιού. Οι καθολικές μεταβλητές είναι σχεδιασμένες ώστε να «κρατούν» ακέραιες τιμές και αναπαρίστανται με 37 διαφορετικούς τρόπους (από όλα τα γράμματα της αγγλικής αλφαβήτου και όλα τα χρώματα που υποστηρίζει το MSKodu)!

Όσον αφορά τις τοπικές μεταβλητές, το χαρακτηριστικότερο ίσως παράδειγμα για να κατανοήσετε τη χρησιμότητα και το ρόλο τους, είναι η *Ενέργεια (Health).* Η ενέργεια ενός αντικειμένου επειδή αφορά ατομικά το κάθε αντικείμενο, θα πρέπει να βρίσκεται «μέσα» σε αυτό, άλλωστε αυτός είναι και ο λόγος που ονομάζεται τοπική η συγκεκριμένη μεταβλητή. Αν και εξακολουθεί να υπάρχει η δυνατότητα να βλέπουν και τα άλλα αντικείμενα μία τοπική μεταβλητή, ο μόνος που μπορεί να την τροποποιήσει είναι το αντικείμενο-ιδιοκτήτης της, δηλαδή το αντικείμενο στο οποίο αναφέρεται. Για να κατανοήσετε καλύτερα τις τοπικές μεταβλητές, φανταστείτε να μειώνεται η ενέργεια του *Μηχανάκια (Cycle)* όταν συγκρούεται με τον *Kodu*. Ο *Kodu* μπορεί να βλέπει την υγεία του *Μηχανάκια (Cycle)* με τη χρήση αισθητήρων, όμως δεν μπορεί να την αλλάξει με τη χρήση ενεργειών. Γι' αυτό και η αντίστοιχη εντολή θα πρέπει να τοποθετηθεί στον *Μηχανάκια (Cycle)* που είναι ο «ιδιοκτήτης» τις μεταβλητής, και όχι στον *Kodu*.

Μετά τη μικρή θεωρητική παρένθεση, ας αξιοποιήσουμε τις μεταβλητές στα παιχνίδια μας!

8.3 Σκορ, Ενέργεια, Χρόνος, Ζωές

8.3.1 Σκορ (Score)

Πάμε να βάλουμε σκορ στα παιχνίδια μας! Το σκορ στο MSKodu είναι μία μεταβλητή η οποία μπορεί να αποθηκεύει πόσους πόντους έχετε, πόσα μήλα έχετε κερδίσει, πόσους εχθρούς έχετε σκοτώσει κλπ. Το MSKodu όμως, σας δίνει τη δυνατότητα να έχετε παραπάνω από ένα σκορ σε ένα παιχνίδι, γεγονός που επιτυγχάνεται μέσα από τα διάφορα χρώματα και τα διάφορα γράμματα της αγγλικής αλφαβήτου που προσφέρει. Η ποικιλία αυτή στα σκορ δεν αποτελεί διακοσμητικό στοιχείο όπως ίσως φαίνεται εκ πρώτης όψεως. Αντιθέτως, σας επιτρέπει στην ουσία να έχετε πολλαπλές μεταβλητές. Με αυτόν τον τρόπο είναι πλέον εφικτό να κρατάτε τιμές για διάφορα πράγματα που συμβαίνουν στον κόσμο σας. Για παράδειγμα, αν θέλετε να κρατήσετε σε κάποιο παιχνίδι σας πόσα μήλα έφαγε ο *Kodu* αλλά και πόσες πέτρες ακούμπησε, αρκεί να κρατήσετε τον αριθμό των μήλων στο κόκκινο σκορ και τον αριθμό των βράχων στο πράσινο σκορ.

Ποιες όμως είναι οι ενέργειες και ποιοι οι αισθητήρες που δίνονται από το MSKodu για να παίζετε με το σκορ;

Α) (Ενέργειες για το σκορ) Οι προσφερόμενες ενέργειες για την αλλαγή του σκορ από τα αντικείμενά μας, είναι η ***Αύξηση (Score)*** *και* ***Μείωση (Subtract)****,* οι οποίες βρίσκονται κρυμμένες πίσω από το εικονίδιο *Παιχνίδι (Game)* στην αρχική πίτα ενεργειών του MSKodu:

Ο ρόλος των δυο ενεργειών είναι η αύξηση ή η μείωση αντίστοιχα της τιμής της μεταβλητής *σκορ*, ενώ πρέπει να τις συνδυάσετε με κάποιο από τα προσδιοριστικά ***Βαθμοί (Points)*** ώστε να δηλώσετε τους πόντους που θέλετε να προστεθούν ή να αφαιρεθούν από το σκορ σας. Για παράδειγμα, αν θέλετε ένας χαρακτήρας, όταν συναντήσει ένα μαύρο μήλο, να αυξάνει το πράσινο σκορ του κατά δύο βαθμούς, τότε μπορείτε να δημιουργήσετε την εξής συμπεριφορά:

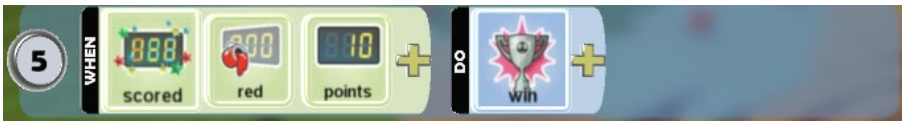
ΟΤΑΝ [πέσεις πάνω][μήλο][μαύρο] **ΤΟΤΕ** [πρόσθεσε][πράσινο σκορ][2 βαθμούς]

Αντίστοιχα, αν θέλετε να μειώνεται κατά δυο βαθμούς το πράσινο σκορ, μπορείτε να το υλοποιήσετε ως εξής:

ΟΤΑΝ [πέσεις πάνω][μήλο][μαύρο] **ΤΟΤΕ** [αφαίρεσε][πράσινο σκορ][2 βαθμούς]

Β) <u>(Αισθητήρας για το σκορ)</u> Ο αισθητήρας που επιτρέπει στα αντικείμενα να ενεργούν ανάλογα με τις τιμές που παίρνουν οι μεταβλητές σκορ είναι ο ***Σκορ (Scored)***. Ο αισθητήρας ***Σκορ (Scored)*** δίνει τη δυνατότητα να αναγνωρίσουμε πότε το σκορ έχει πάρει κάποια τιμή, δηλαδή μας επιτρέπει να ανακαλέσουμε, να συγκρίνουμε και γενικά να χρησιμοποιήσουμε την τιμή του αποθηκευμένου σκορ.

Τα πιο βασικά προσδιοριστικά που μπορούν να ακολουθούν τον αισθητήρα *Σκορ (Scored)* είναι οι ***Βαθμοί (Points)***, *το* ***Πάνω από (Above)*** *και το* ***Κάτω από (Below)***. Αν θέλουμε ο χαρακτήρας μας να αντιλαμβάνεται πότε η τιμή του σκορ του είναι ίση με κάποια άλλη τιμή, αρκεί μετά τον αισθητήρα *Σκορ (Scored),* να τοποθετήσουμε το προσδιοριστικό *Βαθμοί (Points)* με την επιθυμητή τιμή. Στο παρακάτω παράδειγμα, αν το κόκκινο σκορ ισούται με 10 βαθμούς, τότε ο παίκτης κερδίζει:



Αν πάλι επιθυμούμε να προκαλέσουμε μια ενέργεια όταν η τιμή του σκορ να είναι πάνω από κάποιον αριθμό, τότε πρέπει να προσθέσουμε το προσδιοριστικό ***Πάνω από (Above)***, ενώ αν θέλουμε να είναι μικρότερη από μια συγκεκριμένη τιμή χρησιμοποιούμε το προσδιοριστικό ***Κάτω από (Below)***. Για παράδειγμα, αν θέλετε ο χαρακτήρας να κερδίζει όταν το σκορ παίρνει τιμή μεγαλύτερη του δέκα, τότε αρκεί να δημιουργήσετε την εξής συμπεριφορά:

Αν θέλετε να πειραματιστείτε ακόμη περισσότερο με τα προσδιοριστικά του σκορ, μπορείτε να ανατρέξετε στην ανάλογη πίτα που σας παρέχει το MSKodu. Εκεί μπορείτε προσδιοριστικά για τα διαφορετικά χρώματα και γράμματα για το σκορ, το προσδιοριστικό ***Τυχαία (Random)***, το οποίο σας επιτρέπει να χρησιμοποιείτε τυχαίους αριθμούς στο παιχνίδι σας, καθώς και το προσδιοριστικό της άρνησης ***Όχι (Not)***.

Αν επιθυμείτε να εισάγετε στο παιχνίδι σας έναν αριθμό που δεν υποστηρίζεται από την πίτα του MSKodu, μπορείτε να τον εμφανίσετε σαν άθροισμα. Δηλαδή, αν θέλετε να χρησιμοποιήσετε τον αριθμό 60 πρέπει να τοποθετήσετε τα προσδιοριστικά *Βαθμοί10 (Points10)* και *Βαθμοί50 (Points50)* δίπλα δίπλα κάπως έτσι:

]

Δείτε το παράδειγμα 08_01.kodu

Ας δούμε τη χρήση των μεταβλητών στο παράδειγμα *Kodu*: Ο Μηλοφάγος! **[08_01.kodu]**. Υποθέστε λοιπόν ότι έχουμε ένα κόσμο στον οποίο υπάρχει ο *Μηχανάκιας (Cycle),* ένα *Δέντρο*, μία *Πέτρα*, δεκαπέντε κόκκινα και τρία μαύρα *Μήλα*.

Σκοπός του παιχνιδιού είναι συγκεντρώσει ο *Μηχανάκιας (Cycle)* δέκα πόντους. Πώς θα γίνει όμως αυτό; Κάθε φορά που συναντά ένα *Μήλο*, το τρώει. Αν το *Μήλο* που έφαγε είναι κόκκινο, τότε θα πρέπει να προστίθεται ένας πόντος στο αρχικό σκορ, ενώ αν είναι μαύρο, θα πρέπει να αφαιρείται ένας πόντος. Αν ακουμπήσει ο ήρωάς μας την *Πέτρα*, τότε το παιχνίδι τελειώνει, ενώ αν φάει δέκα κόκκινα *Μήλα*, γεγονός που θα του εξασφαλίσει τους δέκα πολυπόθητους βαθμούς, ανακηρύσσεται νικητής.

Οι ΣΥΜΠΕΡΙΦΟΡΕΣ που θα πρέπει να έχει ο *Μηχανάκιας (Cycle)* είναι:

- **ΟΤΑΝ** αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος, **ΤΟΤΕ** κινείται στην αντίστοιχη κατεύθυνση πάνω στην πίστα.
- **ΟΤΑΝ** ακουμπήσει ένα μήλο, **ΤΟΤΕ** το τρώει.
- **ΟΤΑΝ** ακουμπήσει ένα κόκκινο μήλο, **ΤΟΤΕ** αυξάνει το σκορ κατά ένα πόντο.
- **ΟΤΑΝ** ακουμπήσει ένα μαύρο μήλο, **ΤΟΤΕ** μειώνει το σκορ κατά ένα πόντο.
- **ΟΤΑΝ** συγκεντρώσει δέκα πόντους, **ΤΟΤΕ** τερματίζει το παιχνίδι με νίκη για τον παίκτη.
- **ΟΤΑΝ** ακουμπήσει μία πέτρα, **ΤΟΤΕ** τερματίζει το παιχνίδι με ήττα για τον παίκτη.

Προσέξτε ότι ουσιαστικά ταυτίσαμε το σκορ με τον αριθμό μήλων. Δηλαδή χρησιμοποιήσαμε την μεταβλητή σκορ για να αποθηκεύουμε τον αριθμό των μήλων που έχει φάει ο ήρωάς μας. Φαίνεται δηλαδή πόσο ευέλικτα μπορούμε να χρησιμοποιούμε την έννοια της μεταβλητής.

Ας πάμε λοιπόν να προγραμματίσουμε τον πρωταγωνιστή μας:

α) Ο μηχανάκιας πρέπει να αλληλεπιδρά με το πληκτρολόγιο, δηλαδή να μπορεί να κινείται προς όλες τις κατευθύνσεις όταν ο χρήστης χρησιμοποιεί τα βέλη του πληκτρολογίου. Άρα, θα χρησιμοποιήσουμε τον αισθητήρα *Πληκτρολόγιο (Keyboard)* με προσδιοριστικό τα *Βέλη (Arrows)* και ως ενέργεια η *Κινήσου (Move)*:

β) Όταν συναντά κάποιο *Μήλο*, ο *Μηχανάκιας (Cycle)* πρέπει να το τρώει. Άρα, εδώ θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα *Πέφτω πάνω σε (Bump)* και την ενέργεια *Τρώω (Eat).* Προσέχουμε να μην ξεχάσουμε να προσθέσουμε στην ενέργεια *Τρώω (Eat)* το προσδιοριστικό *Αυτό (It)*:

γ) Όταν συναντά ένα κόκκινο *Μήλο,* το σκορ του μηχανάκια πρέπει να αυξάνεται κατά ένα. Αυτό το νούμερο θα πρέπει να είναι αποθηκευμένο και να αλλάζει διαρκώς και γι' αυτόν το λόγο θα χρησιμοποιήσουμε τη μεταβλητή *Σκορ.* Ποιο σκορ όμως; Ας επιλέξουμε το κόκκινο. Άρα, χρειαζόμαστε τον αισθητήρα *Πέφτω πάνω (Bump)* και την ενέργεια ***Αύξησε (Score)***.

ΟΤΑΝ[πέσεις πάνω][σε μήλο][Κόκκινο] **ΤΟΤΕ** [πρόσθεσε][Κόκκινο σκορ][ένα βαθμό].

Παρατηρήστε τα προσδιοριστικά που χρησιμοποιήσαμε στην παραπάνω εντολή. Το ***Κόκκινο Σκορ (Score Red)*** χρησιμεύει για να αναφερθούμε σε ποια μεταβλητή σκορ θέλουμε να αλλάξουμε την τιμή ενώ με το ***Βαθμός 01 (Point 01)*** προσδιορίζουμε το μέγεθος αλλαγής της μεταβλητής.

Δημιουργήστε για το Μηχανάκια την ακόλουθη συμπεριφορά:

δ) Όταν συναντά ο μηχανάκιας ένα μαύρο μήλο, το σκορ του πρέπει να μειώνεται κατά ένα. Θα χρειαστούμε προφανώς την ενέργεια ***Μείωσε (Subtract)***, η οποία πρέπει να εφαρμοστεί στο κόκκινο σκορ:

ΟΤΑΝ[Πέσεις πάνω][σε Μήλο][Μαύρο] **ΤΟΤΕ** [Μείωσε][Κόκκινο Σκορ][ένα βαθμό].

Παρατηρήστε πως χρησιμοποιούμε παραπλήσια προσδιοριστικά με την προηγούμενη συμπεριφορά:

ε) Τέλος, όταν συγκεντρώσει δέκα πόντους ο *Μηχανάκιας (Cycle),* τότε θα πρέπει να ανακηρύσσεται νικητής. Ουσιαστικά θέλουμε να συγκρίνουμε την τρέχουσα τιμή του σκορ με το 10 και αν οι δύο αυτές τιμές είναι ίσες, τότε το παιχνίδι θα πρέπει να τελειώνει με το *Μηχανάκια (Cycle)* να είναι ο μεγάλος νικητής. Άρα, εδώ θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα ***Σκορ (Scored)*** και ως ενέργεια το ***Νίκη (Win)***.

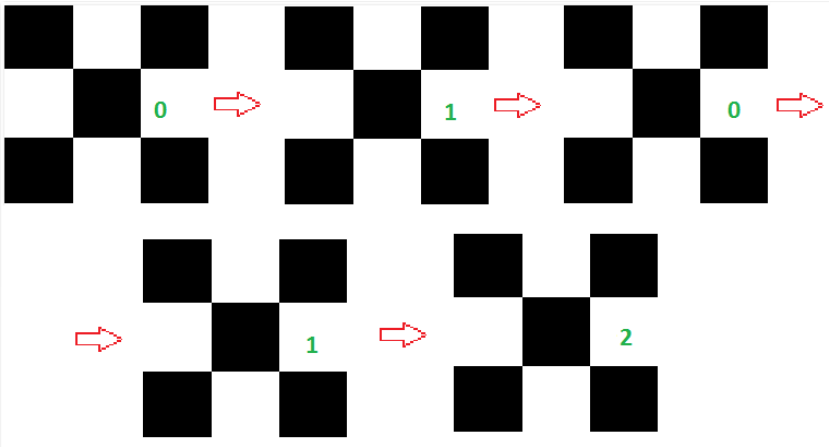
ΟΤΑΝ[η τιμή του σκορ][κόκκινο][10 βαθμοί] **ΤΟΤΕ** [Νίκησε ο παίκτης].

Προσέξτε ότι σε αυτό το παράδειγμα δε χρησιμοποιήσαμε κάποιο προσδιοριστικό σύγκρισης της τιμής, όπως π.χ. ***Πάνω από (Above)*** ή ***Κάτω από (Below)***. Σε αυτή την περίπτωση το MSKodu θεωρεί ως προκαθορισμένο προσδιοριστικό την ισότητα.

στ) Τέλος, όταν ο μηχανάκιας χτυπήσει πάνω σε ένα *Βράχο* τότε θα πρέπει να χάνει οριστικά. Άρα, θα χρειαστούμε το συνδυασμό του αισθητήρα *Χτυπάω σε (Bump)* και της ενέργειας ***Ήττα (End)***.

Ήρθε η ώρα να ξεκινήσετε να παίζετε! Παρατηρήσατε κάτι καινούριο στην εικόνα του παιχνιδιού; Πάνω δεξιά στην οθόνη και με κόκκινο χρώμα εμφανίζεται το σκορ σας το οποίο αρχικά έχει την τιμή μηδέν, ενώ όσο ο χρήστης τρώει *Μήλα* αυτό αυξάνεται.

Για να δούμε όμως, πώς θα είναι η μνήμη του υπολογιστή μας, αν ο *Μηχανάκιας (Cycle)* στην αρχή του παιχνιδιού φάει εναλλάξ ένα κόκκινο και ένα μαύρο *Μήλο* και δύο κόκκινα αμέσως μετά.



Μπορείτε να εξηγήσετε μόνοι σας τις πέντε διαφορετικές καταστάσεις της μνήμης;

Πρόταση για παιχνίδι με σκορ!! Φανταστείτε λοιπόν έναν κόσμο που εκτός από τον *Kodu* των Δασών, υπάρχουν κάποιοι πλούσιοι εχθροί, οι οποίοι προσπαθούν να απομακρυνθούν από αυτόν και ένας μισθοφόρος που προσπαθεί να το σκοτώσει. Σκοπός του σύγχρονου Kodu των Δασών είναι να πετύχει με τους πυραύλους του τόσο τους πλουσίους όσο και το μισθοφόρο και να τους σκοτώσει. Κάθε φορά που σκοτώνει έναν από τους πλούσιους εχθρούς, πετάγονται νομίσματα, τα οποία πρέπει να μαζέψει για να τα δώσει στους φτωχούς. Άρα, για το παιχνίδι μας θα χρειαστούμε δύο σκορ: το πρώτο θα είναι με κόκκινο χρώμα και θα αυξάνει κατά ένα κάθε φορά που ο Ρομπέν σκοτώνει έναν πλούσιο και το δεύτερο θα είναι με πράσινο και θα αυξάνει κατά τρία κάθε φορά που θα κλέβει ένα νόμισμα. Το παιχνίδι τελειώνει όταν ο *Kodu* (Ρομπέν των δασών) καταφέρει να σκοτώσει το μισθοφόρο. Έχοντας μάθει στο κεφάλαιο 4 πώς να φτιάχνετε πίστες, μία καλή ιδέα θα ήταν να διαμορφώσετε την πίστα σας με τέτοιον τρόπο ώστε να μοιάζει με δάσος.

8.3.2 Υγεία (Health)

Στα περισσότερα παιχνίδια δράσης, ο ήρωας ξεκινά με συγκεκριμένη «Υγεία»(πολλές φορές αποκαλείται «Ενέργεια») η οποία αρχίζει να μειώνεται όταν οι αντίπαλοί του καταφέρουν πλήγματα εναντίον του. Στόχος του παίκτη είναι συνήθως να πετύχει το στόχο του παιχνιδιού χωρίς να χάσει όλη την ενέργειά του! Θα θέλατε να βάλουμε και στα δικά μας παιχνίδια την έννοια της *Υγείας-Ενέργειας (Health)*;

Οι σχετικές διαθέσιμες επιλογές στο MSKodu είναι οι εξής:

Α) (Ενέργειες για την «*Υγεία*») Η ***Προκαλώ βλάβη (Damage)*** και η ***Επουλώνω (Heal)***, αποτελούν τις βασικές ενέργειες με τις οποίες αλλάζουμε την κατάσταση της υγείας των αντικειμένων μας. Οι ενέργειες αυτές είναι διαθέσιμες στην επιλογή *Μάχη (Combat)* της αρχικής πίτας ενεργειών. Όπως προδίδουν και οι αντίστοιχοι αγγλικοί όροι (*προκάλεσε ζημιά και επούλωσε*), οι δυο ενέργειες παίζουν αντίστροφους ρόλους, η μία προκαλεί ζημιά στο αντικείμενο και η άλλη διορθώνει τη ζημιά. Δηλαδή, αν σε κάποιο παράδειγμα θέλουμε να μειώνεται η «ενέργεια-υγεία» του *Kodu* όταν πέσει πάνω σε ένα *Μηχανάκια (Cycle)*, τότε μπορούμε να χρησιμοποιήσουμε τη δράση ***Μείωση Ενέργειας (Damage)***. Κατά πόσο όμως θα μειωθεί η ενέργεια; Άλλη ζημιά προκαλεί η σύγκρουση με τον μηχανάκια και άλλη το χτύπημα του Kodu από έναν πύραυλο! Για αυτό το λόγο και οι δυο ενέργειες συνδυάζονται με το προσδιοριστικό ***Βαθμοί (Points)*** που καθορίζει το μέγεθος της αλλαγής της ενέργειας του αντικειμένου (μετριέται σε πόντους όπως το σκορ). Μια σχετική συμπεριφορά για την σύγκρουση του Kodu με τον Μηχανάκια:

Η συμπεριφορά αυτή, έχει ως αποτέλεσμα τη μείωση της ενέργειας του αντικειμένου κατά 1 βαθμό. Αν σε άλλη περίπτωση επιθυμούμε να αλλάξουμε το εύρος της μείωσης αυτής, τότε απλά μετά την ***Μείωση Ενέργειας (Damage)*** πρέπει να επιλέξουμε ***Βαθμούς (Points)*** με τον αντίστοιχο αριθμό. Αν τώρα θελήσουμε να αυξήσουμε την ενέργεια και να επουλώσουμε τα τραύματα, τότε θα χρησιμοποιήσουμε την ***Αύξηση Ενέργειας (Heal)***. Η μόνη διαφορά έγκειται στο γεγονός ότι

πλέον οι αντίστοιχοι ***Βαθμοί (Points)*** προστίθενται στην ήδη υπάρχουσα ενέργεια και δεν αφαιρούνται από αυτήν.

Ωστόσο, και στις δύο περιπτώσεις πρέπει να δηλώσουμε το αντικείμενο του οποίου η ενέργεια αυξάνεται ή μειώνεται. Δηλαδή, από την προηγούμενη συμπεριφορά λείπει ακόμη ένα προσδιοριστικό, το ***Εμένα (Me)*** ή το ***Αυτό (It)***. Τα προσδιοριστικά αυτά καθορίζουν σε ποιο αντικείμενο θα γίνει η αλλαγή της ενέργειας, με το πρώτο να υποδηλώνει το χαρακτήρα που περιέχει τη συμπεριφορά και το δεύτερο, το αντικείμενο που εμφανίζεται στον αισθητήρα της συμπεριφοράς. Έτσι, αν θέλουμε να αυξάνεται η ενέργεια του *Kodu* αν πέσει πάνω σε κάποιο νόμισμα, τότε θα δημιουργήσουμε την εξής συμπεριφορά:

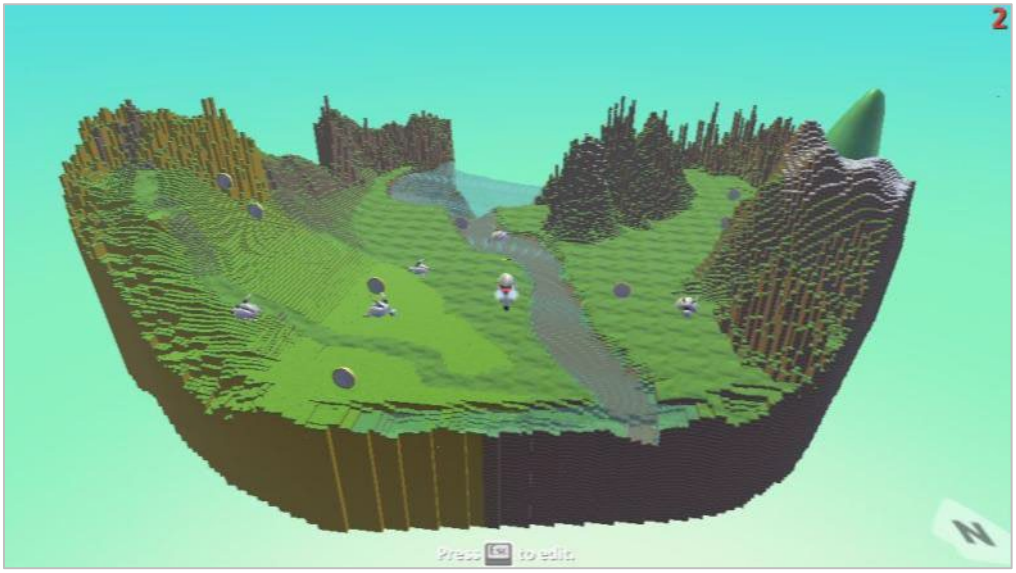
Είναι εξαιρετικά σημαντικό να τονίσουμε, ότι από τη «φύση» τους, τα αντικείμενα στο MSKodu, έχουν συμπεριφορές που αφορούν την Υγεία τους. Π.χ. αν αρχίσετε να πυροβολείτε ένα αντικείμενο, τότε όταν η υγεία του μηδενιστεί, το αντικείμενο θα εξαφανιστεί. Συνεπώς, δεν καλείστε στα παιχνίδια σας να ξανα-δημιουργείτε τις τυπικές συμπεριφορές που έχει η ενέργεια Πυροβολώ (Shoot), αλλά να προσθέσετε πιο σύνθετες συμπεριφορές όπως π.χ. να βελτιώσετε την υγεία του Kodu, όταν αυτός φάει ένα μήλο!

Β) (αισθητήρας για την «Υγεία») Ο αισθητήρας ***Υγεία (Health)***, μας δίνει τη δυνατότητα να δημιουργούμε συμπεριφορές που εξαρτώνται από την τιμή της ενέργειας-υγείας του αντικειμένου. Δεν θα θέλατε, για παράδειγμα, όταν η ενέργεια του κάστρου που προστατεύει ο ήρωάς σας μηδενιστεί, τότε να τελειώσει το παιχνίδι; Ή όταν η ενέργεια ενός αντικειμένου μειωθεί πολύ, το αντικείμενο αυτό να αρχίσει να κινείται πιο αργά; Παρατηρήστε ότι και τα δυο παραδείγματα απαιτούν τη χρήση του αισθητήρα ***Υγεία (Health)*** για να συγκρίνουν την τρέχουσα ενέργεια του αντικειμένου με ένα άλλο επίπεδο ενέργειας. Για αυτό το λόγο και υπάρχουν δυο ειδών προσδιοριστικά που συνοδεύουν τον αισθητήρα ***Υγεία (Health)***: α) οι ***Βαθμοί (Points)***: με το προσδιοριστικό αυτό, ο προγραμματιστικής επιλέγει ποια θα είναι η τιμή σύγκρισης και β) ο *τρόπος σύγκρισης*, όπως ακριβώς και στον αισθητήρα ***Σκορ (Scored)***. Αν δεν προσθέσουμε προσδιοριστικό σύγκρισης, τότε υπονοείται η ισότητα ενώ επιπλέον υπάρχουν τα προσδιοριστικά ***Πάνω από (Above)***, ***Κάτω από (Below)***. Μπορείτε να ερμηνεύσετε την επόμενη συμπεριφορά;

Από τη στιγμή που δεν υπάρχει προσδιοριστικό σύγκρισης, η συμπεριφορά αυτή δηλώνει ότι όταν η ενέργεια-υγεία του αντικειμένου μας γίνει ίση με το 0, τότε θα τελειώσει το παιχνίδι.

Δείτε το παράδειγμα *08_02.kodu*

Ας δούμε όμως την Υγεία-Ενέργεια μέσα σε ένα παράδειγμα **[08_02.kodu]**. Υποθέστε πάλι πως είμαστε σε έναν κόσμο που περιέχει το *Μηχανάκια (Cycle)*, ένα *Δέντρο*, δέκα *Νομίσματα* και πέντε *Χελώνες*.



Σε αυτό το παράδειγμα, στόχος του χρήστη είναι να καταφέρει να πάρει τα δέκα *Νομίσματα* αποφεύγοντας όμως τις *Χελώνες (Turtles)* που κινούνται πάνω σε πολύπλοκα και απρόβλεπτα μονοπάτια και οι οποίες κάθε φορά που αγγίζουν το *Μηχανάκια (Cycle)* του προξενούν βλάβη μειώνοντας την ενέργειά του. Το παιχνίδι θα πρέπει να τελειώνει είτε όταν ο *Μηχανάκιας (Cycle)* δεν έχει ενέργεια είτε όταν μαζέψει και τα δέκα *Νομίσματα*.

Στο συγκεκριμένο παιχνίδι δυο αντικείμενα ουσιαστικά εμφανίζουν συμπεριφορές, ο *Μηχανάκιας (Cycle)* και οι *Χελώνες (Turtles)*.

Οι ΣΥΜΠΕΡΙΦΟΡΕΣ που έχει ο *Μηχανάκιας (Cycle)* είναι:

- **ΟΤΑΝ** αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος, **ΤΟΤΕ** κινείται στην αντίστοιχη κατεύθυνση πάνω στην πίστα.
- **ΟΤΑΝ** ακουμπήσει ένα *Νόμισμα*, **ΤΟΤΕ** το εξαφανίζει.
- **ΟΤΑΝ** ακουμπήσει ένα *Νόμισμα*, **ΤΟΤΕ** αυξάνει το σκορ κατά ένα πόντο.
- **ΟΤΑΝ** ακουμπήσει μία *Χελώνα*, **ΤΟΤΕ** μειώνει την ενέργεια του κατά μία μονάδα.
- **ΟΤΑΝ** συγκεντρώσει δέκα πόντους, **ΤΟΤΕ** τερματίζει το παιχνίδι με νίκη για τον παίκτη.
- **ΟΤΑΝ** η ενέργειά του μηδενιστεί,**ΤΟΤΕ** τερματίζει το παιχνίδι με ήττα για τον παίκτη.

Η ΣΥΜΠΕΡΙΦΟΡΑ που έχει κάθε *Χελώνα* είναι:

- Να μετακινείται για πάντα πάνω στο προκαθορισμένο μονοπάτι.

Ας πάμε να προγραμματίσουμε το *Μηχανάκια (Cycle)*:

α) Ο μηχανάκιας πρέπει να αλληλεπιδρά με το πληκτρολόγιο, δηλαδή να μπορεί να κινείται προς όλες τις κατευθύνσεις όταν ο χρήστης χρησιμοποιεί τα βέλη του πληκτρολογίου:

β) Όταν συναντά κάποιο *Νόμισμα*, ο *Μηχανάκιας (Cycle)* πρέπει να το εξαφανίζει. Άρα, εδώ θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα *Πέφτω πάνω σε (Bump)* σε συνδυασμό με την ενέργεια ***Εξαφανίζω (Vanish)***. Προσέχουμε να μην ξεχάσουμε να προσθέσουμε στην εντολή το προσδιοριστικό ***Αυτό (It)*** ώστε να εξαφανίζεται το νόμισμα και όχι ο ήρωάς μας!

γ) Όταν ο Μηχανάκιας (Cycle) ακουμπά ένα νόμισμα, θα πρέπει να αυξάνεται το σκορ του κατά έναν βαθμό. Τη μεταβλητή σκορ τη χρησιμοποιούμε στο συγκεκριμένο παράδειγμα, για να αποθηκεύσουμε τον αριθμό των νομισμάτων που έχει πάρει ο ήρωάς μας. Έστω ότι θα αξιοποιήσουμε το κόκκινο σκορ:

δ) Όταν ο μηχανάκιας συναντά μία *Χελώνα,* η ενέργεια του θα πρέπει να μειώνεται κατά μία μονάδα. Άρα, θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα *Χτυπάω σε (Bump)* σε συνδυασμό με την ενέργεια ***Προκαλώ βλάβη (Damage)***. Και σε αυτή την περίπτωση δεν πρέπει να ξεχάσουμε το προσδιοριστικό ***Εμένα (Me)*** ώστε να μειωθεί η ενέργεια του Μηχανάκια και όχι της Χελώνας!

Ουπς! Κάποιο λάθος έκανε ο προγραμματιστής μας. Μπορείτε να το διορθώσετε;

ε) Όταν συγκεντρώσει δέκα μήλα ο *Μηχανάκιας (Cycle)* (δηλαδή έχει 10 βαθμούς στο κόκκινο σκορ), ανακηρύσσεται νικητής. Θα πρέπει προφανώς να χρησιμοποιήσουμε τον αισθητήρα ***Σκορ (Scored)*** σε συνδυασμό με την ενέργεια ***Νίκη (Win)***. Μη ξεχνάτε να προσδιορίζετε τον τύπο του σκορ στο οποίο αναφέρεστε (στη δική μας περίπτωση το κόκκινο σκορ).

στ) Όταν η ενέργειά του μηδενιστεί, τότε το παιχνίδι τερματίζει με ήττα για τον παίκτη. Άρα:

ΟΤΑΝ [η ενέργεια][μηδέν] **ΤΟΤΕ** [Ήττα]

Παρατηρείστε ότι δεν χρειάζεται τελεστής σύγκρισης μετά τον αισθητήρα ***Υγεία (Health)***, καθώς η συμπεριφορά θέλουμε να ενεργοποιείται όταν η υγεία-ενέργεια ΙΣΟΥΤΑΙ με μηδέν.

Ας προγραμματίσουμε τώρα τις *Χελώνες*:

α) Να κινούνται πάνω στο προκαθορισμένο μονοπάτι. Στη συγκεκριμένη συμπεριφορά δεν χρησιμοποιούμε κανέναν αισθητήρα, καθώς θέλουμε να εκτελείται διαρκώς. Χρειαζόμαστε όμως την ενέργεια *Κίνηση (Move)* με προσδιοριστικό *Στο Μονοπάτι (On Path)*:

Εάν αρχίσετε να παίζετε το παιχνίδι που μόλις προγραμματίσατε, δεν θα δείτε πουθενά την υγεία του χαρακτήρα. Και πως θα ξέρουμε ποιο είναι το επίπεδο υγείας του χαρακτήρα μας, πόση ενέργεια του απομένει; Όπως μάθατε και στο κεφάλαιο 6, για να κάνετε ορατή την μπάρα που υποδηλώνει το επίπεδο ενέργειας ενός αντικειμένου, πρέπει να πατήσετε δεξί κλικ πάνω στο αντικείμενο και να επιλέξετε *Αλλαγή Ρυθμίσεων (Change Settings)*. ‘Επειτα, από τη σκάλα ιδιοτήτων αρκεί να τσεκάρετε την ιδιότητα *Εμφάνιση Ενέργειας (Show Hit Points)*, προσέχοντας το αντίστοιχο εικονίδιο να γίνει πράσινο. Στην ακριβώς επόμενη ιδιότητα, *Μέγιστη Ενέργεια (Max Hit Points),* μπορείτε να προσδιορίσετε την αρχική ενέργεια που επιθυμείτε για το αντικείμενό σας. Εάν ξεκινήσετε και πάλι το παιχνίδι, θα δείτε μία πράσινη μπάρα πάνω από το *Μηχανάκια (Cycle)*, η οποία δείχνει κάθε στιγμή το επίπεδο ενέργειας που έχει.

Καταλάβατε εδώ ποια είναι η μεταβλητή σας;… Ακριβώς! Η υγεία!! ‘Οπως και πριν με το σκορ, έτσι και τώρα αναφερόμαστε σε ένα καινούριο κουτάκι της μνήμης του υπολογιστή, όπου περιέχεται η αρχική τιμή της υγείας.

Ας μην περιοριστούμε όμως μόνο στα παραδείγματα αυτού του κεφαλαίου. Θυμηθείτε το ενδιαφέρον παράδειγμα του κεφαλαίου 5 με το λαβύρινθο και προσπαθήστε να το αναπτύξετε λίγο περισσότερο. Φανταστείτε δηλαδή πόσο πιο πολύπλοκο αλλά και ευχάριστο θα γινόταν το παιχνίδι αν σε κάθε αδιέξοδο τοποθετούσατε έναν μηχανάκια-εχθρό που θα καραδοκούσε και με το που έβλεπε τον *Kodu* θα άρχιζε να τον κυνηγά. Αν τον ακουμπούσε, τότε θα του μείωνε την ενέργεια, ενώ αν ο *Kodu* κατάφερνε να φτάσει σώος στο τέλος του λαβύρινθου και να φάει το *Μήλο* τότε θα ήταν νικητής και το παιχνίδι θα τελείωνε. Αν όμως η ενέργεια του είχε μηδενιστεί, τότε... *Τέλος Παιχνιδιού (Game Over)*.

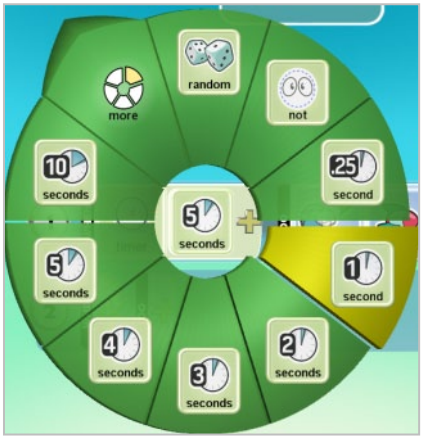
8.3.3 Χρόνος (Time)

Μάλλον λοιπόν ήρθε η ώρα να επαναπροσδιορίσουμε όσα είπαμε ως τώρα, λαμβάνοντας υπ' όψιν μας και χρονικούς περιορισμούς. Τις περισσότερες φορές εγκαταλείπετε ένα παιχνίδι γιατί δεν καταφέρνετε στο διαθέσιμο χρόνο, να πετύχετε το στόχο του π.χ. οι αγώνες αυτοκινήτων έχουν συνήθως ένα κατώτερο χρονικό όριο για να σας επιτρέψουν να κάνετε νέους γύρους στην πίστα του παιχνιδιού. Είναι προφανές ότι ο χρόνος δεν θα μπορούσε να λείπει από το MSKodu. Καθώς ο χρόνος δεν αλλάζει (☺), το MSKodu δεν μας δίνει κάποια σχετική ενέργεια αλλά μας δίνει ένα αισθητήρα με τον οποίο μπορούμε να προγραμματίζουμε ενέργειες που θα συμβούν σε διαφορετικές στιγμές του παιχνιδιού:

Α) Ο αισθητήρας ***Χρονόμετρο (Timer)*** μας επιτρέπει να προκαλούμε ενέργειες σε διαφορετικές στιγμές εξέλιξης του παιχνιδιού. Ουσιαστικά είναι σαν ένας χρονομετρητής που αν το συνδυάσουμε με το προσδιοριστικό ***Δευτερόλεπτα (Seconds)***, μπορούμε να καθορίσουμε με ακρίβεια τη χρονική στιγμή που θέλουμε να συμβεί μια ενέργεια. Για παράδειγμα, αν θέλουμε ο προστάτης μιας πύλης ενός κάστρου, να εκτοξεύει πυραύλους κάθε πέντε δευτερόλεπτα, θα πρέπει να χρησιμοποιήσουμε το προσδιοριστικό ***5 Δευτερόλεπτα (Seconds5)***:

Προσέξτε όμως, ότι ο αισθητήρας δεν ενεργοποιείται μόνο για μια φορά, αλλά <u>*ΚΑΘΕ*</u> 5 δευτερόλεπτα. Δηλαδή ο αισθητήρας δεν μετράει απόλυτο χρόνο (π.χ. 30 δευτερόλεπτα μετά την έναρξη του παιχνιδιού) αλλά σχετικό χρόνο (π.χ. κάθε 30 δευτερόλεπτα). Συνεπώς, ο φύλακας θα ρίχνει πυραύλους για όλη τη διάρκεια του παιχνιδιού. Πως όμως θα μπορούσατε να προκαλέσετε μια ενέργεια μια συγκεκριμένη χρονική στιγμή, χωρίς όμως αυτή να επαναλαμβάνεται; Όπως κάποιοι ίσως ήδη σκέφτηκαν, φυσικά με το προσδιοριστικό ενέργειας ***Για μια φορά (Once)***. Αν, δηλαδή, η προηγούμενη συμπεριφορά είχε την ακόλουθη μορφή,

τότε το αντικείμενο θα έριχνε έναν μόνο πύραυλο πέντε δευτερόλεπτα μετά την έναρξη του παιχνιδιού. Άρα με τη χρήση του αισθητήρα ***Χρονόμετρο (Timer)*** μπορείτε να προκαλείτε τόσο επαναλαμβανόμενες ενέργειες σε τακτά χρονικά διαστήματα, όσο και μεμονωμένες ενέργειες σε συγκεκριμένες χρονικές στιγμές του παιχνιδιού. Παρατηρήστε, επίσης, ότι η πίτα προσδιοριστικών του αισθητήρα έχει πολύ συγκεκριμένες επιλογές χρόνου:



Τι θα κάνουμε αν θέλουμε να εξαφανιστεί ένας εχθρός για να δώσει τη θέση του σε κάποιον ακόμη δυσκολότερο μετά από 17 δευτερόλεπτα; Δεν έχουμε, παρά να συνδυάσουμε προσδιοριστικά χρόνου και να αφήσουμε το MSKodu να τα αθροίσει. Π.χ.:

Χρειάζεται το προσδιοριστικό ***Για μια φορά (Once)*** σε αυτήν την περίπτωση; Δοκιμάστε μόνοι σας τη συγκεκριμένη συμπεριφορά με και χωρίς το προσδιοριστικό. Μπορείτε να εξηγήσετε τι συμβαίνει;

Δείτε το παράδειγμα *08_03.kodu*

Και επειδή όπως γνωρίζετε ο χρόνος είναι χρήμα δεν θα μπορούσε να υπάρχει καλύτερο παράδειγμα για την κατανόηση της έννοιας από μία πίστα με νομίσματα! Έστω ότι έχουμε έναν κόσμο με ένα *Kodu* και δέκα πολύχρωμα *Νομίσματα* **[08_03.kodu]**. Σκοπός του παιχνιδιού είναι να μαζέψει ο *Kodu* όλα τα νομίσματα, έχοντας όμως απέναντί του το χρόνο, δηλαδή θα πρέπει να τα αποκτήσει μέσα σε είκοσι δευτερόλεπτα. Αν πετύχει το στόχο του, τότε ανακηρύσσεται νικητής, ειδάλλως, χάνει.

Μπορείτε εύκολα πλέον να εξάγετε τις συμπεριφορές που θα πρέπει να έχει ο *Kodu*:

- **ΟΤΑΝ** αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος, **ΤΟΤΕ** κινείται στην αντίστοιχη κατεύθυνση πάνω στην πίστα.
- **ΟΤΑΝ** ακουμπήσει ένα *Νόμισμα*, **ΤΟΤΕ** το εξαφανίζει.
- **ΟΤΑΝ** δεν βλέπει κανένα *Νόμισμα* πάνω στην πίστα, **ΤΟΤΕ** τερματίζει το παιχνίδι με νίκη για τον παίκτη
- **ΟΤΑΝ** ο χρόνος φτάσει τα είκοσι δευτερόλεπτα, **ΤΟΤΕ** τερματίζει το παιχνίδι με ήττα για τον παίκτη.

Παρατηρήστε ότι έχουμε προσδιορίσει δυο συμπεριφορές που αφορούν το τέλος του παιχνιδιού: α) η πρώτη αφορά την περίπτωση που ο Kodu δεν βλέπει νομίσματα στην πίστα, οπότε ο παίκτης κερδίζει, και β) η δεύτερη αφορά την περίπτωση που έχουμε φτάσει στα 20 δευτερόλεπτα διάρκειας του παιχνιδιού, γεγονός που σημαίνει ότι ο χρήστης δεν έχει προλάβει να μαζέψει τα νομίσματα και, συνεπώς, χάνει.

Ας πάμε να προγραμματίσουμε τον πρωταγωνιστή μας:

α) Πρέπει να αλληλεπιδρά με το πληκτρολόγιο:

β) Όταν ο *Kodu* συναντά κάποιο *Νόμισμα*, πρέπει να το εξαφανίζει. Προσέχουμε να μην ξεχάσουμε να προσθέσουμε μετά την ενέργεια *Εξαφανίζω (Vanish)* το προσδιοριστικό ***Αυτό (It)***. Αλλιώς θα εξαφανιστεί ο Kodu!

γ) Όταν δεν βλέπει κάποιο *Νόμισμα* στην πίστα τότε να ανακηρύσσεται νικητής. Αν ο Kodu δεν βλέπει νόμισμα πάνω στην πίστα, αυτό σημαίνει ότι τα έχει εξαφανίσει όλα! Η άρνηση στην περιγραφή της συμπεριφοράς μας φέρνει στο μυαλό τη χρήση του προσδιοριστικού ***Όχι (Not)*** στον αισθητήρα ***Βλέπω(See)***:

Θα μπορούσε το παιχνίδι μας να τερματίσει και πάλι με νίκη αλλά με κάποιο διαφορετικό τρόπο; Σκεφτείτε λίγο... Τι μάθαμε στην πρώτη ενότητα αυτού του κεφαλαίου; Το σκορ!! Κάνοντας χρήση των όσων μελετήσαμε παραπάνω, ποια θα ήταν μία τυπική λύση; Μα να κρατάμε ένα σκορ για κάθε κέρμα που εξαφανίζεται και όταν το σκορ αυτό φτάσει τους δέκα πόντους, τότε το παιχνίδι να τερματίζει! Συνειδητοποιήσατε κάτι; Πως ένας στόχος μπορεί να υλοποιηθεί με παραπάνω από έναν τρόπους και να επιφέρει το ίδιο αποτέλεσμα στον κόσμο μας! Επομένως, καμία σκέψη δεν είναι λάθος και καμία λύση δεν είναι μοναδική! Στον προγραμματισμό τις περισσότερες φορές υπάρχουν πολλές λύσεις για το ίδιο πρόβλημα.

δ) Όταν ο χρόνος φτάσει τα είκοσι δευτερόλεπτα, το παιχνίδι θα πρέπει να τελειώνει. Άρα, θα χρησιμοποιήσουμε τον αισθητήρα ***Χρονόμετρο (Timer)*** με προσδιοριστικό τα ***20 Δευτερόλεπτα (20 Seconds)*** και ως ενέργεια την ***Ήττα (End)***:

Τι θα γίνει όμως αν ο χρόνος φτάσει τα 40 δευτερόλεπτα; Θα ξαναεκτελεστεί η ενέργειά μας; Στο συγκεκριμένο παράδειγμα κάτι τέτοιο δε μπορεί να συμβεί, γιατί πολύ απλά, όταν περάσουν τα 20 δευτερόλεπτα, το παιχνίδι θα τελειώσει. Ωστόσο, αν στη θέση της ενέργειας ***Ήττα (End)*** υπήρχε κάποια άλλη ενέργεια, η οποία δεν τερμάτιζε το παιχνίδι, τότε η συμπεριφορά θα ενεργοποιούνταν κάθε 20 δευτερόλεπτα.

Και πως θα μπορούσαμε να βλέπουμε το χρόνο να μετράει αντίστροφα για να κορυφώνεται η αγωνία; Σε περίπτωση αυτή μπορούμε να κάνουμε ένα μικρό τρικ:

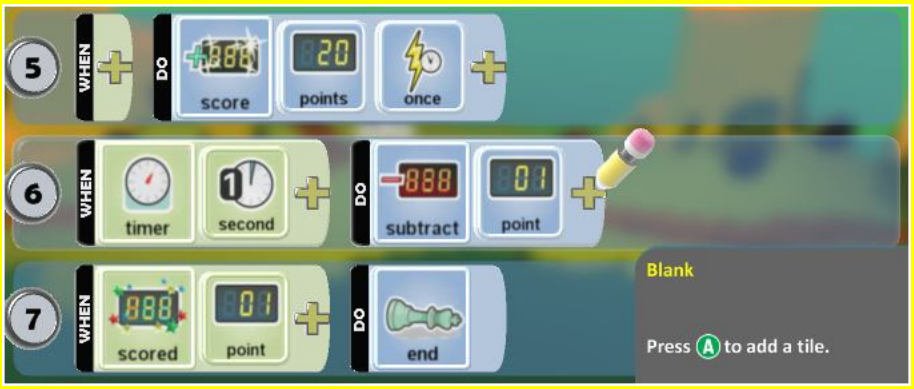
α) Θα εμφανίσουμε μια μεταβλητή σκορ,

β) θα της δώσουμε ως αρχική τιμή τον αριθμό των δευτερολέπτων που θέλουμε να έχει το παιχνίδι μας και

γ) στη συνέχεια κάθε ένα δευτερόλεπτο θα αφαιρούμε μια μονάδα από το σκορ. Ας το δούμε πιο αναλυτικά:

Αρχικά, θα πρέπει να αρχικοποιήσουμε το χρονόμετρό μας, δηλαδή να του δώσουμε μια τιμή από την οποία θα ξεκινά η αντίστροφη μέτρηση, έστω τα 20 δευτερόλεπτα. Στο συγκεκριμένο παράδειγμα, αυτό μπορούμε να το υλοποιήσουμε με τη βοήθεια της ενέργειας *Αύξηση Σκορ (Score)* ακολουθούμενη από το προσδιοριστικό *Βαθμοί 20 (Points 20).* Προσοχή! Η ενέργεια αυτή θα πρέπει να εκτελεσθεί μια φορά, γι' αυτό και στο τέλος θα χρησιμοποιήσουμε το προσδιοριστικό *Μία Φορά (Once)*.

Στη συνέχεια, κάθε ένα δευτερόλεπτο που περνάει, θα πρέπει το σκορ να μειώνεται κατά ένα. Άρα χρειαζόμαστε τον αισθητήρα Χρονόμετρο με προσδιοριστικό το 1 δευτερόλεπτοκαι την ενέργεια *Μείωση (Subtract)* συνοδευόμενη από το προσδιοριστικό *Βαθμός01 (Points01).* Τέλος, όταν το σκορ μας να γίνει ένα, το παιχνίδι θα πρέπει να τερματίζει. Μελετήστε και δοκιμάστε τις παρακάτω εντολές:



Αν το παραπάνω παράδειγμα σας φάνηκε αρκετά εύκολο, ας δούμε μία διαφορετική εκδοχή του, όπου η δυσκολία του παιχνιδιού αυξάνεται με την πάροδο του χρόνου. Καθώς ο *Kodu* προσπαθεί να μαζέψει τα *Νομίσματα*, εμφανίζεται κάθε τρία δευτερόλεπτα μία *Πέτρα*. Αν πέσει πάνω σε μία από αυτές τις *Πέτρες* που δημιουργούνται, τότε το παιχνίδι τερματίζεται.

Οι επιπλέον εντολές που θα χρειαστούμε είναι :

α) Ανά τρία δευτερόλεπτα θα εμφανίζεται μια *Πέτρα*. Θυμηθείτε τη λειτουργία της ενέργειας ***Εκτινάζω (Launch)***:

β) Όταν ο *Kodu* συγκρούεται με μία *Πέτρα*, τότε το παιχνίδι τερματίζει.

Ας πάρετε όμως τώρα εσείς τη σκυτάλη. Για να δούμε πόσο καλά μάθατε να αξιοποιείτε το χρόνο! Θυμηθείτε το παράδειγμα στο κεφάλαιο 6 με το ποδοσφαιράκι. Δεν ήταν αρκετά πρωτότυπο; Τι θα λέγατε αν προσθέτατε επιπλέον χρόνο και σκορ, ούτως ώστε να μοιάζει περισσότερο με έναν πραγματικό αγώνα; Μία εκδοχή είναι να αναπαραστήσετε το τέρμα του αντιπάλου με μία σειρά από κόκκινα *Μήλα* μπροστά από την κίτρινη γραμμή του τέρματος. Αν η *Μπάλα* που κλοτσά ο *Kodu* ακουμπήσει έστω και ένα από τα *Μήλα* αυτά, χρεώνεται με γκολ ο αντίπαλος. Τα γκολ μπορείτε να τα αναπαραστήσετε με το κόκκινο σκορ ενώ το σκορ θα καταγράφεται μόνο στα πρώτα 30 δευτερόλεπτα, γιατί το παιχνίδι μετά θα πρέπει να λήγει.

Μια ενέργεια που μπορεί πολύ εύκολα να συνδυαστεί με το χρόνο είναι η *Επανεκκίνηση Παιχνιδιού (Reset)*. Τι κάνει; Ξαναξεκινάει το παιχνίδι από την αρχή. Για να τη χρησιμοποιήσετε, θα πρέπει να την αναζητήσετε στην κατηγορία *Παιχνίδι (Game)* της αρχικής πίτας ενεργειών του MSKodu. Για παράδειγμα, αν επιθυμούμε κάποιο παιχνίδι μας να αρχίζει ξανά από την αρχή όταν ο χρόνος φτάσει τα 60 δευτερόλεπτα, τότε αρκεί να γράψουμε την εντολή:

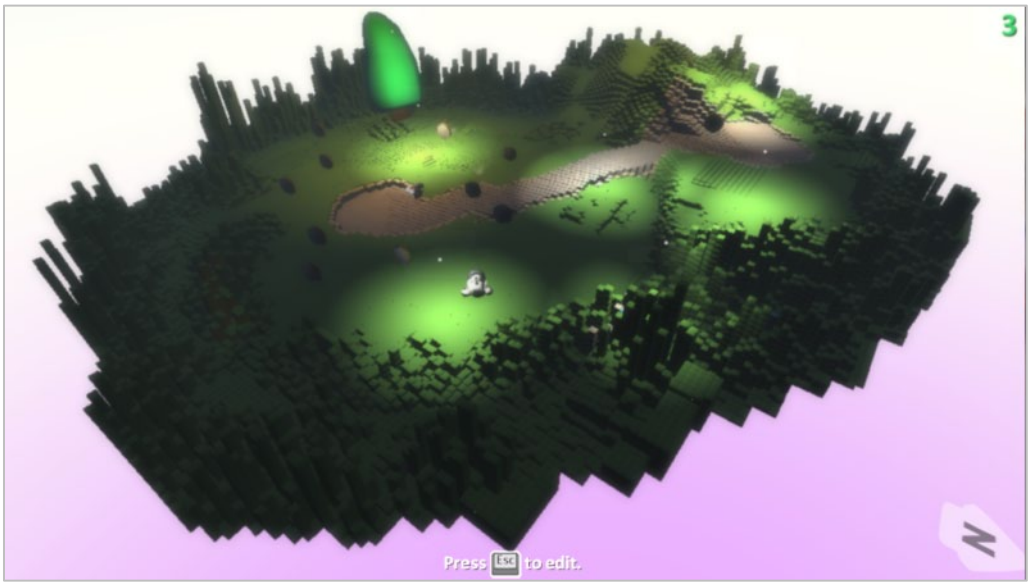
8.3.4 Ζωές

Συνδυάζοντας όσα μάθατε παραπάνω, μπορείτε πλέον να δημιουργήσετε και άλλα στοιχεία που θα πλαισιώσουν τους κόσμους σας. Για παράδειγμα, εάν θα θέλατε να εισάγατε ζωές στα παιχνίδια σας, πώς θα τα καταφέρνατε; Μα οι ζωές δεν είναι ουσιαστικά μία μεταβλητή; Δεν θα μπορούσαμε να χρησιμοποιήσουμε ένα σκορ για να αποθηκεύουμε την τιμή των ζωών και με τη χρήση των αντίστοιχων ενεργειών να αυξάνουμε ή να μειώνουμε την τιμή τους; Η μόνη διαφορά που υπάρχει είναι πως πλέον ο αισθητήρας ***Σκορ (Scored)*** δεν αντιπροσωπεύει σύνολο βαθμών, αλλά ζωών, τις οποίες μας επιτρέπει να ανακαλέσουμε, να συγκρίνουμε και να χρησιμοποιήσουμε κατά τη διάρκεια του παιχνιδιού.

Για να πάρετε μια ιδέα πως θα γίνει αυτό, μελετήστε το παρακάτω παράδειγμα.

Δείτε το παράδειγμα *08_04.kodu*

Kodu: Ο κανονιέρης! **[08_04.kodu]** Έστω ότι έχουμε έναν *Μηχανάκια (Cycle),* αρκετά *Νομίσματα* και ένα *Κανόνι*. Ο σκοπός του παίκτη είναι να συγκεντρώσει όλα τα *Νομίσματα* της πίστας ελέγχοντας το *Μηχανάκια (Cycle)* προκειμένου να αναδειχτεί νικητής. Ωστόσο, ανά τρία δευτερόλεπτα το *Κανόνι* εκτινάσσει πυραύλους, οι οποίοι αν το χτυπήσουν, του αφαιρούν μία ζωή. Αρχικά ο παίκτης έχει 3 ζωές, δηλαδή τρεις ευκαιρίες. Αν χάσει και τις τρεις ζωές του, τότε το παιχνίδι θα πρέπει να τελειώνει.



Ποιες λοιπόν είναι οι συμπεριφορές στο παιχνίδι μας; Μπορέσατε να συγκρατήσετε ορισμένες από αυτές; Χμμ για να δούμε τι καταφέρατε!

Οι ΣΥΜΠΕΡΙΦΟΡΕΣ που θα πρέπει να έχει ο *Μηχανάκιας (Cycle)* είναι:

- Ξεκινώντας το παιχνίδι ο Μηχανάκιας έχει τρεις ζωές.
- **ΟΤΑΝ** αισθανθεί ότι στο πληκτρολόγιο πατιέται ένα βέλος, **ΤΟΤΕ** κινείται στην αντίστοιχη κατεύθυνση.
- **ΟΤΑΝ** ακουμπήσει ένα *Νόμισμα*, **ΤΟΤΕ** το εξαφανίζει.
- **ΟΤΑΝ** ακουμπήσει ένα πύραυλο, **ΤΟΤΕ** μειώνεται κατά ένα ο αριθμός των ζωών του.
- **ΟΤΑΝ** δεν βλέπει κανένα *Νόμισμα* πάνω στην πίστα, **ΤΟΤΕ** τερματίζει το παιχνίδι με νίκη για τον παίκτη.
- **ΟΤΑΝ** ο αριθμός των ζωών του φτάσει το 0, **ΤΟΤΕ** τερματίζει το παιχνίδι με ήττα για τον παίκτη.

Η ΣΥΜΠΕΡΙΦΟΡΑ που έχει το *Κανόνι (Cannon)* είναι:

- **ΟΤΑΝ** περνάνε τρία δευτερόλεπτα, **ΤΟΤΕ** εκτόξευσε πυραύλους.

Ας πάμε λοιπόν να προγραμματίσουμε τις βασικές συμπεριφορές του *Μηχανάκια (Cycle)* που αφορούν τη διαχείριση των ζωών του:

α) Όταν ξεκινήσει το παιχνίδι, θα πρέπει να έχει τρεις ζωές.

Η αντίστοιχη εντολή που μπορούμε να δώσουμε στο *Μηχανάκια (Cycle)* είναι:

ΟΤΑΝ[] **ΤΟΤΕ** [*Αύξησε*][το *Πράσινο Σκορ*][*3 Βαθμούς*][*Μία Φορά*].

Παρατηρήστε πως με αυτή την εντολή δίνουμε στο *Μηχανάκια (Cycle)* τις ζωές που του αναλογούν στην αρχή του παιχνιδιού, καθώς δεν έχουμε χρησιμοποιήσει κάποιον αισθητήρα. Όπως και σε άλλα παραδείγματα, έτσι και τώρα απαραίτητη είναι η χρήση του προσδιοριστικού ***Μία φορά (Once)***, καθώς επιθυμούμε η ανάθεση αυτή των ζωών στο *Μηχανάκια (Cycle)* να πραγματοποιηθεί μία φορά στην έναρξη του παιχνιδιού. Η επιλογή του «πράσινου» σκορ είναι τυχαία.

β) Όταν ο Μηχανάκιας ακουμπήσει έναν πύραυλο, ο αριθμός των ζωών του θα πρέπει να μειώνεται κατά μία ζωή. Άρα, εδώ θα πρέπει να χρησιμοποιήσουμε τον αισθητήρα ***Πέφτω πάνω (Bump)*** και ως ενέργεια τη ***Μείωση (Subtract)***. Κάνουμε χρήση αυτής της ενέργειας γιατί μην ξεχνάτε: οι ζωές δεν είναι τίποτα άλλο, παρά ένα σκορ! Η αντίστοιχη εντολή που μπορούμε να δώσουμε στο *Μηχανάκια (Cycle)* είναι:

γ) Όταν χάσει και την τελευταία του ζωή, τότε πρέπει να τερματίζει το παιχνίδι με ήττα για τον παίκτη. Άρα, θα χρησιμοποιήσουμε τον αισθητήρα ***Σκορ (Scored)*** με προσδιοριστικό το ***Βαθμοί00 (Points00)*** και ως ενέργεια της εντολής θα υπάρχει η ***Ήττα (End)***. Η αντίστοιχη συμπεριφορά που μπορούμε να δώσουμε στο *Μηχανάκια (Cycle)* είναι:

Προσοχή: απαραίτητη προϋπόθεση για να λειτουργήσει σωστά η συγκεκριμένη συμπεριφορά είναι να πραγματοποιηθεί μετά τη συμπεριφορά που δίνει αρχική τιμή στη μεταβλητή μας, δηλαδή στις ζωές. Σε αντίθετη περίπτωση, το παιχνίδι θα τερματιστεί ακαριαία και αυτό γιατί οι αρχικές τιμές των μεταβλητών είναι μηδέν. Πως μπορούμε να ξεπεράσουμε κάτι τέτοιο; Υπάρχει η λύση μέσα στο παράδειγμα 08_04.Kodu

Πώς θα σας φαινόταν αν επεκτείνατε το ήδη υπάρχον παιχνίδι, προσθέτοντας *Αστέρια* που θα αναπλήρωναν τις χαμένες ζωές σας;

Περίληψη

Στο κεφάλαιο αυτό συζητήσαμε για τις μεταβλητές και τον τρόπο με τον οποίο χρησιμοποιούν τη μνήμη του υπολογιστή. Στο MSKodu, οι μεταβλητές είναι συγκεκριμένες, κρατούν ακέραιες τιμές και έχουν διαφορετικές μορφές: Σκορ, Ενέργεια, Χρονόμετρο, κτλ. Παρουσιάσαμε τους αισθητήρες ***Σκορ (Scored)***, ***Χρονόμετρο (Timer)***, ***Ενέργεια (Health)*** που μας επιτρέπουν να ανακαλούμε τις τιμές των μεταβλητών ενώ επιπλέον χρησιμοποιήσαμε ενέργειες που μας επιτρέπουν να αλλάζουμε τις τιμές τους όπως οι ***Αύξηση (Score), Μείωση (Subtract), Προκαλώ βλάβη (Damage), Επουλώνω (Heal)***. Η σημαντικότερη όμως παρατήρηση είναι ότι μπορούμε να χρησιμοποιήσουμε όλες αυτές τις μεταβλητές με το δικό μας τρόπο για να καταγράφουμε οτιδήποτε μας ενδιαφέρει π.χ. πόσα νομίσματα έχει μαζέψει το αντικείμενό μας, ή πόσα υποβρύχια έχει συγκρουστεί. Συνεπώς, τα παιχνίδια μας έχουν τη δυνατότητα αφενός μεν να αριθμούν και να καταγράφουν συμβάντα και αφετέρου να αξιοποιούν τις αντίστοιχες τιμές για να

γίνονται πιο ενδιαφέροντα π.χ. για να αλλάζει κάποια στιγμή η δυσκολία του παιχνιδιού, για να κερδίζει ζωές ο παίκτης, για να αναθέτουμε στους χρήστες πιο σύνθετες αποστολές. Μας μένει μόνο ένα ακόμη βήμα για να καταφέρουμε να δημιουργούμε παιχνίδια όπως ακριβώς τα φανταζόμαστε και αυτό θα το εξετάσουμε στο επόμενο κεφάλαιο.

Ερωτήσεις

1. Σχεδιάστε το στιγμιότυπο που θα έχει η μνήμη του υπολογιστή όταν ο *Kodu*, στο παράδειγμα ο *Kodu* ο κανονιέρης, έχει χτυπηθεί από 2 πυραύλους.

2. Έστω ότι σε ένα από τα παιχνίδια σας, θέλετε ο *Kodu* να πυροβολεί και να καταστρέφει το *Μηχανάκια (Cycle).* Ποιο προσδιοριστικό θα πρέπει να συμπεριλάβετε στο πρόγραμμά σας, ώστε να εξασφαλίσετε ότι αυτός που θα εξαφανιστεί είναι ο *Μηχανάκιας (Cycle)* και όχι κάποιος άλλος χαρακτήρας από τους υπόλοιπους που συμμετέχουν;

3. Αν θέλετε το σκορ σας να αποθηκεύεται στη μεταβλητή σκορ Κ του MSKodu, ποιο προσδιοριστικό θα πρέπει να χρησιμοποιήσετε;

4. Αν θέλετε να τελειώνει ένα από τα παιχνίδια σας όταν ο χρόνος φτάσει τα δύο λεπτά, ποια θα είναι η συμπεριφορά που θα πρέπει να δημιουργήσετε;

Δραστηριότητες

1. Ψάχνοντας το μαύρο χρυσό! Δημιουργείστε μια πίστα με πρωταγωνιστή τον Kodu ο οποίος προσπαθεί να αποκτήσει το ένα και μοναδικό μαύρο νόμισμα που υπάρχει στον κόσμο του. Για να το καταφέρει όμως αυτό πρέπει να περάσει μέσα από ένα δύσβατο και απρόσιτο μονοπάτι γεμάτο βράχους και κρυμμένους εχθρούς. Αν πέσει πάνω σε ένα βράχο τότε πρέπει να χάνει μια ζωή, ενώ αν χτυπηθεί από τις σφαίρες των εχθρών τότε πρέπει να μειώνεται η ενέργεια του κατά δύο μονάδες. Στην πίστα θα υπάρχουν επιπλέον κόκκινα μήλα και αστέρια. Για κάθε δύο κόκκινα μήλα που θα τρώει θα πρέπει να αυξάνεται η ενέργεια του κατά μια μία μονάδα, ενώ για κάθε αστέρι με το οποίο θα έρχεται σε επαφή θα πρέπει να κερδίζει μία ζωή. Το παιχνίδι θα πρέπει να τελειώνει με ήττα για τον παίχτη αν ο χρόνος περάσει τα τριάντα δευτερόλεπτα και ο Kodu δεν έχει αποκτήσει το μαύρο νόμισμα ή αν ο Kodu χάσει όλες του τις ζωές. Αν, από την άλλη πλευρά, ο Kodu φτάσει το μαύρο νόμισμα εντός του χρόνου, τότε το παιχνίδι θα πρέπει να τερματίζει με νίκη. Πρέπει να τοποθετήσετε βράχους, μαύρους Kodu που θα αποτελούν τους εχθρούς του πρωταγωνιστή και θα πυροβολούν προς το μέρος του συνεχώς, κόκκινα μήλα και αστέρια στην πίστα σας.

2. Kodu ο ορειβάτης! Δημιουργείστε μια πίστα με πρωταγωνιστή τον Kodu ο οποίος πρέπει να σκαρφαλώσει σε ένα βουνό για να βρει το σπάνιο δέντρο που βρίσκεται στην κορυφή του. Το βουνό όμως αυτό είναι αρκετά ψηλό και απότομο γεμάτο γκρεμούς και τεράστιους καταρράκτες, γεγονός που έχει αρνητικές επιπτώσεις στην ενέργεια του Kodu. Πιο συγκεκριμένα, κάθε τρία δευτερόλεπτα που περνάνε, η ενέργεια του θα μειώνεται κατά δύο μονάδες, ενώ αν έρθει σε επαφή με κάποιον από τους καταρράκτες τότε θα πρέπει να χάνει μια ζωή. Αν μηδενιστεί η ενέργεια του ή ο αριθμός των ζωών του ή περάσει ένα λεπτό και δεν έχει φτάσει ακόμα στο δέντρο, τότε το παιχνίδι θα πρέπει να τερματίζει με ήττα για τον παίχτη. Αν καταφέρει να φτάσει στο στόχο του εντός του χρόνου, τότε το παιχνίδι πρέπει να τερματίζει με νίκη για τον παίχτη. Προσέξτε η πίστα σας να είναι απότομη με πολλά σκαμπανεβάσματα και αρκετές περιοχές γεμισμένες με νερό.

3. Kodu: Ο Πολιορκητής! Υποθέστε ότι ο παίκτης θα ελέγχει τον *Kodu* τον πολιορκητή! Σκοπός του είναι να φτάσει στο πολυπόθητο *Κάστρο* και να το κατακτήσει. Ο δρόμος του όμως δεν θα είναι εύκολος καθώς θα έχει να αντιμετωπίσει πολλούς εχθρούς. Πιο συγκεκριμένα, οι εχθροί του θα είναι τα *Κανόνια* που θα βρίσκονται μπροστά από το *Κάστρο* και θα το προστατεύουν, πετώντας ανά πέντε δευτερόλεπτα βόμβες προς στον *Kodu* που αν τον πετύχουν θα του αφαιρούν ζωές. Επίσης, εμπόδιο στην πορεία του θα αποτελούν και δύο ακόμα *Μηχανάκια (Cycles)* που θα κινούνται ασταμάτητα σε πολύπλοκα μονοπάτια και τα οποία, αν συγκρουστούν μαζί του, θα επιφέρουν το ίδιο αποτέλεσμα. Οι μοναδικοί σύμμαχοι του *Kodu* θα είναι τα διάφορα *Νομίσματα* που θα βρίσκονται διάσπαρτα στην πίστα. Κάθε φορά που θα καταφέρνει να συγκεντρώνει δέκα από αυτά, θα έχει μπόνους μία επιπλέον ζωή. Το παιχνίδι θα τελειώνει είτε αν ο *Kodu* χάσει όλες

του τις ζωές είτε αν καταφέρει να προσπεράσει όλους του τους εχθρούς και να φτάσει στο *Κάστρο*.

4. *Kodu*: Ο ναυαγός! Έστω ότι έχετε έναν κόσμο με πρωταγωνιστή τον *Kodu*, ο οποίος έχει ναυαγήσει σε ένα νησί και προσπαθεί να φτάσει σε ένα μεγάλο *Καράβι* που εμφανίστηκε στην άλλη πλευρά του νησιού. Όμως, καθώς κατευθύνεται προς τα εκεί, εμφανίζονται από το πουθενά *Ψάρια-Ζόμπι*, τα οποία προσπαθούν να του δυσκολέψουν το δρόμο. Προκειμένου ο *Kodu* να φτάσει στο *Καράβι*, θα πρέπει να τα αποφύγει, αλλά όταν αυτό δεν είναι εφικτό, πρέπει να τους πετάει πυραύλους! Για κάθε ζόμπι που πετυχαίνει, θα κερδίζει δέκα πόντους, ενώ όταν συγκρούεται με τα ζόμπι, θα μειώνεται η ενέργειά του κατά δύο μονάδες. Δυστυχώς οι εχθροί του δεν είναι μόνο αυτοί. Υπάρχει και άλλος ένας και αυτός είναι ο χρόνος! Θα πρέπει δηλαδή να φτάσει στο *Καράβι* μέσα σε διάστημα ενός λεπτού. Αν δεν τα καταφέρει, τότε το παιχνίδι τερματίζει. Όμως, το παιχνίδι δεν τελειώνει μόνο σε αυτή την περίπτωση! Υπάρχει ακόμη μία περίπτωση, όταν ο *Kodu* να χάσει όλη του την ενέργεια. **[08_05.kodu]**

Κεφάλαιο 9ο: Αλλαγή σελίδας και Δημιουργία Κλώνων

Μέχρι αυτό το κεφάλαιο, τα παιχνίδια μας είχαν δυο σταθερά στοιχεία: α) τη συμπεριφορά των αντικειμένων - την προσδιορίζαμε αρχικά και παρέμενε ίδια σε όλη τη διάρκεια του εκάστοτε παιχνιδιού, β) τον αριθμό των αντικειμένων - το παιχνίδι μας είχε όσα αντικείμενα εισάγαμε εμείς κατά τη δημιουργία του, και έτσι, ο αριθμός των εχθρών και των φίλων παρέμενε και αυτός σταθερός. Μήπως όμως τα αγαπημένα μας παιχνίδια δεν είναι ακριβώς έτσι; Σε αυτό το κεφάλαιο θα εξοικειωθείτε με δύο καινούργιες έννοιες, την έννοια της *αλλαγής σελίδας (pages)* και τη *δημιουργία κλώνων (creatables)* ώστε πλέον τα παιχνίδια μας να ξεπεράσουν αυτά τα δυο προβλήματα και να γίνουν ακόμη πιο ελκυστικά και ρεαλιστικά!

9.1 Αλλαγή συμπεριφοράς (Switch Page)

Ως τώρα, η συμπεριφορά που δίνατε στα αντικείμενα του κόσμου σας ήταν σταθερή, δηλαδή, τα αντικείμενα είχαν την ίδια συμπεριφορά σε όλη τη διάρκεια του παιχνιδιού. Πως θα μπορούσαμε να αλλάξουμε τη συμπεριφορά κάποιου αντικειμένου κατά τη διάρκεια του παιχνιδιού π.χ. ο χρήστης να χάνει τον έλεγχο του Kodu για λίγα δευτερόλεπτα όταν αυτός φάει ένα δηλητηριασμένο μήλο; Ένα παράδειγμα, για να καταλάβετε καλύτερα την αλλαγή της συμπεριφοράς είναι να θυμηθείτε το παιχνίδι Super Mario που αναφέρθηκε και στο πρώτο κεφάλαιο. Όταν ο super mario τρώει ένα λουλούδι, αποκτά τη δυνατότητα να πετάει σφαίρες. Πώς θα μπορούσαμε να προσθέσουμε αυτή τη συμπεριφορά σε ένα αντίστοιχο αντικείμενο του δικού μας προγραμματιστικού περιβάλλοντος; Χμ… δεν είναι εύκολο και αυτό γιατί εμείς ορίζουμε συμπεριφορές χρησιμοποιώντας αισθητήρες (*Βλέπω (see)*, *Ακούω (hear)*, *Πέφτω πάνω (bump)* κτλ) που ενεργοποιούνται σε όλη τη διάρκεια του παιχνιδιού μας. Δεν μπορούμε να δώσουμε στις συμπεριφορές μας κάποιο χρόνο ζωής και μετά να τις αλλάξουμε… Ή μήπως μπορούμε;

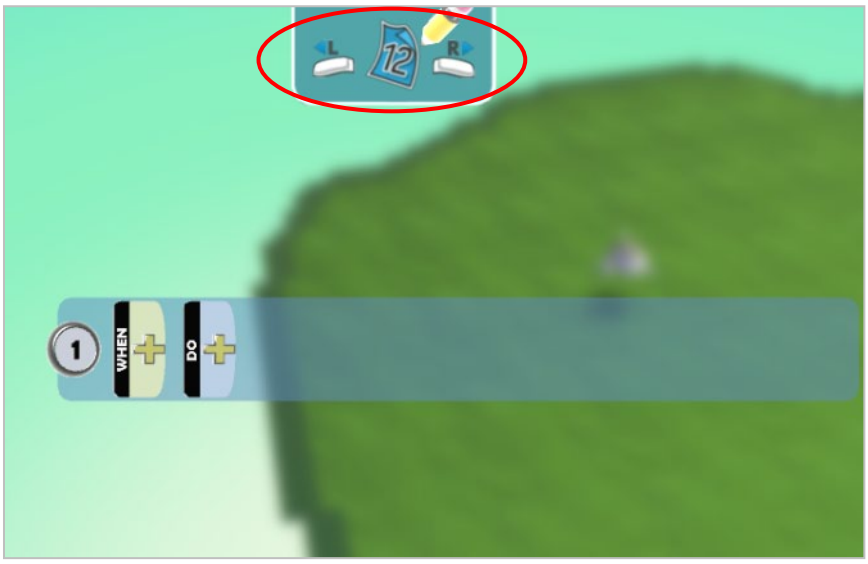
Πράγματι, το MSKodu έχει μηχανισμό που επιτρέπει την αλλαγή συμπεριφοράς των αντικειμένων σε διαφορετικές στιγμές του προγράμματος. Σύμφωνα με το μηχανισμό αυτό:

- **Κάθε αντικείμενο μπορεί να έχει διαφορετικά σύνολα συμπεριφορών δηλαδή ξεχωριστά σύνολα εντολών που μπορούν να ενεργοποιηθούν προγραμματιστικά**. Κάθε σύνολο συμπεριφορών δημιουργείται μέσα σε μια ***«σελίδα» (page) συμπεριφοράς***. Το MSKodu επιτρέπει σε ένα αντικείμενο να έχει μέχρι δώδεκα διαφορετικά σύνολα συμπεριφορών δηλαδή μέχρι δώδεκα σελίδες συμπεριφορών.
- **Κάθε στιγμή στο πρόγραμμα εκτελείται μια και μόνο σελίδα συμπεριφορών για κάθε αντικείμενο**, δηλαδή κάθε αντικείμενο έχει ενεργό ένα από τα δώδεκα δυνατά σύνολα συμπεριφορών.
- **Ο προγραμματιστής μπορεί να ορίζει ποιο σύνολο συμπεριφορών θέλει να ενεργοποιείται κάθε στιγμή με τη χρήση της ενέργειας *«Αλλαγή σελίδας» (Switch Page)* και προσδιοριστικό την αντίστοιχη σελίδα εντολών που επιθυμεί**.

Ας δούμε αυτά τα τρία στοιχεία λίγο πιο αναλυτικά. Όταν προγραμματίζετε κάποιο αντικείμενο έχετε δει στο πάνω μέρος του παραθύρου του MSKodu ένα εικονίδιο που περιέχει έναν αριθμό και δύο βελάκια;

Αυτό το παράθυρο μας υποδηλώνει ότι βρισκόμαστε στη σελίδα συμπεριφορών με τον αριθμό ένα. Δηλαδή σε όλα τα προηγούμενα κεφάλαια, προγραμματίζαμε τα αντικείμενά μας στην πρώτη σελίδα συμπεριφορών τους. Η «σελίδα 1» είναι αυτή που εκτελείται πάντα όταν ξεκινά ένα παιχνίδι. Πατήστε τα βελάκια για να μετακινηθείτε στην επόμενη ή στην προηγούμενη σελίδα. Αν πατήσετε το δεξί βέλος θα δείτε την επόμενη εικόνα, τη σελίδα με τον αριθμό 2:

ενώ αν πατήστε το αριστερό θα μετακινηθείτε στην τελευταία σελίδα συμπεριφορών του αντικειμένου, που είναι η σελίδα 12:



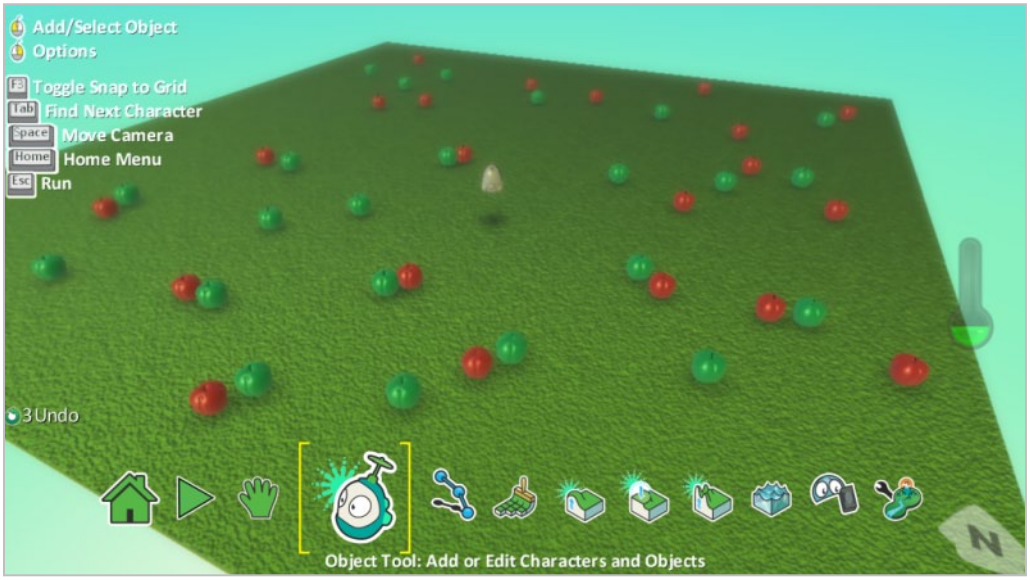
Παρατηρήστε, ότι όταν αλλάζετε σελίδα, ο επεξεργαστής εντολών είναι κενός, ενώ όταν επιστρέφετε στη σελίδα ένα, παρουσιάζει τις εντολές που είχατε εισάγει. Λογικό; Μα φυσικά, αφού δεν έχουμε ορίσει άλλα σύνολα συμπεριφορών μέχρι αυτό το κεφάλαιο. Μέχρι τώρα δημιουργούσαμε συμπεριφορές μόνο στην πρώτη σελίδα όλων των αντικειμένων.

Ας δούμε όμως μέσα από παραδείγματα την ***Αλλαγή σελίδας (Switch page)***. Έστω ότι ο Kodu βρίσκεται σε ένα δάσος με δηλητηριώδη πράσινα Μήλα και αθώα νόστιμα κόκκινα Μήλα. Ο Kodu θα πρέπει να μετακινείται με τα βελάκια του πληκτρολογίου και όταν πέσει πάνω σε ένα κόκκινο μήλο θα πρέπει να το μαζεύει, ενώ αν πέσει πάνω σε ένα δηλητηριασμένο πράσινο μήλο, ο παίχτης θα πρέπει να χάνει τον έλεγχο για λίγο και ο ήρωάς μας θα *Περιφέρεται (wander)* τυχαία στο δάσος. Το παιχνίδι θα είναι δύσκολο, καθώς τα καλά και τα δηλητηριασμένα μήλα θα είναι τόσο κοντά, που θα πρέπει ο παίκτης να είναι ιδιαίτερα προσεκτικός ώστε ο Kodu να πέσει πάνω σε ένα κόκκινο μήλο, αποφεύγοντας τα πράσινα. Στόχος του Kodu είναι να μαζέψει όλα τα κόκκινα μήλα και να αποφύγει τα δηλητηριασμένα μέσα σε ένα λεπτό.

Πως θα μπορούσαμε να υλοποιήσουμε το συγκεκριμένο παιχνίδι; Πως θα μπορούσαμε να προγραμματίσουμε τον Kodu ώστε να μην «αισθάνεται» το πληκτρολόγιο για κάποιο χρονικό διάστημα; Μήπως τώρα βλέπετε την ανάγκη της αλλαγής σελίδας; Ας λύσουμε το πρόβλημα που περιγράφηκε.

Δείτε το παράδειγμα *07_03.kodu*

[09_01.Kodu] *Δημιουργία κόσμου:* Αρχικά βλέπουμε ότι το παιχνίδι μας αποτελείται από τρία αντικείμενα: ωφέλιμα κόκκινα μήλα, τα δηλητηριώδη πράσινα μήλα και τον Kodu. Σε αυτή τη φάση δε μας απασχολεί ο κόσμος του παιχνιδιού, απλά θα τοποθετήσουμε τα αντικείμενα σε μια επίπεδη πίστα. Ξεκινάμε λοιπόν και γεμίζουμε τον κόσμο με μήλα. Θα πρέπει ωστόσο να τηρήσουμε τη μικρή λεπτομέρεια που αναφέραμε παραπάνω, τα πράσινα με τα κόκκινα μήλα να είναι σε σχετικά κοντινή απόσταση μεταξύ τους ώστε να δυσκολευτεί ο παίχτης να πάρει το κόκκινα μήλα χωρίς να ακουμπήσει τα πράσινα. Τέλος τοποθετούμε στον κόσμο και τον Kodu. Οπότε ο κόσμος του παιχνιδιού θα πρέπει να μοιάζει κάπως έτσι:



Στα μήλα δε θα δώσουμε κάποια συμπεριφορά. Έτσι, το επόμενο βήμα είναι να προγραμματίσουμε τον Kodu.

Προγραμματισμός Kodu: Ο Kodu έχει αρχικά τη συμπεριφορά που περιγράψαμε προηγουμένως. Θα πρέπει να μετακινείται με τα βελάκια του πληκτρολογίου και κάθε φορά που θα πέφτει σε ένα κόκκινο μήλο θα το εξαφανίζει (μη ξεχνάτε ότι χρειαζόμαστε το προσδιοριστικό ενέργειας ***It (αυτό)***). Το παιχνίδι θα τελειώνει όταν δεν υπάρχουν άλλα κόκκινα μήλα, άρα όταν ο Kodu δεν βλέπει άλλα μήλα πάνω στην πίστα. Συνεπώς, οι πρώτες συμπεριφορές του Kodu θα είναι οι εξής:

Υπάρχει όμως και η περίπτωση ο Kodu να πέσει πάνω σε πράσινο μήλο. Τότε, θέλουμε να αλλάζει η συμπεριφορά του δηλαδή θέλουμε να αλλάξουμε σελίδα συμπεριφοράς στο αντικείμενό μας! Θα χρειαστεί να χρησιμοποιήσουμε την ενέργεια ***Άλλαξε σελίδα (Switch Page)***.

ΟΤΑΝ[πέσω πάνω σε][πράσινο μήλο] **ΤΟΤΕ** [άλλαξε σελίδα][σελίδα δύο]

Ποια σελίδα εντολών θέλουμε όμως να ενεργοποιήσουμε; Η ενέργεια ***Άλλαξε Σελίδα (Switch Page)*** συνοδεύεται πάντα από το προσδιοριστικό της σελίδας που θέλουμε να ενεργοποιήσουμε.

Για να προσθέσουμε την αντίστοιχη ενέργεια, επιλέγουμε από την πίτα ενεργειών την ενέργεια ***Άλλαξε Σελίδα (Switch Page)***:

και στη συνέχεια επιλέγουμε τη σελίδα στην όποια θα προσθέσουμε τη νέα συμπεριφορά του Kodu. Στη συγκεκριμένη περίπτωση θα περιγράψουμε τη νέα συμπεριφορά του Kodu στη σελίδα δύο:

Έτσι στην τέταρτη γραμμή της πρώτης σελίδας, θα προσθέσουμε την εξής συμπεριφορά:

Μόλις δηλώσαμε στο πρόγραμμα μας ότι, όταν ο Kodu *Πέσει πάνω (bump)* σε πράσινο μήλο, η συμπεριφορά του θα ορίζεται από τη δεύτερη σελίδα συμπεριφορών.

Πλέον, πρέπει να πάμε στη δεύτερη σελίδα για να γράψουμε τη νέα συμπεριφορά του Kodu. Όπως παρατηρείτε είναι κενή. Αν παίξουμε το παιχνίδι μας χωρίς να προσθέσουμε τίποτε στη δεύτερη σελίδα, τότε όταν ο Kodu μόλις ακουμπήσει ένα πράσινο μήλο, θα ακινητοποιείται καθώς δε θα έχει καμία συμπεριφορά! Μη ξεχνάτε ότι οι συμπεριφορές της πρώτης σελίδας δεν ισχύουν πλέον για τον Kodu, μόνο αυτές της δεύτερης, που θα είναι όμως κενή. Δοκιμάστε το.

Τι θα συμπεριλάβουμε στη δεύτερη σελίδα; Όταν ο Kodu *πέφτει πάνω (bump)* σε πράσινο μήλο ο παίχτης θα χάνει τον έλεγχο του Kodu, και για δέκα δευτερόλεπτα θα *περιφέρεται (wander)* μέσα στο κόσμο του παιχνιδιού και ύστερα θα επιστρέφει στην αρχική σελίδα για να συνεχίσει να μαζεύει κόκκινα μήλα. Άρα οι συμπεριφορές της δεύτερης σελίδας:

Στην πρώτη συμπεριφορά, η συνθήκη είναι άδεια, κάτι που ισοδυναμεί με την ενέργεια *Για Πάντα* (*Always*), διότι θέλουμε ο Kodu σε όλη τη διάρκεια της συμπεριφοράς του στη δεύτερη σελίδα να περιφέρεται. Χρησιμοποιώντας τον αισθητήρα του χρόνου και ως προσδιοριστικό τα δέκα δευτερόλεπτα, εκτελούμε την ενέργεια αλλαγής στη σελίδα 1 μετά από δέκα δευτερόλεπτα, οπότε και θα ξανα-αποκτήσει ο Kodu την αρχική του συμπεριφορά.

Μήπως καταλάβατε λίγο καλύτερα τη χρησιμότητα και την ανάγκη της ***Αλλαγής σελίδας (Switch page)***; Ας δούμε άλλο ένα μικρό παράδειγμα.

Δείτε το παράδειγμα *09_02.kodu*

[09_02.Kodu] Αυτή τη φορά ο ήρωας του παιχνιδιού μας δεν είναι ο Kodu αλλά ο Ύπο! Ένα *Υποβρύχιο (sub)* που βρίσκεται σε έναν πολύ όμορφο βυθό λίμνης, γεμάτο από *Αστέρια (star)*, *Καρδιές (heart)* αλλά και πολλά *Ψάρια (fish)*. Ο ήρωας μας, όπως και στο προηγούμενο παιχνίδι, κινείται με τα βελάκια του πληκτρολογίου. Κάθε καρδούλα και κάθε αστέρι δίνει στον Ύπο ένα βαθμό. Στόχος του Ύπο είναι να συγκεντρώσει είκοσι βαθμούς σε χρόνο δύομιση λεπτών. Σε περίπτωση που ο Ύπο ακουμπίσει κάποιο ψάρι, ακινητοποιείται όπως και το ψάρι στο οποίο έπεσε πάνω. Ο Ύπο όμως ακινητοποιείται για δέκα δευτερόλεπτα ενώ το ψάρι μόνιμα. Υπάρχει όμως και ένα μεγάλο πρόβλημα. Κάθε φορά που ο ήρωάς μας πέφτει πάνω σε κάποιο ψάρι, το ψάρι εκείνο εξαφανίζει είτε μία καρδούλα είτε ένα αστέρι από τον βυθό, με αποτέλεσμα ο Ύπο να έχει λιγότερες πιθανότητες να καταφέρει να κερδίσει στο παιχνίδι. Στο βυθό μαζί με τον Ύπο

βρίσκονται είκοσι ψάρια, δεκαοκτώ καρδούλες και δεκατρία αστεράκια. Τι λέτε θα τα καταφέρει ο Ύπο; Πάμε να δημιουργήσουμε το παιχνίδι μας.

Δημιουργία κόσμου: Ο ήρωας μας πρέπει να βρίσκεται σε υδάτινο περιβάλλον, οπότε δημιουργούμε μια επίπεδη πίστα την οποία γεμίζουμε με νερό. Ας τοποθετήσουμε στη συνέχεια τον Ύπο, τα είκοσι ψάρια, τις δεκαοκτώ καρδούλες και τα δεκατρία αστεράκια. Ο κόσμος του Ύπο:

Προγραμματισμός Ύπο: Ο Ύπο θα κινείται με τα βελάκια του πληκτρολογίου. Επίσης όταν θα πέφτει πάνω σε κάποια καρδιά ή αστέρι θα το εξαφανίζει και θα κερδίζει ένα πόντο. Όταν ο Ύπο *πέφτει πάνω* σε κάποιο ψάρι, θα πρέπει να αλλάζει η συμπεριφορά του: θα ακινητοποιείται και θα λάμπει! Αυτή η συμπεριφορά θα ενημερώνει τον παίχτη, πως όση ώρα λάμπει ο ήρωας, δεν θα μπορεί να τον ελέγξει. Ο παίχτης θα ανακτά τον έλεγχο του Ύπο όταν περάσει χρόνος δέκα δευτερολέπτων. Τότε και ο Ύπο θα σταματήσει να λάμπει.

Συνεπώς, για 10 δευτερόλεπτα ο Ύπο θα πρέπει να έχει διαφορετική συμπεριφορά από την αρχική του, οπότε μάλλον χρειαζόμαστε αλλαγή συμπεριφοράς…δηλαδή ***Αλλαγή σελίδας (Switch page)***. Έτσι οι συμπεριφορές του υδρόβιου ήρωα μας εκφράζονται στις παρακάτω γραμμές της πρώτης σελίδας:

Πριν πάμε να δώσουμε στη δεύτερη σελίδα τη νέα συμπεριφορά του 'Υπο ας ολοκληρώσουμε καλύτερα την πρώτη σελίδα. Τώρα οι μόνες εντολές που λείπουν, είναι αυτές που εκφράζουν πότε χάνει και πότε κερδίζει το παιχνίδι ο Ύπο. Κερδίζει, όταν συγκεντρώσει είκοσι βαθμούς και χάνει αν δεν καταφέρει να συγκεντρώσει τους απαραίτητους βαθμούς σε χρόνο δυόμιση λεπτών. Συμπληρώνουμε την προηγούμενη σελίδα με τις παρακάτω συμπεριφορές:

Τώρα που ολοκληρώσαμε την πρώτη σελίδα του Ύπο, ας προσδιορίσουμε τη συμπεριφορά που θα αποκτά όταν ακουμπά ένα ψάρι.

Όταν πέφτει πάνω σε κάποιο ψάρι θα πρέπει να λάμπει και να ακινητοποιείται για 10 δευτερόλεπτα. Δηλαδή δε θα δώσουμε στον παίχτη τη δυνατότητα να κινεί τον Ύπο. Άρα, στη σελίδα 2, δε θα έχουμε καμία ενέργεια κίνησης. Επίσης, ο Ύπο για δέκα δευτερόλεπτα θα λάμπει και έπειτα θα επιστρέφει στην αρχική σελίδα για συνεχίσει:

Και ο Ύπο είναι έτοιμος! Τι θα γινόταν αν αλλάζαμε τη σειρά της δεύτερης και τρίτης συμπεριφοράς; Τι θα αλλάξει και γιατί;

Εκτός από τον Ύπο όμως και τα ψάρια έχουν συμπεριφορά στο παιχνίδι μας. Τα ψάρια όταν έρχονται σε επαφή με το υποβρύχιο ακινητοποιούνται και καλούνται να εξαφανίσουν ένα αστέρι ή μια καρδιά. Ας συμφωνήσουμε ότι κάποια από τα ψάρια θα εξαφανίζουν τις καρδούλες και κάποια τα αστεράκια. Μετά την επαφή με τον Ύπο, τα ψάρια θα πρέπει να παραμένουν ακινητοποιημένα ενώ για λόγους διασκέδασης, όταν ακινητοποιούνται καλό θα ήταν να βγάζουν αστεράκια από το κεφάλι τους!!! Στο παιχνίδι μας από τα είκοσι ψάρια, τα δεκατρία εξαφανίζουν καρδούλες και τα υπόλοιπα αστεράκια.

Λογικά πρέπει να παρατηρήσατε ότι και τα ψάρια αλλάζουν συμπεριφορά. Έτσι σε αυτό το παιχνίδι εκτός από τον Ύπο και τα ψάρια χρησιμοποιούν την ***Αλλαγή σελίδας (Switch page)***. Η πρώτη σελίδα σε όλα τα ψάρια θα είναι ίδια. Τα ψάρια πρέπει να περιφέρονται στην πίστα μας και όταν ακουμπήσουν το υποβρύχιο, θα πρέπει να αλλάζουν συμπεριφορά:

Η δεύτερη σελίδα θα διαφέρει καθώς κάποια ψάρια θα εξαφανίζουν καρδούλες και άλλα αστέρια. Η συμπεριφορά στη δεύτερη σελίδα των ψαριών που εξαφανίζουν καρδιές:

Ενώ η συμπεριφορά των ψαριών που εξαφανίζουν τα αστεράκια:

Και εδώ θα πρέπει να προγραμματίσετε ένα ένα και τα είκοσι ψάρια, όπως είπαμε: δεκαοκτώ εξαφανίζουν καρδούλες και δεκατρία αστεράκια. Τελικά θα κερδίσει ο ΥΠΟ; Για να δούμε…

Μήπως καλύτερα να δημιουργούσαμε ένα ψάρι και στη συνέχεια να αντιγράφαμε τα υπόλοιπα και να τοποθετούσαμε σε διαφορετικές θέσεις; Σε ποια περίπτωση θα χρειαζόταν περισσότερος χρόνος;

9.2 Κατασκευή κλώνων (Creatables)

Οι περισσότεροι από εσάς θα έχετε ήδη παίξει τουλάχιστον ένα παιχνίδι στο οποίο οι εχθροί πολλαπλασιάζονται καθώς εσείς καθυστερείτε να επιτύχετε το στόχο σας! 'Εχετε αναρωτηθεί πώς θα μπορούσαμε να προγραμματίσουμε τη συνεχή δημιουργία νέων αντικειμένων στο περιβάλλον του MSKodu; Μέχρι στιγμής, κάθε αντικείμενο που εμφανιζόταν στα παιχνίδια μας, είχε προηγουμένως δημιουργηθεί από εμάς κατά το σχεδιασμό του παιχνιδιού. Δεν ήμασταν σε θέση να προγραμματίσουμε την εμφάνιση νέων αντικειμένων. Αν ένα αντικείμενο εξολοθρευόταν, τότε απλά έμεναν λιγότεροι εχθροί ή φίλοι για τον ήρωά μας.

Το Kodu όμως μας δίνει τη δυνατότητα να δημιουργήσουμε «***αρχέτυπα***» αντικείμενα, δηλαδή «*πρότυπα*» αντικείμενα τα οποία μπορούμε να αναπαράγουμε προγραμματιστικά μέσα στο παιχνίδι μας! Είμαστε δηλαδή σε θέση να δημιουργήσουμε έναν «πρότυπο» εχθρό ή φίλο και στη συνέχεια να προγραμματίσουμε την εμφάνιση του σε ένα ή περισσότερα αντίγραφα κάτω από οποιεσδήποτε συνθήκες το επιθυμούμε!

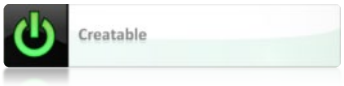
Πρέπει να γνωρίζουμε 2 πράγματα:

Α) **Το αρχέτυπο αντικείμενο**, το αντικείμενο δηλαδή που θέλουμε να εμφανίζεται δυναμικά στον κόσμο μας. Για να φτιάξουμε ένα αρχέτυπο αντικείμενο, το μόνο που έχουμε να κάνουμε είναι να δημιουργήσουμε ένα αντικείμενο, να το προγραμματίσουμε και στη συνέχεια να ενεργοποιήσουμε την ιδιότητά του «**Creatable**». Το αρχέτυπο αντικείμενο δεν φαίνεται μέσα στο παιχνίδι μας, αλλά αρχίζει να αναπαράγεται όταν χρησιμοποιηθεί η ενέργεια ***Δημιούργησε (Create)*** από οποιοδήποτε αντικείμενο.

Β) **Η ενέργεια *Δημιούργησε (Create)*.** Η ενέργεια αυτή μπορεί να χρησιμοποιηθεί από οποιοδήποτε αντικείμενο του παιχνιδιού μας και προκαλεί την εμφάνιση αντιγράφων του αρχέτυπου αντικειμένου και όχι μόνο!

[09_03.Kodu] Ας δούμε πως μπορούμε να αξιοποιήσουμε αυτές τις δυνατότητες μέσα από ένα παράδειγμα! Θα φτιάξουμε ένα εργοστάσιο το οποίο θα παράγει μοτοσυκλέτες! Θα το προγραμματίσουμε έτσι ώστε κάθε 10 δευτερόλεπτα να παράγει μία μοτοσυκλέτα η οποία θα ακολουθεί προκαθορισμένη κυκλική πορεία στο περιβάλλον γύρω από τον Kodu. Ο χρήστης θα μπορεί μόνο να περιστρέφει τον Kodu και εκτοξεύοντας πυραύλους θα πρέπει να εξολοθρεύσει όλα τα μηχανάκια.

Πως θα τα καταφέρουμε;



1) ***Πρέπει να δημιουργήσουμε το αρχέτυπο αντικείμενο***. Πρώτα, τοποθετούμε ένα *Μηχανάκια* στο περιβάλλον μας. Για να ορίσουμε το συγκεκριμένο αντικείμενο ως πρότυπο, θα πρέπει να αλλάξουμε στις ιδιότητές του, την ιδιότητα «*creatable*» πατώντας δεξί κλικ πάνω στο αντικείμενο και επιλέγοντας «*change settings*». Αν πατήσουμε πάνω στη συγκεκριμένη ιδιότητα, το αντίστοιχο εικονίδιο θα γίνει πράσινο και η ιδιότητα θα ενεργοποιηθεί:

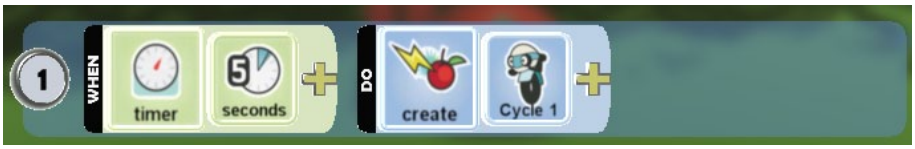
Θα παρατηρήσετε ότι αμέσως η μοτοσυκλέτα-αρχέτυπο (creatable) αποκτά μία πράσινη σκιά, που υποδηλώνει ότι είναι αντικείμενο-πρότυπο.

Αν τρέξετε εκείνη τη στιγμή το παιχνίδι σας, θα παρατηρήσετε ότι ο Μηχανάκιας δεν εμφανίζεται στον κόσμο μας. Δεν θα πρέπει να ξεχνάμε ότι τα αρχέτυπα αντικείμενα δεν εμφανίζονται στα παιχνίδια μας. Μόνο τα αντίγραφά τους εμφανίζονται εφόσον έχει χρησιμοποιηθεί η ενέργεια ***Δημιούργησε (Create)***.

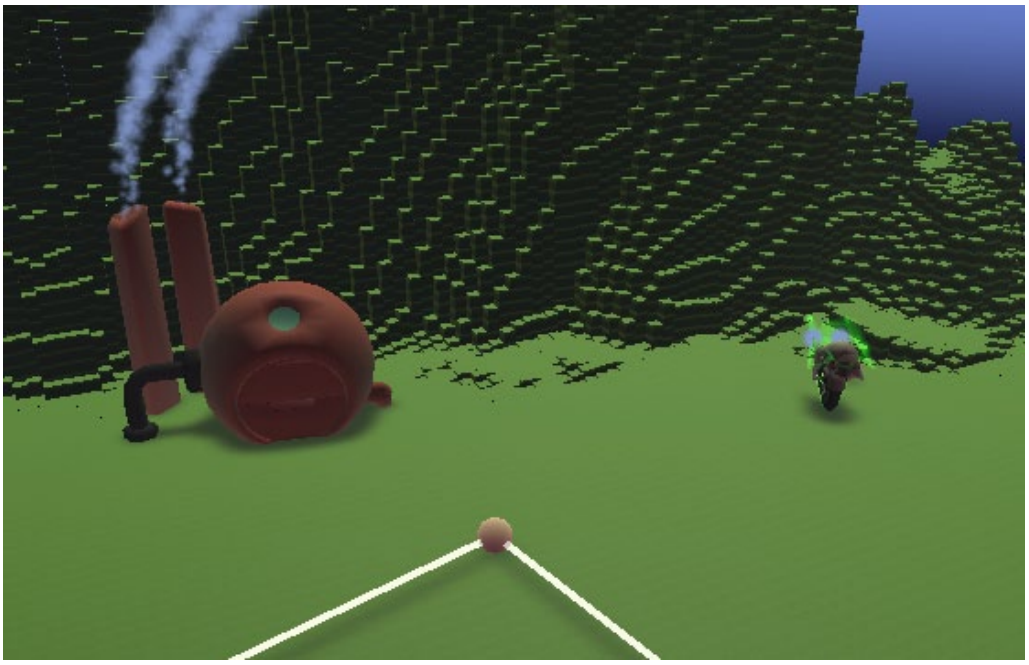
Αφού χαράξουμε την πορεία που θέλουμε να ακολουθούν τα μηχανάκια χρησιμοποιώντας το εργαλείο ***Μονοπάτι (Path)***, προσδιορίζουμε ως παντοτινή ενέργειά τους την κίνηση πάνω σε αυτό. Επίσης προσθέτουμε μια δεύτερη εντολή έτσι ώστε όποτε χάνει την ζωή της μια μοτοσυκλέτα, τότε να αυξάνεται στο σκορ του χρήστη κατά ένα πόντο.

2) ***Πρέπει να αποφασίσουμε ποιο αντικείμενο θα δημιουργεί κλώνους του Μηχανάκια***! Ένα εργοστάσιο μήπως θα ήταν μια αρκετά πιστευτή ιδέα; Εισάγετε ως αντικείμενο στον κόσμο μας ένα εργοστάσιο. Κάτω από ποιες συνθήκες θέλουμε να δημιουργούνται τα αντίγραφα; Συνήθως, κάτι τέτοιο συμβαίνει με το πέρασμα του χρόνου. Στο παράδειγμά μας, χρησιμοποιώντας τον αισθητήρα ***Χρονόμετρο (Timer)*** με προσδιοριστικό τα 5 δευτερόλεπτα, θα δημιουργούμε νέους *κλώνους* που θα κινούνται πάνω στο μονοπάτι μας κάθε 5 δευτερόλεπτα! Για να δημιουργήσουμε τους νέους κλώνους, θα χρησιμοποιήσουμε την ενέργεια ***Δημιούργησε (Create)***. Όταν επιλέξετε να προσθέσετε προσδιοριστικό στην ενέργεια ***Δημιούργησε (Create)***, θα παρατηρήσετε ότι εμφανίζονται όλα τα αντικείμενα που έχουμε προσδιορίσει ως αρχέτυπα (creatables). Στη δική μας περίπτωση θα δείτε το Μηχανάκια!

Άρα η συμπεριφορά του εργοστασίου:



Τρέχοντας το παιχνίδι μπορούμε να παρατηρήσουμε ότι οι κλώνοι εμφανίζονται αρχικά δίπλα στο αντικείμενο που τους παράγει και στη συνέχεια κατευθύνονται προς το μονοπάτι στο οποίο θα κινούνται. Η αρχική θέση του αρχέτυπου δεν παίζει κανέναν ρόλο!



Είναι ενδιαφέρον να δούμε τι επιπλέον προσδιοριστικά έχει η ενέργεια ***Δημιούργησε (Create)***. Πατώντας το «+» για να εμφανιστούν τα επιπλέον προσδιοριστικά, θα δούμε ότι μπορούμε να ορίσουμε τι χρώμα θέλουμε να έχουν οι μοτοσικλέτες. Αυτό φαίνεται να έχει ενδιαφέρον καθώς θα μπορούσαμε με ένα αρχέτυπο αντικείμενο να δημιουργήσουμε κατηγορίες μηχανών με διαφορετικά χρώματα. Επίσης υπάρχει η επιλογή ***Μια φορά (once)***, που έχει ως αποτελέσμα η συγκεκριμένη ενέργεια να εκτελεστεί μόνο μια φορά.

Τρέχοντας το παράδειγμα σε αυτή τη μορφή, θα διαπιστώσουμε ότι μετά από κάποιο χρονικό διάστημα προκύπτει πρόβλημα υπερφόρτωσης του παιχνιδιού μας επειδή δημιουργούνται πολλά αντικείμενα. Θα εμφανιστεί στην οθόνη το *θερμόμετρο χρήσης πόρων του συστήματός μας*, το οποίο θα δείχνει την θερμοκρασία να πιάνει «κόκκινο» (περισσότερες λεπτομέρειες συζητήσαμε στο κεφάλαιο 7). Το παιχνίδι θα γίνει πολύ αργό και κάποια στιγμή που θα εξαντληθούν τελείως οι διαθέσιμοι πόροι του συστήματός μας, θα πάψουν να δημιουργούνται νέες μοτοσυκλέτες από το εργοστάσιο. Αν έχετε ενεργοποιήσει την ιδιότητα *Περιορισμού των Πόρων*, τότε οι μοτοσυκλέτες κάποια στιγμή νωρίτερα θα σταματήσουν να δημιουργούνται ώστε να μη δημιουργηθεί πρόβλημα στο παιχνίδι. Καλό όμως είναι να μην αφήνετε τους κλώνους να πολλαπλασιάζονται ανεξέλεγκτα. Στο παράδειγμα μας ένας τρόπος για να λύσουμε αυτό το πρόβλημα είναι να επιλέξουμε το αρχέτυπο αντικείμενό μας και να προσθέσουμε την επόμενη συμπεριφορά:

Κάθε μοτοσυκλέτα έχει διάρκεια ζωής 2 λεπτά, ενώ αμέσως μετά εκρήγνυται. Αυτό βάζει ένα όριο στον αριθμό των μοτοσυκλετών που μπορούν να υπάρχουν στον κόσμο μας κατά την διάρκεια του παιχνιδιού και δεν υπάρχει κίνδυνος υπερφόρτωσης. Ο μέγιστος αριθμός από Μηχανάκια που μπορούμε να έχουμε κάθε στιγμή στην πίστα μας είναι 24. Γιατί;

Υπάρχει όμως ένα πρόβλημα! Σύμφωνα με τις προηγούμενες συμπεριφορές, κάθε φορά που μια μοτοσυκλέτα εκρήγνυται από μόνη της, ο παίκτης κερδίζει ένα πόντο χωρίς να κάνει καμία ενέργεια! Για να μην αυξάνονται οι πόντοι μας κάθε φορά που εκρήγνυται ένας μηχανάκιας με αυτόν τον τρόπο, μπορούμε να προσθέσουμε μία εντολή όπου κάθε 2 λεπτά θα αφαιρείται ένας πόντος! Οι συμπεριφορές του αρχέτυπου αντικειμένου τελικά θα είναι οι εξής:

Για να αποκτήσει κάποιο νόημα το παράδειγμα ως παιχνίδι μπορείτε πολύ απλά να τοποθετήσετε τον Kodu στο κέντρο και να του δώσετε τη δυνατότητα να περιστρέφετε και να πυροβολεί τους μοτοσικλετιστές μαζεύοντας πόντους:

9.3 Συνδυαστικό παράδειγμα με αλλαγή σελίδας και αρχέτυπα αντικείμενα

Τι λέτε να φτιάξουμε ένα λίγο πιο περίπλοκο παιχνίδι συνδυάζοντας την αλλαγή σελίδας και τα αρχέτυπα αντικείμενα;

Δείτε το παράδειγμα *09_04.kodu*

[09_04.kodu] Έστω ότι ο Kodu περιφέρεται σε ένα μεγάλο δάσος, το οποίο είναι γεμάτο με *Δέντρα (tree), Κέρματα (coin), Μήλα (apple)* και *Πέτρες (rock).* Τα δέντρα και οι πέτρες είναι διακοσμητικά στο χώρο του πάρκου ενώ τα κέρματα δίνουν πόντους στον Kodu. Υπάρχει όμως ένας κίνδυνος για τον Kodu: κάθε δεύτερη φορά που πέφτει πάνω σε κάποιο κέρμα, εμφανίζεται ένας εχθρός που στόχο του έχει να τον εξολοθρεύσει. Και εδώ αποκτούν νόημα τα μήλα. Μόλις ο χρήστης πάρει ένα μήλο, τότε ο Kodu θα πρέπει να κινείται πιο γρήγορα και να αλλάζει χρώμα και από άσπρος να γίνεται μπλε. Μόλις ο Kodu γίνεται μπλε, θα έχει την ικανότητα να πετάει πυραύλους για να σκοτώνει τους εχθρούς του. ΠΡΟΣΟΧΗ όμως, όταν ο Kodu γίνει μπλε δεν μπορεί πια να μαζέψει πόντους. Το μόνο που κάνει είναι να πετάει πυραύλους για να σκοτώσει τους εχθρούς του. Ωστόσο για να συνεχίσει να μαζεύει πόντους θα πρέπει να ξαναπάρει μήλο, να αλλάξει πάλι χρώμα (από μπλε να γίνει άσπρος) και τότε θα έχει και πάλι τη δυνατότητα να συγκεντρώσει πόντους. Για λόγους διασκέδασης, όταν ο Kodu περνάει από ένα βράχο, ο βράχος θα του λέει κάτι. Στόχος του Kodu είναι να μαζέψει 200 πόντους σε λιγότερο από δύο λεπτά και πριν τον σκοτώσουν οι εχθροί του.

Δημιουργία του κόσμου: Μέσα σε έναν κενό κόσμο τοποθετείστε σε τυχαίες θέσεις κάποια δέντρα, κάποια νομίσματα μεγάλου μεγέθους (για να μπορεί ο Kodu να πέφτει πάνω τους), κάποια μήλα επίσης μεγάλου μεγέθους και κάποιους βράχους, έτσι, για πλάκα! Μη ξεχνάτε όταν δημιουργείτε κάτι, να το διασκεδάζετε. Μη ξεχάσετε να προσθέσετε και τον Kodu φυσικά! Τώρα ο κόσμος σας θα πρέπει να μοιάζει κάπως έτσι:

Προγραμματίζοντας τον Kodu: Ας προγραμματίσουμε τώρα τον Kodu. Το πρώτο πράγμα που θέλουμε είναι να κινεί ο παίχτης τον Kodu. Έχουμε δει άπειρες φορές την ίδια συμπεριφορά!

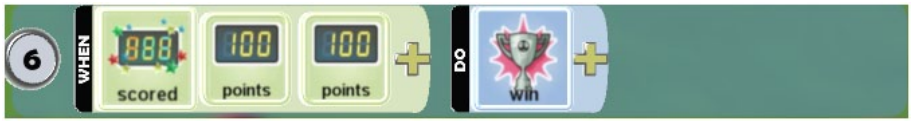
Όπως είπαμε παραπάνω, θέλουμε όταν ο Kodu χτυπάει κάποιο κέρμα, τότε αυτό να εξαφανίζεται και ο παίκτης να κερδίζει πόντους. Έστω ότι από κάθε κέρμα ο παίκτης κερδίζει 10 πόντους:

Όπως μπορείτε να αναγνωρίσετε και από μόνοι σας, σε οποιαδήποτε σειρά και να μπουν οι δυο αυτές συμπεριφορές, δεν θα αλλάξει κάτι. Ο Kodu θα έχει την επιθυμητή συμπεριφορά.

Θυμάστε ποια θέλουμε να είναι η συμπεριφορά του Kodu όταν πέφτει πάνω στα μήλα; Όταν πέφτει πάνω σε κάποιο μήλο ο Kodu θα το τρώει, θα αλλάζει χρώμα και μετά θα έχει τη δυνατότητα να πετάει πυραύλους, για να σκοτώσει τους εχθρούς που εμφανίζονται, αλλά χωρίς να μπορεί να μαζέψει πόντους... Τι παρατηρείτε; Ο Kodu αλλάζει συμπεριφορά! Και τι είχαμε πει ότι κάνουμε όταν θέλουμε να αλλάξουμε από μια συγκεκριμένη συμπεριφορά (πέφτει πάνω σε νομίσματα και μαζεύει πόντους) σε μία άλλη (πέφτει πάνω σε μήλα, αλλάζει χρώμα, μπορεί να εκτοξεύει πυραύλους αλλά δε μπορεί να μαζεύει πόντους); Αλλαγή σελίδας!

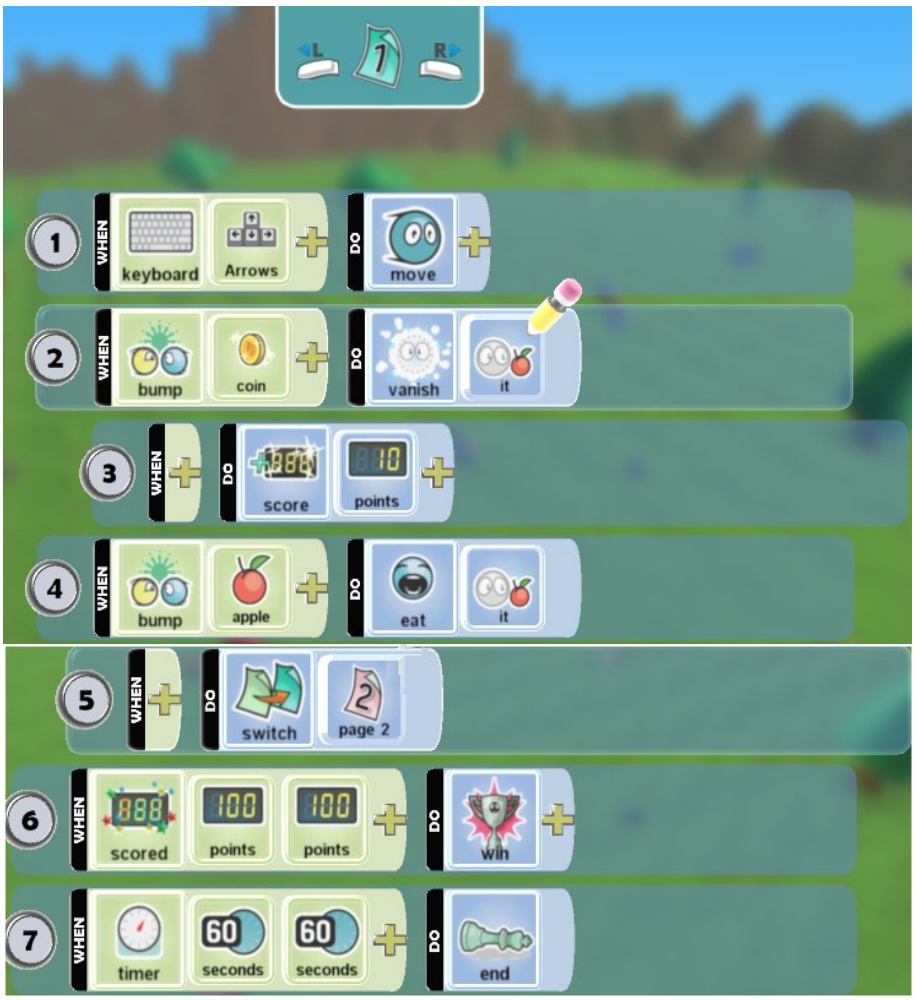
Αρχικά θα πρέπει να πούμε στον Kodu ότι όταν θα πέφτει πάνω σε ένα μήλο, θα πρέπει να το τρώει και μετά να αλλάζει σελίδα για να εκτελεί τη νέα του συμπεριφορά. Έτσι θα προσθέσουμε στις προηγούμενες συμπεριφορές τις παρακάτω:

Πριν πάμε να δημιουργήσουμε τις συμπεριφορές του *Kodu* στη δεύτερη σελίδα, ας ολοκληρώσουμε τις εντολές στην πρώτη σελίδα του παιχνιδιού μας. Σε χρόνο 2 λεπτών, όπως είπαμε, ο *Kodu* πρέπει να μαζέψει 200 πόντους για να κερδίσει το παιχνίδι:



Αν περάσουν τα δύο λεπτά και ο *Kodu* δεν έχει κάνει μαζέψει αρκετούς πόντους, τότε ο παίκτης θα χάνει. Έτσι η τελευταία συμπεριφορά θα έχει τη μορφή:

Και τώρα ήρθε η στιγμή να δούμε ολόκληρη την πρώτη σελίδα του παιχνιδιού μας:



Στη σελίδα 2 θέλουμε να πούμε στον Kodu:

Α) να κινείται πιο γρήγορα καθώς ο παίκτης χρησιμοποιεί τα βελάκια του πληκτρολογίου,

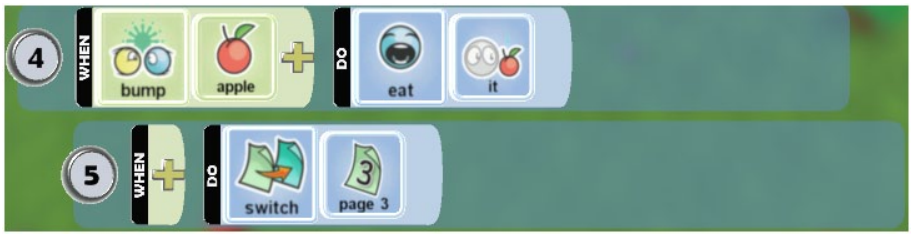
Β) να αλλάξει χρώμα και να γίνει μπλε,

Γ) να μπορεί να εκτοξεύει πυραύλους όταν ο παίκτης πατά το πλήκτρο space του πληκτρολογίου.

Δεν είναι δύσκολο να «μεταφράσουμε» τις απαιτήσεις αυτές σε συμπεριφορές του χαρακτήρα μας για τη σελίδα 2:

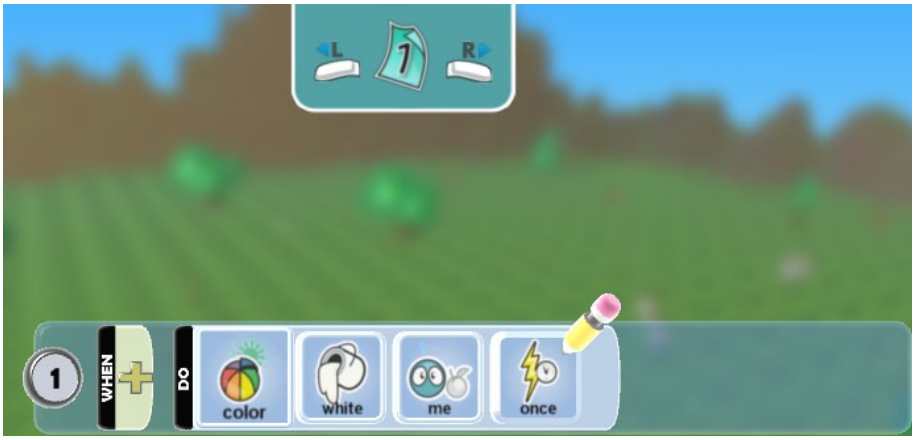
Αν ο Kodu ξαναφάει κάποιο μήλο, τότε το χρώμα του θα πρέπει να γίνεται ξανά άσπρο και να επιστρέφει στην πρώτη σελίδα συμπεριφοράς για να συνεχίσει να μαζεύει πόντους. Παραμένουμε στη σελίδα 2 και πρέπει τώρα να πούμε στον Kodu ότι αν φάει μήλο να αλλάξει

χρώμα. Παρότι θα μπορούσε να γίνει με πολύ εύκολο τρόπο, εμείς αρχικά θα αλλάξουμε το χρώμα στο Kodu σε μια νέα σελίδα συμπεριφορών για να εξοικειωθούμε καλύτερα με τη συγκεκριμένη έννοια. Προσθέστε λοιπόν τις δυο επόμενες συμπεριφορές στη σελίδα 2:



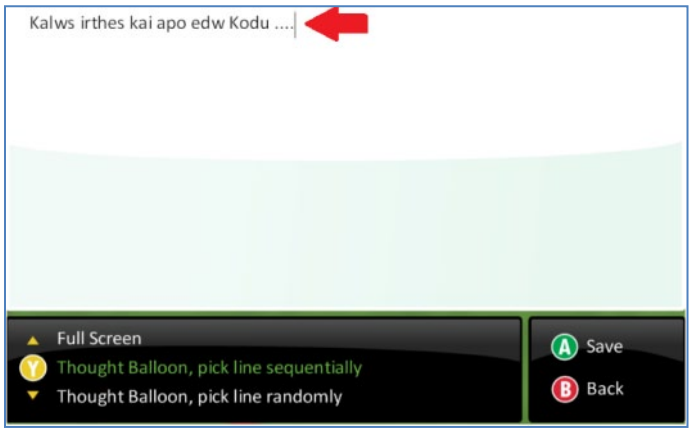
Στη σελίδα 3 θα κάνουμε κάτι διαφορετικό από ότι έχουμε δει μέχρι τώρα. Το μόνο που μας ενδιαφέρει είναι να αλλάξουμε το χρώμα του Kodu από μπλε σε άσπρο και στη συνέχεια να μεταφερθούμε στην πρώτη σελίδα συμπεριφορών:

Μήπως όμως είναι περιττή η χρήση της σελίδας 3 εδώ; Μήπως θα μπορούσαμε να υλοποιήσουμε κάπου αλλού τις εντολές της σελίδας 3; Έχετε καμία ιδέα; Τι θα λέγατε αν στην πρώτη σελίδα προσθέταμε στην αρχή μια εντολή της μορφής:



Θα άλλαζε κάτι στο παιχνίδι μας; Μπορείτε να προσθέσετε μόνοι σας τη συγκεκριμένη συμπεριφορά και να παρατηρήσετε τυχόν διαφορές με την προηγούμενη λύση; Η απάντηση είναι απλή. Η δεύτερη λύση είναι πιο λογική και ταυτόχρονα δε δημιουργεί και το χρονικό κενό των 0.25 δευτερολέπτων που ο Kodu δεν κάνει τίποτε! Υπάρχει όμως και τρίτη λύση στο πρόβλημα του χρωματισμού του Kodu που θα την βρείτε μόνοι σας!

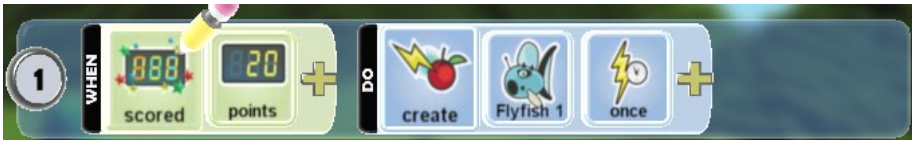
Προγραμματίζοντας τους βράχους: Όπως αναφέραμε και στην αρχική περιγραφή, ο μόνος λόγος που βάλαμε τους βράχους στον κόσμο του Kodu, είναι για να κάνουμε πιο διασκεδαστικό το παιχνίδι μας. Οι βράχοι απλά να μιλούν στο Kodu με τη χρήση της ενέργειας ***Πες (Say)***:



Καλό είναι να θυμηθούμε ότι μπορούμε να γράψουμε σε διαφορετικές γραμμές όσες προτάσεις θέλουμε στο παράθυρο της ενέργειας ***Πες (Say)*** και να επιλέξουμε να εμφανίζονται οι φράσεις αυτές με τυχαία σειρά. Δυστυχώς, το MSKodu δεν διαθέτει ελληνικά ακόμη, οπότε οι φράσεις μας αναγκαστικά θα είναι γραμμένες με λατινικούς χαρακτήρες. Ξεκινήστε και κάντε τους βράχους σας να μιλάνε…

Προγραμματίζοντας τους εχθρούς: Δημιουργήστε ένα ιπτάμενο ψάρι και προσδιορίστε το ως *αντικείμενο αρχέτυπο* ενεργοποιώντας την ιδιότητα «*creatable*». Πρέπει να προγραμματίσουμε το αρχέτυπο αντικείμενο ώστε όταν βλέπει τον Kodu να τρέχει προς τα πάνω του, κι όταν τον ακουμπάει να τερματίζεται το παιχνίδι. Για να γίνει πιο ελκυστικό το παιχνίδι μας, καλό θα ήταν προσθέσουμε μια επιπλέον συμπεριφορά για να αποφεύγει το ιπτάμενο ψάρι τα δέντρα που συναντά στο δρόμο του! Αυτό θα το πετύχουμε χρησιμοποιώντας το προσδιοριστικό ***Απέφυγε (Avoid)*** της ενέργειας ***Κινήσου (Move)***:

Τέλος, θα πρέπει να επιλέξουμε από ποια αντικείμενα θα δημιουργούνται οι κλώνοι του αρχέτυπου ψαριού και πότε! Επιλέξτε μερικά δέντρα για να κάνουν αυτήν την δουλειά και κάθε κλώνος να βγαίνει από διαφορετικό σημείο. Κάθε πότε όμως θα πρέπει να δημιουργούνται; Ο παίκτης κερδίζει 10 πόντους για κάθε μήλο που τρώει ο Kodu. Άρα κάθε 20 βαθμούς (δηλαδή 2 μήλα) θα πρέπει να δημιουργείτε έναν κλώνο. Στο πρώτο δέντρο θα δώσετε την παρακάτω συμπεριφορά.



Στο δεύτερο δέντρο θα επαναλάβετε την ίδια συμπεριφορά με προσδιοριστικό τους 40 πόντους, στο τρίτο τους 60 πόντους κτλ. Συνολικά, θα πρέπει να μπορούν να εμφανιστούν μέχρι 9 κλώνοι αφού έχουμε θέσει τους 200 πόντους ως στόχο του παιχνιδιού. Δημιουργήστε τις αντίστοιχες συμπεριφορές και …έτοιμο και αυτό το παιχνίδι μας!

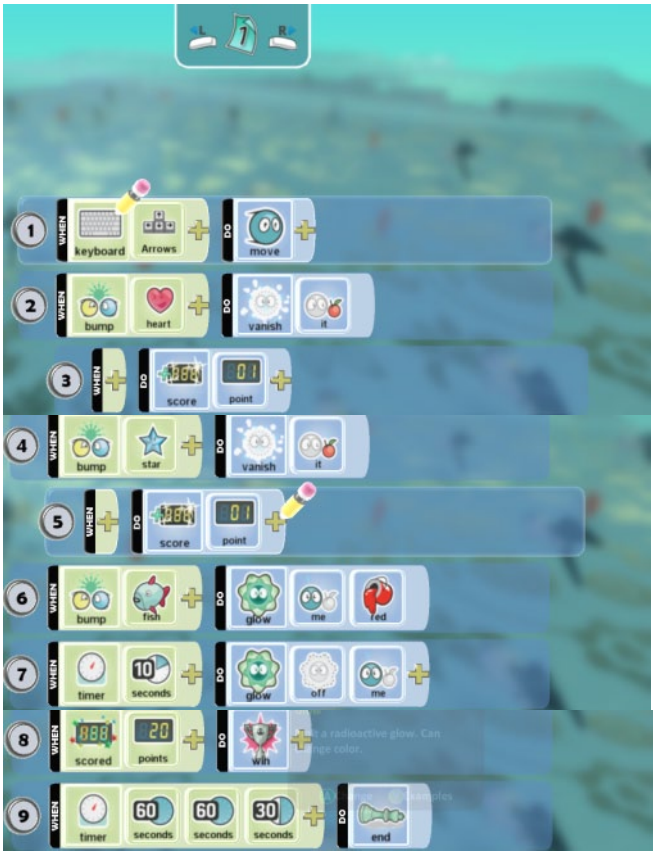
Περίληψη

Σε αυτό το κεφάλαιο ήρθαμε σε επαφή με δυο καινούριες έννοιες. Την έννοια της ***Αλλαγής σελίδας (Switch page)*** και την έννοια των ***Αρχέτυπων Αντικειμένων (Creatable)***. Ενώ μέχρι τώρα δίναμε στα αντικείμενα των παιχνιδιών μας σταθερές συμπεριφορές, είδαμε ότι στο MSKodu μπορούμε να αλλάζουμε τον τρόπο που φέρονται τα αντικείμενά μας κάτω από διαφορετικές συνθήκες με τη χρήση της ενέργειας ***Αλλαγή σελίδας (Switch page)***. Πλέον, αν ο πρωταγωνιστής μας φάει ένα μήλο, μπορεί να τρέχει πιο γρήγορα, αν πέσει πάνω σε ένα δέντρο μπορεί να ακινητοποιείται για κάποια δευτερόλεπτα, αν μαζέψει 100 πόντους μπορεί να αποκτήσει ένα επιπλέον όπλο κτλ.! Επιπλέον με τα ***Αρχέτυπα Αντικείμενα (Creatable)***, γνωρίσαμε μια πολύ χρήσιμη ιδιότητα που μας προσφέρει το MSKodu. Ενώ μέχρι τώρα, όλα τα αντικείμενα που εμφανίζονταν στα παιχνίδια μας, είχαν δημιουργηθεί από εμάς κατά τον σχεδιασμό του παιχνιδιού, πλέον είμαστε σε θέση να προγραμματίζουμε την εμφάνιση νέων αντικειμένων καθώς ο παίκτης παίζει το παιχνίδι μας. Αρκεί να δημιουργήσουμε ένα αρχέτυπο αντικείμενο και να αποφασίσουμε ποιο αντικείμενο θα δημιουργεί τους κλώνους του, χρησιμοποιώντας την ενέργεια ***Δημιούργησε (Create)***. Οι εχθροί και οι φίλοι πλέον είναι ατελείωτοι!

Ερωτήσεις

1. Αναφέρετε τους λόγους που είναι χρήσιμη η ενέργεια ***Αλλαγή Σελίδας (Switch Page)*** και δώστε ένα μικρό δικό σας παράδειγμα για να δικαιολογήσετε τη χρησιμότητα της.

2. Αν στο δεύτερο παιχνίδι που δημιουργήσαμε με τις καρδούλες και τα αστεράκια, δεν δημιουργούσαμε δεύτερη σελίδα συμπεριφορών αλλά την εξής σελίδα:



θα άλλαζε το παιχνίδι μας; Τα αντικείμενα μας θα είχαν την ίδια συμπεριφορά με αυτή που ορίσαμε μέσα στο κεφάλαιο; Τι διαφορές παρατηρείτε ανάμεσα στο αρχικό παιχνίδι και σε αυτό που δημιουργήσαμε μόλις τώρα; Γιατί χρειαζόμασταν την ***Αλλαγή Σελίδας (Switch Page)***;

3. Το MSKodu όπως είπαμε διαθέτει δώδεκα σελίδες, δηλαδή μας επιτρέπει να δώσουμε στα αντικείμενα μας δώδεκα διαφορετικά σύνολα συμπεριφορών. Τι μπορεί να συμβεί αν στο μεγάλο μας παιχνίδι, αντί να πάμε από τη σελίδα ένα στη σελίδα δύο, πάμε από τη σελίδα ένα στη σελίδα πέντε; Θα δημιουργηθεί κάποιο πρόβλημα στο παιχνίδι μας;

4. Περιγράψτε 5 παιχνίδια που μπορείτε να φτιάξετε χρησιμοποιώντας την ιδιότητα των ***Αρχέτυπων Αντικειμένων (creatable)***.

5. Στο παράδειγμα με τις μοτοσυκλέτες, μπορείτε να βρείτε άλλους τρόπους να λύσετε το πρόβλημα της υπερφόρτωσης εκτός από το να εκρήγνυνται οι μοτοσυκλέτες μετά από κάποιο χρονικό διάστημα;

Δραστηριότητες

1. Καλείστε να δημιουργήσετε ένα παιχνίδι με το γνωστό μας *Μηχανάκια (cycle)*. Ο κόσμος σας θα είναι γεμάτος με *Μπάλες (ball)*, *Νάρκες (mine)* και δύο *Κανόνια (cannon)*. Η πίστα θα είναι της επιλογής σας. Ο *Μηχανάκιας (cycle)* θα κινείται ευθεία με το γράμμα W του πληκτρολογίου, δεξιά με το γράμμα D, αριστερά με το γράμμα A, και πίσω με το γράμμα Z. Το πλήθος των *ναρκών (mine)* θα είναι διπλάσιο από το πλήθος των *μπαλών (ball)*. Στόχος του Μηχανάκια είναι να εξαφανίσει όλες τις *μπάλες (ball)* χωρίς να *πέσει πάνω* σε κάποια *νάρκη (mine)*. Όταν πέφτει πάνω σε κάποια *νάρκη (mine)* γίνεται πορτοκαλί και ακινητοποιείται για πέντε δευτερόλεπτα. Ταυτόχρονα τα *κανόνια (cannon)* μόλις δουν το *Μηχανάκια (cycle)* αρχίζουν και κινούνται προς το μέρος του. Το παιχνίδι τελειώνει είτε όταν ο μηχανάκιας μαζέψει όλες τις *μπάλες (ball)* είτε όταν κάποιο από τα κανόνια *πέσει πάνω* στο *Μηχανάκια (cycle)*.

2. Κατασκευάστε έναν κόσμο σαν αυτόν που φαίνεται στην επόμενη εικόνα. Τοποθετείστε στην μία πλευρά πέτρες και στην άλλη ένα μεγάλο χελιδονόψαρο (Fly Fish). Οι πέτρες θα παράγουν κλώνους ενός μικρού χελιδονόψαρου (Fly Fish) σε τυχαία χρονικά διαστήματα. Τα μικρά χελιδονόψαρα (Fly Fish) θα κινούνται προς την πλευρά του μεγάλου χελιδονόψαρου (Fly Fish) πολύ γρήγορα. Το μεγάλο χελιδονόψαρο (Fly Fish) θα κινείται με τα βελάκια και θα τρώει τα μικρά ψάρια συλλέγοντας πόντους. Σκοπός του παιχνιδιού είναι το μεγάλο χελιδονόψαρο (Fly Fish) να φάει τα μικρά μαζεύοντας όσους περισσότερους πόντους μπορεί, πριν αυτά πέσουν εκτός του κόσμου σε διάστημα 2 λεπτών. Προφανώς δε πρέπει να έχετε γυάλινο τοίχο στον κόσμο σας.

Κεφάλαιο 10ο: Από τη φαντασία στην υλοποίηση-σχεδίαση παιχνιδιών

Μέσα από το σύντομο αυτό κεφάλαιο, θα προσπαθήσουμε να ανακαλύψουμε μαζί ποια είναι τα «μυστικά συστατικά» που έχουν τα παιχνίδια και μας κάνουν να μη μπορούμε να τους αντισταθούμε! Με άλλα λόγια, θα δούμε ποιες είναι οι βασικές αρχές που πρέπει να έχετε στο νου σας για να σχεδιάσετε ένα ενδιαφέρον και ελκυστικό παιχνίδι για εσάς και τους φίλους σας. Για το λόγο αυτό, θα παρουσιάσουμε κάποια στοιχεία από τη «Μηχανική Παιχνιδιών» («Game Mechanics»), ένα πεδίο έρευνας που μπορεί να σας βοηθήσει να αποφασίσετε πώς να δομήσετε το παιχνίδι σας, πώς να σχεδιάσετε τους βασικούς κανόνες, τους απαραίτητους στόχους και γενικότερα πώς να ικανοποιήσετε ψυχολογικά χαρακτηριστικά των παικτών, ώστε να «χαθούν» στο μοναδικό μαγικό κόσμο των παιχνιδιών σας!

10.1 Βασικές αρχές οργάνωσης και σχεδιασμού των παιχνιδιών

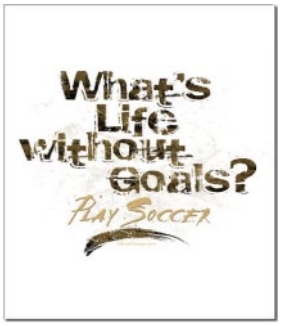
Τι θέμα θα έχει το παιχνίδι μας; Ποια θα είναι η δομή και η μορφή του κόσμου μας; Πόσους χαρακτήρες θα έχουμε; Τι ιδιότητες θα έχει καθένας τους; Ποια αντικείμενα θα κοσμούν τον κόσμο μας; Θα εισάγουμε περιορισμούς χρόνου ώστε να αυξάνεται η αγωνία; Θα απαρτίζεται το παιχνίδι από επιμέρους επίπεδα ώστε να γίνει και πιο ανταγωνιστικό; Ποιος θα είναι ο τελικός στόχος του παίκτη;

Τα ερωτήματα που καλούμαστε να απαντήσουμε κάθε φορά που σχεδιάζουμε ένα παιχνίδι είναι πολλά και ανάλογα με το τι είμαστε διατεθειμένοι να υλοποιήσουμε, καλούμαστε να ακολουθήσουμε και ένα διαφορετικό μονοπάτι σχεδίασης.

Πριν αρχίσετε να σχεδιάζετε το παιχνίδι σας, καλό είναι να σκεφτείτε προσεκτικά τα εξής:

- *Τι είδος παιχνιδιού θα είναι;* Θα είναι παιχνίδι ψυχαγωγίας, εκπαιδευτικού περιεχομένου, βίας και δράσης, αθλητικού περιεχομένου;
- *Σε ποιες κατηγορίες παικτών θα απευθυνθείτε;* Θα στοχεύσετε στους «εξερευνητές» που λατρεύουν να ανακαλύπτουν κρυφά πράγματα και να πιέζονται για να σκεφτούν έξυπνα όταν υπάρχει συγκεκριμένο χρονικό όριο; Σε αυτούς που επιθυμούν να απαντούν διαρκώς σε νέες προκλήσεις, να συλλέγουν όσο το δυνατόν περισσότερους πόντους, και να προχωρούν σε επόμενα επίπεδα δυσκολίας; Στους «ανταγωνιστές» που τους αρέσει να συγκρίνουν τις δεξιότητές τους με άλλα άτομα και ικανοποιούνται περισσότερο όταν τους ξεπεράσουν;

Τα δυο αυτά ερωτήματα θα σας επιτρέψουν να αποκτήσετε ένα γενικό προσανατολισμό για το σχεδιασμό του παιχνιδιού σας, θα σας επιτρέψουν να σχηματίσετε μια γενική εικόνα. Αμέσως μετά, μελετήστε προσεκτικά τα επόμενα 12 σημεία-κλειδιά που έχουν συνήθως τα επιτυχημένα παιχνίδια και προσπαθήστε να τα αξιοποιήσετε στο δικό σας:



[1. Επιτεύγματα] Αν θέλετε να δεσμεύσετε το χρήστη με το παιχνίδι σας, αν θέλετε να νιώσει ικανοποίηση για ότι έχει καταφέρει μέχρι στιγμής, αν θέλετε να τον προκαλέσετε να συνεχίσει, τότε θα πρέπει να βάλετε στο παιχνίδι σας διαφορετικά *«Επιτεύγματα» (Achievements)*. Τα επιτεύγματα είναι μορφές αναγνώρισης των επιτυχιών του χρήστη σε συγκεκριμένες δοκιμασίες. Άλλωστε, όλα τα παιχνίδια αποτελούνται από μια σειρά από δοκιμασίες. Μη ξεχνάτε όμως ότι τα επιτεύγματά σας πρέπει:

- να είναι κατανοητά,
- να εξάπτουν την περιέργεια του παίκτη,
- να μην είναι εύκολα στην επίτευξή τους, ώστε να δημιουργούν αγωνία,
- να διακρίνονται από εξυπνάδα και χιούμορ,
- να είναι διαφορετικών τύπων ώστε να ικανοποιούν όλες τις κατηγορίες παικτών.

Κάθε παιχνίδι πρέπει να έχει ένα κύριο επίτευγμα και συνήθως θα περιέχει και άλλα ενδιάμεσα επιτεύγματα. Στο Super Mario, για παράδειγμα, επίτευγμα είναι να καταφέρει ο Super Mario να σώσει την πριγκίπισσα μέσα στον προκαθορισμένο χρόνο και με τη χρήση των «ζωών» που έχει

στη διάθεσή του ο παίκτης. Στο ProEvolution, το κύριο επίτευγμα είναι να καταφέρει ο παίκτης να επικρατήσει του συμπαίκτη του στο τελικό σκορ, βάζοντας περισσότερα γκολ από τον δεύτερο.

[2. Κανόνες] Σε κάθε παιχνίδι απαιτείται προσθήκη «κανόνων», διότι με τους κανόνες μπορείτε να ορίσετε τις δυνατότητες και τα όρια των παικτών μέσα στο παιχνίδι. Είναι γνωστό στους περισσότερους από εσάς ότι οι κανόνες είναι βασικό συστατικό της περιγραφής του παιχνιδιού και πρέπει να είναι ξεκάθαροι και σαφείς σε όλους, δίκαιοι και ίδιοι για κάθε χρήστη και να είναι δομημένοι με τρόπο που να κάνουν το παιχνίδι ελκυστικό. Στον Γκρινιάρη για παράδειγμα ένας κανόνας είναι ότι κάθε παίκτης έχει 4 πιόνια ίδιου χρώματος, τα οποία τοποθετούνται στη δική του βάση. Για να βγει κάθε πιόνι από τη βάση, πρέπει ο παίκτης να φέρει 6 στο ζάρι.

[3. Επιβραβεύσεις] Οι επιβραβεύσεις (ή μπόνους) έρχονται συνήθως απρόβλεπτα στους παίκτες και τους ξαφνιάζουν θετικά. Δίνονται όταν πετύχουν μια σειρά από προκλήσεις ή κατακτήσουν κάποια αντικείμενα. Ως επιβραβεύσεις μπορούν να θεωρηθούν οι επιπλέον βαθμοί ή μία επιπλέον ζωή ή μια ενδιαφέρουσα παροιμία! Παράδειγμα επιβράβευσης στο SuperMario είναι όταν συλλέξει ο ήρωας το αντικείμενο «λουλούδι», οπότε και τροφοδοτείται με σφαίρες/μπάλα.

[4. Βαθμολογία] Είναι μια αριθμητική ανταμοιβή που λαμβάνει κάθε παίκτης για μια δράση που κατάφερε να φέρει εις πέρας. Η βαθμολογία πρέπει να διαφοροποιείται ανάλογα με τη δυσκολία των ενεργειών. Μια πιο δύσκολα υλοποιήσιμη ενέργεια πρέπει να ανταμείβεται με πιο πολλούς πόντους απ' ότι μια εύκολη. Έτσι, οι παίκτες θα παρακινούνται να επιμένουν εντονότερα για να ξεπεράσουν τις συγκεκριμένες πιο δύσκολες προκλήσεις.

[5. Δράση – Κίνδυνοι] Όταν προσθέσετε ή τονίσετε το στοιχείο του κινδύνου ή της δράσης στο παιχνίδι σας, το παιχνίδι σας θα γίνει πιο συναρπαστικό και διασκεδαστικό για τον παίκτη. Η δράση για παράδειγμα τονίζεται στο Pacman με τα Φαντασματάκια να τρεμοπαίζουν όταν βρίσκονται στο όριο του να επανέλθουν στην απειλητική-κανονική τους κατάσταση που μπορούν να εξολοθρεύσουν τον πρωταγωνιστή του παιχνιδιού, ενώ την εκείνη τη στιγμή ισχύει το αντίστροφο.

[6. Ανακάλυψη – εξερεύνηση] Είναι καλό να δίνετε τη δυνατότητα στους παίκτες να ανακαλύπτουν σε απρόσμενες χρονικές στιγμές πράγματα που θα τους εκπλήξουν. Δώστε τους δηλαδή τη δυνατότητα να ανακαλύψουν κατά τη διάρκεια του παιχνιδιού νέους αντιπάλους, νέες λειτουργίες του παιχνιδιού, νέα επίπεδα, νέες προκλήσεις, νέες δυνατότητες του πρωταγωνιστή. Στους αγώνες αυτοκινήτων, για παράδειγμα, η μέγιστη ταχύτητα του κάθε οχήματος πιστεύεις ότι είναι συγκεκριμένη. Κατά τη διάρκεια του παιχνιδιού ωστόσο, μπορεί να διαπιστώσεις ότι η συλλογή κάποιου συγκεκριμένου αντικειμένου, την ώρα που βρίσκεται σε εξέλιξη η κούρσα, αυξάνει κι άλλο τη μέγιστη ταχύτητα με την οποία μπορεί να κινηθεί το όχημά σου.

[7. Εξέλιξη/Πρόοδος] Σε κάθε παιχνίδι πρέπει να υπάρχει ορατή πρόοδος για τον παίκτη σε όλη τη διάρκειά του. Η πρόοδος του παίκτη μπορεί να αποκαλύπτεται από την εμφάνιση του σκορ του, του επιπέδου δυσκολίας που βρίσκεται, των αντικειμένων που έχει συλλέξει, των εχθρών που του απομένουν κτλ. Ο χρήστης με τον τρόπο αυτό αντιλαμβάνεται διαρκώς σε ποια φάση του παιχνιδιού βρίσκεται, καταλαβαίνει τι έχει καταφέρει κάθε στιγμή.

[8. Τύχη] Δεν παίζετε σχεδόν ποτέ ένα παιχνίδι μια φορά μόνο. Κάθε φορά όμως αυτό εμφανίζεται ελαφρώς διαφορετικό. Δεν θα ήταν πολύ βαρετό να είναι απολύτως προβλέψιμη η πλοκή την επόμενη φορά που θα ξαναπαίξετε το παιχνίδι; Για αυτό το λόγο, προσπαθήστε να εισάγετε τον παράγοντα της τύχης μέσα στα παιχνίδια σας. Για παράδειγμα, τοποθετήστε με τυχαίο τρόπο τους αντιπάλους του παίκτη ή/και επιτρέψτε τους να κινούνται και να πυροβολούν με τυχαίο τρόπο. Με τον τρόπο αυτό το παιχνίδι σας θα είναι καινούριο διαρκώς για τους παίκτες σας ανεξάρτητα με το πόσες φορές το έχουν ξαναπαίξει.

[9. Χρονικά όρια] Εισάγετε χρονικούς περιορισμούς στο παιχνίδι σας. Αν οι περιορισμοί που έχετε θέσει ξεπεραστούν, τότε ο παίκτης συνήθως χάνει μια ζωή ή κάποιο πλεονέκτημα που απέκτησε κατά τη διάρκεια του παιχνιδιού. Η έννοια του χρόνου προσδίδει μεγαλύτερη αγωνία. Ο παίκτης αναγκάζεται να σκέφτεται και να αντιδρά γρήγορα, είναι συγκεντρωμένος απόλυτα στην εκπλήρωση του στόχου του, και πρέπει να αναγνωρίζει προσεκτικά τα λάθη του ώστε την επόμενη φορά που θα ξαναπαίξει το παιχνίδι να προλάβει να τερματίσει το παιχνίδι πριν τελειώσει ο χρόνος.

[10. Επίπεδα δυσκολίας] Με τα επίπεδα δυσκολίας οι παίκτες ανταμείβονται για την πορεία τους σταδιακά. Έχει αποδειχθεί ότι η ύπαρξη αυτού του στοιχείου στα παιχνίδια αποτελεί ένα πολύ καλό κίνητρο τόσο για τους παίκτες, όσο και για τους ίδιους τους δημιουργούς του, καθώς τους

βοηθάει να σπάσουν το παιχνίδι σε μικρά επιμέρους κομμάτια, δίνοντας στον παίκτη ενδιάμεσα επιτεύγματα. Στους μηχανοκίνητους αγώνες, για παράδειγμα, κάθε επίπεδο έχει και μία διαφορετική πίστα στην οποία αγωνίζεσαι. Όσο υψηλότερο είναι το επίπεδο δυσκολίας, τόσο πιο απαιτητική είναι και η διαδρομή της πίστας.

[11. Καταστάσεις πρωταγωνιστή] Προσθέστε στο παιχνίδι σας πολλές «Καταστάσεις» για τους πρωταγωνιστές. Οι δυνατότητες του πρωταγωνιστή σας ή ακόμη και τα φυσικά του χαρακτηριστικά μπορεί να αλλάζουν κατά τη διάρκεια του παιχνιδιού. Στο Super Mario για παράδειγμα, υπάρχουν αρκετές διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί ο παίκτης. Μπορεί να είναι ψηλός, κοντός, με σφαίρες, χωρίς σφαίρες κ.λπ.

[12. Τιμωρία] Δεν θα είναι λίγες οι φορές που όταν συλλέξατε το λάθος αντικείμενο, χάσατε μια ζωή. Δεν είναι πάντα αρνητικό το αποτέλεσμα της τιμωρίας. Η τιμωρία επισημαίνει ένα λάθος και επιτρέπει στον παίκτη να εστιάσει στα πράγματα που πρέπει να αποφύγει.

Λογικά τώρα θα αναρωτιέστε: «μήπως θα μπορούσατε να μας δώσετε ένα λίγο πιο συγκεκριμένο παράδειγμα παιχνιδιού που αξιοποιεί τα παραπάνω στοιχεία»;

Ας εξετάσουμε τη «MONOPOLY» που είναι ένα από τα παιχνίδια που πιθανότατα όλοι μας έχουμε παίξει κάποια στιγμή, τουλάχιστον μία φορά. Πως εφαρμόζει ορισμένα από τα στοιχεία-κλειδιά που αναλύσαμε;

Επίτευγμα/στόχος: Ο κάθε παίκτης πρέπει να αποκτήσει όσο το δυνατόν περισσότερα χρήματα μπορεί, αγοράζοντας, νοικιάζοντας και πουλώντας οικόπεδα και κτίρια, διότι τελικά κερδίζει ο παίκτης με τα περισσότερα χρήματα.

Επιβραβεύσεις: Ένας ιδιοκτήτης οικοπέδου, αν έχει όλες τις κάρτες με το ίδιο χρώμα, τότε εισπράττει διπλάσιο ενοίκιο από το κανονικό από κάθε παίκτη που θα σταθεί στο τετράγωνο αυτό.

Βαθμολογία: Σχετίζεται άμεσα με τα χρήματα που συγκεντρώνει ο κάθε παίκτης. Αρχικά ξεκινάει ο κάθε παίκτης με 1000€ για παράδειγμα. Στη συνέχεια, κάθε παίκτης λαμβάνει χρήματα (βαθμολογία) από οποιονδήποτε άλλον παίκτη που στέκεται στο οικόπεδό του. Ακόμη, χρήματα μπορεί να λάβει ο κάθε παίκτης από την "Πώληση Ακινήτων" που έχει στην κατοχή του.

Δράση-Κίνδυνος: Στο παιχνίδι της MONOPOLY υπάρχει ένας και βασικός κίνδυνος, η χρεωκοπία όταν θα κληθεί ο παίκτης να πληρώσει ένα ποσό που δεν έχει στη διάθεσή του. Ο κίνδυνος αυξάνεται με το πέρασμα του παιχνιδιού, καθώς οι παίκτες χτίζουν τα οικόπεδά τους και τα ενοίκια αυξάνονται δραματικά. Έτσι, ο χρόνος κυλάει ευχάριστα καθώς η δράση διαρκώς αυξάνεται.

Πρόοδος: Ακόμη και στη MONOPOLY υπάρχουν εμφανή σημάδια της προόδου των παικτών! Οι παίκτες παρατηρούν τη στοίβα των χρημάτων που κατέχουν οι αντίπαλοί τους, το πλήθος των καρτών που κρατούν καθώς και το πλήθος των ξενοδοχείων που έχουν χτίσει. Αυτό δημιουργεί ποικιλία συναισθημάτων: Κάποιοι νιώθουν ικανοποίηση και σιγουριά για την πορεία τους ενώ ταυτόχρονα άλλοι παίκτες αισθάνονται αγχωμένοι και προσπαθούν να ανατρέψουν την κατάσταση.

Τύχη: Ο παράγοντας τύχη υπάρχει αφού έχουμε στο παιχνίδι τα ζάρια. Κάθε φορά που πετάτε τα ζάρια, προχωράτε με το πιόνι σας θέσεις ίσες με το άθροισμα των ζαριών και καταλήγετε σ' ένα τετράγωνο όπου μπορεί να χρειαστεί να πληρώσετε νοίκι ή φόρους ή θα μπορείτε να αγοράσετε το οικόπεδο. Μπορεί επίσης να πέσετε σε τετράγωνο που να επιβάλει να τραβήξετε κάρτα (Εντολή ή Απόφαση). Πριν όμως ρίξετε το ζάρι, δεν ξέρετε τι σας περιμένει.

Τιμωρία: Τιμωρία στη MONOPOLY είναι η φυλακή. Στη φυλακή οδηγείται ένας παίκτης σε 3 περιπτώσεις: α) Όταν το πιόνι του σταματήσει στο τετράγωνο με τον αστυφύλακα που γράφει «ΠΗΓΑΙΝΕ ΣΤΗ ΦΥΛΑΚΗ» β) Όταν τραβήξει μια κάρτα (Εντολή ή Απόφαση) που γράφει «ΠΗΓΑΙΝΕ ΣΤΗ ΦΥΛΑΚΗ» γ) εάν ρίξει διπλές τρεις φορές στη σειρά.

Παρατηρήστε πως κάθε ένα από τα παραπάνω στοιχεία παίζει το δικό του ιδιαίτερο ρόλο για να γίνει το παιχνίδι πραγματικά διασκεδαστικό, κάθε ένα από τα παραπάνω στοιχεία συνεισφέρει στο να διατηρεί τον παίκτη σε εγρήγορση, να του προκαλεί αγωνία, να κρατά αμείωτο το ενδιαφέρον του. Συνεπώς για να δημιουργήσουμε ένα παιχνίδι, αρχικά πρέπει να συλλάβουμε τη βασική δομή του και στη συνέχεια να το εμπλουτίσουμε με όσο το δυνατόν περισσότερο από τα παραπάνω στοιχεία.

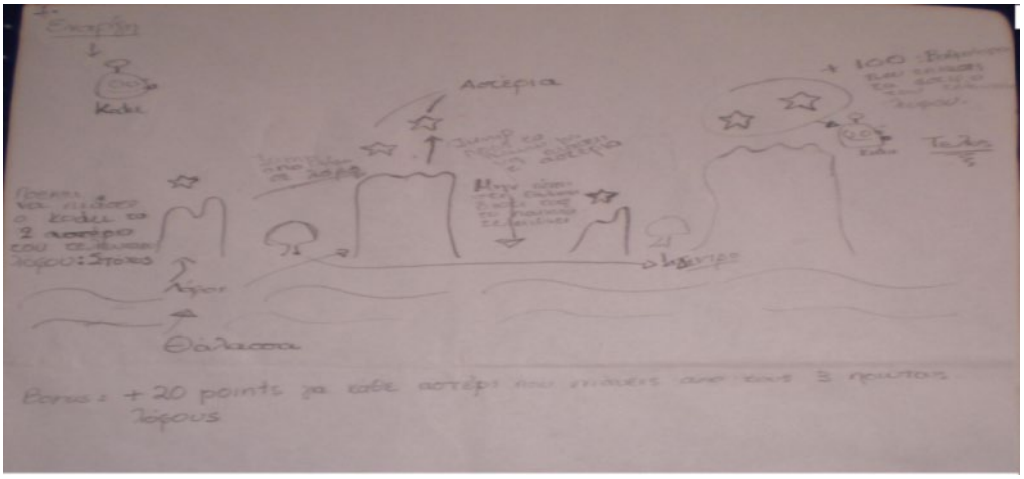
10.2 Δημιουργώ τα παιχνίδια μου στο MSKodu

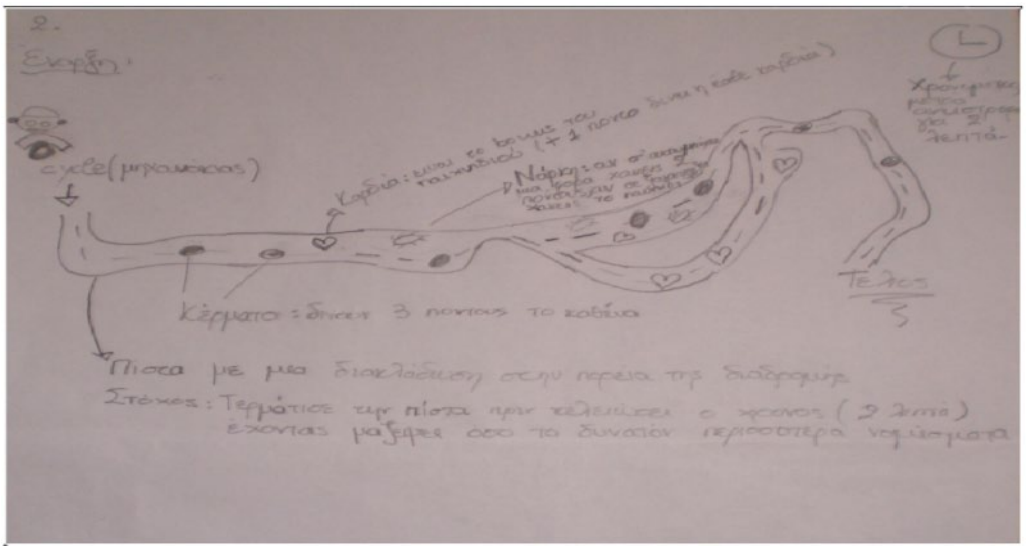
Πολλοί προγραμματιστές θεωρούν ότι για να αναπτύξουν ένα παιχνίδι, μπορούν να βασιστούν αρχικά σε μια γενική ιδέα και σιγά-σιγά να την αναπτύξουν στο MSKodu. Η τακτική αυτή δε θα υποστηρίξουμε πως είναι λάθος, ωστόσο θα λέγαμε πως δεν είναι και η πιο αποδοτική. Μπορεί κατά την πορεία υλοποίησης να προκύψουν διάφορα ζητήματα που δεν μπορέσαμε να προβλέψουμε εκ των προτέρων και να αναγκαστούμε να αλλάξουμε το παιχνίδι μας σε μεγάλο βαθμό, εάν όχι ολοκληρωτικά. Στον προγραμματισμό πάντα συστήνεται:

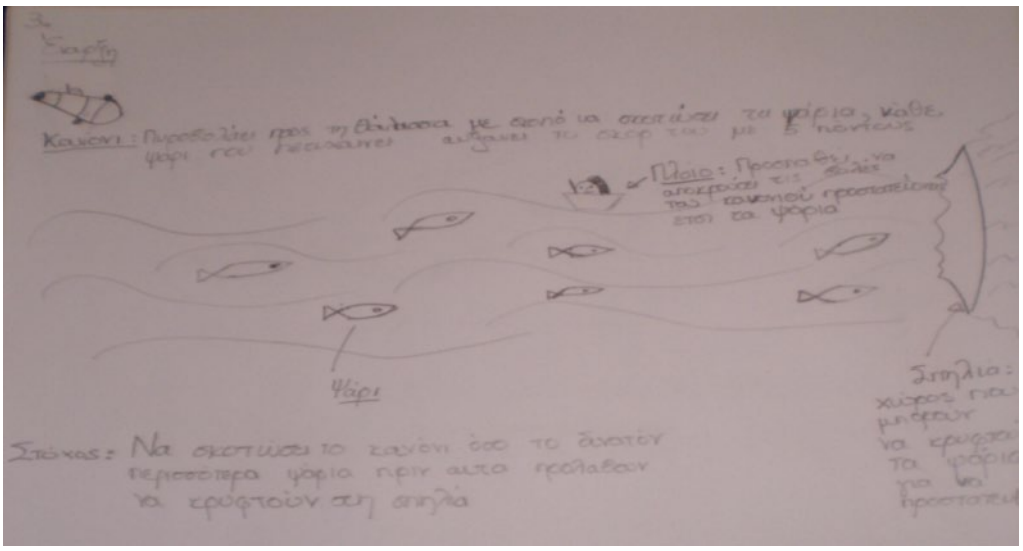
πρώτα σχεδίασε τη λύση, μετά υλοποίησέ την

Μια πιο ασφαλής μέθοδος είναι να σχεδιάσω πρώτα το παιχνίδι μου στο χαρτί, να προδιαγράψω τη μορφή του κόσμου μου, να αναγνωρίσω τα διάφορα αντικείμενά μου και τις ιδιότητές τους, να αποφασίσω προσεκτικά για τις συμπεριφορές τους και γενικά να δημιουργήσω μια ολοκληρωμένη εικόνα του τι θα υλοποιήσω. Με τον τρόπο αυτό, στη φάση υλοποίησης του παιχνιδιού, το ζητούμενό μας θα είναι συγκεκριμένο και θα αναφέρεται στο πώς θα αναπτύξουμε τις επιδιωκόμενες συμπεριφορές με βάσει τις διαθέσιμες εντολές του MSKodu. Δεν είναι εύκολο να προσπαθούμε να σχεδιάζουμε ταυτόχρονα τις συμπεριφορές και το παιχνίδι.

Ας δούμε μερικά παραδείγματα πιθανών πλοκών. Παρακάτω παρουσιάζονται τρία διαφορετικά σενάρια που σχεδιάσαμε στο χαρτί.



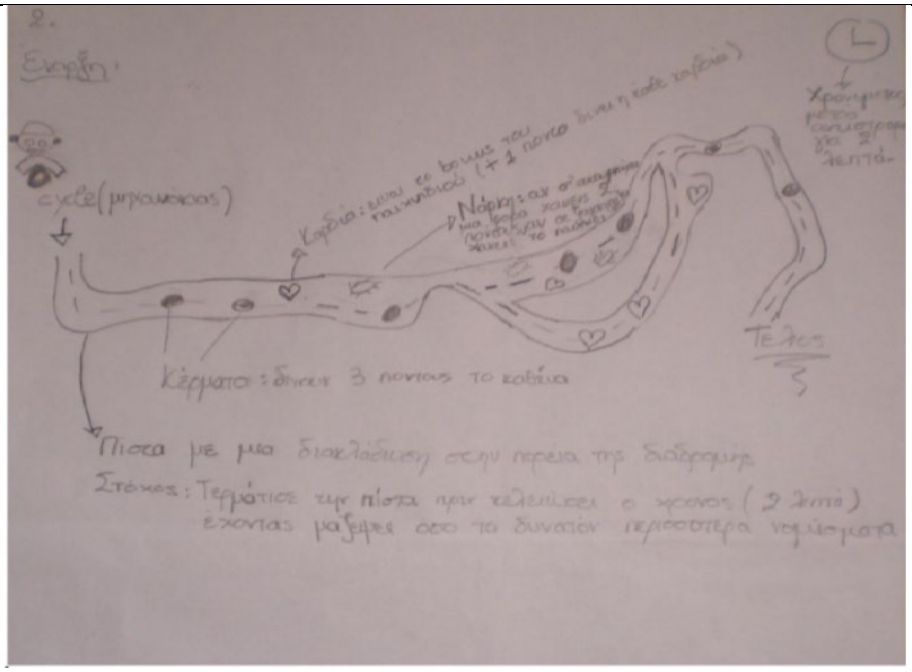




Στα παραπάνω τρία σκίτσα φτιάξαμε στο χαρτί το βασικό κόσμο των παιχνιδιών μας. Σχεδιάσαμε δηλαδή τον άξονα γύρω απ' τον οποίο θα κινηθούμε. Αξίζει να παρατηρήσουμε πως οι παραπάνω κόσμοι είναι αρκετά απλοϊκοί, έχουμε ωστόσο τη δυνατότητα να εμπλουτίσουμε και να επεκτείνουμε τις δυνατότητές τους. Ανάλογα, μπορεί ο καθένας από εσάς να κάνει μια πρόχειρη κατασκευή του κόσμου που θέλει ο ίδιος να υλοποιήσει στο MSKodu και στη συνέχεια να συνθέσει βήμα-βήμα όλα τα επιμέρους στοιχεία ώστε το δικό του προσωπικό παιχνίδι να είναι πλέον γεγονός!

Ας υποθέσουμε πως θέλουμε να κάνουμε ένα παιχνίδι και πως καταλήξαμε στην πλοκή που παρουσιάζεται στο δεύτερο από τα παραπάνω σκίτσα. Πάμε λοιπόν να φτιάξουμε το παιχνίδι μας!

ΣΧΕΔΙΑΣΜΟΣ ΤΟΥ ΚΟΣΜΟΥ ΣΟΥ ΚΑΙ ΤΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΤΟΥ



Αφού έχουμε μια πρώτη εικόνα για το παιχνίδι μας, πρέπει να αναγνωρίσουμε προσεκτικά ποια είναι τα αντικείμενά μας και ποια θα είναι η λειτουργία τους στο παιχνίδι μας

ΠΕΡΙΕΓΡΑΨΕ ΤΟ ΠΑΙΧΝΙΔΙ ΣΟΥ	ΠΟΙΑ ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ ΣΟΥ	ΠΟΙΕΣ ΟΙ ΙΔΙΟΤΗΤΕΣ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ
Ο Μηχανάκιας πρέπει σε συγκεκριμένο χρονικό διάστημα να διασχίσει μια πίστα από την αρχή μέχρι το τέλος συλλέγοντας όσο το δυνατόν περισσότερους πόντους.	1. Κέρματα	1. Κέρματα: δίνουν 3 πόντους το καθένα.
	2. Καρδούλες	2. Καρδούλες: δίνουν 1 πόντο η μία.
	3. Νάρκες	3. Νάρκες: αφαιρούν 2 πόντους από τη βαθμολογία σου.
	4. Δρόμος	4. Δρόμος: εάν βγεις εκτός ορίων μειώνεται η ταχύτητά σου.
	5. Μηχανάκιας	5. Μηχανάκιας: το χειρίζεται ο χρήστης με τα βελάκια.

Επιλέξαμε το *Μηχανάκια (Cycle)* για κεντρικό ήρωα και αποφασίσαμε πως τα αντικείμενα που θα συλλέγει θα είναι κέρματα. Για κάθε *Κέρμα (Coin)* που θα καταφέρει να συλλέξει θα προστίθενται 3 πόντοι στη συνολική βαθμολογία του. Με τη χρήση των πόντων, διαφορετικοί χρήστες θα μπορούν να συγκρίνουν τις επιδόσεις τους και έτσι να εισαχθεί και το στοιχείο του συναγωνισμού.

Και η πίστα; Με μια πρώτη σκέψη θέλαμε ο ήρωας μας να συλλέγει τα Κέρματα απλά από ένα χώρο όπου τα τελευταία θα ήταν διασκορπισμένα. Με μία δεύτερη όμως σκέψη αποφασίσαμε πως είναι καλύτερο να φτιάξουμε μία πίστα δρόμου πάνω στην οποία θα κινείται με σκοπό να τον δυσκολέψουμε λίγο, καθώς για να μαζεύει τα Κέρματα θα πρέπει να ακολουθεί συγκεκριμένη διαδρομή και να κάνει επομένως τους απαραίτητους ελιγμούς. Ωραία! Ο Μηχανάκιας πρέπει να ακολουθήσει μία συγκεκριμένη διαδρομή για να μαζέψει τα κέρματα και τελικά ξεκινώντας από την έναρξη να φτάσει στο τέλος! Είναι όμως αυτό αρκετό; Μάλλον όχι... Ανιαρό και άχρωμο... Πρέπει να το εμπλουτίσουμε.

Μία καλή ιδέα είναι να προσθέσουμε χρόνο στο παιχνίδι μας. Έτσι ο παίκτης θα έχει την αγωνία για το αν θα προλάβει να φτάσει στον προορισμό του μέσα στον προβλεπόμενο χρόνο. Και πάλι όμως αυτό το στοιχείο από μόνο του δεν είναι αρκετό. Είναι πιθανόν η πίστα μας να είναι αρκετά εύκολη και σε κάθε περίπτωση ο χρόνος να είναι υπέρ αρκετός. Μήπως θα ήταν καλό να έχουμε έναν παράγοντα που να κάνει ακόμη πιο απαιτητικό το παιχνίδι; Έτσι και αποφασίσαμε ότι όταν ο Μηχανάκιας βγαίνει εκτός των ορίων του δρόμου, η ταχύτητά του να ελαττώνεται! Τώρα είμαστε καλύτερα απ' ότι ήμασταν πριν!

Μία πίστα όμως χωρίς κάποιο ουσιαστικό εμπόδιο, δε λέει και κάτι. Σωστά; Μας ήρθε επομένως η ιδέα να τοποθετήσουμε σε διάφορα διάσπαρτα σημεία της διαδρομής Νάρκες! Όταν ο Μηχανάκιας έρθει σε επαφή με κάποια *Νάρκη (Mine)* θα χάνει 2 από τους πόντους του και έτσι η βαθμολογία του θα ελαττώνεται! Με την τοποθέτηση των Ναρκών δεν εισάγαμε μόνο πιθανή ποινή για το Μηχανάκια αλλά κάναμε και το έργο του πιο δύσκολο, καθότι στην προσπάθειά του να αποφύγει κάποια Νάρκη, πιθανόν να χάνει την ευκαιρία να συλλέξει κάποιο κέρμα. Ωραία λοιπόν, βρήκαμε δύο τρόπους να δυσκολέψουμε λίγο τη "ζωή" του ήρωά μας.

Ένας παίκτης όμως δεν πρέπει μόνο να τιμωρείται. Πρέπει και να επιβραβεύεται! Πρέπει επομένως να δώσουμε στο Μηχανάκια μας ένα κίνητρο για να συνεχίσει να προσπαθεί. Σκεφτήκαμε λοιπόν να τοποθετήσουμε στην αγωνιστική πίστα καρδούλες! Η κάθε *Καρδούλα (Heart)* ευεργετεί το Μηχανάκια με επιπλέον 1 πόντο, εάν φυσικά τη συλλέξει! Η διαδρομή τώρα δεν του επιφυλάσσει μόνο δυσάρεστες εκπλήξεις, αλλά και ευχάριστες. Μένει τώρα να προσθέσουμε κάτι ακόμη στο παιχνίδι μας έτσι ώστε να έχουμε ανατροπή! Σκεφτήκαμε πως το παιχνίδι μας θα δυσκολέψει αρκετά εάν τη δεύτερη φορά που ο Μηχανάκιας ακουμπήσει Νάρκη, δεν χάνει απλά άλλους 2 πόντους από τη συνολική του βαθμολογία, αλλά θα επιστρέφει στην αρχή της πίστας και θα είναι υποχρεωμένος να ξανακάνει όλη τη διαδρομή από την αρχή, χωρίς φυσικά ο χρόνος του να ανανεωθεί.

Επιπλέον, αποφασίσαμε στην κύρια διαδρομή να δώσουμε κάποιους παράδρομους ώστε να έχει ο παίκτης δυνατότητα επιλογής. Θα έχουμε ένα παράδρομο πιο μεγάλο, αλλά πιο εύκολο (δηλαδή με μικρότερο ρίσκο) και έναν παράδρομο πιο σύντομο, αλλά πιο δύσκολο (δηλαδή πιο ριψοκίνδυνη διαδρομή) και περισσότερους διαθέσιμους πόντους. Επομένως, ο παίκτης του παιχνιδιού θα έχει τη δυνατότητα να εξερευνήσει τις διάφορες διαδρομές και να δοκιμάσει τις

εναλλακτικές που του δίνονται. Το ουσιαστικό κομμάτι του παιχνιδιού μας ολοκληρώθηκε! Μπορείτε να εξετάσετε τη λίστα με τα κύρια χαρακτηριστικά που συζητήσαμε προηγουμένως και να συμπληρώσετε ποια από τα στοιχεία-κλειδιά των παιχνιδιών έχει το συγκεκριμένο παιχνίδι; Γιατί;

ΣΤΟΙΧΕΙΑ - ΚΛΕΙΔΙΑ	**ΤΟ ΠΑΙΧΝΙΔΙ ΜΟΥ ΕΧΕΙ**	**ΓΙΑΤΙ;**
1. Επίτευγμα		**….**
2. Κανόνας		
3. Επιβραβεύσεις		
4. Βαθμολογία		
5. Κίνδυνος/Δράση		
6. Ανακάλυψη/Εξερεύνηση		
7. Εξέλιξη/Πρόοδος		
8. Τύχη		
9. Χρονικό Περιορισμό		
10. Επίπεδα		
11. Ενδιάμεσα Επιτεύγματα		
12. Τιμωρία		

Παρατηρήστε ότι μόλις συμπληρώσαμε τον πίνακα, νέες ιδέες αρχίζουν να προκύπτουν. Πως θα μπορούσαμε να προσθέσουμε το στοιχείο της τυχαιότητας; Ίσως να κινούνται οι καρδούλες με τυχαίο τρόπο; Θα μπορούσε ο Μηχανάκιας να αποκτά νέες καταστάσεις π.χ. να έχει τη δυνατότητα να πυροβολεί τις νάρκες; Συνεπώς, μόλις δημιουργούμε την πρώτη αποτύπωση του παιχνιδιού μας, είναι σημαντικό να εξετάζουμε αναλυτικά την προηγούμενη λίστα με τα στοιχεία-κλειδιά ώστε να προσθέσουμε νέα χαρακτηριστικά.

Από όλα όσα αναφέρθηκαν παραπάνω, μπορούμε να καταλήξουμε με ασφάλεια στο συμπέρασμα ότι το δύσκολο δεν είναι να σκεφτούμε ένα παιχνίδι, αλλά να το υλοποιήσουμε με τέτοιον τρόπο ώστε να γίνει θελκτικό και ενδιαφέρον για αυτούς που θα το παίξουν. Αξίζει ωστόσο να σημειωθεί πως εμείς σε αυτή την υποενότητα του κεφαλαίου μας επιλέξαμε να υλοποιήσουμε ένα αρκετά απλό παιχνίδι έτσι ώστε να μπορέσουμε πιο εύκολα να επισημάνουμε τα σημεία που θέλαμε και κρίναμε απαραίτητα. Τώρα είσαστε σε θέση και εσείς, ο καθένας προσωπικά, να δοκιμάσει να υλοποιήσει ένα παιχνίδι ολοκληρωμένο και να μπορέσει να του δώσει όσο το δυνατόν περισσότερα στοιχεία-κλειδιά.

Αξίζει να σημειώσουμε πως ο "στολισμός" του κόσμου σας είναι σημαντικός, καθώς ευχαριστεί το μάτι του παίκτη και τον προδιαθέτει ευχάριστα. Αφού ολοκληρώσετε την υλοποίηση του παιχνιδιού σας, προσπαθήστε να βελτιώσετε την αισθητική του αλλάζοντας τη μορφή της πίστας αλλά και προσθέτοντας επιπλέον διακοσμητικά αντικείμενα που θα κάνουν πιο αληθοφανή το σενάριό σας.

Περίληψη Κεφαλαίου

Στο κεφάλαιο σας δείξαμε σύντομα ποια είναι τα πιο σημαντικά στοιχεία που πρέπει να λαμβάνουμε υπόψη στο παιχνίδι που σκοπεύουμε να υλοποιήσουμε. Η «Μηχανική Παιχνιδιών» προσδιορίζει πολλά στοιχεία-κλειδιά που διαφοροποιούνται ανάλογα με τον τύπο παιχνιδιού που θέλουμε να φτιάξουμε αλλά και ανάλογα με την κατηγορία ή τις κατηγορίες των παικτών που θέλουμε να προσελκύσουμε. Μη ξεχνάτε ότι είναι σημαντικό πρώτα να σχεδιάσετε το γενικό περίγραμμα του παιχνιδιού σας στο χαρτί, να προσδιορίσετε τα αντικείμενα και τις συμπεριφορές τους και στη συνέχεια να υλοποιείτε το παιχνίδι στο MSKodu. Πρώτα σχεδιάζουμε και μετά υλοποιούμε.

Ερωτήσεις

1. Ποια είναι τα 12 στοιχεία κλειδιά που πρέπει να λαμβάνουμε υπόψη μας κατά τη δημιουργία ενός παιχνιδιού;

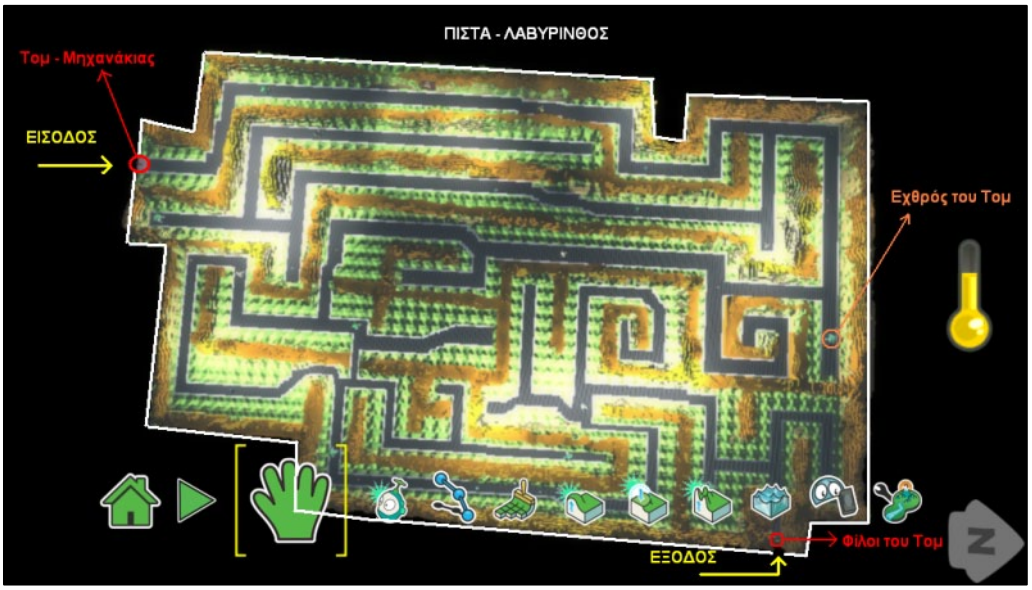
2. Ποια από τα 12 στοιχεία κλειδιά μπορείτε να αναγνωρίσετε σε γνωστά παιχνίδια, όπως το Pacman και το Age of Empires;

3. Μπορείτε να προτείνετε επιπλέον στοιχεία που κάνουν τα παιχνίδια πιο ενδιαφέροντα; Ψάξτε στο διαδίκτυο για «Game Mechanics».

Κεφάλαιο 11ο: Mission Impossible

11.1 Περιγραφή

Οι φίλοι του Τομ είναι φυλακισμένοι και αγωνιούν για την απελευθέρωσή τους. Ένας λαβύρινθος με πολλά εμπόδια και δυσκολίες τους χωρίζουν. Ο Τομ ξεκινώντας από την μία πλευρά του λαβύρινθου θα πρέπει να βρει τον κατάλληλο δρόμο που θα τον οδηγήσει στη άλλη άκρη της πίστας που βρίσκονται φυλακισμένοι οι φίλοι του. Σε αυτή του την προσπάθεια, όμως θα έχει να αντιμετωπίσει εχθρούς και εμπόδια που θα κάνουν την προσπάθειά του ακόμη πιο δύσκολη. Δείτε στην επόμενη εικόνα το λαβύρινθο που έχει δημιουργηθεί με το συνδυασμό λόφων και δρόμων. Μια μόνο συγκεκριμένη διαδρομή οδηγεί τον Τομ στην έξοδο:



Δείτε το παράδειγμα
11_01.kodu

Ο χρήστης θα πρέπει να κινηθεί στους δρόμους του λαβύρινθου ψάχνοντας την έξοδο. Σε εκείνο το σημείο επίσης θα βρίσκεται και ένα αστέρι όπου η απόκτησή του θα σημαίνει και το τέλος του παιχνιδιού. Κατά την διάρκεια όμως της προσπάθειά του, ο χρήστης θα βρεθεί αντιμέτωπος με άλλα αντικείμενα (εχθρούς) που θα τον εμποδίσουν να εκτελέσει επιτυχώς την αποστολή του. Αυτά θα είναι κανόνια, περισκόπια και κουμπιά πίεσης. Οι εχθροί πυροβολώντας τον Τομ θα επιδιώξουν να μειώσουν την ενέργειά του και όταν αυτή μηδενιστεί ο παίκτης θα χάνει. Από την άλλη πλευρά όμως, στην πίστα θα υπάρχουν και αντικείμενα που θα προσφέρουν στον παίκτη βοήθεια στην προσπάθεια για την εκπλήρωση του στόχου του. Αυτά θα είναι οι καρδιές και τα κέρματα, τα οποία θα αυξάνουν την ενέργεια του και θα του δίνουν ένα επιπλέον όπλο. Ο παίκτης πυροβολώντας τους εχθρούς του θα κερδίζει πόντους εξόντωσης (σε ένα κόκκινο σκορ)! Ξεπερνώντας ένα όριο από πόντους εξόντωσης, ο παίκτης θα επιβραβεύεται με μπόνους στην μπάρα ενέργειάς του.

Συνοψίζοντας τους στόχους του παίκτη:

- κίνηση στους δρόμους του λαβύρινθου
- αποφυγή πυροβολισμού από αντίπαλο
- συλλογή καρδιών
- συλλογή κερμάτων
- συγκέντρωση πόντων εξόντωσης
- συλλογή αστεριού

11.2 Αντικείμενα και συμπεριφορές

Σε αυτό το σημείο θα παρουσιάσουμε τα αντικείμενα που θα χρειαστούμε για το παιχνίδι μας.

Τομ

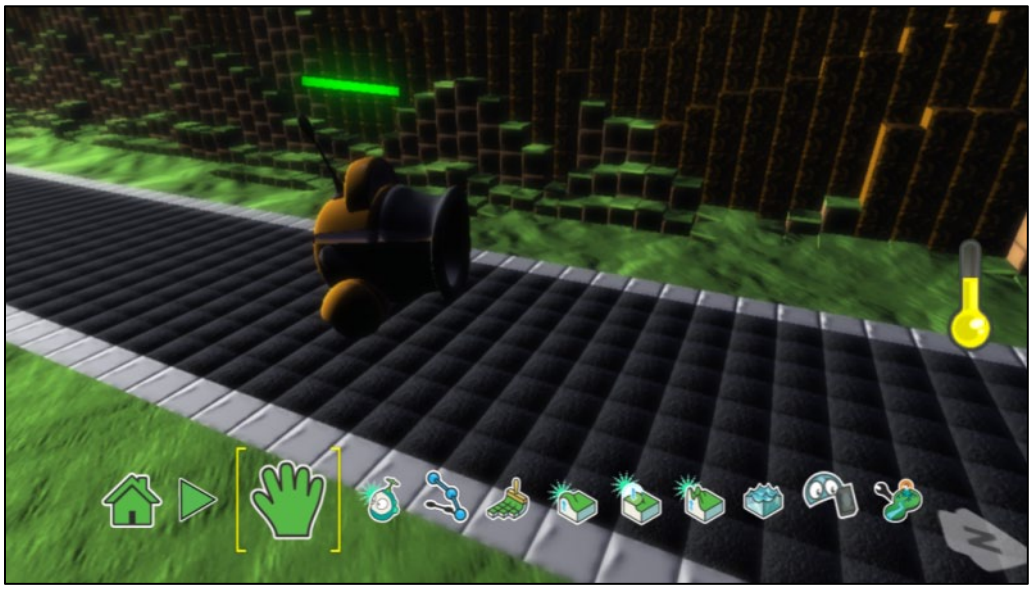
Αρχικά θα έχουμε τον βασικό μας ήρωα, τον Μηχανάκια Τομ, όπως μας παρουσιάζεται στην παρακάτω εικόνα:

Ο χρήστης θα μπορεί να τον μετακινεί από το πληκτρολόγιο και πιο συγκεκριμένα με τα βέλη, ενώ θα έχει αρχική ενέργεια ίση με 500 μονάδες. Επίσης, θα μπορεί να πυροβολεί με το κουμπί κενό (space) από το πληκτρολόγιο. Κάθε φορά που πετυχαίνει έναν αντίπαλο θα αυξάνονται οι πόντοι εξόντωσης. Όταν χτυπήσει μια καρδιά θα κερδίζει ενέργεια 20 πόντων. Όταν χτυπήσει ένα κέρμα θα ανεβάζει κατά ένα τον κίτρινο μετρητή του. Αν χτυπήσει το αστέρι, το παιχνίδι θα τερματίζει επιτυχώς. Αν μηδενιστεί η ενέργειά του, θα χάνει.

Οι φίλοι του Τομ, τους οποίους θα πρέπει να σώσει.

Οι αντίπαλοί του θα είναι:

Κανόνια (cannon)



Τα κανόνια μόλις μπορέσουν να δουν τον Τομ (έχουν δηλαδή οπτική επαφή με τον Τομ) θα τον πυροβολούν και θα αφαιρούν 5 μονάδες από την ενέργειά του κάθε φορά που τον πετυχαίνουν. Όταν τελειώσει η δική τους ενέργεια από τους πυραύλους του Τομ, θα καταστρέφονται.

Περισκόπια (stick)

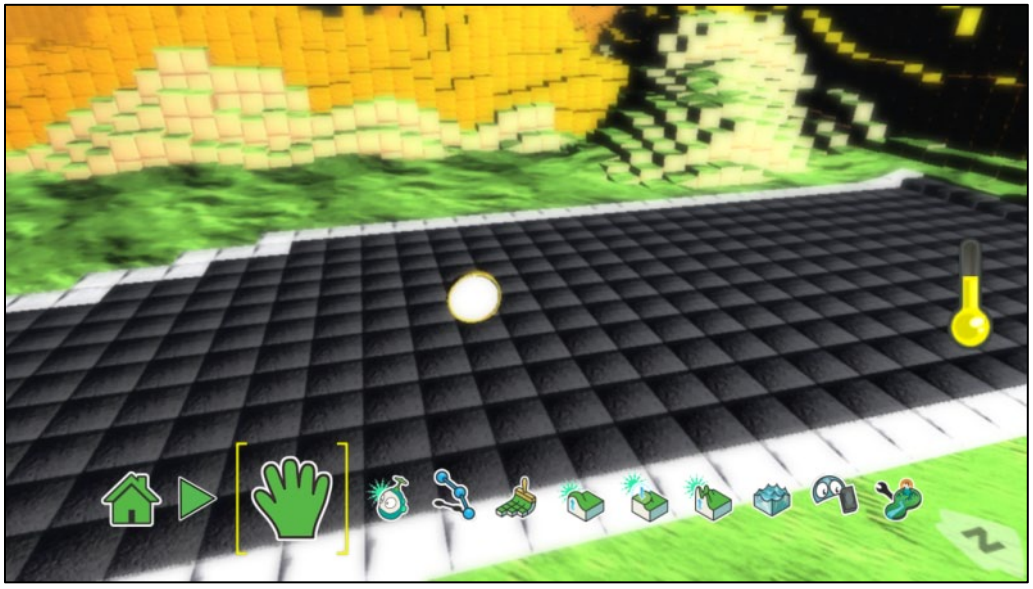
Τα περισκόπια θα είναι κρυμμένα στο έδαφος. Όταν τα πλησιάσει ο Τομ, θα ανοίγουν και θα πυροβολούν στριφογυρίζοντας. Όταν απομακρυνθεί ο Τομ, θα ξανακρύβονται. Και αυτά καταστρέφονται όταν τελειώσει η ενέργειά τους.

Κουμπί πίεσης (push pad)

Χρειάζονται προσοχή γιατί καταστρέφοντας ένα push pad, θα δημιουργείται ένα κανόνι. Έτσι ο Τομ θα έχει έναν ακόμα μπελά να αντιμετωπίσει.

Αρχικά το κουμπί πίεσης θα αρχίζει να πυροβολεί τον Τομ όταν έχει οπτική επαφή μαζί του. Αν τον πετύχει, θα αφαιρεί από την ενέργειά του 5 μονάδες. Όταν το κουμπί πίεσης χάσει όλη την δική του ενέργεια, μια δυσάρεστη έκπληξη θα περιμένει τον παίκτη. Στο ίδιο ακριβώς σημείο θα γεννηθεί ένα νέο κουμπί πίεσης, σαν μία άλλη λερναία Ύδρα, που θα θελήσει να ολοκληρώσει το έργο του προηγούμενου, δηλαδή την εξολόθρευση του Τομ. Μόλις «γεννηθεί» το καινούργιο κουμπί πίεσης, θα εμφανιστεί από πάνω του ένα μήνυμα (SUPRISE!).

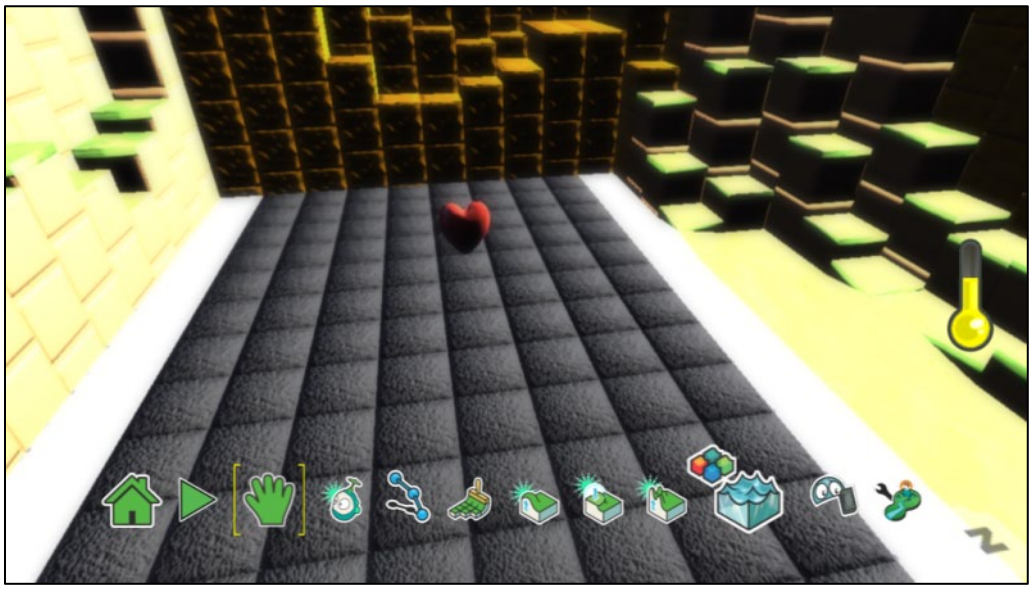
Τα κέρματα



Τα κέρματα θα αποκτούνται απλά από τον χρήστη. Με την απόκτηση ενός κέρματος, ο κίτρινος μετρητής θα αυξάνεται. Τι πρέπει να προσέξουμε όμως; Τι θα συμβεί αν κοντά στο σημείο που έχει τοποθετηθεί ένα κέρμα γίνεται μία μάχη μεταξύ του Τομ και ενός αντιπάλου; Οι σφαίρες μπορεί να χτυπήσουν το κέρμα με αποτέλεσμα να το καταστρέψουν και να το εξαφανίσουν. Το κέρμα όμως θα πρέπει να χάνει την ενέργειά του μόνο όταν το *χτυπήσει* ο χρήστης.

Οι καρδιές

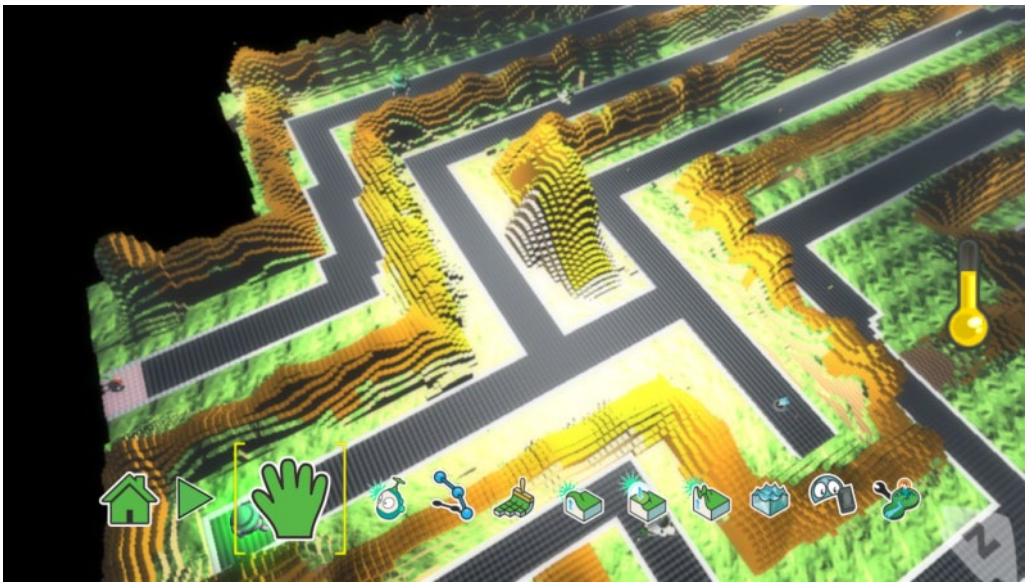
Οι καρδιές θα αποκτούνται από τον χρήστη και θα προσφέρουν μονάδες ενέργειας.



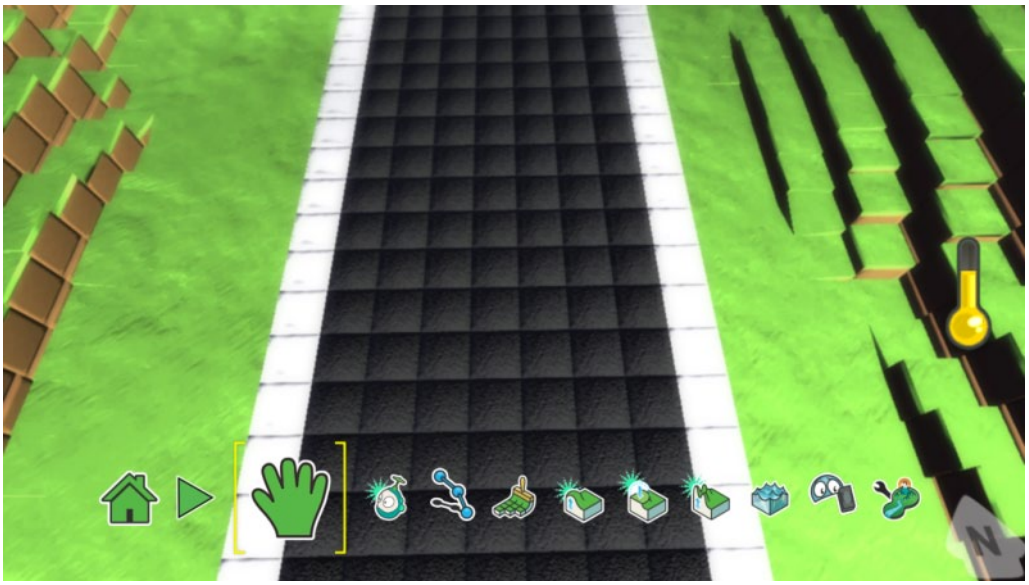
11.3 Δημιουργώντας τον κόσμο

Αρχικά θα πρέπει να σκεφτούμε πως θα θέλαμε να είναι η πίστα μας. Μπορούμε να σκεφτούμε οποιαδήποτε μορφή λαβύρινθου θέλουμε ενώ στη συνέχεια πρέπει να δημιουργήσουμε τα περιγράμματα των διαδρόμων μέσα από τα οποία θα κινηθεί ο παίκτης. Στην υλοποίησή μας, σχεδιάσαμε περιγράμματα από λόφους προσθέτοντας πρώτα έδαφος (τύποι εδάφους 20 και 78), μετά με την επιλογή *Δημιουργήστε Λόφους ή Κοιλάδες* ανυψώσαμε το έδαφος μέχρι να δημιουργηθούν αρκετά ψηλοί λόφοι και, τέλος, με την επιλογή *Λείανση Εδάφους* κάναμε τους λόφους πιο λείους. Για την κατασκευή των δρόμων χρησιμοποιήσαμε έδαφος τύπου 25 και στην άκρη των δρόμων προσθέσαμε μία άσπρη γραμμή έτσι ώστε να είναι πιο ευδιάκριτοι και από αισθητικής πλευράς πιο ελκυστικοί. Είναι βέβαια στην ευχέρεια του κάθε χρήστη να πειραματιστεί με διάφορους τύπους εδάφους προσθέτοντας έτσι την δική του πινελιά! Ακόμα σχεδιάσαμε το σημείο από το οποίο ξεκινάει ο χρήστης με μία διαφορετική επιλογή εδάφους (τύπος 29) και το σημείο στο οποίο βρίσκονται οι φίλοι του με ένα άλλο τύπο εδάφους (τύπος 32). Τέλος, κάναμε το ουρανό του κόσμου μας μαύρο, προσθέτοντας, όμως, αντικείμενα-φώτα για να δημιουργήσουμε μια πιο ενδιαφέρουσα ατμόσφαιρα! Πάμε να δούμε σε εικόνες τα όσα περιγράψαμε.

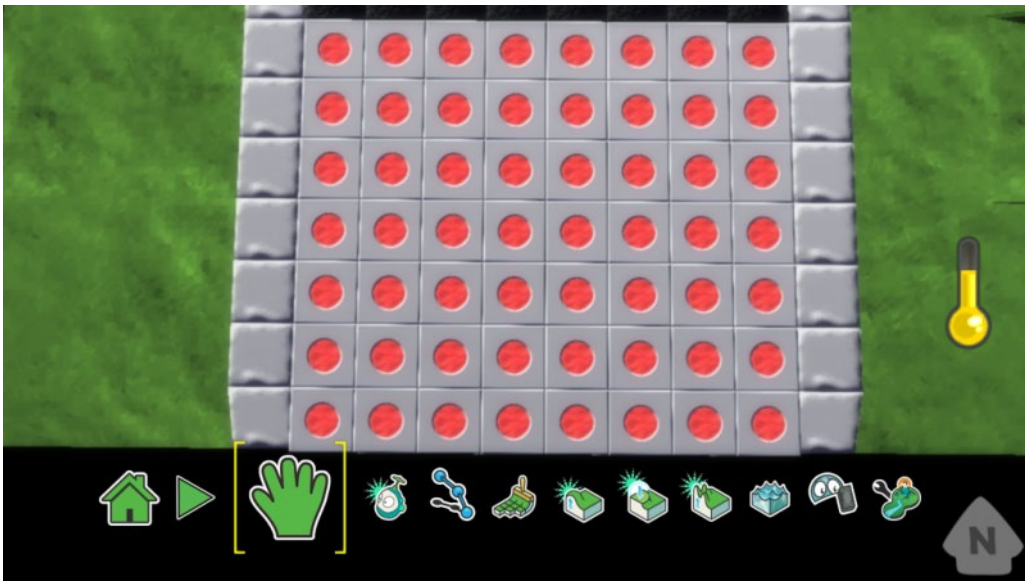
Οι λόφοι:



Οι δρόμοι:

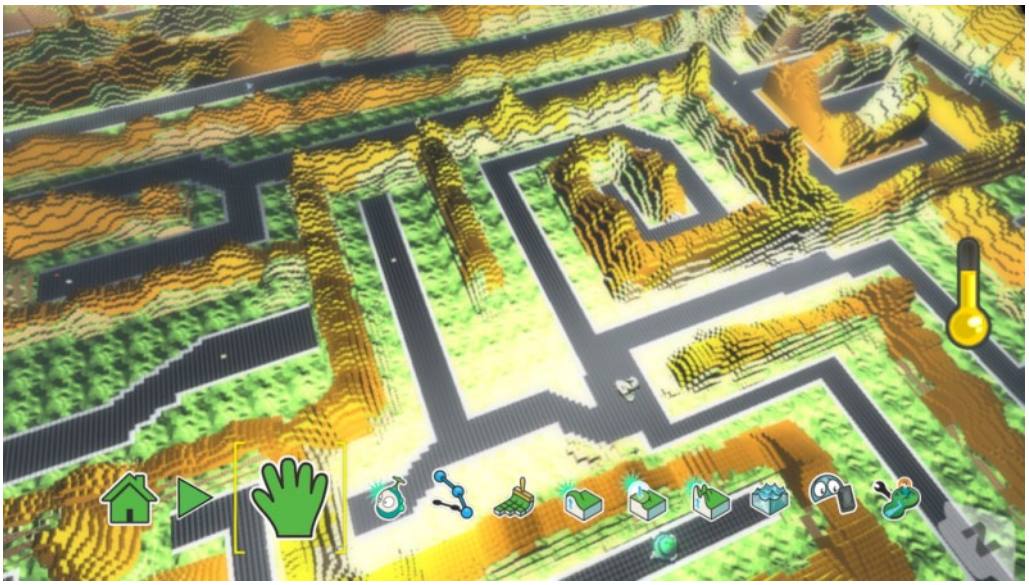


Η έναρξη (είσοδος):



Σημείο που βρίσκονται οι φίλοι του Μηχανάκια (έξοδος):

Γενική άποψη της πίστας:



11.4 Συμπεριφορές αντικειμένων και ανάλυση εντολών

Τομ, συμπεριφορά και ανάλυση εντολών

Σύμφωνα με τη γενική περιγραφή, μπορούμε να διακρίνουμε τρεις διαφορετικές συμπεριφορές για τον πρωταγωνιστή μας, τις οποίες και θα αντιστοιχίσουμε σε τρεις σελίδες συμπεριφορών:

- Σελίδα 1: τυπικές ενέργειες, κινούμαι και πυροβολώ
- Σελίδα 2: με την απόκτηση μιας καρδιάς…
- Σελίδα 3: με την απόκτηση ενός κέρματος (δυνατότητα εκτόξευσης πυραύλων)…

Στην πρώτη σελίδα, θα πρέπει να εξασφαλίσουμε ότι ο πρωταγωνιστής θα κινείται με τα βελάκια του πληκτρολογίου και θα πυροβολεί *σφαιρίδια (blip)* με το πλήκτρο κενό (space). Τα παραπάνω μπορούν να υλοποιηθούν με τις εξής εντολές:

Όταν ο Τομ «φάει» μια καρδιά θα πρέπει να συμβαίνουν μια σειρά από ενέργειες:

- θα πρέπει να φωτίζεται με κόκκινο χρώμα και μετά από δύο δευτερόλεπτα θα πρέπει να επιστρέφει στο αρχικό του χρώμα. Αυτό θα είναι μία ένδειξη ότι ο παίκτης πήρε την καρδιά.

- ταυτόχρονα θα πρέπει να παίρνει 20 μονάδες ενέργειας.

Πως θα δημιουργήσουμε τις κατάλληλες συμπεριφορές; Θα εμφωλεύσουμε τις αντίστοιχες συμπεριφορές μέσα στην εντολή

Όταν Φάω Καρδιά

Στην παρακάτω εικόνα φαίνεται συγκεντρωτικά η ακολουθία των εντολών:

Αρχικά, θέλουμε να γίνεται κόκκινος και να παραμένει μέχρι να του δώσουμε εμείς εντολή να σβήσει. Δηλαδή,

ΟΤΑΝ *Πάντα (Always)* **ΤΟΤΕ** *Φώτισε (Glow) κόκκινο (Red).*

Προσέξτε ότι δεν συμπληρώνουμε κάποιον αισθητήρα καθώς η εντολή είναι εμφωλευμένη μέσα στην προηγούμενη. Με αυτή την πρόταση ο Τομ γίνεται κόκκινος και παραμένει κόκκινος μέχρι να του πούμε να σβήσει.

Στη συνέχεια θέλουμε να αυξάνει κατά 20 μονάδες την ενέργειά του**:**

ΟΤΑΝ *Πάντα (Always)* **ΤΟΤΕ** *Αύξησε ενέργεια (Heal) Εμένα (Me) 20 βαθμούς (20 points).*

Τέλος, μετά από 2 δευτερόλεπτα θέλουμε ο Τομ να σβήσει. Η παρακάτω έκφραση φαίνεται να ικανοποιεί την προηγούμενη πρόταση:

ΟΤΑΝ *χρονόμετρο (timer) 2 δευτερόλεπτα (2 seconds)* **ΤΟΤΕ** *μην φωτίζεις (Glow off).*

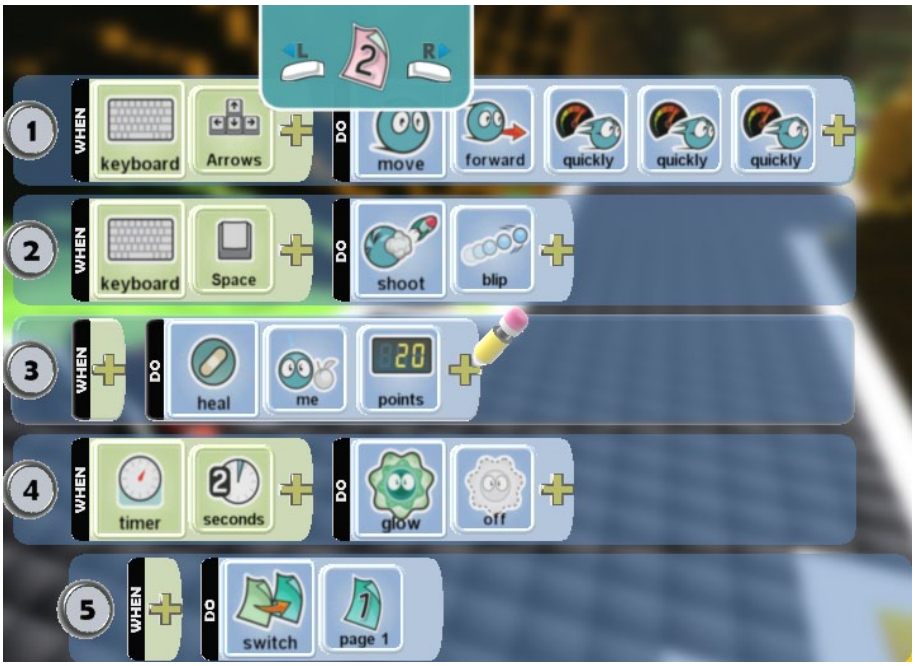
Είναι όμως σωστή αυτή η ακολουθία συμπεριφορών; Τι πιστεύετε θα γίνει εάν τρέξουμε το πρόγραμμά μας με αυτό τον τρόπο; Αν δοκιμάσετε να εκτελέσετε τις εντολές αυτές, θα διαπιστώσετε ότι ο Μηχανάκιας θα έμενε κόκκινος όλη την ώρα χωρίς να σβήσει. Γιατί; Γιατί δεν ενεργοποιείται το χρονόμετρο; Ποιες συνθήκες τίθενται για να ενεργοποιηθεί; Μήπως δεν έχουμε σκεφτεί κάτι ολοκληρωμένα;

Τώρα μπορούμε να σκεφτούμε ως εξής: δεν είναι μία αλλαγή συμπεριφοράς οι μπόνους μονάδες ενέργειας καθώς και το σβήσιμο του Τομ όταν έχει γίνει κόκκινος; Στην αρχή (σελίδα 1) του δώσαμε μία συμπεριφορά, γίνε κόκκινος. Μετά ζητάμε μία άλλη συμπεριφορά, να αυξήσει την ενέργειά του κατά 20 βαθμούς και να σταματήσει να είναι κόκκινος μετά από 2 δευτερόλεπτα. Στο κεφάλαιο 9 είδαμε πως αλλάζουμε τη συμπεριφορά ενός χαρακτήρα χρησιμοποιώντας τις *σελίδες*. Αυτό λοιπόν θα κάνουμε και εδώ.

Στη σελίδα 2, όπως αναφέραμε, θέλουμε να αυξήσουμε την ενέργειά του κατά 20 βαθμούς και επιπλέον ο Τομ θα πρέπει να σταματήσει να φωτίζει μετά από 2 δευτερόλεπτα. Πότε και πως θα γυρίσουμε τον πρωταγωνιστή μας στην προηγούμενή του συμπεριφορά, δηλαδή πίσω στην σελίδα 1;

Πρώτον θέλουμε να γυρίζει πίσω στην σελίδα 1 μετά το τέλος των 2 δευτερολέπτων (απαντάμε στο πότε). Άρα η επιστροφή θα πρέπει να συνδεθεί με την εντολή όπου ο Τομ μετά από 2 δευτερόλεπτα θα σβήσει. Μπορεί ο Μηχανάκιας να κινηθεί στη σελίδα 2; Μπορεί να πυροβολήσει; Όχι αφού στη σελίδα αυτή δε περιέχονται αντίστοιχες συμπεριφορές. Εμείς όμως

θέλουμε να συνεχίζει να κινείται και να πυροβολεί και κατά την διάρκεια αυτών των 2 δευτερολέπτων που μεσολαβούν. Έτσι προσθέτουμε και εδώ τις εντολές 1, 2 της σελίδας 1.



Έχουμε ολοκληρώσει τη σελίδα 2 και συνεχίζοντας τις σκέψεις μας για τη 1. Έχουμε δώσει μέχρι στιγμής τη δυνατότητα στο χρήστη να κινείται, να πυροβολεί και να τρώει καρδιές αυξάνοντας την ενέργειά του. Πάμε τώρα να του δώσουμε τη δυνατότητα να τρώει κέρματα.

Πότε θα τρώει τα κέρματα; Θέλουμε να τρώει τα κέρματα όταν τα ακουμπήσει. Άρα μπορούμε να πούμε:

ΟΤΑΝ *Πέσει πάνω σε (Bump)* κέρμα (Coin) **ΤΟΤΕ** *Φάε (Eat) Αυτό (It).*

Αυτό μπορούμε να το δούμε στην εντολή 6 στην παρακάτω εικόνα. Τι θέλουμε να γίνεται ταυτόχρονα τότε; Θέλουμε να αυξάνεται ένας κίτρινος μετρητής:

Ο Μηχανάκιας με το που αποκτά 3 κέρματα, θα πρέπει να αλλάζει συμπεριφορά και αφενός μεν να χρωματίζεται με το πορτοκαλί χρώμα και αφετέρου να είναι σε θέση να εκτοξεύει πυραύλους. Μπορούμε δηλαδή να πούμε ότι όταν έχεις στο κίτρινο μετρητή τον αριθμό τρία άναψε σε πορτοκαλί:

ΟΤΑΝ *Σκορ (Scored) Πάνω από (Above) 2 βαθμούς (2 Points) Κίτρινους (Yellow)* **ΤΟΤΕ** *Φώτισε (Glow) πορτοκαλί (Orange)*

Προσοχή! Όταν στον αισθητήρα προσδιορίζουμε σκορ «με πάνω από 2 βαθμούς», ουσιαστικά εννοούμε τον αμέσως επόμενο αριθμό του 2 που είναι το 3. Αυτό το κάνουμε γιατί δεν μας δίνεται η δυνατότητα της ισότητας. Για να αρχίσει να πετά πυραύλους ο Τομ, μήπως πρέπει να μεταφερθούμε σε μια άλλη σελίδα;

Εμείς εδώ θα μεταφερθούμε στην σελίδα 3, δηλαδή:

Μεταφερόμαστε τώρα στην σελίδα 3 όπου ο Μηχανάκιας θα έχει μία διαφορετική συμπεριφορά ρίχνοντας αυτή την φορά πυραύλους. Και σε αυτή την σελίδα, θα χρειαστεί να κινείται ο πρωταγωνιστή μας. Παρατηρήστε πως στην πρώτη γραμμή βάζουμε την εντολή κίνησης (εντολή 1 της παρακάτω εικόνας) όπως και στις δύο προηγούμενες σελίδες. Στη συνέχεια θα θέλαμε ο Μηχανάκιας να εκτοξεύει πυραύλους. Πώς μπορεί να γίνει αυτό; Αντίστοιχα με ότι κάναμε στην σελίδα 1 για να πυροβολούμε σφαιρίδια. Θα παρατηρήσετε ότι στο τέλος της αντίστοιχης συμπεριφοράς έχουμε προσθέσει το «once». Γιατί; Η ενέργεια μετά το **DO** θέλουμε να εκτελεστεί μία φορά μόνο με το πάτημα του πλήκτρου (αλλιώς κρατώντας το κουμπί πατημένο ο ήρωάς μας θα επαναλαμβάνει την εκτόξευση πυραύλων).

Από την στιγμή που ο χρήστης χρησιμοποιήσει τον πύραυλο, ο κίτρινος μετρητής θα πρέπει να μειωθεί κατά 3 (τόσο ήταν και το σκορ που έπρεπε να έχουμε με την συγκομιδή κερμάτων για να μπορέσουμε να χρησιμοποιήσουμε τον πύραυλο). Την επόμενη φορά που θα μαζέψει 3 κέρματα και το κίτρινο σκορ θα φτάσει τον αριθμό 3 τότε θα έχει και πάλι τη δυνατότητα να ρίξει πύραυλο. Άρα αφού ο Μηχανάκιας ρίξει ένα πύραυλο, πάντα το κίτρινο σκορ θα πρέπει να μειώνεται κατά 3. Που λοιπόν θα βάλουμε την μείωση του σκορ; Αφού προϋπόθεση για τη μείωση του κίτρινου σκορ είναι ο χρήστης να έχει ρίξει τον πύραυλο, τότε η συμπεριφορά θα πρέπει να μπει από κάτω από την εντολή 2 της παρακάτω εικόνας.

Πάμε να δούμε τώρα κάτι ακόμα πιο ενδιαφέρον. Το μάζεμα των κερμάτων και η αύξηση του κίτρινου μετρητή συμβαίνουν στην πρώτη σελίδα. Έστω ότι μαζέψαμε 3 κίτρινους βαθμούς από τα κέρματα και έχουμε μεταβεί στη σελίδα 3. Αν ο χρήστης δεν εκτοξεύσει το πύραυλο, το κίτρινο σκορ δεν μειώνεται. Δεν θα θέλαμε ο χρήστης να συνεχίσει να έχει τη δυνατότητα να μαζεύει κέρματα; (δηλαδή και άλλους κίτρινους βαθμούς;) Γιατί όχι! Άρα μπορούμε να χρησιμοποιήσουμε, όπως στην σελίδα 1, τις εντολές για το κίτρινο μετρητή κατά τον ίδιο ακριβώς τρόπο και με την ίδια λογική. Αυτό φαίνεται με τις εντολές 4 και 5 της παρακάτω εικόνας. Τι ακριβώς πετυχαίνουμε με την αύξηση εντός της σελίδας 3; Έχουμε την δυνατότητα να χρησιμοποιήσουμε και άλλο πύραυλο μαζεύοντας και άλλα κέρματα. Μαζεύοντας 3 επιπλέον κέρματα, ο χρήστης θα έχει την δυνατότητα να πετάξει δύο πυραύλους! Με 9 κέρματα 3 πυραύλους. Με κάθε πολλαπλάσιο του 3 θα μπορεί να ρίξει τους ανάλογους πυραύλους. Πάμε να δούμε πως!

Και πότε θα φεύγουμε από τη σελίδα 3; Όταν δεν θα έχουμε αρκετούς πόντους για να ρίξουμε πυραύλους, όταν θα έχουμε δηλαδή λιγότερους από 3 βαθμούς. Τι ακριβώς θα γίνει τότε; Ο Μηχανάκιας θα σβήσει (είχε φωτίσει στην πρώτη σελίδα όταν μάζεψε 3 κέρματα) και θα επιστρέψει στην αρχική σελίδα συμπεριφορών.

Ένα ακόμα ενδιαφέρον σημείο που θα πρέπει να σκεφτούμε, είναι τι γίνεται όταν ο χρήστης φάει μία καρδιά και μεταβεί στην σελίδα 2. Είδαμε ότι θα Μηχανάκιας θα παραμείνει στη σελίδα δύο για 2 δευτερόλεπτα. Μέσα στα δύο αυτά δευτερόλεπτα, τι θα γίνει αν βρεθεί ένα κέρμα κοντά του; Σύμφωνα με την μέχρι τώρα υλοποίηση των εντολών, δεν θα μπορεί να φάει το κέρμα. Πως λοιπόν μπορούμε να προσθέσουμε και αυτή την δυνατότητα; Και που θα την προσθέσουμε; Προφανώς στη σελίδα 2:

Γυρνώντας πάλι στην σελίδα 1, μας μένει να υλοποιήσουμε τη δυνατότητα του χρήστη χτυπώντας τους αντιπάλους του να αυξάνει ένα κόκκινο μετρητή, το μετρητή εξολόθρευσης. Θυμίζουμε ότι όταν ο μετρητής ξεπεράσει ένα όριο, ο χρήστης θα πρέπει να επιβραβεύεται με μπόνους ενέργειας και ο κόκκινος μετρητής θα πρέπει να μειώνεται αντίστοιχα.

Πάμε να δούμε σταδιακά πως θα δημιουργήσουμε και αυτή τη δυνατότητα. Καταρχήν οι αντίπαλοι του παίκτη είναι τρεις στον αριθμό. Τα κανόνια, τα κουμπιά πίεσης και τα περισκόπια. Για το κάθε εχθρό ξεχωριστά, θέλουμε εφόσον το χτυπήσει ο χρήστης, να αυξάνεται ο κόκκινος μετρητής κατά ένα:

Στη συνέχεια θέλουμε αφού έχει φτάσει το σκορ σε ένα αριθμό πόντων, τότε να αυξάνεται η ενέργεια του Τομ. Μπορούμε να σκεφτούμε όπως με την εντολή 8 της πρώτης σελίδας.

ΟΤΑΝ *Σκορ (Scored) Πάνω από (Above) 100 βαθμούς (100 Points) Κίτρινους (Yellow)* **ΤΟΤΕ** *Αύξησε ενέργεια (Heal) Εμένα (Me) 120 βαθμούς (120 points)*

Αφού όμως συμβεί αυτό θα πρέπει να μειώνεται το κόκκινο σκορ κατά 100 βαθμούς:

Σε αυτό το σημείο πρέπει να σκεφτούμε τι θα συμβεί αν ο παίκτης χάσει όλη του την ζωή. Προφανώς, θα πρέπει να τερματίσουμε το παιχνίδι.

Και πότε θα τερματίζει επιτυχώς το παιχνίδι; Όταν ο χρήστης καταφέρει και αρπάξει το μπλε αστέρι. Τότε ο παίκτης θα έχει νικήσει.

Σε ποιες σελίδες πρέπει να βάλουμε τις δυο τελικές εντολές; Μόνο στην πρώτη; Ή και στις υπόλοιπες; Τις προηγούμενες εντολές;

Κανόνι

Η συμπεριφορά του κανονιού θέλουμε να είναι η εξής: όταν δει έναν Μηχανάκια χωρίς όμως κάτι να τον εμποδίζει, όπως για παράδειγμα ένας λόφος, θέλουμε να ρίχνει προς αυτόν πυραύλους. Παράλληλα όταν πετύχει ένα Μηχανάκια θέλουμε να του προκαλείται ζημιά πέντε βαθμών, δηλαδή να του αφαιρούνται πέντε μονάδες από την ενέργειά του:

Περισκόπιο

Η συμπεριφορά του περισκοπίου θέλουμε να είναι η εξής: Καταρχάς, πρέπει να είναι κλειστό και να ανοίγει μόνο όταν δει ένα Μηχανάκια. Στην παραπάνω πρόταση έχουμε δύο ξεχωριστές συμπεριφορές προς υλοποίηση:

1. περισκόπιο πάντα κλειστό2.

2. περισκόπιο ανοίγει μόνο όταν δει έναν Μηχανάκια και ο Μηχανάκιας είναι κοντά.

Στη συνέχεια, πρέπει να σκεφτούμε τι θέλουμε να κάνει το περισκόπιο όταν αυτό παραμένει ανοικτό (θυμηθείτε ότι παραμένει ανοικτό όσο ο Μηχανάκιας είναι κοντά σε αυτό). Αυτό που θέλουμε να κάνει είναι:

1. να γίνεται κόκκινο

2. να εκτοξεύει σφαιρίδια

3. να στριφογυρίζει

Τέλος, όταν απομακρύνεται ο ήρωάς μας, το περισκόπιο πρέπει να κλείνει. Οι εντολές της επόμενης εικόνας υλοποιούν όλα τα προηγούμενα:

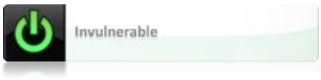
Κουμπί πίεσης

Η συμπεριφορά του κουμπιού πίεσης θέλουμε να είναι παρόμοια με αυτή του κανονιού, δηλαδή όταν δει ένα Μηχανάκια θέλουμε να ρίχνει πυραύλους. Αντίστοιχα, όταν πετύχει το Μηχανάκια θέλουμε να μειώνει την ενέργεια του παίκτη κατά πέντε μονάδες. Επιπλέον όμως, όπως αναφέραμε στην περιγραφή του προβλήματος, θέλουμε όταν το κουμπί πίεσης καταστρέφεται, θέλουμε να κλωνοποιείται, δηλαδή να ξαναγεννιέται. Εδώ έρχεται η έννοια των κλώνων (Creatables). Όπως αναφέραμε στο Κεφάλαιο 9, μπορούμε να δημιουργήσουμε «αρχέτυπα», δηλαδή «πρότυπα» αντικείμενα τα οποία μπορούμε να αναπαράγουμε στο παιχνίδι μας μέσα από άλλα αντικείμενα. Στην περίπτωσή μας ενεργοποιούμε την επιλογή Creatable σε ένα κουμπί πίεσης. Από που όμως θέλουμε να κλωνοποιηθεί το νέο κουμπί πίεσης, δηλαδή ποιό αντικείμενο θα το γεννήσει; Θα το «γεννήσει» το αρχικό κουμπί πίεσης που έχουμε εισάγει μέσα στην πίστα, το οποίο βέβαια δεν θα έχει χαρακτηρίσει ως Creatable. Με την ενέργεια *Δημιούργησε (Create)* θα προκαλείται η δημιουργία του «πρότυπου» κουμπιού πίεσης, όταν το κουμπί πίεσης που συμμετέχει στο παιχνίδι μας δεν έχει ενέργεια.

Το κλωνοποιημένο κουμπί πίεσης τι συμπεριφορά θα έχει; Θα έχει και αυτό την ίδια συμπεριφορά με το αρχικό κουμπί πίεσης που είχαμε στην πίστα μας, δηλαδή θα πυροβολεί μόλις έχει οπτική επαφή με τον Μηχανάκια και θα του προκαλεί ζημιά είκοσι μονάδων. Επιπλέον, θα εμφανίζει το μήνυμα SUPRISE!

Καρδιά

Η καρδιά θα αποκτιέται από το χρήστη δίνοντας του μπόνους ενέργειας. Ας δώσουμε λίγη παραπάνω προσοχή και ας σκεφτούμε τι θα γίνει αν, κατά λάθος, πυρά από μία μάχη πέσουν πάνω σε μια καρδιά. Αν αυτά είναι αρκετά, τότε η καρδιά θα καταστραφεί. Θέλουμε εμείς να συμβαίνει αυτό; Προφανώς όχι! Θέλουμε η καρδιά να παραμένει ακέραια μέχρι να την ακουμπήσει ο Μηχανάκιας και να την φάει. Πως μπορούμε να κάνουμε την καρδιά ακέραια, δηλαδή να μην καταστρέφεται; Πηγαίνοντας στις επιλογές του αντικειμένου (Change Settings) ενεργοποιούμε την επιλογή *Άτρωτος (Invulnerable)*. Με αυτό τον τρόπο η καρδιά δεν θα καταστρέφεται.



Κέρματα

Ακριβώς την ίδια διαδικασία ακολουθούμε και με το κέρματα. Και το κέρματα δεν θέλουμε να καταστρέφονται, άρα ενεργοποιούμε την επιλογή *Άτρωτος (Invulnerable)*.

Πλέον είστε έτοιμοι να απολαύσετε το παιχνίδι σας

Παραλλαγές

1. «Κλωνωποιήστε» τα κανόνια της πίστας δημιουργώντας στη θέση τους ένα καινούργιο κανόνι κάθε φορά που καταστρέφονται. Κάντε όμως το δεύτερο κανόνι πιο επιθετικό.

2. Εισάγετε στο παιχνίδι αντικείμενα που κάτω υπό προϋποθέσεις αποκαλύπτουν στον Τομ ποια κατεύθυνση να ακολουθήσει. Οι προϋποθέσεις μπορεί να είναι η συγκέντρωση αρκετών μήλων ή το ποσοστό ενέργειας που του έχει απομείνει.

Κεφάλαιο 12ο: Σώστε την Kodula

12.1 Περιγραφή

Πιάσανε τη φίλη του Kodu!! Στο παιχνίδι που θα σχεδιάσουμε θα πρέπει ο Kodu να περάσει από μια σειρά δοκιμασιών για να σώσει τη φίλη του, τη Kodula!

Έστω ότι βρισκόμαστε σε έναν κόσμο που αρχικά περιέχει μία πίστα για αγώνα δρόμου, ένα κλουβί στο οποίο μπορούμε να κρύψουμε αντικείμενα, ένα βουνό, μία λιμνούλα πίσω από το βουνό και το μέρος όπου είναι φυλακισμένη η Kodula. Έχουμε στο μυαλό μας την πίστα που θα σχεδιάσουμε, κάπως έτσι:

Στόχος του παίκτη είναι να σώσει την Kodula περνώντας από διάφορα εμπόδια και στάδια δυσκολίας και έχοντας ως πρωταγωνιστή τον Kodu.

Στάδιο Α: Αγώνας δρόμου

Η πρώτη πρόκληση αφορά έναν αγώνα ταχύτητας: Πώς θα σας φαινόταν αν αρχικά ο Kodu έκανε έναν αγώνα με τη μηχανή του; Θα σκεφτείτε αμέσως... μα ο Kodu δεν έχει μηχανή! Σωστά, αλλά θα υπάρχει κάποιος τρόπος για να αποκτήσει! Ο χρήστης λοιπόν, αφού αποκτήσει τον έλεγχο του Μηχανάκια, θα πρέπει να τρέξει 3 γύρους στην πίστα και να τερματίσει πρώτος με αντίπαλο τον μαύρο Μηχανάκια, που έχει ακριβώς τον ίδιο στόχο. Ο χρήστης πρέπει να κάνει τους γύρους στην πίστα αριστερόστροφα (αντίθετα των δεικτών του ρολογιού). Αν κάποιος προσπαθήσει να κλέψει κάνοντας ανάποδα γύρους και περνώντας πάνω από τη γραμμή τερματισμού, θα πρέπει να τιμωρείται εγκαταλείποντας το παιχνίδι. Στην πίστα θα υπάρχουν 3 Πιατάκια σε σταθερά σημεία τα οποία θα πυροβολούν τον Kodu όταν τον βλέπουν και θα μειώνουν την ενέργειά του. Όταν όμως ο Kodu ακουμπά μία κόκκινη Καρδιά, θα ανακτά μέρος της χαμένης του ενέργειας! Χαμός! Φυσικά, αν η ενέργεια μηδενιστεί, τότε ο παίκτης χάνει. Υπάρχει όμως ακόμη ένα πρόβλημα για τον Kodu! Ο μαύρος Μηχανάκια είναι πιο γρήγορος από αυτόν! Γι' αυτό, κατά τη διάρκεια του αγώνα ο χρήστης θα πρέπει να μαζεύει υγρό το οποίο θα του δίνει επιτάχυνση όταν πατηθεί το κατάλληλο πλήκτρο!

Βασικά χαρακτηριστικά αγώνα δρόμου:

- 3 γύροι
- μόνο αριστερόστροφα
- επιτάχυνση στον Kodu
- γρηγορότερος ο μαύρος Μηχανάκιας
- τιμωρία σε όποιον κλέψει
- Πιατάκια που ρίχνουν πυραύλους στον Kodu
- Καρδιές που δίνουν ενέργεια στον Kodu

Δείτε το παράδειγμα *12_01.kodu*

- νικάει όποιος τερματίσει πρώτος
- χάνει όποιος δεν έχει ενέργεια.

Στάδιο Β: Μάχη για την Μπάλα

Αν ο χρήστης ξεφύγει από τον μαύρο Μηχανάκια στον αγώνα, τότε θα πρέπει να πάρει τη Μπάλα από το κλουβί και να την πετάξει στο κάστρο μπροστά από την Kodula για να την ελευθερώσει. Για να μπει στο κλουβί ο Kodu θα αφήσει το μηχανάκι και θα πάρει το αεροπλάνο, πιο συγκεκριμένα το Τζετ! Τύφλα να' χει ο πράκτορας 007! Άρα, μόλις βγει από την πίστα αγώνα ο Kodu και μετατραπεί σε Τζετ (αυτόματα), αρχίζει το δεύτερο μέρος του παιχνιδιού. Δεν θα είναι όμως τόσο εύκολα τα πράγματα, καθώς θα πρέπει να αποφύγει τους εχθρικούς Δορυφόρους, που εμφανίζονται και εξαφανίζονται σε τυχαία χρονικά διαστήματα πυροβολώντας τον Kodu. Κρατήστε αυτό το χαρακτηριστικό της τυχαιότητας, θα μας απασχολήσει ιδιαίτερα στον προγραμματισμό του παιχνιδιού.

Τι λέτε; Θα αφήσουμε μόνο του το Τζετ να παλεύει με τόσους Δορυφόρους; Όχι φυσικά! Ο παίκτης θα μπορεί να ανεβάσει τους πόντους ζωής του πηγαίνοντας κοντά σε ένα φιλικό Αερόπλοιο που προσφέρει δωρεάν ενέργεια! Επίσης θα μπορεί να τους πυροβολεί και να τους σκοτώνει για να ξεμπερδεύει μια και καλή! Αφού φύγουν οι Δορυφόροι από τη μέση ο δρόμος θα είναι ελεύθερος για να κουβαλήσουμε με το Τζετ τη μεγάλη Μπάλα που θα ρίξει το Κάστρο! Το παιχνίδι τελειώνει μόλις πέσει το Κάστρο και ελευθερωθεί η Kodula!

Βασικά χαρακτηριστικά 2^{ου} σταδίου:

- χειρισμός Τζετ
- τυχαία εμφάνιση και εξαφάνιση Δορυφόρων
- γέμισμα ενέργειας του Τζετ από το Αερόπλοιο
- απογείωση - προσγείωση Τζετ και κουβάλημα Μπάλας
- νίκη παιχνιδιού αν πέσει το Κάστρο
- ήττα αν τελειώσει η ενέργεια

12.2 Αντικείμενα και συμπεριφορές

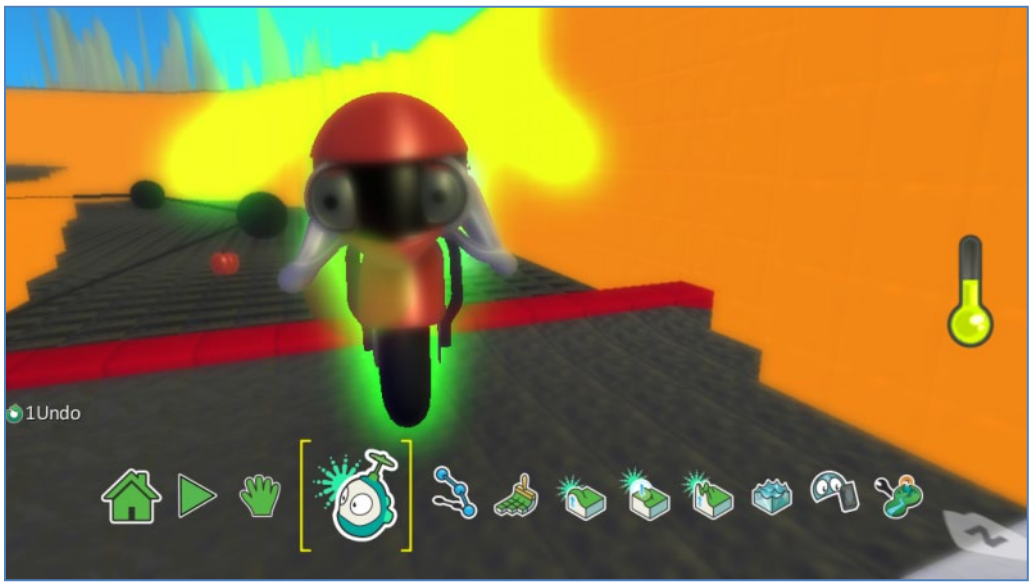
Ποια θα είναι τα αντικείμενα στο παιχνίδι μας;

Ας προσθέσουμε αρχικά τους δύο βασικούς μας ήρωες, την *Kodula* (γι' αυτήν γίνονται όλα):

και τον *Kodu*! (μάλλον δεν του αρέσει η δημοσιότητα):

Ο Kodu θα μετακινείται με τα βέλη του πληκτρολογίου. Θα πρέπει μόλις ακουμπήσει το πρώτο Εργοστάσιο να μετατρέπεται σε κόκκινο Μηχανάκια. Ουσιαστικά δηλαδή δε θα τον χρειαστούμε και για πολύ στο παιχνίδι μας. Απλά θα βοηθήσει στην αλυσιδωτή αντίδραση των μετατροπών Kodu > κόκκινος Μηχανάκια > Τζετ.

Ο *κόκκινος Μηχανάκιας* (ο Kodu με κόκκινη μηχανή και κράνος):

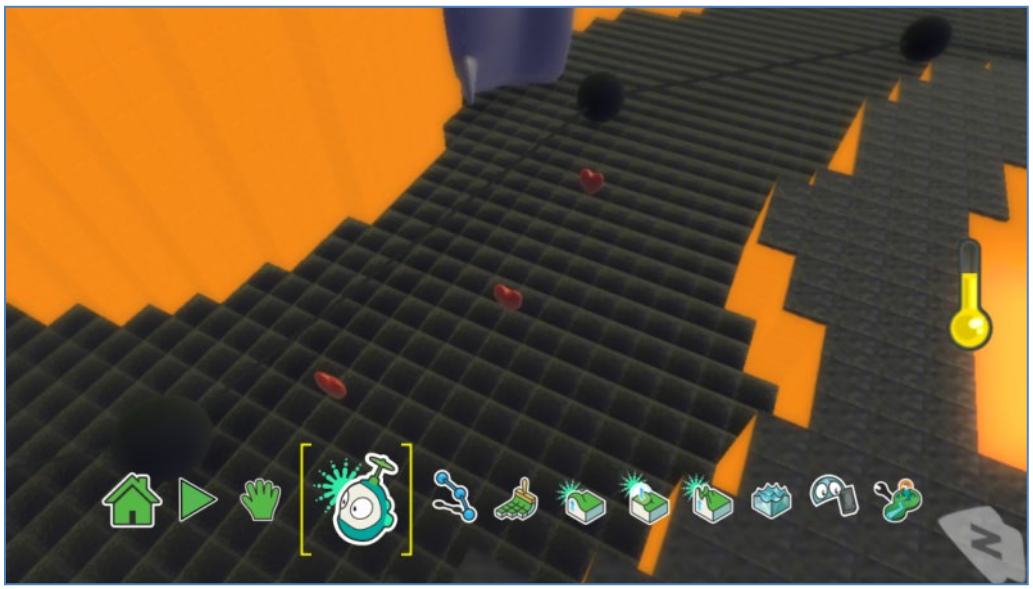


θα χειριζόμαστε τον κόκκινο Μηχανάκια από το πληκτρολόγιο και θα μπορεί να πυροβολεί με το γράμμα Z του πληκτρολογίου. Όταν περάσει πάνω από τη γραμμή τερματισμού με τη σωστή φορά (αριστερόστροφα) πρέπει να προστίθεται ένας πόντος στο κόκκινο σκορ. Όταν χτυπήσει μία Καρδιά, θα αυξάνεται η ενέργειά του κατά 10 πόντους. Όταν ακουμπήσει την Καλύβα, θα μετατρέπεται σε κόκκινο Τζετ. Αν βρίσκεται πάνω από το κίτρινο υγρό της πίστας, θα παίρνει πόντους επιτάχυνσης. Την επιτάχυνση θα μπορεί να τη χρησιμοποιήσει ο χρήστης πατώντας το αριστερό πλήκτρο Shift. Αν μηδενιστεί η ενέργειά του, θα καταστρέφεται.

Το *κόκκινο Τζετ*:

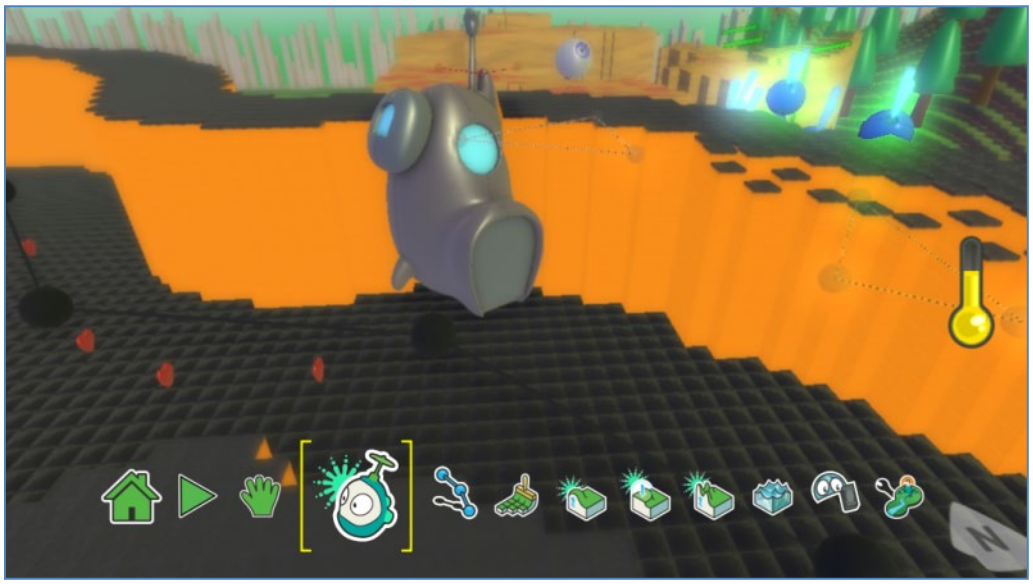
θα το χειριζόμαστε από το πληκτρολόγιο και θα μπορεί να απογειώνεται και να προσγειώνεται με τα πλήκτρα «αριστερό Shift» και «αριστερό Ctrl» αντίστοιχα. Όταν ακουμπά την Μπάλα θα την κουβαλά και όταν πατάμε Space θα την αφήνει. Όταν πλησιάζει το Αερόπλοιο θα παίρνει +100 πόντους ενέργειας. Θα πυροβολεί και αυτό με το πλήκτρο Z.

Οι *Καρδιές*:



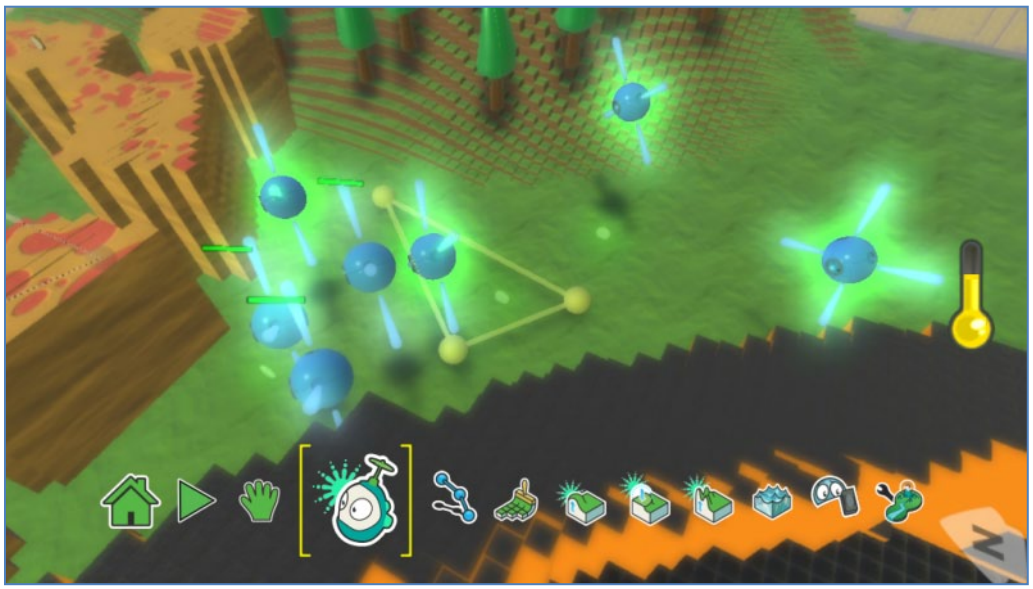
θα είναι τυχαία διατεταγμένες στη διαδρομή της πίστας αγώνα, θα δίνουν +10 πόντους ζωής στον κόκκινο Μηχανάκια και μετά θα εξαφανίζονται.

Η *Καλύβα* η οποία θα χρησιμοποιηθεί για τη μετατροπή του κόκκινου Μηχανάκια σε Τζετ μόλις την ακουμπήσει:



Έπειτα θα εξαφανίζεται για να ελευθερώσει την έξοδο προς τον υπόλοιπο κόσμο. Προσοχή! Αν ο κόκκινος Μηχανάκιας την ακουμπήσει πριν τελειώσει ο αγώνας, δεν πρέπει να αλλάζει τίποτα!

Οι *Δορυφόροι*:



θα κινούνται μπροστά από το κλουβί σε ένα μονοπάτι. Θα εμφανίζονται και θα εξαφανίζονται σε τυχαία διαστήματα. Σκοπός τους είναι να ρίχνουν πυραύλους στο Τζετ έτσι ώστε να το καταρρίψουν.

Τα *Πιάτα*:

θα τοποθετηθούν στην πίστα αγώνα και θα πυροβολούν τον κόκκινο Μηχανάκια όταν τον βλέπουν προς το μέρος τους. Σκοπός τους θα είναι να μειώσουν την ενέργεια του κόκκινου Μηχανάκια.

Το *Αερόπλοιο*:

θα κινείται σε ένα συγκεκριμένο κυκλικό μονοπάτι και θα δίνει ενέργεια στο Τζετ μόλις αυτό βρεθεί κοντά του. Αν χτυπηθεί από έναν πύραυλο ή αν πυροβοληθεί από το Τζετ καταστρέφεται.

12.3 Δημιουργώντας την πίστα

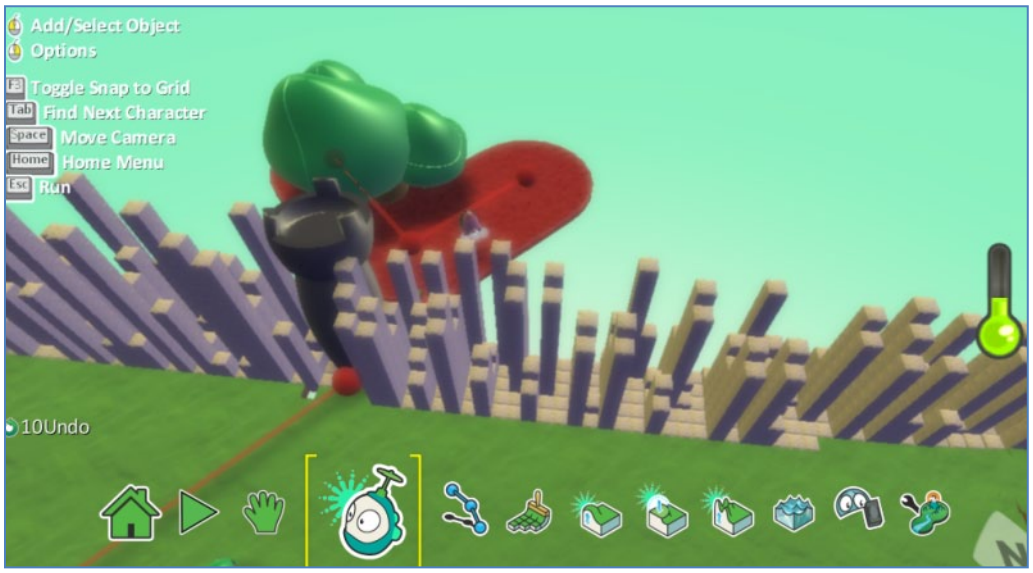
Σε έναν κενό κόσμο αρχίζουμε να σχεδιάσουμε την πίστα του αγώνα. Είδαμε ότι ο κόκκινος Μηχανάκιας πρέπει πρώτα να ολοκληρώσει τον αγώνα και μετά να περάσει στον υπόλοιπο κόσμο. Αν θυμάστε, στο 4.2 είχαμε σχεδιάσει μία πίστα αγώνα και αμέσως μετά ένα Κάστρο. Θα χρησιμοποιήσουμε ένα συνδυασμό των δύο τεχνικών για να σχεδιάσουμε μία κλειστή πίστα αγώνα! Μπορούμε αρχικά να ζωγραφίσουμε και μετά να σηκώσουμε τα τείχη έτσι ώστε να έχουμε ένα καλύτερο και πιο γρήγορο αποτέλεσμα.

Τώρα, μέσα στην πίστα πρέπει να προσθέσουμε το υγρό που μας δίνει πόντους επιτάχυνσης. Σε ποιο σημείο θα δημιουργούσατε τη λιμνούλα εσείς; Θα τη βάζατε σε μία στροφή της πίστας; Μάλλον όχι, γιατί έτσι θα δυσκολεύαμε αρκετά το χρήστη. Ας την προσθέσουμε σε μία ευθεία της πίστας. Το ίδιο πρόβλημα υπάρχει και με τα Πιατάκια. Αν τα βάλουμε στο δρόμο της πίστας, θα χάνει πολύ εύκολα ενέργεια ο χρήστης, άρα σχεδιάζουμε την πίστα έτσι ώστε να υπάρχει κενός χώρος στη μέση για να τοποθετήσουμε εκεί τα Πιατάκια.

Ο υπόλοιπος κόσμος μας περιέχει τα κλειδιά για να καταφέρουμε να ανοίξουμε το Κάστρο και να σώσουμε την Kodula. Άρα, έξω από την πίστα αγώνα σχεδιάζουμε ένα μέρος όπου ο Kodu μπορεί να πάει μόνο από αέρος (ναι! θα τον κάνουμε να πετάξει!) και το κλείνουμε με κοφτερές κορυφές.

Προσθέτουμε και το βουνό που βλέπετε στο βάθος της προηγούμενης εικόνας με τέτοιον τρόπο ώστε από πίσω να κάνουμε μία λίμνη όπου θα μπορούμε να κρύψουμε στοιχεία που θα χρειαστεί ο Kodu για την απελευθέρωση!

Τέλος, εισάγουμε τη θέση που θα φυλάσσεται η Kodula (απομονωμένη με μοναδική είσοδο/έξοδο μπροστά από το Κάστρο):



Θαυμάστε τον τελικό κόσμο μας και φύγαμε για προγραμματισμό!

12.4 Συμπεριφορές αντικειμένων και ανάλυση εντολών

Kodu, συμπεριφορά και ανάλυση εντολών

Ο αρχικός χαρακτήρας Kodu θα μας απασχολήσει για λίγο στο παιχνίδι καθώς θα μετατραπεί σε κόκκινο Μηχανάκια μόλις ακουμπήσει το Εργοστάσιο. Αφού βάλουμε τις εντολές κίνησης, πρέπει να τον προγραμματίσουμε ώστε μόλις ακουμπήσει το Εργοστάσιο, να εξαφανίζεται και να δημιουργεί στη θέση του τον κόκκινο Μηχανάκια. Είστε σε θέση πλέον να κατανοήσετε μόνοι σας τις συμπεριφορές που εμφανίζονται στην επόμενη εικόνα:

Λείπει κάτι; Λειτουργεί όπως θέλουμε ο Kodu; Αν όχι, πως μπορείτε να βελτιώσετε τις συμπεριφορές αυτές;

Κόκκινος Μηχανάκιας

Όπως φαίνεται από τη συμπεριφορά του Kodu, ο κόκκινος Μηχανάκιας είναι ένα αρχέτυπο αντικείμενο. Για να χρησιμοποιήσουμε αυτό το αντικείμενο, όπως μάθαμε στο κεφάλαιο 9, το μαρκάρουμε ως *κλώνο (creatable)*. Από την αρχική περιγραφή του προβλήματος συμπεραίνουμε ότι στον κόκκινο Μηχανάκια θα χρειαστεί να υλοποιήσουμε αρκετές ενέργειες και συμπεριφορές τις οποίες μπορούμε να οργανώσουμε ως εξής:

- Σελίδα 1: κυρίως πρόγραμμα, βασικές ενέργειες, μεταβάσεις προς τις άλλες σελίδες
- Σελίδα 2: μετρητής γύρων
- Σελίδα 3: μετρητής πόντων επιτάχυνσης
- Σελίδα 4: τιμωρία σε κλέψιμο γύρων
- Σελίδα 5: μετατροπή σε Τζετ
- Σελίδα 6: χρήση επιτάχυνσης με Shift

Αρχικά, για να μπορεί να κινείται με τα βέλη και να πυροβολεί με το Z:

Πώς θα καταφέρουμε να υλοποιήσουμε ένα ακριβή μετρητή γύρων; Είναι σωστό να βάλουμε απλά μία άσπρη γραμμή και να προγραμματίσουμε τα Μηχανάκια με τη συμπεριφορά:

ΟΤΑΝ είσαι πάνω από έδαφος τύπου άσπρο, ΤΟΤΕ αύξησε την τιμή του σκορ κατά ένα;

Όχι, γιατί μπορεί κάποιος χρήστης να κλέψει, αυξάνοντας το μετρητή κάνοντας μπρος πίσω ή να περιμένει πάνω στην άσπρη γραμμή ή ακόμα και να κάνει τους γύρους ανάποδα!

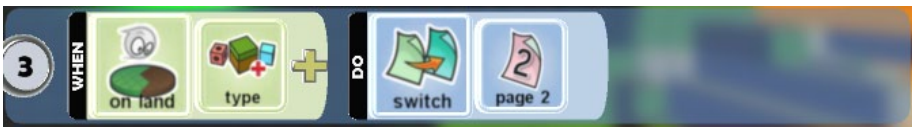
Μια λύση είναι να βάλουμε μία δικλείδα ασφαλείας, ζωγραφίζοντας μία κόκκινη γραμμή μετά την άσπρη έτσι ώστε αν ο χρήστης προσπαθήσει να κάνει ανάποδα το γύρο, να αναγκαστεί να περάσει πρώτα από την κόκκινη γραμμή. Αυτό το χαρακτηριστικό θα το εκμεταλλευτούμε σε μία *σελίδα (page)* εισάγοντας τις κατάλληλες εντολές!

Επίσης, για να μην επιτρέψουμε στο χρήστη να παίρνει πόντους απλά περιμένοντας πάνω στην άσπρη γραμμή, πρέπει να πάρουμε προγραμματιστικά τον έλεγχο του χαρακτήρα και να τον αναγκάσουμε να φύγει από την άσπρη γραμμή. Για παράδειγμα μπορούμε να ζητήσουμε από τον πρωταγωνιστή μας να ακολουθήσει ένα μονοπάτι μόλις βρεθεί πάνω στην άσπρη γραμμή που θα τον οδηγεί μετά από αυτή. Έπειτα θα του ξαναδώσουμε τον έλεγχο.

Στο παραπάνω σκεπτικό βοηθά η εικόνα της πίστας στο σημείο τερματισμού: (παρατηρήστε τα διαφορετικά είδη εδαφών: άσπρο, γκρι, μαύρο)

Σύμφωνα με τα παραπάνω, πρέπει να προγραμματίσουμε τον κόκκινο Μηχανάκια να αυξάνει το κόκκινο σκορ (μετρητής γύρων) κατά ένα, όταν περνά από την άσπρη γραμμή και ταυτόχρονα να μεταβαίνει ο έλεγχος προσωρινά από το χρήστη στο σύστημα.

Μεταβαίνουμε στη σελίδα 2, όταν είμαστε πάνω από την άσπρη γραμμή τερματισμού. Στη σελίδα 2 θέλουμε να πάρουμε τον έλεγχο από το χρήστη, να αυξήσουμε το κόκκινο σκορ κατά ένα και να του ξαναδώσουμε τον έλεγχο όταν περάσει στο έδαφος μετά την άσπρη γραμμή επιστρέφοντας στη σελίδα 1. Η σελίδα 1 θα διατηρήσει όλα τα χαρακτηριστικά που είχε και πριν με την προσθήκη της παρακάτω συμπεριφοράς:



Η σελίδα 2:

Σκεφτείτε τι πρόβλημα θα είχαμε αν αφήναμε μόνο την πρώτη εντολή. Όταν η ροή εκτέλεσης περάσει στη σελίδα 2, το σκορ θα αυξηθεί κατά ένα, αλλά ο χρήστης δε θα μπορεί να ελέγχει το Μηχανάκια! Το μόνο που θα ήξερε ο Μηχανάκιας στη σελίδα 2 είναι να αυξάνει το σκορ κατά ένα! Δε θα είχε προγραμματιστεί για κάτι διαφορετικό.

Τώρα πρέπει να αποτρέψουμε το χρήστη από το να κλέβει γύρους. Είχαμε βρει έναν τρόπο βάζοντας μία κόκκινη γραμμή μετά την άσπρη. Πώς θα προγραμματίσουμε το Μηχανάκια έτσι ώστε, αν περάσει πρώτα την κόκκινη γραμμή χωρίς να έχει περάσει την άσπρη, να τιμωρείται ο παίκτης; Θα τον ακινητοποιήσουμε! Δηλαδή, κάθε φορά που θα πατάει την κόκκινη γραμμή, θα το στέλνουμε στη σελίδα 4 όπου θα χάνει τον έλεγχο, εκτός κι αν αμέσως μετά πατήσει σε έδαφος της πίστας (μαύρο). Προσθέτουμε στο κυρίως πρόγραμμα (σελίδα 1) μία μετάβαση για τη σελίδα 4

ΟΤΑΝ είσαι σε έδαφος <τύπος κόκκινο> ΤΟΤΕ μετάβαση σε σελίδα 4

και στη σελίδα 4 μια συμπεριφορά

ΟΤΑΝ είσαι σε έδαφος <τύπος μαύρο> ΤΟΤΕ μετάβαση σε σελίδα 1

όπως φαίνεται στις δυο παρακάτω εικόνες:

Τι κερδίσαμε; Δείτε το σενάριο όπου ο χρήστης προσπαθεί να κλέψει κάνοντας ανάποδα τους γύρους: Πατάει στην κόκκινη γραμμή, άρα πάει στη σελίδα 4 και αμέσως μετά πατάει σε γκρι έδαφος ανάμεσα στην κόκκινη και άσπρη γραμμή. Μένει για πάντα στη σελίδα 4 όπου δεν μπορεί να κουνηθεί! Δεν επιστρέφει ποτέ! Τον τιμωρήσαμε γιατί πήγε να παραβιάσει τους κανόνες του παιχνιδιού! Σκεφτείτε το λίγο, είναι απλό και δουλεύει!

Η επόμενη ενέργεια που θα προγραμματίσουμε για το Μηχανάκια είναι να παίρνει πόντους επιτάχυνσης όταν περνάει πάνω από το νερό της πίστας. Πρέπει να αποθηκεύουμε κάπου αυτούς τους πόντους. Θα χρειαστούμε μία μεταβλητή (έστω το πράσινο σκορ) η οποία θα αυξάνει τιμή μόλις ο τύπος του υγρού είναι ίδιος με αυτόν που έχουμε ορίσει για την επιτάχυνση. Επίσης θα δώσουμε αρχικά 100 μονάδες επιτάχυνσης στο χρήστη.

Όπως και στα προηγούμενα παραδείγματα, πρέπει να αλλάξουμε συμπεριφορά στο χαρακτήρα μας, άρα θα χρησιμοποιήσουμε μία νέα σελίδα. Παρατηρήστε ότι κατά τη χρονική διάρκεια (εδώ 3 δευτερόλεπτα) στην οποία αυξάνουμε το μετρητή, δεν μπορούμε να επιταχύνουμε με το Shift.

Στην πρώτη σελίδα βάζουμε μία μετάβαση για την τρίτη όταν ο Μηχανάκιας είναι πάνω στο υγρό:

Όταν περάσουν τα 3 δευτερόλεπτα που χρειάζεται για να γεμίσει ενέργεια ο Μηχανάκιας, επιστρέφουμε πίσω στην αρχική σελίδα συμπεριφορών:

Επίσης, οι πόντοι επιτάχυνσης πρέπει να μειώνονται όταν ο χρήστης πατάει το Shift. Θα σκεφτείτε... εύκολο αυτό, ας βάλουμε στην αρχική σελίδα μία εντολή

ΟΤΑΝ πατηθεί στο Πληκτρολόγιο το Shift, ΤΟΤΕ Μείωσε το πράσινο σκορ κατά ένα.

Φυσικά, αυτός ο τρόπος δε μας επιτρέπει να κινούμαστε γρηγορότερα! Για να λύσουμε το πρόβλημα θα χρησιμοποιήσουμε μία νέα σελίδα στην οποία θα υλοποιήσουμε την επιτάχυνση. Πρέπει όμως να προσέξουμε το πότε επιστρέφουμε στη σελίδα 1 και την κανονική ροή εκτέλεσης. Σκεφτείτε: επιστρέφουμε στη σελίδα 1 όταν δεν πατάμε το Shift ή όταν οι πόντοι πέσουν κάτω από το μηδέν. Στην πρώτη περίπτωση είναι απαίτηση του χρήστη να μη χρησιμοποιήσει τους πόντους του, ενώ στη δεύτερη δεν έχει δικαίωμα να επιταχύνει.

Σε αυτό το σημείο μπορεί κάποιος να αναρωτηθεί τι θα γίνει αν ο χρήστης έχει συνεχώς πατημένο το Shift ενώ οι πόντοι επιτάχυνσης είναι μηδέν; Θα μεταπηδάμε συνεχώς από την πρώτη σελίδα στην τρίτη, με αποτέλεσμα να μειώνονται οι πόντοι και κάτω από το μηδέν!

Η μετατροπή του Μηχανάκια σε Τζετ όταν χτυπήσει την Καλύβα είναι πια κάτι απλό για εμάς! Με τόση εξάσκηση στην αλλαγή συμπεριφορών και στην ανακατεύθυνση σε άλλα κομμάτια κώδικα η υλοποίηση είναι πια παιχνιδάκι!

Τέλος, δύο ακόμα χαρακτηριστικά που είχαμε βάλει στην περιγραφή είναι η αύξηση της ενέργειας του Μηχανάκια όταν παίρνει μία Καρδιά και το Ζάλισμα (*Stun*) στο μαύρο Μηχανάκια όταν τον πυροβολούμε.

Παρατηρήστε ότι από τα παραδείγματα που κάναμε μέχρι τώρα προκύπτει πως, αν οι χαρακτήρες μας θέλουμε να αποκτούν ιδιαίτερα χαρακτηριστικά σε κάποια στιγμή του παιχνιδιού, αρκεί να αντιγράψουμε τις βασικές συμπεριφορές τους σε μια νέα σελίδα, να προσθέσουμε τις νέες συμπεριφορές και μετά προγραμματιστικά να ελέγχουμε πότε θα τρέχει κάθε σελίδα!

Καρδιές

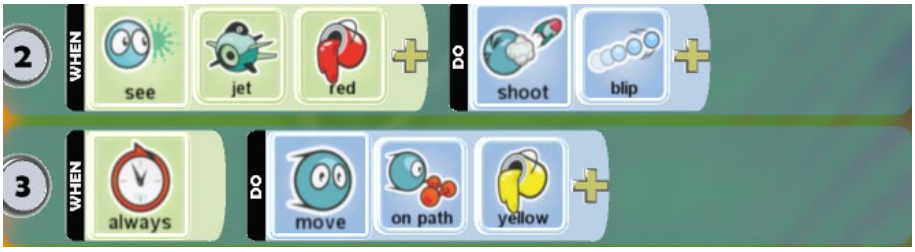
Ένα ζήτημα που δεν έχουμε ασχοληθεί μέχρι τώρα είναι το πού θα εισάγουμε τις συμπεριφορές που θα επιτρέπουν στο Τομ να αποκτά ενέργεια από τις Καρδιές. Αν τις βάλουμε στις Καρδιές, θα εισάγουμε μια φορά τις συμπεριφορές, ενώ αν τις εισάγαμε στον Μηχανάκια θα έπρεπε να τις επαναλαμβάνουμε σε κάθε σελίδα συμπεριφορών του. Χμ! Καλύτερα στις καρδιές και στην συνέχεια με αντιγραφή και επικόλληση, θα τοποθετήσουμε Καρδιές σε όποια σημεία της πίστας επιθυμούμε. Επομένως, στην πρώτη σελίδα μιας Καρδιάς:

Δορυφόρος

Πώς θα προγραμματίσουμε τους Δορυφόρους, έτσι ώστε να εμφανίζονται και να εξαφανίζονται σε τυχαία χρονικά διαστήματα, πυροβολώντας τον Kodu; Σίγουρα θα χρειαστούμε τις εντολές *Χρονόμετρο (Timer)*, *Τυχαία (Random)* και *Έκρηξη (Boom)* (ή *Εξαφανίζω (Vanish)*). Αν όμως κάνουμε το χαρακτήρα μας να εξαφανίζεται, πώς θα τον επαναφέρουμε στη ζωή και σε τυχαίο μάλιστα χρονικό διάστημα;

Η λύση ακούει στο όνομα... *κλώνοι (creatables)*! Δημιουργούμε δύο χαρακτήρες (π.χ. ένα Νόμισμα και το Δορυφόρο μας) σε μορφή *κλώνου (creatable)* και τους προγραμματίζουμε έτσι ώστε όταν καταστρέφουν τον εαυτό τους, να δημιουργεί νέους!

Καταρχήν, όταν ο Δορυφόρος μας είναι "εν ζωή", πρέπει να πυροβολεί το κόκκινο Τζετ που είναι κοντά του. Επίσης, θα το βάλουμε να περιφέρεται σε ένα τρίγωνο κίτρινο μονοπάτι έτσι ώστε να προστατεύει την περιοχή:



Για παράδειγμα, έστω ότι ένας δορυφόρος δημιουργεί ένα αόρατο Νόμισμα (γράφουμε εκεί τις εντολές προς εκτέλεση!):

Νόμισμα (σε συνδυασμό με το Δορυφόρο)

Στο Νόμισμα πρέπει να εισάγουμε την τυχαιότητα. Σίγουρα θα σας έχει τύχει παίζοντας διάφορα παιχνίδια «να μην ξέρετε από πού σας έρχονται»! Και όχι μόνο αυτό, αλλά κάθε φορά που παίζετε την ίδια ακριβώς πίστα στο ίδιο ακριβώς σημείο, «οι εχθροί» σας να είναι σε διαφορετικές θέσεις! Τυχαία! Αυτό θα κάνουμε και εμείς. Όσο ο Δορυφόρος είναι εξαφανισμένος (άρα εκτελείται το πρόγραμμα του Νομίσματος) και με τυχαία καθυστέρηση (ανάμεσα σε 0.25 και 2 δευτερόλεπτα), το Νόμισμα θα δημιουργεί το Δορυφόρο:

Παρατηρείτε κάποια έλλειψη; Στον προγραμματισμό πάντα ένα σημαντικό κομμάτι είναι η αρχικοποίηση. Εδώ ποιο αντικείμενο θα τρέξει πρώτο; Ο Δορυφόρος ή το Νόμισμα;

Αφού και τα δύο είναι *κλώνοι (creatables)* και δεν υπάρχουν στο παιχνίδι, πρέπει κάποιος να τα δημιουργήσει. Γι' αυτό το σκοπό δημιουργούμε άλλο ένα αντικείμενο (έστω άλλο ένα φανερό Νόμισμα) το οποίο θα κάνει την αρχή της αλυσιδωτής μας αντίδρασης με το που πλησιάσει το Τζετ κοντά του.

Τζετ

Επιτέλους και ένα αντικείμενο που δε θα μας ταλαιπωρήσει! Οι ενέργειες του Τζετ είναι σχετικά απλές. Το Τζετ πρέπει να μετακινείται με τα βέλη του πληκτρολογίου και να ανεβοκατεβαίνει όποτε το επιθυμεί ο χρήστης (επίσης με το πληκτρολόγιο). Το ιδιαίτερο χαρακτηριστικό του είναι ότι μπορεί να κουβαλήσει την Μπάλα και να την πετάξει όταν του ζητηθεί!

Είστε έτοιμοι.

Παραλλαγές

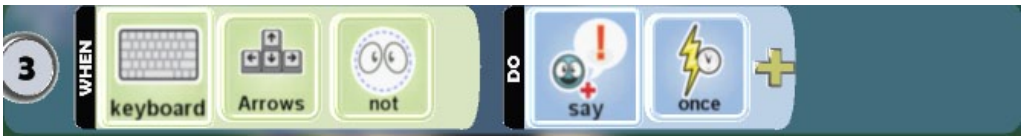
1. Στα διάφορα αντικείμενα έχουμε εισάγει περισσότερες σελίδες συμπεριφορών από αυτές που πραγματικά χρειαζόμασταν. Μπορείτε να βρείτε τους αντίστοιχους πλεονασμούς και να τους διορθώσετε;

2. Αποσφαλμάτωση (Debugging) του κώδικα. Στο παιχνίδι μας προγραμματίσαμε τον κόκκινο Μηχανάκια να μετατρέπεται σε Τζετ όταν ακουμπά την Καλύβα και αφού έχει νικήσει στον αγώνα ταχύτητας. Όμως υπάρχει ένα σενάριο εκτέλεσης όπου ο κώδικας δε συμπεριφέρεται σύμφωνα με τις προδιαγραφές που έχουμε θέσει. Στον προγραμματισμό, αυτή η περίπτωση αποτελεί ένα **σφάλμα λογισμικού (software bug)** του προγράμματος. Συγκεκριμένα στην παραπάνω υλοποίηση του παιχνιδιού υπάρχει ένα σενάριο εκτέλεσης του προγράμματος όπου ο κόκκινος Μηχανάκιας καταφέρνει να βγει εκτός πίστας αγώνα χωρίς να μετατραπεί σε Τζετ. Καλείστε λοιπόν να βρείτε αυτό το σενάριο, να εξηγήσετε γιατί συμβαίνει και να το διορθώσετε.

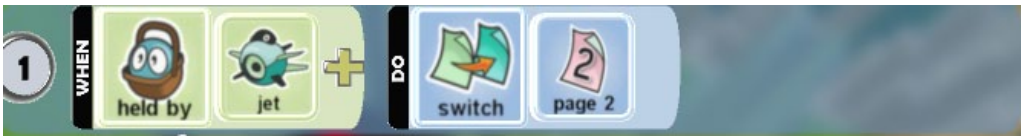
Παράρτημα – Παραδειγματικές συμπεριφορές

Μόλις η ενέργεια του χαρακτήρα πέσει κάτω από τις δυο μονάδες, θα δείξει λυπημένος!

Όταν πέσεις πάνω στο εργοστάσιο, μεταφέρσου στις συμπεριφορές της σελίδας δύο.

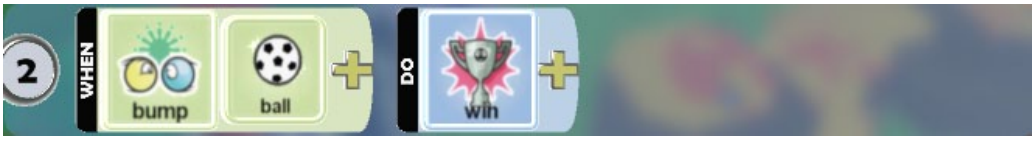


Όσο δεν πατάμε κάποιο από τα βελάκια κατεύθυνσης, ο χαρακτήρας μας εμφανίζει ένα μήνυμα για μια φορά.



Αν ο χαρακτήρας μας κουβαλιέται από το Τζετ, μεταφερόμαστε στις συμπεριφορές της σελίδας 2.

Μόλις ο χαρακτήρας μας πέσει πάνω στη μπάλα, θα εκραγεί.



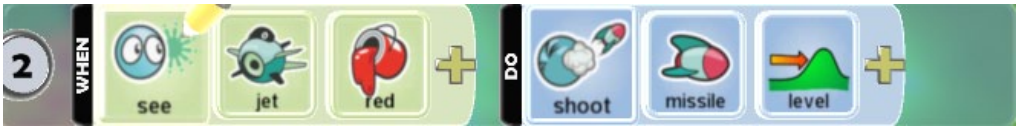
Νίκη παιχνιδιού μόλις ο χαρακτήρας πέσει πάνω στη μπάλα.

Όταν ο χαρακτήρας μας δει το κόκκινο Τζετ κοντά του, του δίνει 100 πόντους ενέργειας.

Όταν πάρουμε ένα αστέρι, τότε δημιουργούμε ένα νόμισμα.

Σε δύο δευτερόλεπτα, μετάβαση για στη δεύτερη σελίδα συμπεριφορών.

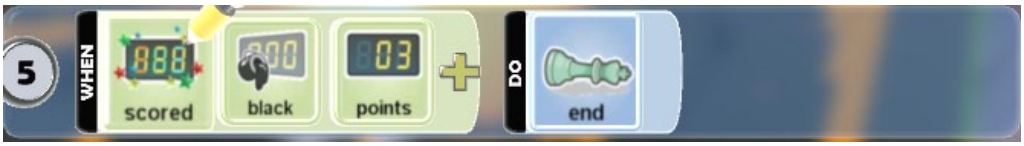
Μόλις πετύχουμε τον δορυφόρο, του δίνουμε 10 πόντους ζωής!



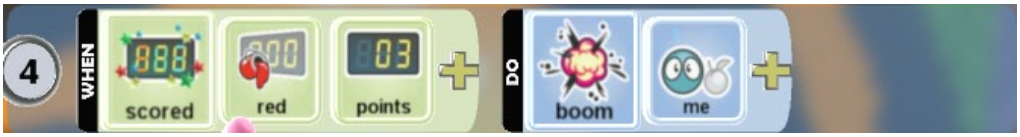
Αν ο χαρακτήρας μας δει το κόκκινο Τζετ, του ρίχνει πύραυλο.

Όταν ο χαρακτήρας μας ακούσει οτιδήποτε, εξαφανίζεται.

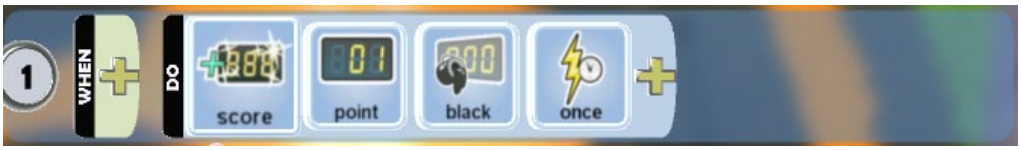
Όταν δούμε κοντά μας και από αριστερά τον κόκκινο μηχανάκια, του ρίχνουμε έναν πύραυλο.



Όταν η μαύρη ομάδα έχει τρεις πόντους, τελειώνει το παιχνίδι.

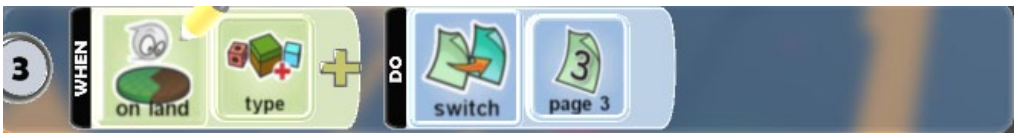


Μόλις ο χαρακτήρας μαζέψει τρεις πόντους στο κόκκινο σκορ, εκρήγνυται.



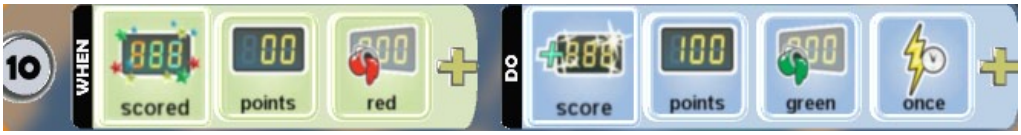
Δίνουμε έναν πόντο στην μαύρη ομάδα.

Ο χαρακτήρας μας κινείται στο μαύρο μονοπάτι διαρκώς.



Μόλις περάσουμε πάνω από έναν συγκεκριμένο τύπο εδάφους, μεταβαίνουμε στην τρίτη σελίδα συμπεριφορών.

Κάθε τρία δευτερόλεπτα, ο χαρακτήρας μας εμφανίζει ένα μήνυμα.



Όταν το κόκκινο σκορ είναι μηδέν, πρόσθεσε 100 πόντους στο πράσινο σκορ.

Μόλις χτυπήσουμε τον μαύρο μηχανάκια από πίσω, ζάλισέ τον για λίγο.

Αν πατήσουμε το πλήκτρο Z, ρίχνουμε έναν πύραυλο μπροστά.

Σε τυχαίο χρόνο ανάμεσα σε 0.25 και 2 δευτερολέπτων, εκτέλεσε τις εντολές της σελίδας δύο.

Κάθε 60 δευτερόλεπτα δημιούργησε ένα αστέρι.

Κάθε 120 δευτερόλεπτα εκτόξευσε ένα μήλο με δυτική κατεύθυνση και χαμηλά.

Όταν το μαύρο σκορ γίνει 7 τότε, μεταφέρσου στη δεύτερη σελίδα συμπεριφορών.

(Συνέχεια) κινήσου πάνω στο μονοπάτι πολύ γρήγορα.

Όταν ακούσεις τον μπλε Kodu, κινήσου προς το μέρος του.

Όταν οι πόντοι ζωής είναι κάτω από 200, θα ζαλιστείς για μια φορά.

Όταν πέσεις πάνω στον Kodu, άρπαξε τον.

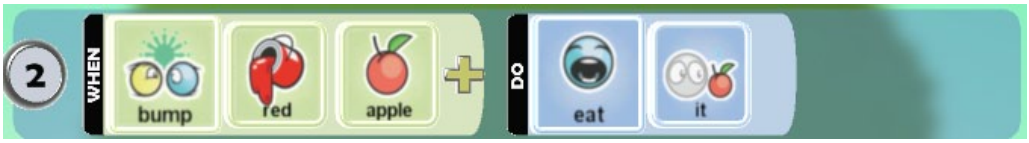
Πάντα η κάμερα να ακολουθεί τον παίκτη.

Όταν είσαι σε ένα συγκεκριμένο τύπο νερού, μειώνεται η ενέργεια σου κατά 3 πόντους.

Όταν πατήσεις το αριστερό κουμπί του ποντικιού, πυροβόλησε ένα σφαιρίδιο.

Όταν πατήσουμε τα πλήκτρα W,A,S,D κινήσου προς την αντίστοιχη κατεύθυνση πολύ γρήγορα.

Όταν δεις μπλε μπαλόνι, πες κάτι.



Όταν πέσεις πάνω σε κόκκινο μήλο, να το φας

Όταν πατήσω το πλήκτρο M, παίξε έναν οποιοδήποτε ήχο.

Όταν δεις κανόνι, γύρνα αριστερά γρήγορα.

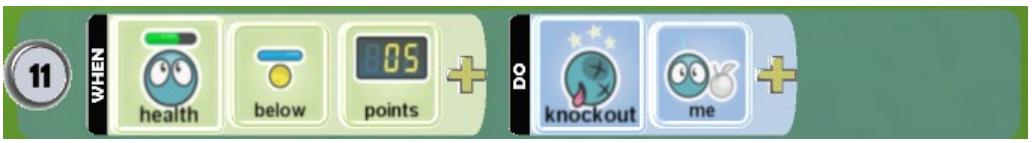
Όταν το μπλε σκορ είναι 100, επανεκκίνησε το παιχνίδι.

Όταν πέσεις πάνω σε καρδιά, αύξησε την ενέργεια σου κατά 100 πόντους.

Όταν δεις τον ροζ Kodu κοντά σου, εμφάνισε καρδούλες γύρω μου.

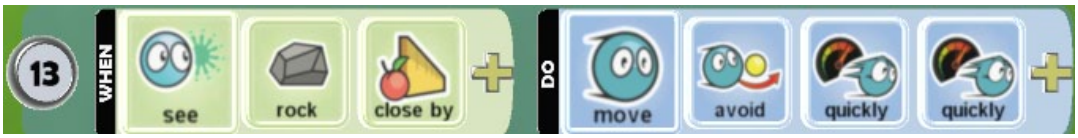
Όταν πατήσεις το πλήκτρο F, σταμάτα να κινείσαι.

Πάντα θα περιφέρεσαι πολύ γρήγορα.



Όταν η ενέργειά σου είναι κάτω από 5 μονάδες, τότε βγαίνεις νοκ άουτ.

Όταν ακούσεις το μαύρο Kodu κοντά σου, φύγε μακριά του γρήγορα.



Όταν δεις ένα βράχο κοντά σου, απέφυγε τον, πολύ γρήγορα.

Όταν δεις οποιοδήποτε μαύρο αντικείμενο, πυροβόλησε ένα πύραυλο κρουζ.