ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΙΣ ΨΗΦΙΑΚΕΣ ΕΠΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ

ΑΝΑΠΤΥΞΗ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΟΥ ΣΕ ΜΗΧΑΝΗ UNITY ΓΙΑ ANDROID ΣΥΣΚΕΥΕΣ

ΚΑΣΒΙΚΗΣ ΗΛΙΑΣ: ΜΨΕ 1607

Επιβλέπων Καθηγητής : Απόστολος Μηλιώνης

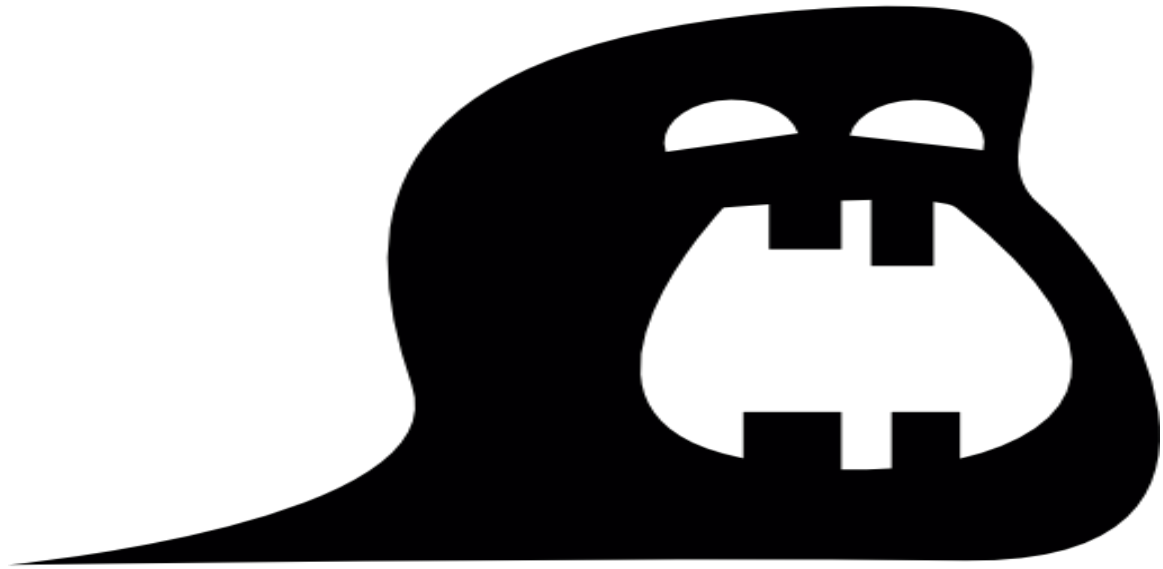
Διπλωματική Εργασία υποβληθείσα στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς ως μέρους των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στις Ψηφιακές Επικοινωνίες και Δίκτυα

Πειραιάς, Οκτώβριος 2018

ΑΝΑΠΤΥΞΗ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΟΥ ΣΕ ΜΗΧΑΝΗ UNITY ΓΙΑ ANDROID ΣΥΣΚΕΥΕΣ

Περίληψη

Σκοπός αυτής της πτυχιακής εργασίας είναι η γνωριμία και η ανάπτυξη βιντεοπαιχνιδιού σε πλατφόρμα Unity. Παρακάτω γίνεται παρουσίαση του παιχνιδιού καθώς και των βημάτων που πραγματοποιήθηκαν για την υλοποίηση του. Αυτή περιελάμβανε εγκατάσταση του απαραίτητου λογισμικού, σχεδίαση γραφικών, ανάπτυξη κώδικα προγραμματισμού και δοκιμή της εφαρμογής σε android συσκευή. Για την καλύτερη περιγραφή του θέματος, της εμφάνισης και λειτουργικότητας της εφαρμογής παρατίθονται screenshots καθώς και κώδικας με λεπτομερή επεξήγηση. Επίσης αναφέρονται προβλήματα που εμφανίστηκαν κατά τη δειάρκεια της υλοποίησης και ο τρόπος αντιμετώπισής τους.



Εικόνα 1-Το icon του παιχνιδιού

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

1.ΕΙΣΑΓΩΓΗ

Λόγω έλλειψης προγενέστερης εμπειρίας στην ανάπτυξη εφαρμογών σε πλατφόρμα Unity, αποφάσισα να επεκτείνω ένα υπάρχον παιχνίδι. Το παιχνίδι αυτό ονομάζεται ‘Flappy Bird’, στο οποίο ο παίχτης κατά τη διάρκεια της πτήσης προσπαθεί να περάσει ανάμεσα απο δύο σωλήνες οι οποίοι αλλάζουν συνεχώς μήκος. Είναι πολύ εύκολο στη χρήση του, καθώς απαιτεί το πάτημα ενός μόνο πλήκτρου καθόλη τη διάρκειά του. Ο σκοπός του είναι να διανυθεί όσο το δυνατόν μεγαλύτερη απόσταση και κατά συνέπεια να επιτευθεί υψηλότερο σκορ.

Υπάρχουν διάφοροι λόγοι που επέλεξα το συγκεκριμένο παιχνίδι για τη διπλωματική μου εργασία. Αρχικά, όταν είχε κυκλοφορήσει είχε τεράστια απήχηση και ήμουν κι εγώ ένας απο τους πολλούς που είχαν αφιερώσει αρκετό χρόνο προσπαθώντας να σημειώσουν όσο το δυνατόν υψηλότερο σκορ. Επίσης ήμουν περίεργος για τον τρόπο και τη δυσκολία της υλοποίησής του. Τέλος, μου φάνηκε πολύ ενδιαφέρουσα η δημιουργία μιας διαφορετικής εκδοχής ενός τόσο δημοφιλούς παιχνιδιού, με τις αλλαγές που είχα σκεφτεί ενώ ήμουν ακόμα παίκτης.

Οι τροποποιήσεις σε σχέση με το αρχικό παιχνίδι αφορούν τόσο την εμφάνιση όσο και τη λειτουργικότητα. Συγκεκριμένα:

- Τα γραφικά.
- Η σταδιακή αλλαγή της ταχύτητας του παιχνιδιού και η απόσταση μεταξύ των σωλήνων, αυξάνωντας έτσι και τη δυσκολία.
- Η προσθήκη μικρής διάρκειας κατά την οποία οι σωλήνες αλλάζουν χρώμα και ο χαρακτήρας μπορεί να περάσει από μέσα τους χωρίς να χάσει.

1.1 Τι είναι το λογισμικό android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις, αυτοκίνητα και ρολόγια χειρός. Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ και σε άλλες ηλεκτρονικές συσκευές.Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Ιρίνα Μπλόκ.

Η τελευταία έκδοση καλείται Android 9.0 Pie και φέρνει σημαντικές αλλαγές, όπως το Adaptive Battery και το Adaptive Brightness. Επίσης, στο Android 9.0 Pie υπάρχει ένα νέο σύστημα πλοήγησης με ένα μεμονωμένο κουμπί home.

Επίσημες εκδόσεις android και ημερομηνίες πρώτης κυκλοφορίας

N/A	1.0	23 Σεπτεμβρίου 2008	1
	1.1	9 Φεβρουάριου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5 – 7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9 – 10
Honeycomb	3.0 – 3.2.6	22 Φεβρουάριου 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16 – 18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19 – 20
Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21 – 22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	7.0 – 7.1.2	22 Αυγούστου 2016	24 – 25
Oreo	8.0 – 8.1	21 Αυγούστου 2017	26 – 27
Pie	9.0	6 Αυγούστου 2018	28

Η Πρώτη έκδοση του android κυκλοφόρησε στις 26/9/2009 και τα εξής χαρακτηρισικα:

- Προεγκατεστημένες εφαρμογές όπως playStore και gmail

Η Δεύτερη έκδοση του android κυκλοφόρησε στις 26/9/2009 με την ονομασία Éclair και τα εξής χαρακτηρισικα:

- Περισσότερη μνήμη
- Προσθήκη Javascript Engine στον Chrome
- Μεγαλύτερη ασφάλεια κωδικών
- Υποστήριξη για το adobe Flash
- Υποστήριξη σε μεγαλύτερες οθόνες
- Δυνατότητα copy/paste
- Ευκολότερη συγγραφή κειμένου

Η Τρίτη έκδοση του android κυκλοφόρησε στις 18/10/2011 με την ονομασία Honeycomb και τα εξής χαρακτηρισικα:

- Συνδεσιμότητα με USB εξαρτήματα
- Υποστήριξη εξωτερικού πληκτρολογίου
- Βελτιωμένη υποστήριξη στο Hardware
- Μεγαλύτερο έλεγχο στους προγραμματιστές

Η Τέταρτη έκδοση του android κυκλοφόρησε στις 18/10/2011 με την ονομασία Ice Cream και τα εξής χαρακτηρισικα:

- Απευθείας πρόσβαση σε εφαρμογές απο την οθόνη κλειδώματος
- Βελτιώσεις στο UI του hardware
- Απαλότερο user interface
- Βελτιστοποίηση του κλειδώματος οθόνης με τη χρήση widget support
- Ομαδοποίηση μηνυμάτων
- Μεγαλύτερη χωρητικότητα στις μορφές ειδοποιήσεων

Η Πέμπτη έκδοση του android κυκλοφόρησε στις 12/11/2014 με την ονομασία Lollipop και τα εξής χαρακτηρισικα:

- Πρόσβαση στο menu γρήγορων ρυθμίσεων, από την οθόνη κλειδώματος
- Διαμόρφωση γρήγορων ρυθμίσεων
- Προσθήκη προστασίας συσκευής
- Πραγματοποίηση κλήσεων με ήχο HD
- Ήχος υψηλής ευκρίνειας,
- Ενεργοποίηση της λειτουργίας προτεραιότητας

Η Έκτη έκδοση του android κυκλοφόρησε στις 5/10/2015 με την ονομασία Marshmallow και τα εξής χαρακτηρισικα:

.

- Υποστήριξη USB Type-C
- Επιστροφή του Silence Mode
- Μόνιμη εμφάνιση των εργαλείων Copy-Paste επάνω από το κείμενο
- Δυνατότητα ρύθμισης του ήχου σε διαφορετικά επίπεδα για multitasking (ήχος κλήσης, ήχος εφαρμογής, ήχος ειδοποιήσεων)

- Κάθετη εμφάνιση εφαρμογών με δυνατότητα αναζήτησης και επισήμανσης αγαπημένων
- Εμφάνιση όλων των επιλογών στις Ρυθμίσεις
- Νέα λίστα εμφάνισης της κατανάλωσης μνήμης RAM για κάθε εφαρμογή

Η Έβδομη έκδοση του android κυκλοφόρησε στις 22/8/2016 με την ονομασία Nougat και τα εξής χαρακτηρισικα:

- μενού γρήγορων ρυθμίσεων και οι ειδοποιήσεις
- Πολλαπλά παράθυρα
- Λειτουργία απόδοσης

Η Όγδοη έκδοση του android κυκλοφόρησε στις 21/8/2017 με την ονομασία Oreo και τα εξής χαρακτηρισικα:

- Picture-in-picture
- Autofill API
- Κανάλια ειδοποιήσεων
- Snooze ειδοποιήσεων
- Background limits
- Νεο design στο μενού ρυθμίσεων
- Νεο design στο μενού μπαταρίας
- Έξυπνη επιλογή κειμένου
- Αυτόματο Wi-Fi
- Γρηγορότερο boot
- Καλύτερο performance
- Καλύτερο Bluetooth

Η Ένατη και τελευταία μέχρι σήμερα έκδοση του android κυκλοφόρησε στις 7/8/2018 με την ονομασία Lollipop και τα εξής χαρακτηρισικα:

- Τη λειτουργία Adaptive Battery, η οποία δίνει προτεραιότητα στην κατανάλωση της μπαταρίας μόνο για τις εφαρμογές και τις υπηρεσίες που χρησιμοποιεί περισσότερο ο χρήστης, κλείνοντας ταυτόχρονα όλες τις υπόλοιπες εφαρμογές που "τρέχουν" στο background.
- Τη λειτουργία Adaptive Brightness, η οποία μαθαίνει πως να προσαρμόζει την φωτεινότητα της οθόνης ανάλογα με το περιβάλλον.
- Τα App Actions που βοηθούν το χρήστη να πραγματοποιεί την επόμενη ενέργεια του πιο γρήγορα επειδή θα αντιλαμβάνονται τι ακριβώς θέλεις να κάνει.
- Η εμφάνιση και η εμπειρία χρήσης γίνεται πιο εύκολη με το νέο σύστημα πλοήγησης.
- Το Android Dashboard, το οποίο θα δείχνει στον χρήστη αναλυτικά πόσο χρόνο ξοδεύει σε συγκεκριμένες εφαρμογές, πόσες φορές, ξεκλείδωσε το smartphone και πόσες ειδοποιήσεις έλαβε κατά τη διάρκεια της ημέρας.
- Το App Timer είναι ένα εργαλείο που θα του επιτρέπει να θέτει χρονικό περιορισμό στην χρήση μιας εφαρμογής, θα τον ειδοποιεί όταν φτάνει στο όριο που έχει θέσει και θα μετατρέπει το εικονίδιο της εφαρμογής σε ασπρόμαυρο για να του το υπενθυμίσει
- Η λειτουργία Wind Down θα ενεργοποιεί το Night Light όταν σκοτεινιάζει, θα βάζει την συσκευή σε Do Not Disturb mode και θα μετατρέπει την οθόνη σε ασπρόμαυρη την ώρα που έχει δηλώσει ο χρήστης ότι βρίσκεται στο κρεβάτι.

1.2 Unity και κατηγορίες παιχνιδιών

Η Unity είναι μια μηχανή ανάπτυξης ηλεκτρονικών παιχνιδιών, δισδιάστατων ή τρισδιάστατων, συμβατών με πολλές πλατφόρμες, όπως υπολογιστές και κινητά με λογισμικό Android ή iOS.

Ο χρήστης μπορεί να βλέπει όσα υλοποιεί σε πραγματικό χρόνο και έτσι μπορεί και ζει το παιχνίδι την ώρα που το δημιουργεί. Ένα μεγάλο πλεονέκτημα είναι ότι προσφέρει πολλές επιλογές και δυνατοτήτες ώστε τα αντικείμενα και οι χαρακτήρες να αποκτήσουν τη συμπεριφορά που θέλει ο χρήστης. Επίσης η σύνδεσή του με code editors όπως το Visual Studio και ο ενσωματωμένος MonoDevelop που προσφέρει ευκολία στη εγγραφή κώδικα για τη σωστή λειτουργία του παιχνιδιού.

Εικόνα 2-Το λογότυπο του προγράμματος unity

Η πρώτη έκδοση της Unity παρουσιάστηκε το 2005 και ήταν διαθεσιμη μόνο για λογισμικά OS X.

Το 2007 παρουσιάστηκε η δεύτερη έκδοση με νέες δυνατότητες και με τον Unity Asset Server απο τον οποίο οι χρήστες μπορούν να κατεβάζουν υλικό το οποίο είναι συμβατό με το Unity.

Το 2009 η Unity υποστηρίζει παιχνίδια συμβατά με τα Windows της Microsoft

Το 2010 παρουσιάστηκε η Τρίτη έκδοση η οποία είχε συμβατότητα με android συσκευές και λίγο αργότερα οι εκδόσεις 3.4 και 3.5 υποστήριζαν γραφικα 3D

Το 2012 κυκλοφόρησε η τέταρτη έκδοση η οποία μπορούσε να υποστηρίξει 64 bit εφαρμογές στο IOS

ΤΟ 2015 κυκλοφόρησε η Πέμπτη έκδοση και πλέον υπάρχουν πάρα πολλές εκδόσεις οι οποίες έχουν την ονομασία της χρονολογίας που κυκλοφόρησαν και ανανεώνονται κάθε μήνα.

Η τελευταία έκδοση του Unity είναι συμβατή με πάρα πολλές πλατφόρμες διαφορετικού είδους οι οποίες ανήκουν στις κατηγορίες των υπολογιστών, των κινητών συσκευών, και των βιντεοπαιχνιδιών σε κονσόλες.

Σε αυτές τις κατηγορίες η Unity έχει συμβατότητα στα εξής:

Κινητές συσκευές: Apple- iOS- Google- Android- Microsoft-Windows phone-Blackberry

Υπολογιστές:Windows-MAC-Linux

Κονσόλες:Playstation 3-Playstation 4-Playstation Vita-Xbox One-Xbox 360-Wii

Έτσι λοιπόν με τόσες επιλογές που μας δίνει η Unity αντίστοιχα υπάρχουν πολλές κατηγορίες video games.Οι πιο διαδεδομένες είναι οι παρακάτω:

- Sport games
- Role-playing games
- Action games
- Strategy games
- Indie games
- Adventure games
- Vehicle simulation games
- Serious Games
- Lan and online games
- Pc games
- Console games
- Mobile games

1.3 Οδηγίες εγκατάστασης της πλατφόρμας

Αρχικά χρειάστηκε να εγκαταστήσω τη πλατφόρμα του Unity. Όταν θα δοκιμάσει κάποιος να εγκαταστήσει το πρόγραμμα υπάρχουν οι επιλογές Personal, Plus, Pro και Enterprise Edition. Η επιλογή Personal Edition είναι δωρεάν και είναι αρκετή για το στήσιμο ενός παιχνιδιού. Αυτό που πρέπει να προσέξουμε είναι στις επιλογές που μας δίνει για την εγκατάσταση να διαλέξουμε τα Unity 5, 5, 1f1, το Standard Assets και για να εγκατασταθεί το παιχνίδι σε ένα κινητό ή tablet με android λογισμικό την επιλογή Android Build Support. Αν κάποιος θέλει να δημιουργήσει ένα παιχνίδι σε λογισμικό IOS θα πρέπει να επιλέξει το IOS Build Support αντίστοιχα.

Αφού τελειώσει η εγκατάσταση το δεύτερο βήμα ειναι να κατεβάσουμε το JDK (Java Development Kit). Όταν εγκατέστησα το JDK 9 παρατήρησα ότι υπήρχε δυσλειτουργία όταν προσπαθούσα να περάσω την εφαρμογή στο κινητό μου, κάτι που διορθώθηκε με το JDK 8. Ίσως αυτό συνέβαινε γιατί το JDK 9 είχε μόλις κυκλοφορήσει και δεν είχαν γίνει γνωστά τέτοιου είδους προβλήματα. Μετά την εγκατάσταση του JDK υπάρχουν ρυθμίσεις που πρέπει να γίνουν σχετικά με αυτό. Αρχικά λοιπόν μπαίνουμε στο φάκελο του JDK που βρίσκεται μέσα στα Program files ή όπου έχουμε επιλέξει να το αποθηκεύσουμε και ανοίγουμε τον φάκελο bin. Έπειτα κάνουμε δεξί κλικ στο εικονίδιο appletviewer και πατάμε properties. Εκεί λοιπόν κάνουμε αντιγραφή το location που είναι το (C:\Program Files\Java\jdk1.8.0_161\bin) και μετα πρέπει να δημιουργήσουμε δυο νέα environments στον υπολογιστή μας. Για να γίνει αυτό πηγαίνουμε στο My Computer πατάμε το Properties, στη συνέχεια το Advanced System Settings και τέλος το Environment Variables. Εκεί μέσα λοιπόν δημιουργούμε τα εξής environments:

1.Variable name: path

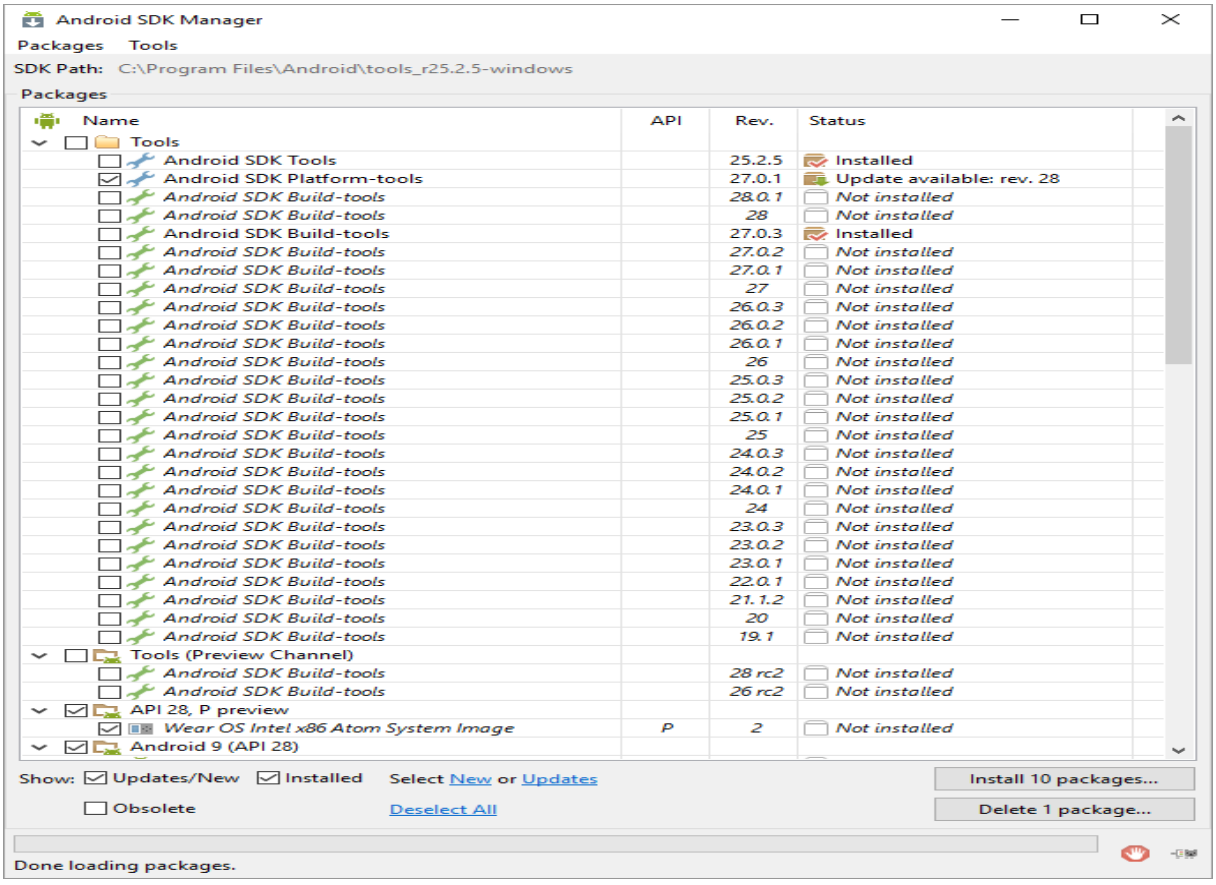
Variable value: C:\Program Files\Java\jdk1.8.0_161\bin

2. Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jdk1.8.0_161

Τρίτο βήμα είναι να εγκαταστήσουμε τον Android SDK. Ένας τρόπος για να δουλέψει ο SDK και να γίνουν τα κατάλληλα updates είναι αφού κατεβάσουμε το αρχείο μπαίνουμε στον φάκελο tools\lib κάνουμε copy το αρχείο SDK MANAGER και στη συνέχεια paste στον φάκελο tools. Με αυτή τη κίνηση αν τώρα κάνουμε διπλό κλικ στο αρχείο android.bat που βρίσκεται στον ίδιο φάκελο μας ανοίγει το παρακάτω παράθυρο στην εικόνα 3 στο οποίο μπορούμε να επιλέξουμε όλα τα updates που μας είναι απαραίτητα. Αυτά που πρέπει οπωσδήποτε να κάνουμε install είναι τα εξής: Android SDK tools, Android SDK platform tools, Android SDK build tools, Android SDK platform και Google USB driver.

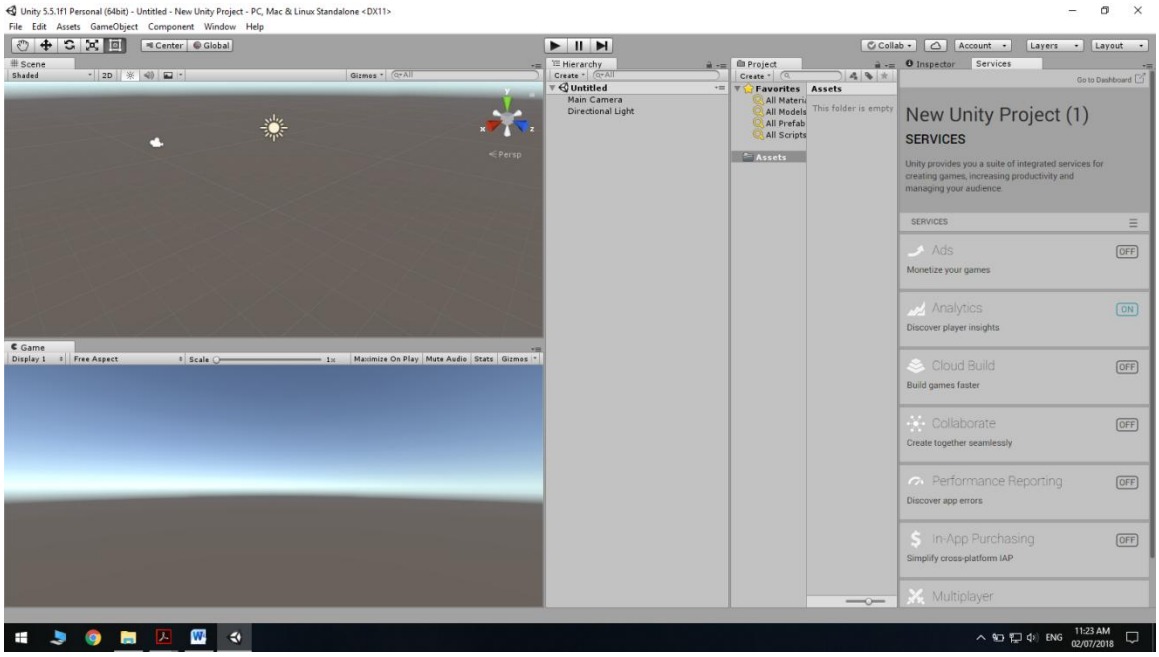
Τέλος για να μπορέσουμε να εγκαταστήσουμε την εφαρμογή στο κινητό μας πρέπει οπωσδήποτε να έχουμε ενεργοποιημένες τις επιλογές ‘Εντοπισμός σφαλμάτων USB’ και ‘Εγκατάσταση μέσω USB’. Αυτά φυσικά διαφέρουν απο κινητό σε κινητό ως προς το μενού που βρίσκονται.



Εικόνα 3-Updates για τον SDK Manager

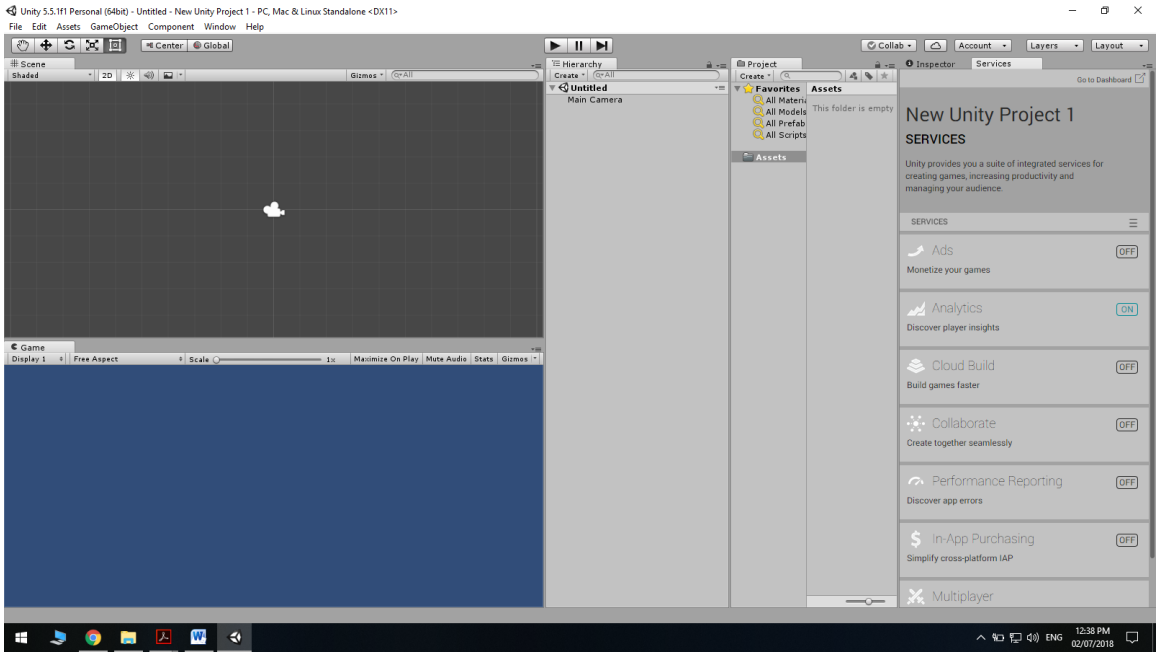
1.4 Περιβάλλον εργασίας

3D



Εικόνα 4-Περιβάλλον εργασίας Unity 3D

2D

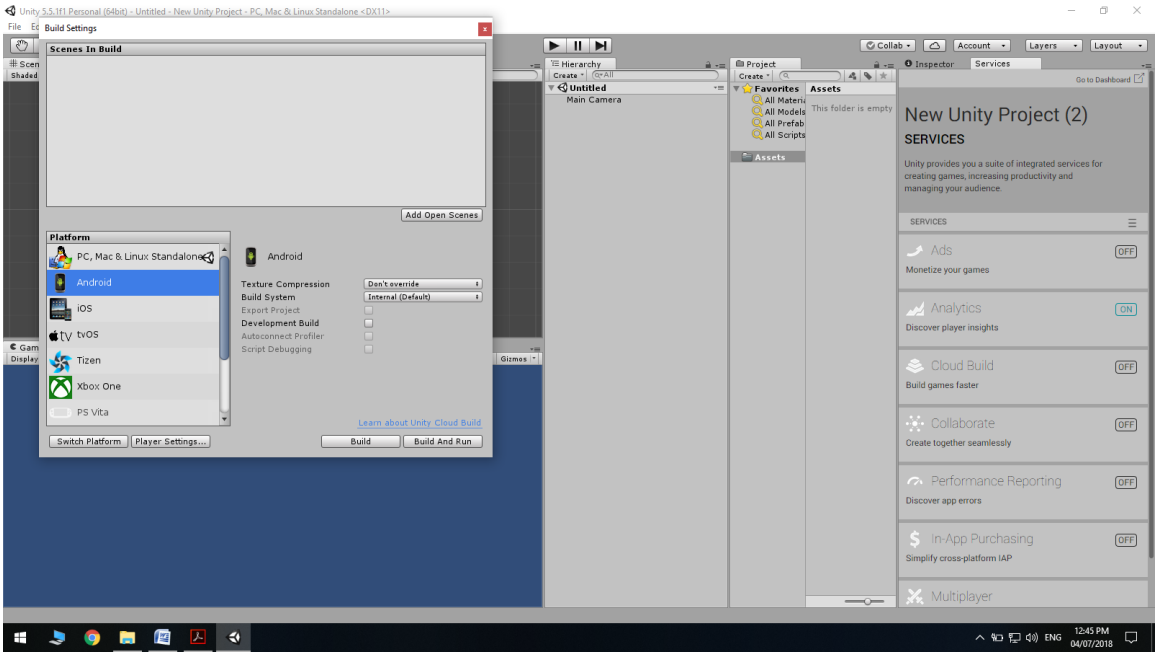


Εικόνα 5-Περιβάλλον εργασίας Unity 2D

1.5 Ρυθμίσεις της πλατφόρμας Unity

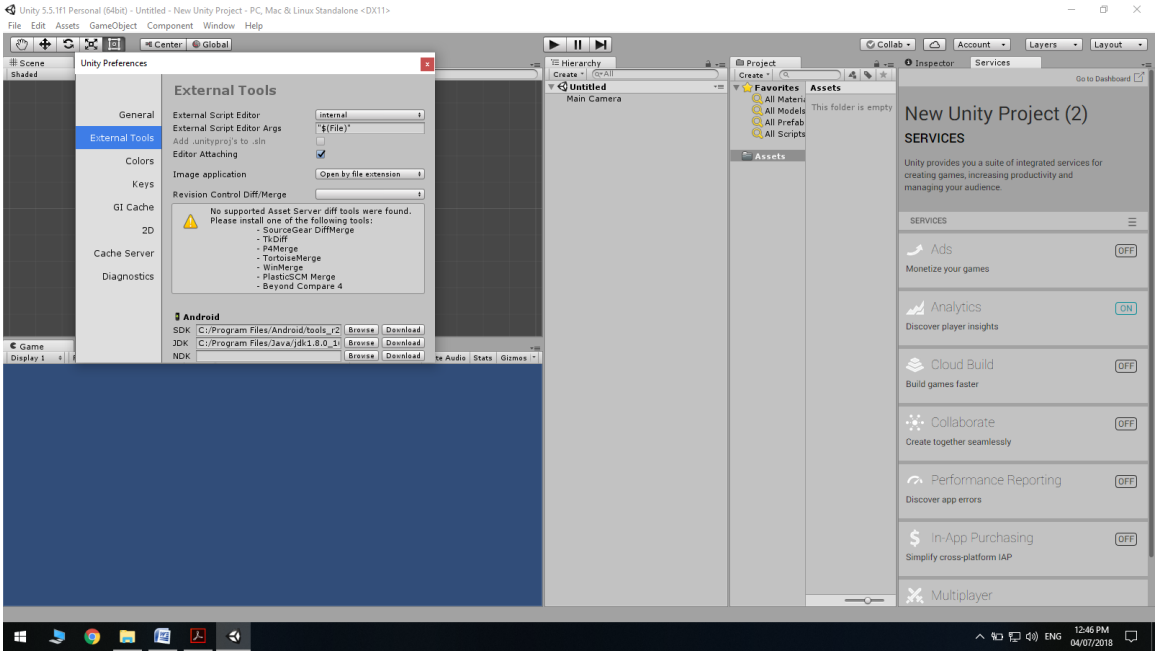
Σε αυτό το κεφάλαιο θα δούμε τι χειάζεται να κάνουμε πριν ξεκινήσουμε να φτιάχνουμε το παιχνίδι μας.

ΒΗΜΑ 1: Ανοίγουμε το Unity και επιλέγουμε File-Build Settings και Android έτσι ώστε να αλλάξει το περιβάλλον και να είναι συμβατό για την εφαρμογή που θέλουμε να φτιάξουμε σε Android.



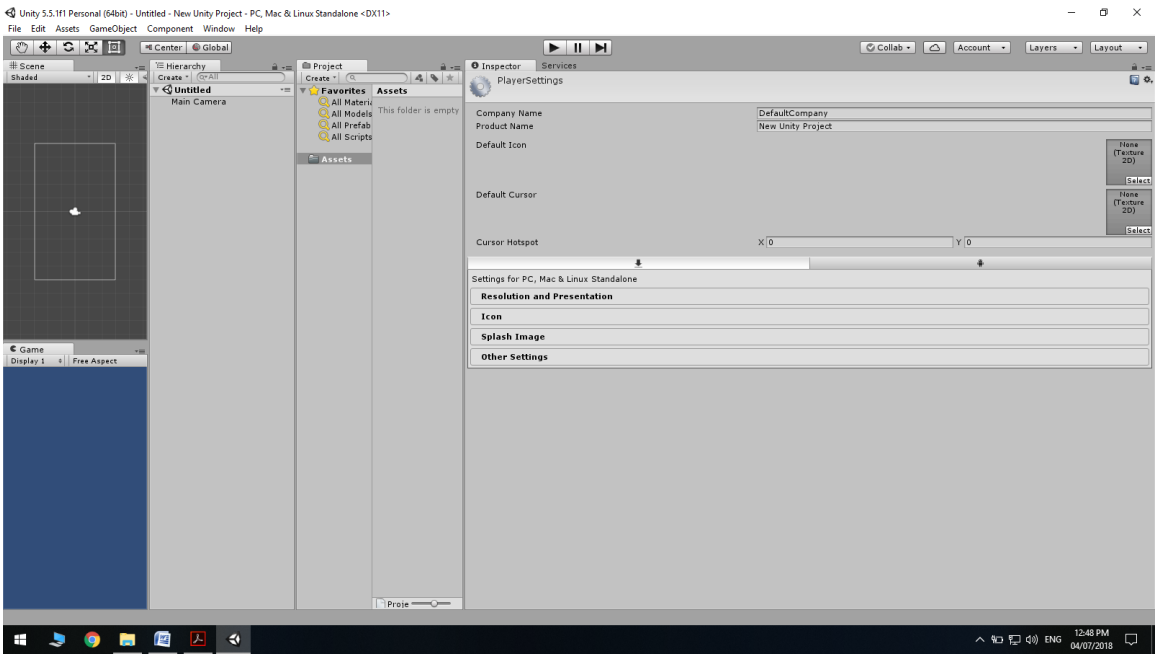
Εικόνα 6-Ρυθμίσεις Unity για περιβάλλον συμβατό με android

ΒΗΜΑ 2: Για να δώσουμε στο Unity Editor τον JDK και τον SDK που έχουμε κατεβάσει στον υπολογιστή, πρέπει να επιλέξουμε μέσα από το Unity τους φακέλους στους οποίους βρίσκονται. Αυτό γίνεται ως εξής: Edit > Preferences > External tools και εκεί κάνουμε Browse τους αντίστοιχους φακέλους.



Εικόνα 7-Ρυθμίσεις εισαγωγής JDK και SDK στην πλατφόρμα

ΒΗΜΑ 3: Για να εκδωθεί το παιχνίδι υπάρχουν κάποιες αλλαγές που πρέπει να γίνουν. Επιλέγουμε λοιπόν File > Build Settings > Player > Settings, συμπληρώνουμε αρχικά το όνομά μας και το όνομα της εφαρμογής και στη συνέχεια στα Other settings αλλάζουμε το bundle identifier το οποίο πρέπει να είναι αυτής της μορφής: com.IliasKasvikis.FlappyMonster.

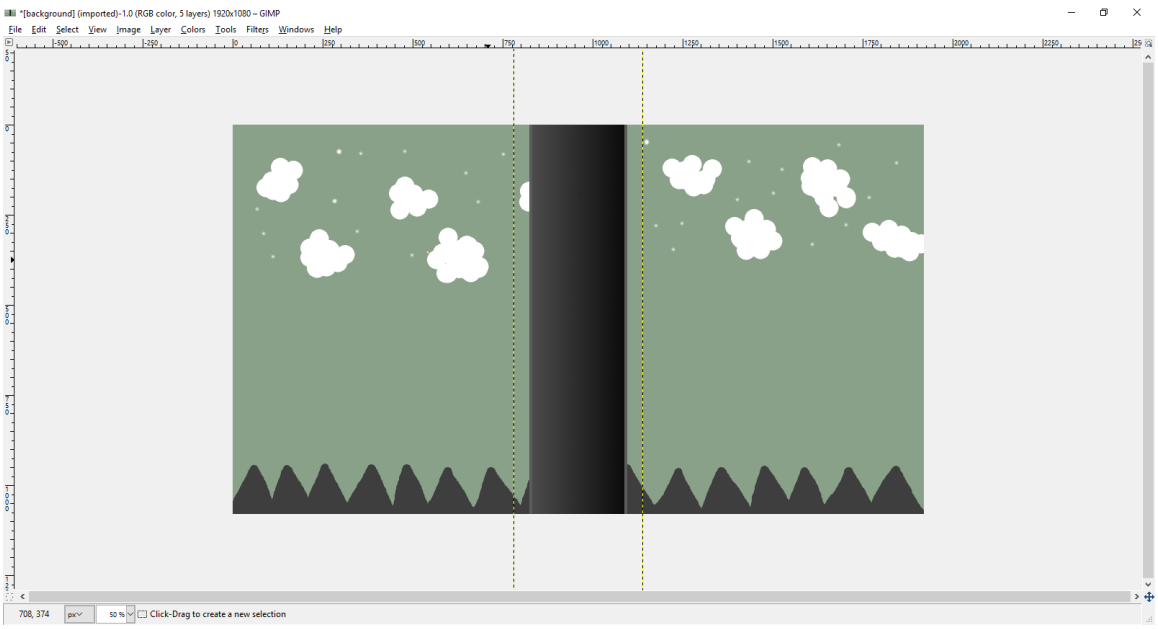


Εικόνα 8-Ρυθμίσεις έκδοσης του παιχνιδιού

2. FLAPPY MONSTER

2.1 Σχεδιασμός γραφικών

Το πρώτο στάδιο για την ανάπτυξη του παιχνιδιού, περιελάμβανε τον σχεδιασμό των γραφικών. Ο σχεδιασμός έγινε με το πρόγραμμα gimp που είναι κάτι αντίστοιχο του photoshop αλλά ευκολότερο στη χρήση. Στην παρακάτω εικόνα φαίνεται το φόντο, το έδαφος, ο αρχικός σωλήνας, τα σύννεφα και τα αστέρια (όλα σχεδιασμένα με το χέρι).



Εικόνα 9-Γραφικά του παιχνιδιού

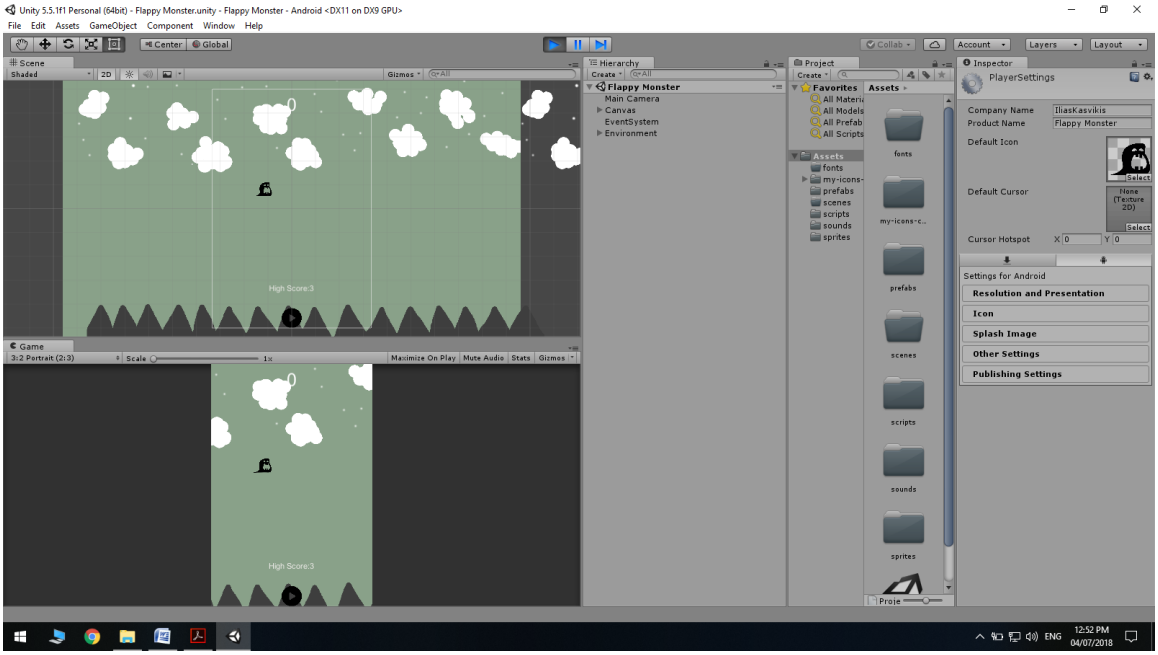
Τα υπόλοιπα γραφικά του παιχνιδιού είναι 3 εικόνες από το flaticon και αυτά είναι: το monster του παιχνιδιού, το κουμπί για το play και το κουμπί για το replay.

2.2 Ρυθμίσεις παιχνιδιού στην Unity

Για το παιχνίδι αρχικά δημιούργησα έναν Canvas από το μενού UI με screen space camera και μετά τοποθέτησα την main camera στην επιλογή render camera. Έπειτα ρύθμισα το UI scale mode σε scale with screen size και το screen match mode σε expand. Μετά δημιούργησα τις οθόνες του παιχνιδιού δηλαδή την StartPage την GameoverPage και την Countdown Page καθώς και το ScoreText του παιχνιδιού το οποίο το έβαλα στην κορυφή με λευκό χρώμα και 100 height και 100 width. Το αρχικό νούμερό του φυσικά το όρισα σε 0. Στη συνέχεια δημιούργησα μια κατηγορία την οποία ονόμασα environment και έβαλα μέσα όλα τα γραφικά του παιχνιδιού τα οποία και πέρασα στην οθόνη μου. Για κάθε μία απο τις οθόνες δημιούργησα ένα text το οποίο βγάζει ένα μήνυμα ανάλογα με την οθόνη. Στην StartPage ενσωμάτωσα το PlayButton και στην GameoverPage το ReplayButton το οποία είναι έτοιμες εικόνες όπως έχει ήδη αναφερθεί στο προηγούμενο κεφάλαιο. Σε αυτές τις δύο εικόνες είναι πολύ σημαντικό να επιλέξουμε το preserve aspect για να μην αλλάζει το μέγεθός τους σε περίπτωση που αλλάξουμε την εμφάνιση της οθόνης. Για τους σωλήνες τοποθέτησα την αρχική που ήταν η bottom pipe και έπειτα δημιούργησα ένα αντίγραφο της με αντίθετες διαστάσεις η οποία ήταν η top pipe. Τέλος τα layers, τα οποία καθορίζουν ποιο αντικείμενο θα φαίνεται πάνω από κάποιο άλλο. Έτσι λοιπόν δημιούργησα ένα layer το οποίο το ονόμασα UI και το όρισα στον canvas και ένα layer το οποίο ονόμασα Game και το όρισα στο monster, background, stars, clouds και pipes τα οποία μεταξύ τους έχουν διαφορετικό αριθμό layer για να φαίνεται πάντα κάποιο πάνω από κάποιο άλλο.

2.3 Το παιχνίδι

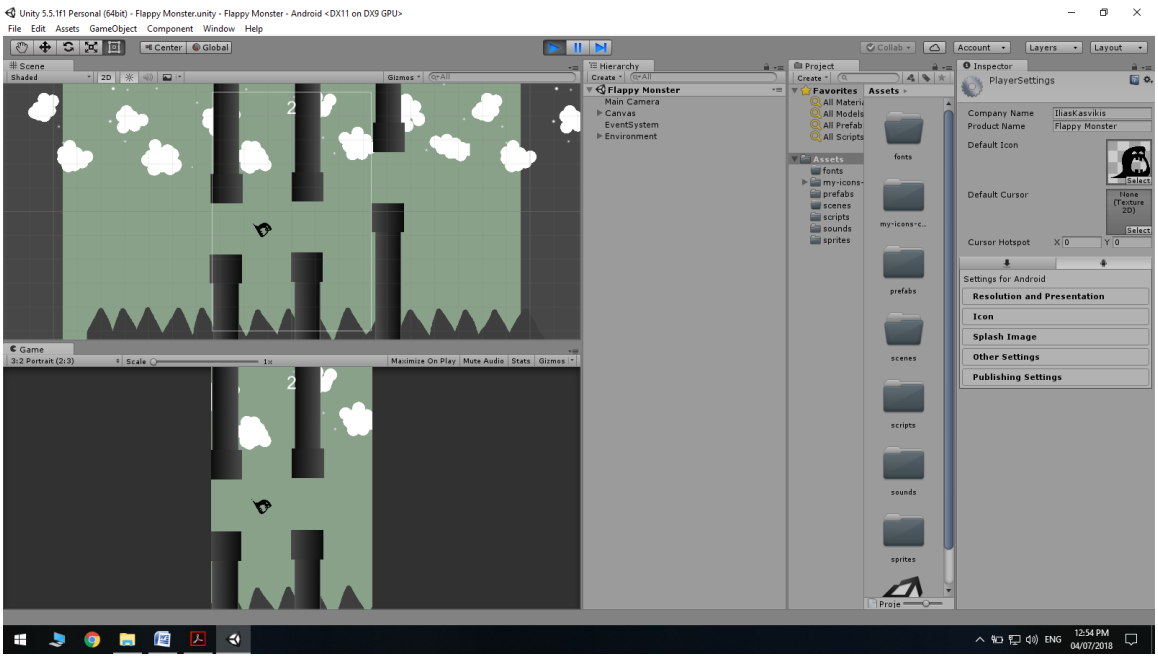
Στην παρακάτω εικόνα φαίνεται η οθόνη του παιχνιδιού πριν πατήσουμε το κουμπί play. Στην πάνω οθόνη φαίνεται όλη η εικόνα του παιχνιδιού και στην κάτω οθόνη βλέπουμε πως θα φαίνεται η εφαρμογή όταν θα παίζουμε το παιχνίδι από ένα κινητό ή ένα tablet .



Εικόνα 10-Το παιχνίδι σε οθόνη κινητού

2.4 Σκοπός του παιχνιδιού

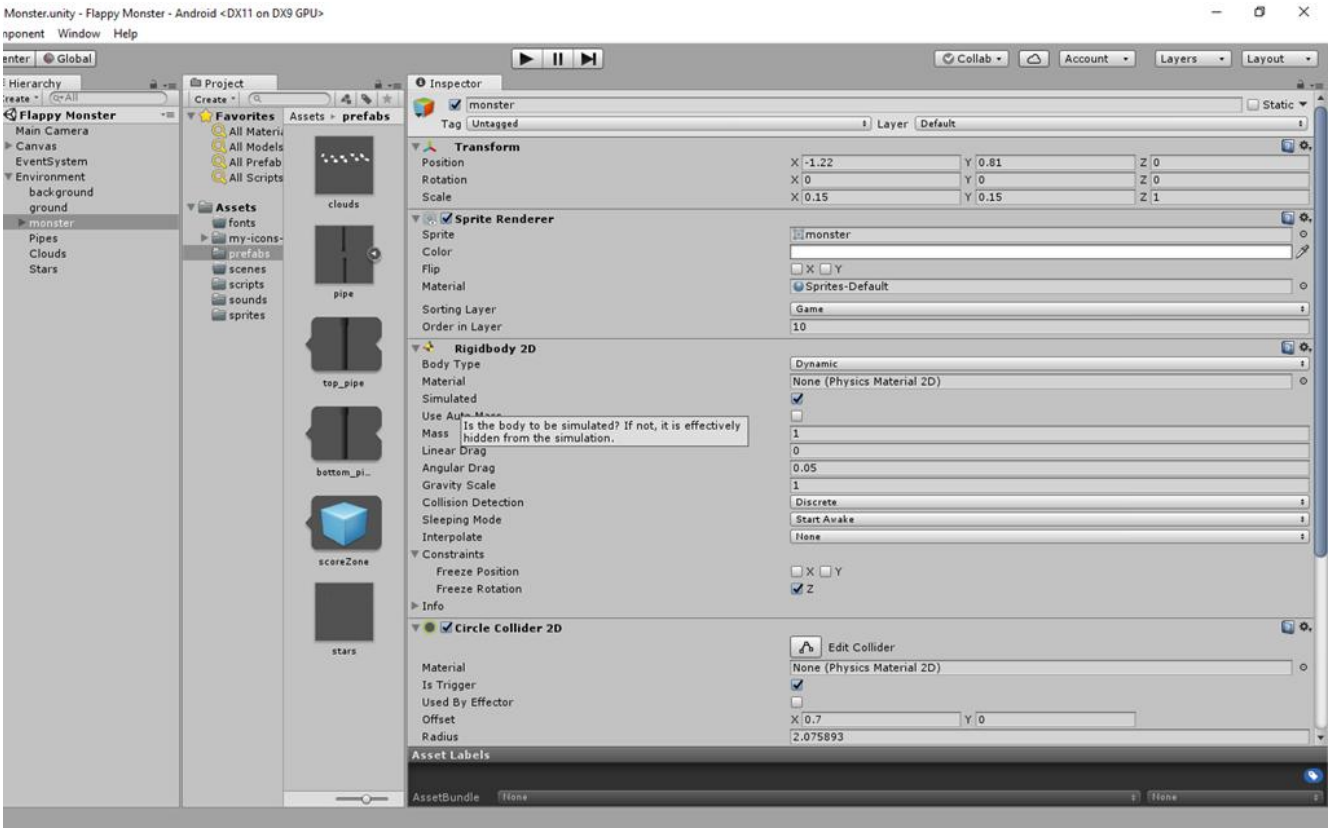
Όταν ξεκινήσει το παιχνίδι οι σωλήνες αρχίζουν να κινούνται προς το monster έχοντας ένα κενό ανάμεσά τους. Ο σκοπός του είναι να περνάει το monster ανάμεσα από τους σωλήνες χωρίς να τους ακουμπήσει. Κάθε φορά που περνάει ανάμεσα από δύο σωλήνες το σκορ μεγαλώνει κατά μία μονάδα. Το κενό των σωλήνων είναι σε διαφορετικό επίπεδο κάθε φορά και υπάρχει η δυνατότητα να αυξήσουμε την ταχύτητα που κινούνται για να γίνεται όλο και πιο δύσκολο όσο ο παίκτης κάνει μεγαλύτερο σκορ.



Εικόνα 11-Προσομοίωση του παιχνιδιού

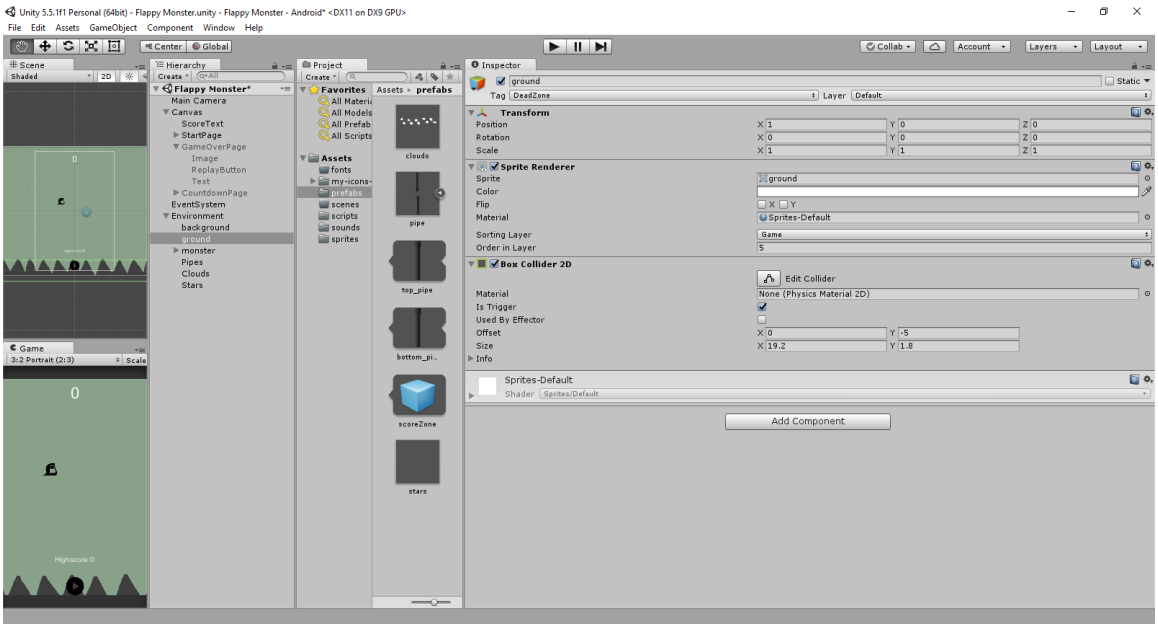
2.5 Ιδιότητες και οθόνες

Σε αυτό το κεφάλαιο εξηγούνται ιδιότητες κάποιων αντικειμένων. Στην παρακάτω εικόνα βλέπουμε τις ιδιότητες του αντικειμένου monster. Το rigidbody 2D κάνει το monster να πέφτει όταν ξεκινάει το παιχνίδι και πρέπει να πατάμε συνεχώς το αριστερό κλικ του ποντικιού για να του δίνουμε ώθηση έτσι ώστε να μην πέσει στο έδαφος ή πάνω σε κάποιον σωλήνα. Το circle Collider 2D δημιουργεί ένα κύκλο γύρω απο το monster του οποίου τα όρια έχουν ρυθμιστεί σύμφωνα με το μεγεθός του. Αυτό σημαίνει ότι αν το monster ακουμπήσει κάποιο σωλήνα ή το έδαφος ο παίχτης χάνει και το παιχνίδι ξεκινάει από την αρχή αν πατήσουμε το κουμπί replay.

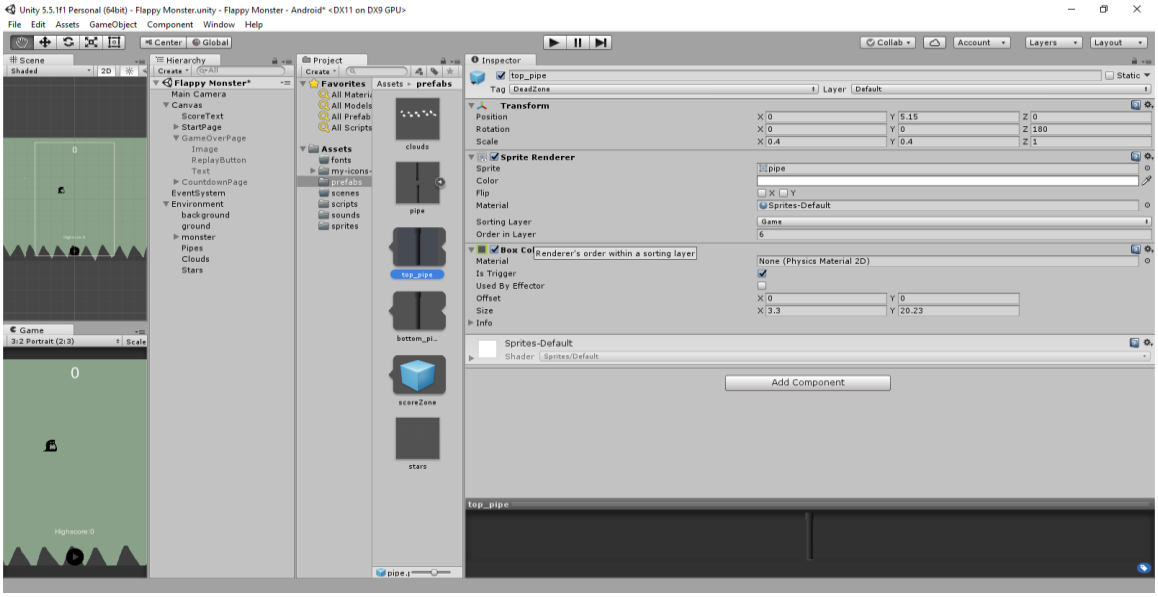


Εικόνα 12-Ιδιότητες του αντικειμένου monster

Εδώ μπορούμε να διακρίνουμε μία ιδιότητα που λέγεται Box Collider 2D και έχει χρησιμοποιηθεί για τους σωλήνες και το έδαφος έτσι ώστε να ρυθμίστούν τα όρια αυτών των αντικειμένων και να τελείωνει το παιχνίδι όταν το monster ακουμπήσει επάνω τους.

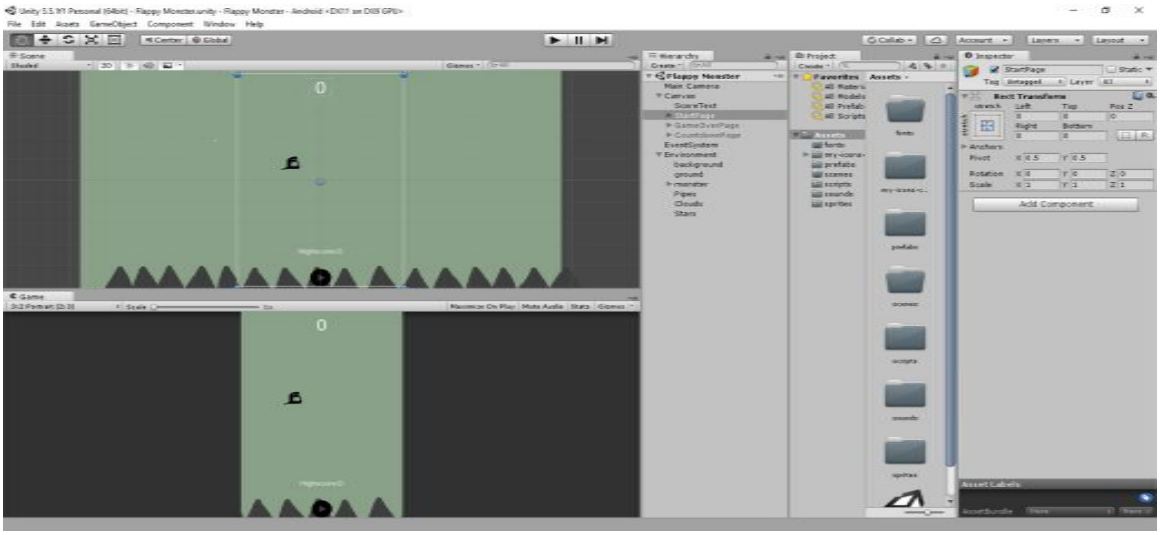


Εικόνα 13-Ιδιότητες του εδάφους του παιχνιδιού

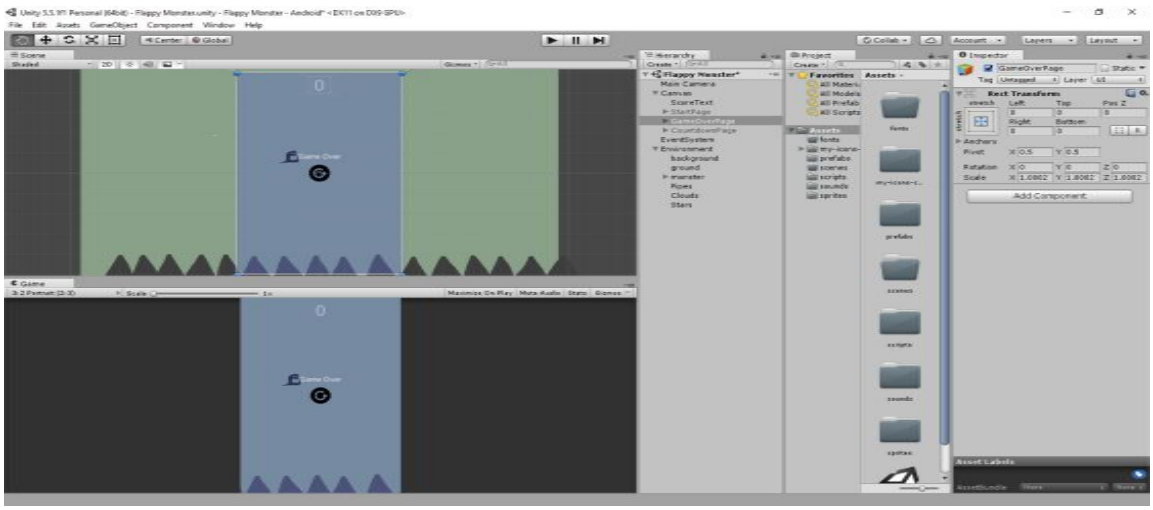


Εικόνα 14-Ιδιότητες των σωληνών του παιχνιδιού

Στις παρακάτω δύο εικόνες βλέπουμε την αρχική και την τελική οθόνη του παιχνιδιού. Στην αρχική οθόνη βλέπουμε το play button και ένα κείμενο που μας λέει ποιο είναι το highscore που έχουμε κάνει. Στην τελική οθόνη βλέπουμε μία εικόνα με μπλε φόντο που δηλώνει ότι το παιχνίδι έχει σταματήσει επειδή ο παίχτης έχασε, το replay button το οποίο αν το πατήσουμε ξεκινάμε από την αρχή και ένα κείμενο το οποίο μας λέει game over.



Εικόνα 15-Αρχική οθόνη του παιχνιδιού



Εικόνα 16-Τελική οθόνη του παιχνιδιού

2.6 Προγραμματισμός παιχνιδιού

Ο κώδικας δημιουργήθηκε σε γλώσσα C# και για το παιχνίδι χρειάστηκαν 5 scripts τα οποία είναι τα εξής:

Tap Controller script:

Ο παρακάτω κώδικας έχει να κάνει με το αντικείμενο που χειρίζεται ο παίχτης και στη συγκεκριμένη περίπτωση το monster του παιχνιδιού μας. Δίνει την αρχική του θέση, ορίζει τη συμπεριφορά του όταν πατάμε αριστερό κλικ αλλά και την κίνηση της πτώσης. Επίσης με αυτόν τον κώδικα ορίζονται οι ήχοι όταν το monster περνάει ανάμεσα από δύο σωλήνες και σκοράρει αλλά και όταν προσκρούει πάνω σε αυτούς ή στο έδαφος. Όταν γίνεται reset του παιχνιδιού το αντικείμενό μας επανέρχεται στην αρχική του θέση.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(Rigidbody2D))]//this tells unity that if
 i drag a chat app controller on my object it's going to  automat
ically crate a rigidbody 2d for me
public class tapController : MonoBehaviour {

    public delegate void PlayerDelegate();
    public static event PlayerDelegate OnPlayerDied;
    public static event PlayerDelegate OnPlayerScored;

    public float tapForce = 10;
    public float tiltSmooth = 5;
    public Vector3 startPos;//when we reset the game we're going
to set the bird back

    public AudioSource tapAudio;
    public AudioSource scoreAudio;
    public AudioSource dieAudio;

    Rigidbody2D rigidbody;
    Quaternion downRotation;//this is what we are slowly going to
 be moving towards


 Quaternion forwardRotation;//every time we tap will snap right b
ack up to that forward rotation
```

```
GameManager game;
// Called when game starts

     void Start() {
         rigidbody = GetComponent<Rigidbody2D>();
         downRotation = Quaternion.Euler (0, 0, -90);
         forwardRotation = Quaternion.Euler (0, 0, 35);
         game = GameManager.Instance;
         rigidbody.simulated = false;
     }


// Called when page state is active

     void OnEnable(){
         GameManager.OnGameStarted += OnGameStarted;
         GameManager.OnGameOverConfirmed += OnGameOverConfirmed;
     }
// Called when page state is inactive

     void OnDisable(){
         GameManager.OnGameStarted -= OnGameStarted;
         GameManager.OnGameOverConfirmed -= OnGameOverConfirmed;
     }
// Called when game has started

     void OnGameStarted (){
         rigidbody.velocity = Vector3.zero;
         rigidbody.simulated = true;
     }
// Called when replay button is pushed

    void OnGameOverConfirmed(){
         transform.localPosition = startPos;
         transform.rotation = Quaternion.identity;
     }
```

```
// Called every time the user taps during the game

    void Update() {
        if (game.GameOver) return;

        if (Input.GetMouseButtonDown(0)){
            tapAudio.Play ();
            transform.rotation = forwardRotation;
            rigidbody.velocity = Vector3.zero;
            rigidbody.AddForce (Vector2.up * tapForce, ForceMode2
D.Force);//every time we tap we want to add an upwards force on o
ur rigidbody
        }

        transform.rotation = Quaternion.Lerp (transform.rotation,
 downRotation, tiltSmooth * Time.deltaTime);
    }

   // Called when scoreZone or DeadZone touched

    void OnTriggerEnter2D(Collider2D col) {
        if (col.gameObject.tag == "ScoreZone") {
            //register a score event
            OnPlayerScored();//event sent to GameManager;
            //play a sound
            scoreAudio.Play();
        }

        if (col.gameObject.tag == "DeadZone") {
            rigidbody.simulated = false;
            //register a dead event
            OnPlayerDied();//event sent to GameManager;
            //play a sound
            dieAudio.Play();
```

Game manager script:

Αυτός ο κώδικας έχει όλες τις οθόνες του παιχνιδιού και ανάλογα με τον χειρισμό που κάνουμε εμφανίζεται η κατάλληλη οθόνη στο παιχνίδι. Όταν μπούμε μέσα στο παιχνίδι εμφανίζεται η αρχική οθόνη (Start Page) και αναγράφεται το μεγαλύτερο σκορ που έχουμε επιτύχει. Πατώντας το PlayButton περνάμε στην Countdown Page όπου αρχίζει η αντίστροφη μέτρηση για να ξεκινήσει το παιχνίδι. Κατά τη διάρκεια του παιχνιδιού όσο σκοράρουμε μεγαλώνει ο ScoreText το οποίο αναγράφεται στο πάνω μέρος της οθόνης. Τέλος όταν χάσουμε εμφανίζεται η GameOver Page το παιχνίδι τελειώνει και επίσης εμφανίζεται το μήνυμα Game Over που έχουμε ορίσει για αυτήν την οθόνη. Πατώντας το ReplayButton το παιχνίδι ξεκινάει πάλι.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour {

    public delegate void GameDelegate();
    public static event GameDelegate OnGameStarted;
    public static event GameDelegate OnGameOverConfirmed;

    public static GameManager Instance;

    public GameObject startPage;
    public GameObject gameOverPage;
    public GameObject countdownPage;
    public Text scoreText;

    enum PageState{
        None,
        Start,
        GameOver,
        Countdown
    }

    int score = 0;
    bool gameOver = true;

    public bool GameOver{ get { return gameOver; } }
```

```
  public int Score { get { return score; } }

 // Returns the current instance


   void Awake(){
      Instance = this;
  }

 // Called when page state is active

 void OnEnable(){
      CountdownText.OnCountdownFinished += OnCountdownFinished;
      tapController.OnPlayerDied += OnPlayerDied;
      tapController.OnPlayerScored += OnPlayerScored;
  }


// Called when page state is inactive

 void OnDisable(){
      CountdownText.OnCountdownFinished += OnCountdownFinished;
      tapController.OnPlayerDied -= OnPlayerDied;
      tapController.OnPlayerScored -= OnPlayerScored;
  }


// Set state when countdown is finished

 void OnCountdownFinished(){
      SetPageState (PageState.None);
      OnGameStarted ();//event sent TapController
      score = 0;
      gameOver = false;
  }

 // Set state when player dies

 void OnPlayerDied(){
      gameOver = true;
      int savedScore = PlayerPrefs.GetInt ("HighScore");
      if (score > savedScore) {
          PlayerPrefs.SetInt ("Highscore", score);
```

```
        }
        SetPageState (PageState.GameOver);
    }


// Score addition when player scores


  void OnPlayerScored(){
        score++;
        scoreText.text = score.ToString ();
    }

   // Set related state according the game status

 void SetPageState(PageState state){
        switch (state) {
            case PageState.None:
                startPage.SetActive (false);
                gameOverPage.SetActive (false);
                countdownPage.SetActive (false);
                break;
            case PageState.Start:
                startPage.SetActive (true);
                gameOverPage.SetActive (false);
                countdownPage.SetActive (false);
                break;
            case PageState.GameOver:
                startPage.SetActive (false);
                gameOverPage.SetActive (true);
                countdownPage.SetActive (false);
                break;
            case PageState.Countdown:
                startPage.SetActive (false);
                gameOverPage.SetActive (false);
                countdownPage.SetActive (true);
                break;
        }
    }
// Replay button pushed when game is over
```

```
    public void ConfirmGameOver(){

        OnGameOverConfirmed();//event sent TapController
        scoreText.text="0";
        SetPageState (PageState.Start);

 }
// Start the countdown when play button is pushed
    public void StartGame(){

        SetPageState(PageState.Countdown);
```

Parallaxer script:

Είναι ο κώδικας που ρυθμίζει τη συμπεριφορά των αντικειμένων που κινούνται μέσα στο παιχνίδι. Αυτά τα αντικείμενα είναι οι σωλήνες, τα σύννεφα και τα αστέρια. Τα δύο τελευταία έχουν μία αρχική θέση μέσα στην οθόνη πριν ξεκινήσει το παιχνίδι ενώ οι σωλήνες εμφανίζονται αφού ξεκινήσει και όλα μαζί κινούνται προς τη μεριά του παίχτη με διαφορετική ταχύτητα. Οι σωλήνες κινούνται πιο γρήγορα, μετά ακολουθούν τα σύννεφα και τέλος τα αστέρια. Όσο ανεβαίνει το σκορ όλα τα αντικείμενα κινούνται πιο γρήγορα για να ανεβαίνει το επίπεδο δυσκολίας του παιχνιδιού. Επίσης οι σωλήνες είναι ρυθμισμένοι να αλλάζουν χρώμα ανά τακτά χρονικά διαστήματα και τότε το αντικείμενο του παιχνιδιού μας μπορεί να περάσει από μέσα τους χωρίς να χάσουμε. Αυτό γίνεται για να διευκολυνθεί ο παίχτης σε κάποια σημεία που το παιχνίδι έχει γίνει πολύ δύσκολο.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Parallaxer : MonoBehaviour {

    class PoolObject{
        public Transform transform;
        public bool inUse;
        public PoolObject(Transform t) { transform = t; }
        public void Use(){inUse = true;}
        public void Dispose(){inUse = false;}
    }
```

```
[System.Serializable]
public struct YSpawnRange{
    public float min;
    public float max;
}

public GameObject Prefab;
public int poolSize;
public float shiftSpeed;
public float spawnRate;
public int overSpeed;

public YSpawnRange ySpawnRange;
public Vector3 defaultSpawnPos;
public bool spawnImmediate;//particle prewarm
public Vector3 immediateSpawnPos;
public Vector2 targetAspectRatio;

float spawnTimer;
float targetAspect;
PoolObject[] poolObjects;
GameManager game;

// Returns the current instance
void Awake(){
    Configure ();
}

// Called when game starts
void Start(){
    game = GameManager.Instance;
}

// Called when page state is active
void OnEnable(){
    GameManager.OnGameOverConfirmed += OnGameOverConfirmed;
}

// Called when page state is inactive
void OnDisable(){
    GameManager.OnGameOverConfirmed -= OnGameOverConfirmed;
}

// Called when replay button has been pushed
void OnGameOverConfirmed() {
```

```
        for (int i = 0; i < poolObjects.Length; i++) {
            poolObjects [i].Dispose ();
            poolObjects [i].transform.position = Vector3.one * 10
00;
        }
        if (spawnImmediate) {
            SpawnImmediate ();
        }
    }

    // Called every time the game updates
    void Update(){
        if (game.GameOver) {
            GameObject[] box = GameObject.FindGameObjectsWithTag
("DeadZone");
            foreach (GameObject go in box) {
                go.GetComponent<BoxCollider2D> ().enabled = true;
                go.GetComponent<SpriteRenderer> ().color = Color.
white;
            }
            overSpeed = 0;
            shiftSpeed = 1.5f;

            return;
        }

        Shift ();
        spawnTimer += Time.deltaTime;
        if (spawnTimer > spawnRate) {
            Spawn ();
            GameObject[] box = GameObject.FindGameObjectsWithTag
("DeadZone");
            spawnTimer = 0;
            overSpeed = overSpeed + 1;
            if (overSpeed == 5) {
                shiftSpeed = shiftSpeed * 1.5f;
                overSpeed = 0;
                foreach (GameObject go in box) {
            go.GetComponent<BoxCollider2D> ().enabled = false;
            go.GetComponent<SpriteRenderer> ().color = Color.red;
                }
            } else {
                foreach(GameObject go in box)
                {
          go.GetComponent<BoxCollider2D>().enabled = true;
          go.GetComponent<SpriteRenderer> ().color = Color.white;
```

```
                }
            }
        }
    }

    // Configure the position of pool objects
    void Configure() {
        targetAspect = targetAspectRatio.x / targetAspectRatio.y;
        poolObjects = new PoolObject[poolSize];
        for (int i = 0; i <poolObjects.Length; i++) {
            GameObject go = Instantiate (Prefab) as GameObject;
            Transform t = go.transform;
            t.SetParent (transform);
            t.position = Vector3.one * 1000;
            poolObjects [i] = new PoolObject (t);
        }

        if(spawnImmediate) {
            SpawnImmediate ();
        }
    }

    // Spawn the game objects
    void Spawn(){
        Transform t = GetPoolObject ();
        if (t == null)return;//if true, this indicates that poolS
ize is too small
        Vector3 pos = Vector3.zero;
        pos.x = (defaultSpawnPos.x * Camera.main.aspect) / target
Aspect;
        pos.y = Random.Range (ySpawnRange.min, ySpawnRange.max);
        t.position = pos;
    }

    // Spawns constantly the game objects
    void SpawnImmediate() {
        Transform t = GetPoolObject ();
        if (t == null)return;//if true, this indicates that poolS
ize is too small
        Vector3 pos = Vector3.zero;
        pos.x = (immediateSpawnPos.x * Camera.main.aspect) / targ
etAspect;
        pos.y = Random.Range (ySpawnRange.min, ySpawnRange.max);
        t.position = pos;
        Spawn ();
    }
```

```
    // Set the movement speed
    void Shift (){
        for(int i = 0; i <poolObjects.Length; i++) {
            poolObjects [i].transform.localPosition += -
Vector3.right * shiftSpeed * Time.deltaTime;
            CheckDisposeObject (poolObjects [i]);
        }
    }

    // Check if the object is available
    void CheckDisposeObject(PoolObject poolObject){
        if (poolObject.transform.position.x<(-
defaultSpawnPos.x * Camera.main.aspect) / targetAspect){
            poolObject.Dispose ();
            poolObject.transform.position = Vector3.one * 1000;
        }
    }

    // Get the available objects
    Transform GetPoolObject(){
        for (int i = 0; i < poolObjects.Length; i++) {
            if (!poolObjects [i].inUse) {
                poolObjects [i].Use ();
                return poolObjects[i].transform;
            }
        }
        return null;
    }
}
```

HighscoreText script:

Κώδικας που εμφανίζει το μεγαλύτερο σκορ που έχει επιτευχθεί στο παιχνίδι. Αν δεν το ξεπεράσουμε παραμένει το ίδιο. Αν το ξεπεράσουμε μας εμφανίζει το καινούριο ρεκόρ.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Text))]
```

```
public class HighscoreText : MonoBehaviour {

    Text highscore;


// Shows the highscore on starting screen

 void Start(){
        highscore = GetComponent<Text> ();
        highscore.text = "High Score:" + PlayerPrefs.GetInt ("Hig
hscore").ToString();
    }
}
```

CountdownText script:

Τέλος αυτός ο κώδικας μας δίνει τη δυνατότητα να περιμένουμε κάποια δευτερόλεπτα ενώ έχουμε πατήσει το PlayButton καθώς ξεκινά η αντίστροφη μέτρηση και μόλις τελείωσει μπορούμε να κουνήσουμε το monster του παιχνιδιού. Στη συγκεκριμένη περίπτωση έχει ρυθμιστεί στα 3 δευτερόλεπτα.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Text))]
public class CountdownText : MonoBehaviour {

    public delegate void CountdownFinished ();
    public static event CountdownFinished OnCountdownFinished;

    Text countdown;


 // Set the default value for countdown


    void OnEnable() {
        countdown = GetComponent<Text> ();
```

```
        countdown.text = "3";
        StartCoroutine ("Countdown");
    }

// Start the countdown

    IEnumerator Countdown(){
        int    count = 3;
        for (int i = 0; i < count; i++) {
            countdown.text = (count - i).ToString ();
            yield return new WaitForSeconds (1);
        }

        OnCountdownFinished ();
    }
```

2.7 Ήχοι παιχνιδιού

Για τους ήχους του παιχνιδιού χρησιμοποιήθηκε η εφαρμογή BFX με την οποία υπάρχει δυνατότητα δημιουργίας custom ήχων. Χρειάστηκαν λοιπόν 3 ήχοι για το παιχνίδι που είναι οι εξής:

Die audio - όταν το monster πεθαίνει.

Tap audio - όταν πατάμε το κλικ για να σηκωθεί το monster στον αέρα.

Score audio - όταν το monster περνάει ανάμεσα απο δύο σωλήνες και ανεβαίνει το σκορ.

3. ΣΥΜΕΡΑΣΜΑΤΑ

Τα ηλεκτρονικά παιχνίδια είναι από τους πιο αναπτυγμένους κλάδους της πληροφορικής. Σκοπό έχουν την ψυχαγωγία των χρηστών καθώς όσο προχωράει η τεχνολογία τόσο και τα παιχνίδια γίνονται πιο άρτια όσον αφορά την εμφάνιση, το θέμα και τον σχεδιασμό. Βέβαια, λόγω της συνεχούς εξέλιξης των ηλεκτρονικών παιχνιδιών, οι απαιτήσεις των χρηστών είναι συνεχώς μεγαλύτερες και αρκετοί δοκιμάζουν να φτάξουν μόνοι τους ένα παιχνίδι όπως το φαντάζονται.

Η Unity προσφέρει στον κάθε ενδιαφερόμενο την ευκαιρία να μπει στη θέση των δημιουργών διάφορων αγαπημένων του παιχνιδιών και να δημιουργήσει ο ίδιος τον αγαπημένο του χαρακτήρα, να του δώσει ζωή και να πραγματοποιήσει το δικό του φανταστικό περιβάλλον. Παρέχει οτιδήποτε χρειάζεται για την ανάπτυξη παιχνιδιών, από οδηγούς στον ιστοχώρο της μέχρι και μεγάλη ποικιλία χαρακτηριστικών και επιλογών στο πρόγραμμά της. Τέλος, το εύρος επιλογών ως προς τις πλατφόρμες στις οποίες θα τρέξουν οι εφαρμογές των παιχνιδιών την κάνει πιο προσιτή αφού οι κονσόλες, οι υπολογιστές και γενικά μία συσκευή αναπαραγωγής παιχνιδιών δε λείπει από τις ζωές καθενός μας.

Για τη δημιουργία ενός καινούριου παιχνιδιού χρειάζεται καλή γνώση προγραμματισμού και οφείλω να αναφέρω πως αντιμετώπισα αρκετές δυσκολίες σε αυτό το κομμάτι και στο στήσιμο της πλατφόρμας. Μεγάλη βοήθεια καθιστά το διαδίκτυο όπου υπάρχει μεγάλος αριθμός οδηγιών και βίντεο. Κατά τη διάρκεια της εγγραφής της διπλωματικής μου βρήκα εντυπωσιακό το μέγεθος της κοινότητας των προγραμματιστών και την ποικιλία των forums και των απαντήσεων που αντιστοιχεί σε κάθε bug που μπορεί να προκύψει στον κώδικα.

Συμπερασματικά, παρά το γεγονός ότι η ανάπτυξη ενός παιχνιδιού δεν είναι απλή ασχολία, η κατάλληλη έρευνα, η απαιτούμενη μελέτη και κάποια εξειδικευμένα και εύχρηστα εργαλεία μπορούν να καταστήσουν τη διαδικασία ευχάριστη και να μεγιστοποιήσουν την ικανοποίηση που φέρνει το τελικό αποτέλεσμα.