ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ (Σ.Δ.Ο.)

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΜΜΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και ανάπτυξη παιχνιδιού σε λειτουργικό σύστημα Android

---

ΚΑΣΤΡΙΝΑΚΗΣ ΠΑΥΛΟΣ -

ΤΣΑΚΝΑΚΗ ΒΑΛΕΝΙΑ -

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: ΚΑΘ. ΛΑΜΠΡΟΣ ΔΡΟΣΟΣ

ΠΥΡΓΟΣ 2015

Περίληψη

Η συγκεκριμένη πτυχιακή εργασία πραγματοποιήθηκε τον Σεπτέμβριο του 2015, στο Τεχνολογικό Εκπαιδευτικό Ίδρυμα, Δυτικής Ελλάδας, στο τμήμα Πληροφορικής και Μ.Μ.Ε., παραρτήματος Πύργου Ηλείας.

Το θέμα της πτυχιακής εργασίας μας, αφορά το “Σχεδιασμό και την ανάπτυξη παιχνιδιού σε λειτουργικό σύστημα Android”. Ο λόγος επιλογής αυτού του θέματος, είναι η συνεχής ανάπτυξη της τεχνολογίας και η ανταπόκριση που έχουν οι χρήστες απέναντι στις εφαρμογές. Μέσα από την εργασία μας, θα αναλύσουμε την ιστορική αναδρομή, τόσο του Android και του Unity, όσο και της C#, που είναι η βασική γλώσσα προγραμματισμού ανάπτυξης εφαρμογών στο περιβάλλον του Unity.

Στόχος μας είναι η εις βάθος εκμάθηση όλων όσων δομούν σχεδιαστικά και λειτουργικά τις εφαρμογές και το τι είναι αυτό που τις κάνει περιζήτητες σε ένα τεράστιο ποσοστό των χρηστών του διαδικτύου παγκοσμίως.

Πίνακας Περιεχομένων

Εισαγωγή

Η παρούσα πτυχιακή εργασία έχει ως στόχο την ανάλυση της συνεχούς αναπτυξιακής πορείας που λαμβάνει η τεχνολογία και πιο συγκεκριμένα, από την περίοδο δημιουργίας του Unity (2005) και της συμβατότητας του με το λογισμικό του Android (2010) για το σχεδιασμό και την ανάπτυξη εφαρμογών για κινητές συσκευές που τρέχουν το εν λόγο λειτουργικό σύστημα.

Στο πρώτο κεφάλαιο, κάνουμε μία ιστορική αναδρομή στη γέννηση του Android, τις δοκιμαστικές και επίσημες εκδόσεις που κυκλοφόρησαν καθώς και σε εργαλεία που αφορούν τους προγραμματιστές εφαρμογών Android OS.

Στο δεύτερο κεφάλαιο θα αναλύσουμε το Unity3d. Θα αναφερθούμε σε ιστορικά στοιχεία, τις επίσημες εκδόσεις του που έχουν κυκλοφορήσει μέχρι σήμερα, τη συμβατότητα του με πλατφόρμες άλλων εταιρειών, το format αρχείων που υποστηρίζει, καθώς και την προσφορά του στη βιομηχανία. Επίσης, θα μιλήσουμε για τους τρόπους εκμάθησης του προγράμματος, θα αναλύσουμε το ίδιο το πρόγραμμα και το περιβάλλον εργασίας του, καθώς και τη συμβατότητα του με το λογισμικό του Android.

Στο τρίτο κεφάλαιο θα κάνουμε αναφορά στην C#, η οποία είναι μία από τις γλώσσες ανάπτυξης του Unity και η γλώσσα προγραμματισμού που χρησιμοποιήσαμε στην δημιουργία των εφαρμογών μας. Πιο συγκεκριμένα θα αναφέρουμε ιστορικά στοιχεία, στοιχεία που αφορούν την εξέλιξη της καθώς και διάφορα κομμάτια κώδικα που αφορούν τη δομή της.

Στο τέταρτο κεφάλαιο, συνδέουμε το Unity με τη C#, που αποτελούν τα βασικά εργαλεία υλοποίησης της εφαρμογής μας. Εν συνεχεία θα αναλύσουμε τις πιο σημαντικές κλάσεις, συναρτήσεις και θα αναφέρουμε παραδείγματα ανάπτυξης κώδικα που αφορούν τις εφαρμογές σε Unity.

Στο πέμπτο κεφάλαιο, γίνεται αναφορά στα παιχνίδια που σχεδιάσαμε και αναπτύξαμε στα πλαίσια της πτυχιακής μας εργασίας. Θα μιλήσουμε για το περιεχόμενο του παιχνιδιού, τα γραφικά και τον κώδικα που προσδίδει λειτουργικότητα στην εφαρμογή μας. Στο τέλος παραθέτουμε και την βιβλιογραφία της εργασίας μας.

Κεφάλαιο 1 : Τι είναι το Android;

Εικόνα 1: www.androidiletisim.blogspot.gr

Το Android αποτελεί ένα ανοιχτού κώδικα λειτουργικό σύστημα (open source operating system) για κινητές συσκευές το οποίο αναπτύχθηκε από την Google.

Η Google στηρίχθηκε πάνω στο λειτουργικό σύστημα της Linux για να αναπτύξει το Android, το οποίο παρουσιάστηκε στην αγορά στις *23 Σεπτεμβρίου του 2008*.

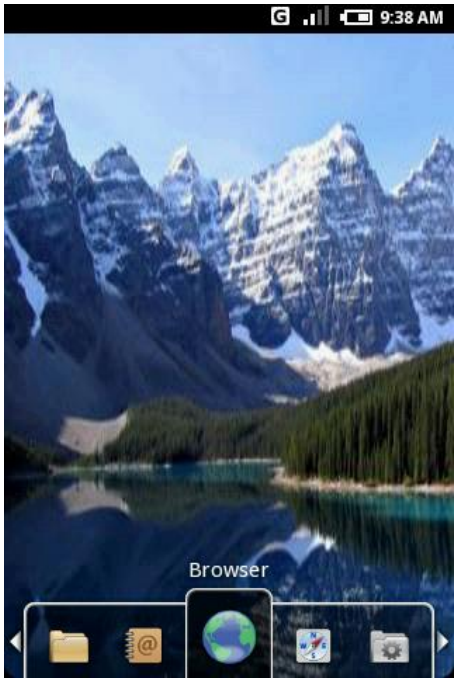
Το Android είναι γραμμένο στις γλώσσες προγραμματισμού C(core), *C++* και *JAVA*(UI), είναι διαθέσιμο σε 70 γλώσσες παγκοσμίως, ενώ μέχρι στιγμής έχουν κυκλοφορήσει 5 εκδόσεις του λογισμικού (χωρίς τις αναβαθμίσεις), με πιο πρόσφατη την *Android 5.0 Lollipop*.

1.1 Οι εκδόσεις του Android (BETA)

Πριν την επίσημη παρουσίαση του λογισμικού στην αγορά, η Google κυκλοφόρησε μία δοκιμαστική έκδοση του Android, *beta,* στις *5 Νοεμβρίου του 2007* και μία εβδομάδα αργότερα κυκλοφόρησε και το *Software Development Kit (SDK)*.

Η έκδοση του Android beta κυκλοφόρησε σε 7 διαφορετικές δοκιμαστικές version,

- m3 – rc20a (milestone 3, release code 20a)
- m3 – rc22a (milestone 3, release code 22a)
- m3 – rc37a (milestone 3, release code 37a)
- m5 – rc14 (milestone 5, release code 14)
- m5 – rc15 (milestone 5, release code 15)
- 0.9
- 1.0 – r1



Εικόνα 2: www.en.wikipedia.org

1.2 Οι εκδόσεις του Android (Επίσημες Εκδόσεις)

Η **πρώτη επίσημη έκδοση** του λογισμικού Android, η *version 1.0*, κυκλοφόρησε στην αγορά στις 23 Σεπτεμβρίου του 2008 και η πρώτη συσκευή για την οποία ήταν διαθέσιμο, ήταν η *HTC Dream*.

Στην πρώτη έκδοση του OS της Google, ήταν προεγκατεστημένες αρκετές εφαρμογές της εταιρείας, όπως το *Android Market*, το ευρέως γνωστό σήμερα ως *Play Store*, το *Gmail*, το *Google Maps*, το *YouTube* κτλ.

Οι αναβαθμισμένες εκδόσεις της πρώτης επίσημης έκδοσης του λογισμικού που κυκλοφόρησαν στην αγορά, ήταν,

- Η *version 1.1* (*09/02/2009*) η οποία ενσωμάτωνε περαιτέρω χαρακτηριστικά, ενώ διόρθωνε και τα bugs τα οποία υπήρχαν στην version 1.0,
- Η *version 1.5* (*27/04/2009*), εμπεριείχε καινούρια χαρακτηριστικά αλλά και τροποποιήσεις στο User Interface,
- Η *version* ***1.6 Donut*** (*15/09/2009*), αποτελεί την πρώτη από τις εκδόσεις του Android που θα κυκλοφορήσουν εμπεριέχοντας κάποιο γλυκό στο όνομα τους. Αυτή η έκδοση ενσωμάτωνε καινούρια χαρακτηριστικά στο λογισμικό, ενώ εμπεριείχε και αναβαθμίσεις για ορισμένες λειτουργίες του.

Η **δεύτερη επίσημη έκδοση** του λογισμικού, η *version* ***2.0 Eclair***, κυκλοφόρησε στις *26 Οκτωβρίου 2009* και εμπεριείχε πάρα πολλές αναβαθμίσεις σε όλες τις λειτουργίες του Android.

Οι αναβαθμισμένες εκδόσεις της δεύτερης επίσημης έκδοσης του λογισμικού που κυκλοφόρησαν στην αγορά ήταν,

- Η *version 2.0.1* (*03/12/2009*) η οποία εμπεριείχε μερικές μικρές αλλαγές στις εφαρμογές και διόρθωση των bugs,
- Η *version 2.1* (*12/01/2010*) η οποία ήταν παρόμοια με την ακριβώς προηγούμενη έκδοση 2.0.1,
- Η *version* ***2.2 Froyo*** (*20/05/2010*) η τρίτη σε σειρά έκδοση με την αναφορά γλυκού στο όνομα της, η οποία ανέπτυξε αρκετά το λογισμικό αλλά και τη συσκευή μέχρι εκείνο το σημείο. Η συσκευή έγινε πιο γρήγορη, προστέθηκε περισσότερη μνήμη και βελτιστοποιήθηκαν οι αποδόσεις της. Προστέθηκε Javascript Engine στον Chrome for devices που μέχρι τότε δεν υπήρχε, μεγαλύτερη ασφάλεια στο επίπεδο της διαχείρισης προσωπικών κωδικών, ενώ υπήρξε και υποστήριξη για το Adobe Flash κ.ο.κ.,
- Για την έκδοση 2.2 Froyo, κυκλοφόρησαν κάποιες αναβαθμίσεις, οι *versions 2.2.1 – 2.2.2 – 2.2.3*, οι οποίες εμπεριείχαν κάποιες βελτιώσεις, περισσότερη ασφάλεια και διόρθωση των υπάρχοντών bugs.
- Η *version* ***2.3 Gingerbread*** (*06/12/2010*) εμπεριείχε πολλές αλλαγές τόσο στην εμφάνιση, όσο και στον τρόπο λειτουργίας. Μερικές από αυτές ήταν, η αναβάθμιση του *user interface design*, *υποστήριξη σε μεγαλύτερες οθόνες και καλύτερη ανάλυση*, *δυνατότητα copy / paste,* και *γρηγορότερη και ευκολότερη συγγραφή κειμένου στο εικονικό πληκτρολόγιο*.
- Παρεμφερείς εκδόσεις της version 2.3 που κυκλοφόρησαν, ήταν οι *version 2.3.1 – 2.3.2 – 2.3.3 – 2.3.4 – 2.3.5 – 2.3.6 – 2.3.7* με τις οποίες προστέθηκαν επιπλέον χαρακτηριστικά, βελτιώθηκαν υπάρχοντα χαρακτηριστικά και μειώθηκαν τα bugs.

Η **τρίτη επίσημη έκδοση** του Android, *version **3.0 Honeycomb***, κυκλοφόρησε στις *22 Φεβρουαρίου 2011* και εμπεριείχε, *υποστήριξη για tablet*, *απλοποιημένο multitasking*, *πιο γρήγορο hardware, ικανότητα στο χρήστη για κρυπτογράφηση των δεδομένων του* κτλ.

Οι αναβαθμισμένες εκδόσεις της τρίτης επίσημης έκδοσης του λογισμικού που κυκλοφόρησαν στην αγορά ήταν,

- Η *version 3.1* (*10/05/2011*) η οποία εμπεριείχε *συνδεσιμότητα με USB εξαρτήματα, υποστήριξη εξωτερικού πληκτρολογίου – joystick – FLAC audio playback – HTTP proxy* κ.ο.κ.,
- Η *version 3.2* (*15/07/2011*) εμπεριείχε *βελτιωμένη υποστήριξη στο hardware, συμβατότητα με συσκευές οι οποίες δεν λειτουργούσαν καλά σε tablet, μεγαλύτερο έλεγχο στους προγραμματιστές διαφορετικών Android συσκευών* κτλ.
- Για την έκδοση 3.2 βγήκαν κάποιες αναβαθμίσεις και πιο συγκεκριμένα οι *versions 3.2.1 – 3.2.2 – 3.2.3 – 3.2.4 – 3.2.5 – 3.2.6,* οι οποίες εμπεριείχαν *αναβαθμίσεις* και *διορθώσεις* σε εφαρμογές του λογισμικού καθώς και διορθώσεις στα bugs που υπήρχαν.

Η **τέταρτη επίσημη έκδοση** του λειτουργικού συστήματος του Android, ήταν η *version **4.0 Ice cream***, η οποία κυκλοφόρησε στις *18 Οκτωβρίου του 2011*. Η συγκεκριμένη έκδοση, είναι η πρώτη απ' όλες μέχρι στιγμή τις εκδόσεις, η οποία "θεωρητικά" ήταν συμβατή με όλες τις συσκευές οι οποίες υποστήριζαν λογισμικό Android. Η έκδοση 4.0 είχε πάρα πολλές βελτιώσεις *στη λειτουργικότητα της*, *νέους τρόπους πρόσβασης*, όπως την απευθείας πρόσβαση σε εφαρμογές κατευθείαν από την οθόνη κλειδώματος, *βελτιώσεις στο UI του Hardware* κ.α.

Οι αναβαθμισμένες εκδόσεις της τέταρτης επίσημης έκδοσης του λογισμικού που κυκλοφόρησαν στην αγορά ήταν,

- Για την έκδοση 4.0 κυκλοφόρησαν κάποιες αναβαθμισμένες version και πιο συγκεκριμένα οι *versions 4.0.1 – 4.0.2 – 4.0.3 – 4.0.4*, οι οποίες κατά κύριο λόγο εμπεριείχαν *διορθώσεις στα bugs* και *βελτίωση των διάφορων εφαρμογών*.
- Η *version **4.1 Jelly Bean*** (*09/07/2012*) εμπεριείχε *βελτιωμένη προσβασιμότητα*, *αυτόματο re-size ή μετατόπιση των apps προκειμένου να χωρέσουν και άλλες εφαρμογές στην οθόνη, πιο απαλό User Interface* κ.τ.λ.
- Αναβαθμισμένες εκδόσεις της Jelly Bean versions που κυκλοφόρησαν, ήταν οι *4.1.1* και *4.1.2* οι οποίες εμπεριείχαν *βελτιώσεις* και *διορθώσεις των bugs*.
- Η *version 4.2* (13/11/2012) εμπεριείχε νέες λειτουργίες σε σχέση με την προηγούμενη της έκδοση, κάποιες από τις οποίες ήταν, *η βελτιστοποίηση με τη χρήση widget support του κλειδώματος οθόνης, μεγαλύτερη χωρητικότητα στις μορφές ειδοποιήσεων που στηρίζει το λογισμικό, ομαδοποίηση των μηνυμάτων* κτλ.,
- Οι διορθωτικές εκδόσεις της συγκεκριμένης version, *4.2.1* και *4.2.2,* εμπεριείχαν *βελτιώσεις* και *διορθώσεις* πάνω στην έκδοση 4.2 του Android.
- Η *έκδοση 4.3* (*24/07/2013*) κυκλοφόρησε με το σλόγκαν "Ένα ακόμα πιο γλυκό Jelly Bean" καθώς εμπεριείχε, *μεγαλύτερη εξοικονόμηση μπαταρία κατά τη χρήση του*

Bluetooth, την OpenGL ES 3.0 για πιο βελτιωμένα gaming graphics, αλλαγή γλώσσας με μία κίνηση (right-to-left) κ.α.,

- Η συγκεκριμένη έκδοση ήταν πολύ καλά αναπτυγμένη και για το λόγο αυτό κυκλοφόρησε μόνο μία version βελτιστοποίησης, η *4.3.1,* η οποία αποτέλεσε διορθωτική version για τα *bugs* στο *Nexus 7 LTE*.
- Η έκδοση ***4.4 Kit Kat*** (*31/10/2013*) αποτέλεσε μία πιο ανανεωμένη version η οποία εμπεριείχε πολλά *στοιχεία του μπλε στο user interface της, εγκαταλείφθηκε το bold που υπήρχε στο ρολόι του Android και πλέον το text έγινε thin, νέα Frameworks για το UI design, το Wi-fi και τα δεδομένα του κινητού μεταφέρθηκα στις γρήγορες επιλογές* κ.τ.λ.
- Οι βελτιωμένες εκδόσεις του Kit Kat που κυκλοφόρησαν, ήταν οι εκδόσεις *4.4.1 – 4.4.2 – 4.4.3 – 4.4.4* οι οποίες βελτίωσαν τις λειτουργίες των εφαρμογών της τρέχουσας έκδοσης του λογισμικού ή πρόσθεσαν νέες λειτουργίες και οι εκδόσεις *4.4W – 4.4W.1 – 4.4W.2* οι οποίες πρόσθεσαν ή βελτίωσαν κάποιες λειτουργίες στις εφαρμογές.

Η **πέμπτη και τελευταία επίσημη έκδοση** που έχει κυκλοφορήσει μέχρι στιγμής για το Android, είναι η *version* ***5.0 Lollipop*** (*12/11/2014*). Κάποια από τα στοιχεία τα οποία περιλαμβάνει η τελευταία και πολύ βελτιωμένη έκδοση του Android, είναι *η προσθήκη της OpenGL ES 3.1 για καλύτερα gaming graphics, μεγαλύτερη διάρκεια της μπαταρία χάρις στο Project Volta, εισαγωγή και εξαγωγή ήχου μέσω USB συσκευών, προσθήκη 15 ακόμα γλωσσών, καινούριο έξυπνο κλείδωμα* κ.ο.κ.

Οι αναβαθμισμένες εκδόσεις της πέμπτης επίσημης έκδοσης του λογισμικού που κυκλοφόρησαν στην αγορά ήταν,

- Οι εκδόσεις *5.0.1* και *5.0.2* με τις οποίες *βελτιώθηκαν κάποιες εφαρμογές* και *εξαλείφθηκαν τα υπάρχοντα bugs.*
- Η έκδοση *5.1* (*09/03/2015*) η οποία προσφέρει *μεγαλύτερη προστασία στη συσκευή, υψηλής ποιότητας κλήσεις, official support στις συσκευές με περισσότερες από μία SIM* κ.τ.λ.
- Η τελευταία έκδοση για το Android (και οι επίσημες και οι update versions) που έχει κυκλοφορήσει μέχρι στιγμής είναι η έκδοση βελτίωσης της version 5.1, η έκδοση *5.1.1,* η οποία έχει διορθώσει πάρα πολλά bugs τα οποία υπήρχαν.

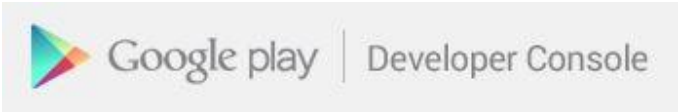
1.3 Google Play (Play Store)

Εικόνα 3: www.mobilectrl.de

Το Play Store, αποτελεί το βασικό Market που έχει δημιουργηθεί από την Google, όσον αφορά τις εφαρμογές και τα παιχνίδια τα οποία έχουν δημιουργηθεί για το λογισμικό του Android.

Το Google Play άνοιξε τις διαδικτυακές του πύλες στις *22 Οκτωβρίου του 2008* και μέχρι σήμερα, περιλαμβάνει πάνω από 1.500.000 εφαρμογές. Το επίσημο site είναι το http://play.google.com από το οποίο οι χρήστες μπορούν να κατεβάζουν εφαρμογές στο smartphone τους, ενώ οι προγραμματιστές λογισμικού Android, μπορούν να ανεβάσουν στο Play Store τις εφαρμογές που σχεδιάζουν και αναπτύσσουν.

1.4 Google Developer Console



Εικόνα 4: PrintScreen από www.developer.android.com

Το Google Developer Console αποτελεί τον επίσημο λογαριασμό των προγραμματιστών οι οποίοι θέλουν να ανεβάσουν κάποια εφαρμογή στο Play Store και να τη μοιραστούν με τους χρήστες.

Η εγγραφή γίνεται με τον λογαριασμό που έχουν δημιουργήσει ήδη στη Google, είτε για το Gmail, είτε για κάποια άλλη εφαρμογή της εταιρείας, ή σε περίπτωση που δεν υπάρχει λογαριασμός Google, ο προγραμματιστής μπορεί να εγγραφεί εκείνη τη στιγμή.

Η εγγραφή είναι πληρωτική και κοστίζει 25$ Αμερικής.

Τα βήματα εγγραφής είναι τα εξής,

Εικόνα 5: PrintScreen από www.developer.android.com

Σύνδεση με το Google Account: Αυτό το αναφέραμε στην προηγούμενη παράγραφο, κάνουμε είτε σύνδεση με τον λογαριασμό μας, ή αν δεν έχουμε δημιουργούμε εκείνη τη στιγμή έναν λογαριασμό στη Google.

Αποδοχή όρων: Υπάρχουν κάποιες προϋποθέσεις για την εγγραφή με τις οποίες οι developers πρέπει να συμφωνήσουν προκειμένου να συνεχιστεί η διαδικασία.

Πληρωμή: Οι developers πληρώνουν τα 25$ Αμερικής για να περάσουν στο επόμενο βήμα.

Ολοκλήρωση στοιχείων: Το τελευταίο βήμα περιλαμβάνει την δημιουργία του λογαριασμού μας στο Google Console και την ολοκλήρωση την διαδικασίας.

Κεφάλαιο 2 : Τι είναι το Unity3d;

Εικόνα 6: www.unity3d.com

"Το Unity3d, αποτελεί μία πολύ δυνατή και ευέλικτη πλατφόρμα, ανάπτυξης και σχεδιασμού, δημιουργικών παιχνιδιών." [1]

Το Unity, αποτελεί μία ευέλικτη και πολύ δυνατή (παιχνιδό)πλατφόρμα για τη δημιουργία *2D* και *3D* παιχνιδιών. Δημιουργήθηκε από τη *Unity Technologies*, τον *Ιούνιο του 2005* και πιο συγκεκριμένα στις 8 του μηνός. Είναι ανεπτυγμένο στις γλώσσες προγραμματισμού *C*, *C++* και *C#*, όντας, μέχρι και σήμερα, διαθέσιμο μόνο στην *αγγλική γλώσσα*. Το περιβάλλον του χωρίζεται σε δύο μέρη,

- Το *καθαρά σχεδιαστικό*, μέσα στο οποίο ο χρήστης στήνει το παιχνίδι του, κάνοντας χρήση δικών του γραφικών, αλλά και έτοιμων γραφικών, από το asset store του προγράμματος.
- Το *περιβάλλον ανάπτυξης*, μέσω του οποίου δίνει ζωντάνια και διαδραστικότητα στα γραφικά του με τη συγγραφή κώδικα C# ή Javascript.

2.1 Οι εκδόσεις του Unity3d

Η πρώτη έκδοση του Unity, η *version 1.0,* παρουσιάστηκε το 2005, στο ετήσιο (παγκόσμιο) συνέδριο της Apple και ήταν διαθέσιμο μόνο για λογισμικά OS X.

Δύο χρόνια αργότερα, και πιο συγκεκριμένα στις *11 Οκτωβρίου 2007*, στο πρώτο συνέδριο του Unity, έγινε και η παρουσίαση της δεύτερης έκδοσης του προγράμματος, της *version 2.0*. Στο συνέδριο παρουσιάστηκαν οι νέες δυνατότητες οι οποίες προστέθηκαν στο πρόγραμμα, καθώς και ο Unity Asset Server, από τον οποίο οι χρήστες μπορούν να κατεβάζουν υλικό, το οποίο είναι συμβατό με το Unity. Το 2009, το πρόγραμμα πλέον υποστηρίζει και παιχνίδια, τα οποία είναι συμβατά με τα Windows της Microsoft.

[1] Unity Official Website, 2005. Unity, από http://unity3d.com

Στις *4 Οκτωβρίου του 2010*, έγινε η επίσημη παρουσίαση της *version 3.0* του Unity. Στην νέα έκδοση του προγράμματος, ενσωματώθηκε και η δυνατότητα *preview,* καθώς και η συμβατότητα με συσκευές Android, η οποία ανακοινώθηκε και επίσημα τον *Μάρτιο του 2011*. Μέχρι τον Φλεβάρη του επόμενου χρόνου, είχαν βγει στην αγορά και οι εκδόσεις *3.4* και *3.5*, οι οποίες άρχισαν να υποστηρίξουν 3D γραφικά, καθώς και τη δυνατότητα του rendering.

Η *version 4.0* του Unity, παρουσιάστηκε επίσημα για πρώτη φορά στις *13 Νοεμβρίου του 2012,* για να ακολουθήσουν οι επόμενες εκδόσεις

- *Version 4.1* (*13.03.2013*): Υποστήριξη του AirPlay στο Operating System (OS) του iOS.
- *Version 4.2* (*22.07.2013*): Υποστήριξη Windows Phone 8, Windows Store, BlackBerry.
- *Version 4.3* (*12.11.2013*): Υποστήριξη 2D Frameworks, 2D rendering, 2D physics engine.
- *Version 4.5* (*27.05.2014*): Επιδιόρθωση περισσότερων από 450 bugs.
- *Version 4.6* (*26.11.2014*): Εισαγωγή νέου UI Framework.
- *Version 4.6.2* (*29.01.2015*): Υποστήριξη 64bit applications στο iOS.

Η τελευταία έκδοση που κυκλοφόρησε και είναι διαθέσιμη αυτή την περίοδο στην αγορά, είναι η *Version 5.0*, η οποία ανακοινώθηκε στις *3 Μαρτίου του 2015* και είναι διαθέσιμη για 22 πλατφόρμες.

2.2 Σε ποιες πλατφόρμες υποστηρίζεται σήμερα το Unity3d;

Η τελευταία έκδοση του Unity3d είναι συμβατή με πάνω από 20 διαφορετικού είδους πλατφόρμες, οι οποίες έχουν να κάνουν τόσο με κινητές συσκευές και υπολογιστές, όσο και με κονσόλες βιντεοπαιχνιδιών.

Όσον αφορά τις *κινητές συσκευές*, η πλατφόρμα του Unity είναι συμβατή με το λειτουργικό σύστημα της Apple, *iOS*, με το λειτουργικό σύστημα της Google, *Android,* με το λειτουργικό σύστημα της Microsoft, *Windows Phone*, και με τις κινητές συσκευές, *BlackBerry 10* και *Tizen*.

Στα *desktop* των Ηλεκτρονικών Υπολογιστών (Η/Υ), μπορούμε να βρούμε συμβατότητα του Unity με τα λειτουργικά συστήματα Windows και τους Η/Υ Mac, αλλά και με το *Store Apps* των Windows και το *OS* της Linux.

Οι *παιχνιδομηχανές* στις οποίες υποστηρίζεται η πλατφόρμα του Unity, είναι οι κονσόλες της Sony, *PlayStation 3, PlayStation 4 & Morpheus, PlayStation Vita*, οι κονσόλες της Microsoft, *Xbox One* και *Xbox 360* καθώς και η παιχνιδομηχανή *Wii,* της Nintendo.

Οι *Web Player* που βρίσκουμε στο διαδίκτυο, είναι συμβατοί με την τελευταία έκδοση της πλατφόρμας του Unity, όπως και η *WebGL* της Mozilla Foundation.

Η Version 5.0 του Unity, είναι συμβατή με συσκευές Augmented Reality και Virtual Reality, όπως είναι η *Oculus Rift,* η *Gear VR* και η Microsoft Hololens.

Αν σε όλες τις παραπάνω συσκευές και τα λειτουργικά συστήματα, προσθέσουμε και τον κόσμο της τηλεόραση με τις νέες πρωτοποριακές συσκευές της Google και της Samsung με την *Android TV* η πρώτη και με την *Samsung SMART TV* η δεύτερη, καταλήγουμε ότι η νέα έκδοση της πλατφόρμας του Unity, είναι συμβατή με 22 διαφορετικές συσκευές και λειτουργικά συστήματα, συνολικά.

2.3 Τι format αρχείων υποστηρίζει ο editor του Unity3d;

FORMAT ΑΡΧΕΙΩΝ ΕΙΚΟΝΑΣ				
.psd	.jpg	.png	.gif	.bmp
.tga	.tiff	.iff	.pict	.dds

FORMAT ΑΡΧΕΙΩΝ ΗΧΟΥ			
.mp3	.ogg	.aiff	.wav
.mod	.it	.sm3	

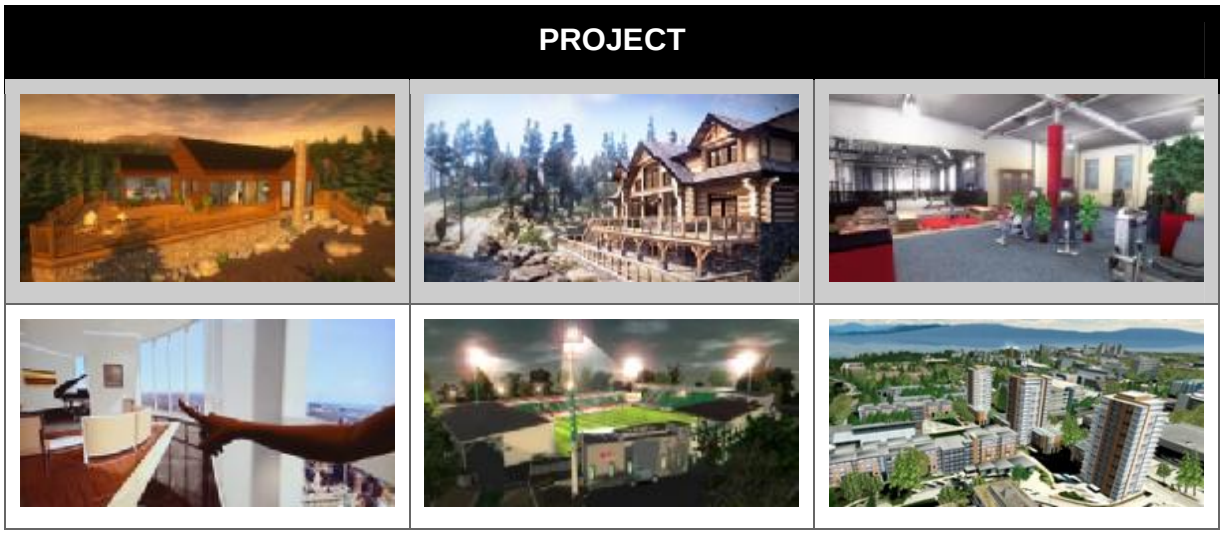
FORMAT ΑΡΧΕΙΩΝ ΒΙΝΤΕΟ		
.mov	.avi	.asf
.mpg	.mpeg	.mp4

FORMAT ΑΡΧΕΙΩΝ ΚΕΙΜΕΝΟΥ		
.txt	.htm	.html
.xml	.bytes	

2.4 Η προσφορά του Unity3d στη βιομηχανία

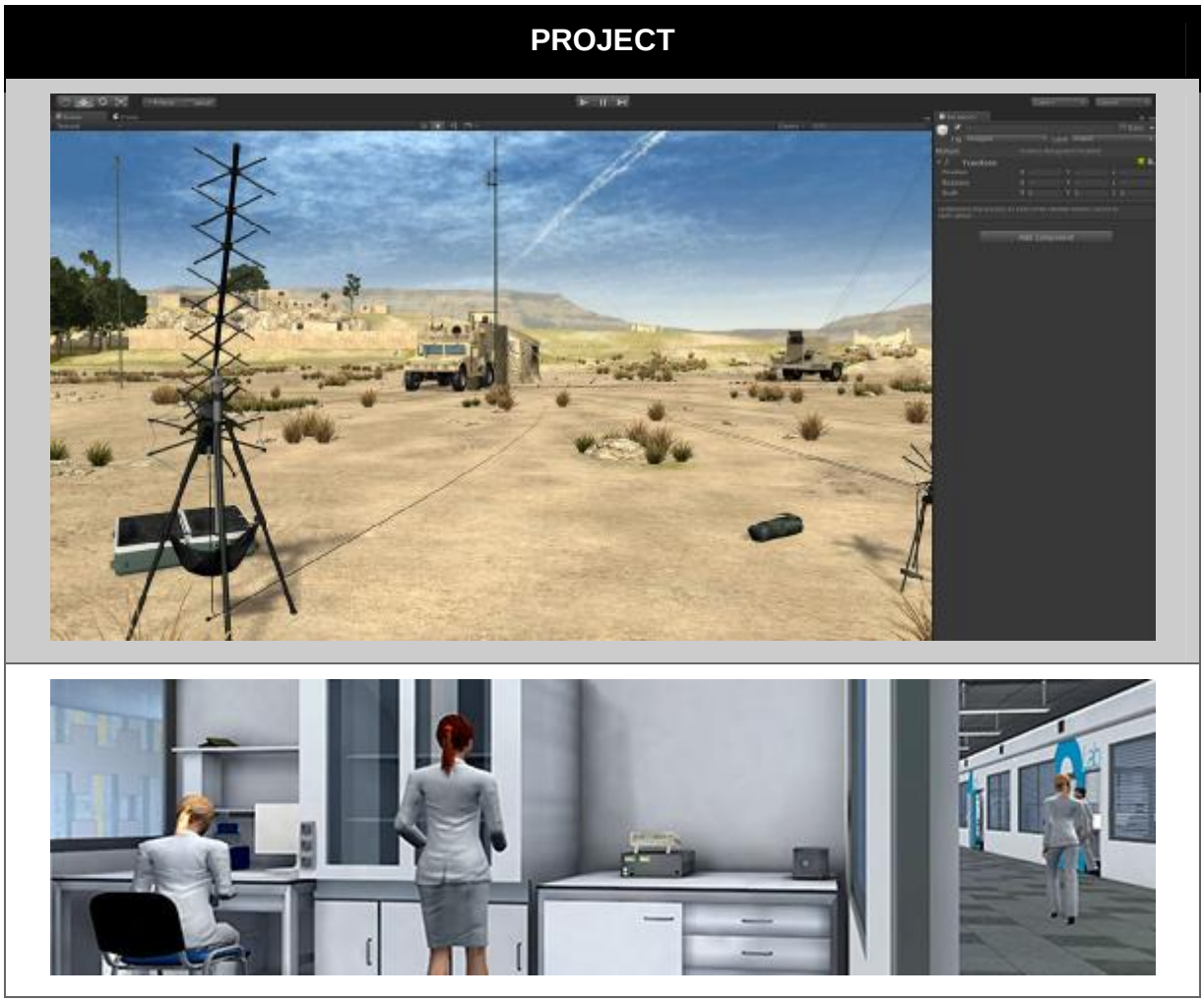
Το Unity ως πλατφόρμα, μπορεί αρχικά να δημιουργήθηκε προκειμένου να βοηθήσει τους χρήστες του στην ανάπτυξη παιχνιδιών, αλλά στην πορεία συνείσφερε και σε άλλους βιομηχανικούς τομείς.

Ένας από αυτούς τους τομείς, είναι ο τομέας της *Αρχιτεκτονικής*, της *Κατασκευής* και της *Μηχανικής*, όπου το Unity, βοήθησε στην ανάπτυξη διαδραστικών mocap, τα οποία λειτουργούσαν σε πραγματικό χρόνο, προκειμένου να παρουσιαστεί ένας χώρος, ένα σχέδιο, μία ιδέα σε virtual reality.



Εικόνα 7: www.unity3d.com

Βοήθεια προσέφερε στους τομείς της *Ιατρικής* και της *Ασφάλειας*, όπου δούλεψε σαν simulator, για την εκπαίδευση και την σωστή κατάρτιση προσωπικού κάτω υπό συγκεκριμένες συνθήκες.



Εικόνα 8: www.unity3d.com

Η πλατφόρμα του Unity, αποτελεί ένα από τα αγαπημένα προγράμματα ανάπτυξης των εταιρειών τυχερών παιχνιδιών, τόσο σε online mode, όσο και στο mobile gaming.

2.5 Μαθαίνοντας το Unity3d

Ο τρόπος εκμάθησης του περιβάλλοντος του Unity, μπορεί να γίνει με ποικίλους τρόπους, καθώς μέσα στο επίσημο site του *unity3d.com*, οι χρήστες έχουν τη δυνατότητα μέσω,

- *Video Tutorials*, τα οποία έχουν δημιουργηθεί από την ομάδα του Unity,
- *Documentation,* με τη βοήθεια του Unity Manual το οποίο προσφέρετε δωρεάν σε όλους του χρήστες,
- *Live Training,* που γίνεται σε συγκεκριμένες μέρες και ώρες και
- του *Community,* του forum δηλαδή που υπάρχει για το πρόγραμμα, στο οποίο οι χρήστες μπορούν να ρωτήσουν και να απαντήσουν σε ότι γνωρίζουν,

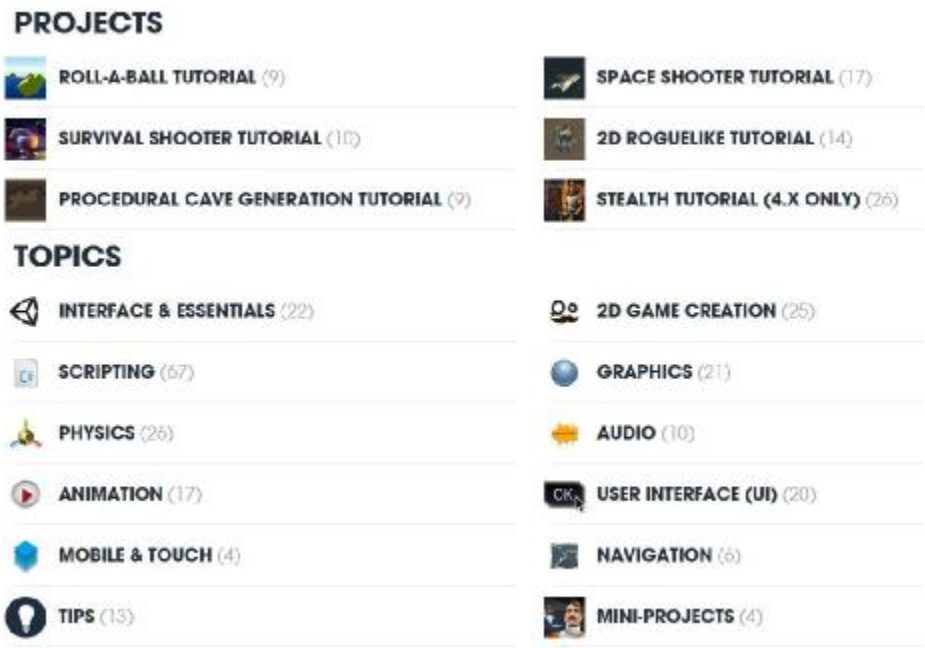
να μάθουν το πρόγραμμα, να σχεδιάσουν και να αναπτύξουν ότι επιθυμούν, έχοντας την υποστήριξη, τόσο των δημιουργών, όσο και τον μελών της κοινότητας του Unity.

2.5.1 Video Tutorials

Τα video tutorials, αποτελούν εκπαιδευτικά βίντεο τα οποία σου δείχνουν με πιο τρόπο θα χειριστείς το πρόγραμμα, θα δημιουργήσεις γραφικά, θα αναπτύξεις κώδικα και θα χρησιμοποιήσεις έτοιμα προϊόντα που σου δίνει το Unity.

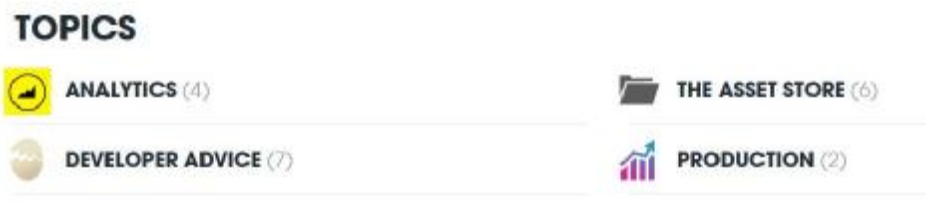
Χωρίζονται σε δύο κατηγορίες,

- Την κατηγορία *ENGINE*, τα tutorials της οποίας σε βοηθάνε να δημιουργήσεις παιχνίδια, να τα σχεδιάσεις και να αναπτύξεις κώδικα που θα τους δώσει ζωή και θα τα κάνει διαδραστικά.



Εικόνα 9: PrintScreen από www.unity3d.com

- και την κατηγορία, *SERVICES & PRODUCTION,* μέσα από την οποία λαμβάνεις συμβουλές για καλύτερη υλοποίηση παιχνιδιών και προγραμμάτων, αναλύσεις, συμβουλές για χρήση έτοιμων γραφικών και κώδικα από το Asset Store κτλ.



Εικόνα 10: PrintScreen από www.unity3d.com

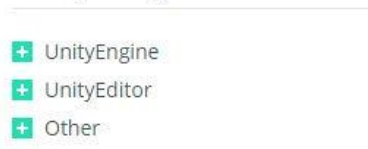
2.5.2 Documentation

Μέσα στο documentation του Unity, μπορούμε να βρούμε το Manual και το Scripting API του Unity, τα οποία παρουσιάζονται, όπως βλέπουμε στην εικόνα αριστερά μας, σαν ένα κυρίως μενού, όπου πατώντας πάνω σε κάθε επιλογή μας εμφανίζονται δευτερεύοντα και τριτεύοντα μενού, μέσα στα οποία εμπεριέχονται πληροφορίες σχετικές με τις κατηγορίες της οποίες διακρίνουμε στο κυρίως μενού των manual. Οι πληροφορίες αυτές, δίνονται με τη χρήση κειμένου, εικόνων αλλά και κομματιών κώδικα, τα οποία έχουν πάντοτε την ανάλογη επεξήγηση για τον τρόπο χρήσης τους, αλλά και για το αποτέλεσμα το οποίο παράγεται μετά τη χρήση των εκάστοτε επιλογών, ή των εκάστοτε κομματιών κώδικα, τον οποίο αναπτύσσουμε.

Εικόνα 11: PrintScreen από www.unity3d.com

Εικόνα 12: PrintScreen από www.unity3d.com

2.5.3 Live Training



Μιλώντας για live training, μιλάμε για online training courses, δηλαδή για ζωντανά μαθήματα μέσω διαδικτύου. Τα sessions τα οποία διδάσκονται είναι συγκεκριμένα, σε ορισμένες ώρες και ημερομηνίες και από συγκεκριμένους ομιλητές, ανάλογα πάντα

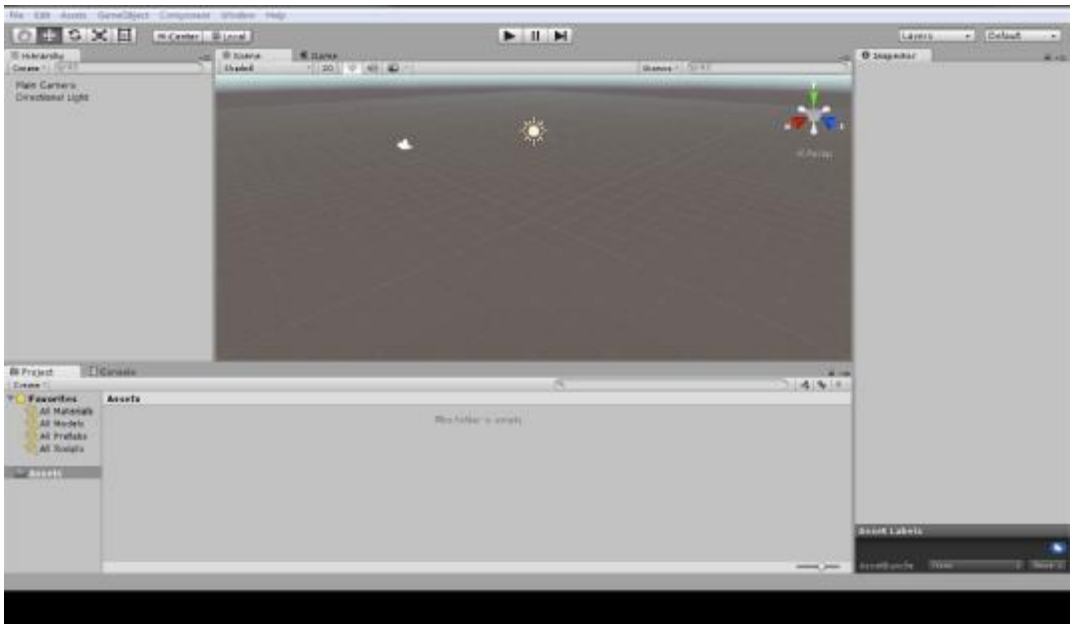
με το περιεχόμενο του εκάστοτε μαθήματος. Μέσα στον ιστότοπο του Unity, ο χρήστης μπορεί να βρει σε μορφή βίντεο και τα προηγούμενα μαθήματα και τις θεματολογίες, οι οποίες έχουν διδαχθεί.

2.5.4 Unity Resources

Το Unity3d, μέσω του Asset Store του, προσφέρει στους χρήστες του, πέρα από εργαλεία τα οποία θα τους βοηθήσουν να υλοποιήσουν τα project τους, έτοιμα projects, προκειμένου να βοηθήσει τους game developers, να βρουν λύσεις σε προβλήματα τα οποία πιθανότατα να παρουσιαστούν κατά τη διάρκεια της υλοποίησης ενός project.

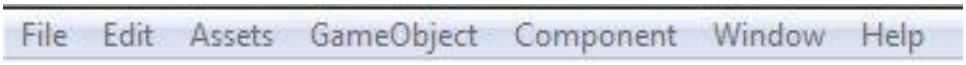
2.6 Το περιβάλλον του Unity3d

Το περιβάλλον του Unity, είναι κατά κύριο λόγο σχεδιαστικό. Δημιουργείς το περιβάλλον του παιχνιδιού σου, εισάγεις τα γραφικά σου και με τη βοήθεια του code editor του Unity και τη συγγραφή κώδικα, τα κάνεις διαδραστικά, δίνοντας τους ζωή.



Εικόνα 13: PrintScreen από το περιβάλλον του Unity

2.6.1 Το main menu του Unity



Εικόνα 14: PrintScreen από το περιβάλλον του Unity

Όπως όλα τα προγράμματα λογισμικού, έτσι και το Unity, έχει το βασικό μενού του, με το οποίο διαχειρίζεσαι τα projects, αλλά και άλλες λειτουργίες του προγράμματος.

File: Μέσω της επιλογή *file*, ο χρήστης έχει τη δυνατότητα να ανοίξει κάποιο project, να σώσει κάποιο project, να εξάγει κάποιο project, να σώσει ή να ανοίξει κάποια σκηνή (scene) η οποία σχετίζεται με αυτό που δημιουργεί κτλ. Η συγκεκριμένη επιλογή, κατά κύριο λόγο έχει να κάνει με τη διαχείριση και την αποθήκευση των σκηνών που δημιουργούμε στο παιχνίδι μας, αλλά και ολόκληρων των project καθώς και με το export για συμβατά με το Unity συστήματα.

Edit: Η συγκεκριμένη επιλογή, έχει να κάνει με συνήθης λειτουργίες οι οποίες εμπεριέχονται σε όλα τα προγράμματα λογισμικού, όπως είναι οι επιλογές αναίρεσης, undo – redo, οι επιλογές αποκοπής, αντιγραφής, επικόλλησης, copy – cut – paste κ.ο.κ. Επίσης εμπεριέχει λειτουργίες οι οποίες αφορούν αποκλειστικά το Unity, όπως η επιλογής συγκεκριμένων σκηνών, η αναζήτηση μέσα στο πρόγραμμα, η έναρξη, παύση και συνέχιση του παιχνιδιού, αλλά και άλλες λειτουργίες που έχουν να κάνουν με το project, την δικτύωση και τα γραφικά.

Asset: Η επιλογή, *asset*, έχει να κάνει εξ ολοκλήρου με το asset store του unity, την εισαγωγή και την εξαγωγή γραφικών, κομματιών κώδικα, από το asset store στις σκηνές που έχουμε δημιουργήσει στο πρόγραμμα μας, αλλά και το αντίθετο.

GameObject: Μία από τις πιο σημαντικές επιλογές του μενού στο Unity, είναι η επιλογή *GameObject*. Μέσω αυτής της επιλογής, ο χρήστης εισάγει στη σκηνή του, objects, δηλαδή αντικείμενα, τα οποία μπορεί να έχουν 2D ή 3D μορφή. Ο χρήστης μπορεί επίσης να εισάγει, φωτισμό, κάμερα παρατήρησης, ήχο κτλ., τα οποία θα βοηθήσουν στη συνολική διαμόρφωση του περιβάλλοντος το οποίο θέλουμε να δημιουργήσουμε. Τα αντικείμενα μπορούν να σταθούν στο χώρο αυτόνομα, είτε να εξαρτιόνται από άλλα αντικείμενα, να είναι δηλαδή child κάποιων αντικειμένων και εκείνα τα αντικείμενα με τη σειρά τους να είναι parents εκείνων των αντικειμένων.

Component: Κάτι αντίστοιχο με την επιλογή GameObject, είναι και η επιλογή *component*. Η εισαγωγή φίλτρων, εφέ, collider αντικειμένων, rendering αντικειμένων κτλ., είναι μερικές από τις επιλογές που μπορείς να εισάγεις στο σκηνικό σου από την επιλογή component του βασικού μενού του unity.

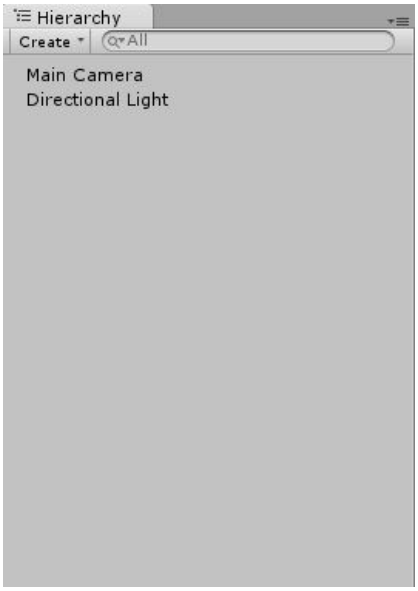
Window: Μέσω της επιλογής *window*, ο χρήστης μπορεί να κάνει εναλλαγές στις προβολές παραθύρων που είναι εμφανή στο πρόγραμμα, ή να εμφανίσει παράθυρα, τα οποία by default δεν εμφανίζονται με την έναρξη του προγράμματος, αλλά μόνο όταν ο χρήστης επιλέξει την εμφάνιση τους.

Help: Μέσω της τελευταίας επιλογής του μενού, την επιλογή *help,* οι χρήστες μπορούν να αναζητήσουν λύσεις εσωτερικά του προγράμματος, αλλά και εξωτερικά, όπως π.χ. στην κοινότητα του Unity, προκειμένου να βρουν λύσεις σε προβλήματα τα οποία έχουν παρουσιαστεί κατά την ανάπτυξη και το σχεδιασμό του project που έχουν αναλάβει.

2.6.2 Το Hierarchy window ή window προβολής των Object του Unity

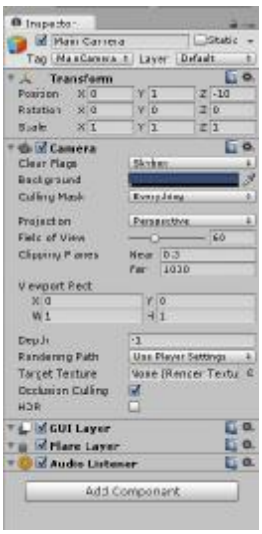
Τα αντικείμενα τα οποία υπάρχουν κατά την έναρξη του προγράμματος, αλλά και τα αντικείμενα τα οποία δημιουργούμε εμείς, εμφανίζονται στην αριστερή πλευρά του προγράμματος και το Hierarchy Window, όπως φαίνεται και από την εικόνα.

Στη συγκεκριμένη εικόνα, βλέπουμε πως στο παράθυρο μας έχουμε εισάγει μία κάμερα και ένα κατευθυνόμενο φως. Αυτά τα αντικείμενα μπορούμε να τα δούμε στη σκηνή μας και πατώντας πάνω σε κάποιο από αυτά, μπορούμε να το επεξεργαστούμε μέσω του inspector window (όπως θα δούμε παρακάτω) ή να τα ορίσουμε σαν parent ή child object. Από την επιλογή create την οποία διακρίνουμε πάνω αριστερά στην εικόνα μας, μπορούμε να εισάγουμε άμεσα αντικείμενα, χωρίς να χρησιμοποιήσουμε την επιλογή GameObject από το βασικό μενού πλοήγησης.



Εικόνα 15: PrintScreen από το περιβάλλον του Unity

2.6.3 Το Inspector window του Unity



Εικόνα 16: PrintScreen από το περιβάλλον του Unity

Όπως αναφέραμε παραπάνω, στο παράθυρο inspector του Unity, μπορούμε να κάνουμε τροποποιήσεις στις παραμέτρους των επιλεγμένων αντικειμένων στο Unity, να εισάγουμε ή να διαγράψουμε παραμέτρους κτλ., ανάλογα με τις ιδιότητες και τις λειτουργίες τις οποίες θέλουμε να έχει το αντικείμενο μας. Όταν θα επιλέξουμε ένα αντικείμενο (στη συγκεκριμένη περίπτωση έχουμε επιλέξει την Main Camera που έχουμε εισάγει στο σκηνικό μας), εμφανίζονται οι παράμετροι, πάνω στις οποίες μπορεί να δεχτεί τροποποιήσεις το εκάστοτε αντικείμενο.

Από την επιλογή Add Component που βλέπουμε στο κάτω μέρος της εικόνας μας, μπορούμε να εισάγουμε περαιτέρω παραμέτρους που θέλουμε να έχει το αντικείμενο μας και να τις τροποποιήσουμε ανάλογα με τις λειτουργίες τις οποίες θέλουμε να του θέσουμε.

2.6.4 Το Project window του Unity

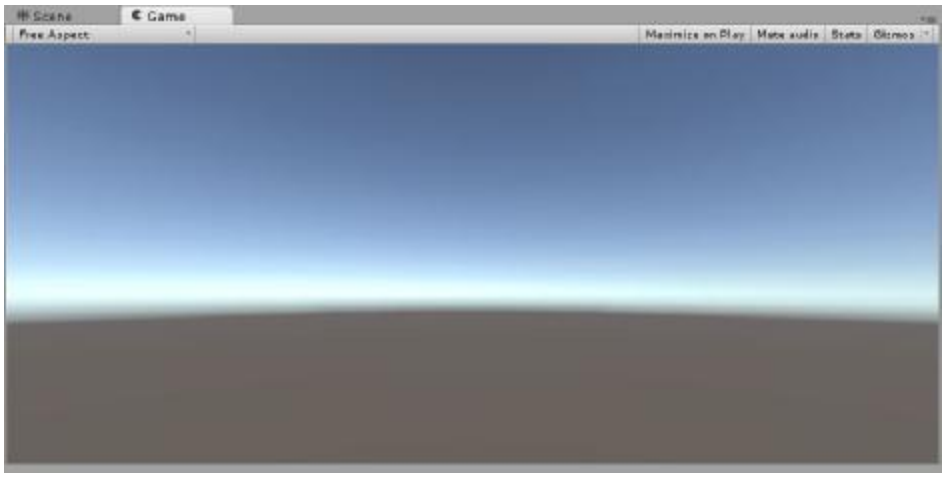
Το παράθυρο project του Unity, αποτελεί το μέρος στο οποίο εμφανίζονται τα διάφορα υλικά, μοντέλα κτλ., τα οποία έχουμε κατεβάσει από το Asset Store ή τα έχουμε εισάγει στο Unity, προκειμένου να τα ενσωματώσουμε στα αντικείμενα τα οποία έχουμε εισάγει στο σκηνικό μας. Σε αυτό το παράθυρο εμφανίζονται και τα κομμάτια κώδικα, τα οποία δίνουν τη διαδραστικότητα και ορίζουν διάφορες λειτουργίες των αντικειμένων μέσα στο σκηνικό μας. Στο παράθυρο των project μπορούμε να εισάγουμε ή να κατεβάσουμε υφές υλικών (ξύλο, μάρμαρο, πλακάκι) τα οποία θα προσθέσουμε στο αντικείμενο μας, ώστε να πάρει ανάλογη μορφή. Επίσης, μπορούμε να κατεβάσουμε έτοιμα αντικείμενα, τα οποία έχουν πάνω τους ήδη κάποια υφή και να τα εισάγουμε κατευθείαν στο σκηνικό μας.

2.6.5 Το παράθυρο του σκηνικού μας στο Unity

Εικόνα 17: PrintScreen από το περιβάλλον του Unity

Το παράθυρο του σκηνικού μας χωρίζεται σε δύο μέρη. Το πρώτο μέρος (εικόνα 17), είναι το *παράθυρο σκηνής*, όπου μπορούμε να διακρίνουμε τα αντικείμενα τα οποία έχουμε εισάγει μέσα στο σκηνικό, όπως κάμερα, φως, σχήματα κτλ.,

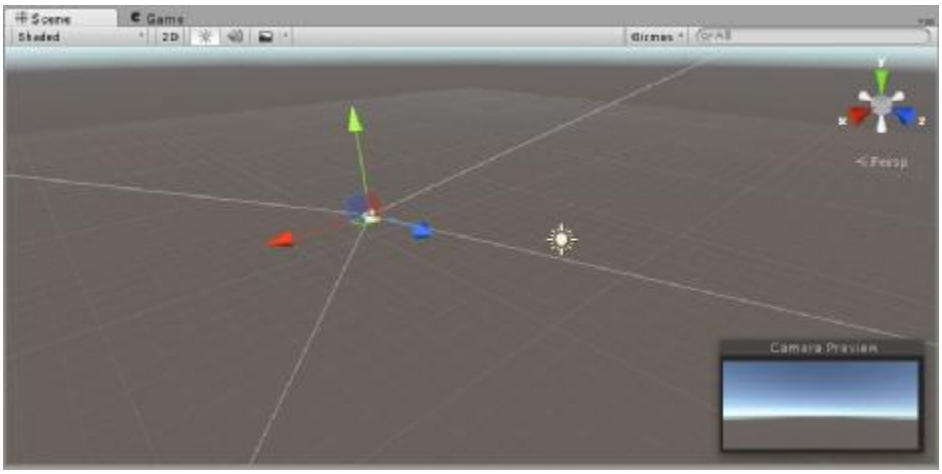
και το δεύτερο (εικόνα 18), το *παράθυρο προβολής του παιχνιδιού* μας, στο οποίο μπορούμε να διακρίνουμε το σκηνικό μας, ζωντανά πλέον, σε επίπεδο χρήσης και όχι σε επίπεδο επεξεργασίας.



Εικόνα 18: PrintScreen από το περιβάλλον του Unity

Στο παράθυρο προβολής του παιχνιδιού, η κάμερα την οποία έχουμε εισάγει στο σκηνικό μας, γίνεται τα μάτια μας και μέσα από αυτή βλέπουμε το σκηνικό ανάλογα με τις τροποποιήσεις (τρόπους προβολής ανάλογα με τους άξονες, χρήση κώδικα στην κάμερα) τις οποίες έχουμε κάνει στις παραμέτρους της.

Στην περίπτωση του παραθύρου του σκηνικού μας (εικόνα 19), αν επιλέξουμε την κάμερα μας, θα δούμε πως κάτω δεξιά, εμφανίζεται ένα preview window, κάτι το οποίο μας βοηθάει ώστε να μπορούμε να διακρίνουμε τις αλλαγές που έχουμε κάνει και στο παράθυρο σκηνικού αλλά και στο παράθυρο προβολής του παιχνιδιού.



Εικόνα 19: PrintScreen από το περιβάλλον του Unity

2.6.6 Το μενού τροποποίησης και διαχείρισης αντικειμένων στο σκηνικό

Εικόνα 20: PrintScreen από το περιβάλλον του Unity

Μέσω του μενού το οποίο βλέπουμε στην παραπάνω εικόνα, μπορούμε να τροποποιήσουμε τις θέσεις των αντικειμένων μέσα στο σκηνικό, τους τρόπους προβολής, τις θέσεις τους σε σχέση με το σκηνικό ή τις θέσεις τους σε σχέση με τα άλλα αντικείμενα του σκηνικού κ.ο.κ.

Με την επιλογή με *το χεράκι*, μετακινούμε και τοποθετούμε ολόκληρο το σκηνικό μας, προκειμένου να κάνουμε τις τροποποιήσεις που θέλουμε σε συγκεκριμένα μέρη ή αντικείμενα μέσα στο χώρο.

Η επιλογή με *το σταυρό*, μας βοηθάει να επιλέξουμε τα αντικείμενα του σκηνικού μας και να τα τοποθετήσουμε σε όποιο μέρος της σκηνής μας θέλουμε εμείς.

Αν διαλέξουμε την επιλογή με *τα δύο βελάκια*, μπορούμε να αλλάξουμε τη θέση ή την προβολή των αντικειμένων ως προς του άξονες x και y, αν μιλάμε για 2D σκηνικό ή προς τους άξονες x, y και z, εφόσον μιλάμε για σκηνικό σε 3D μορφή.

Με την τέταρτη επιλογή η οποία μοιάζει με ένα μεγάλο X το οποίο έχει ένα τετράγωνο στη μέση, μπορούμε να αλλάξουμε το scale των αντικειμένων μέσα στο σκηνικό μας, χωρίς να παρεμβούμε στις παραμέτρους τους από το Inspector window.

Η τελευταία επιλογή, *το μεγάλο τετράγωνο με τον κύκλο στη μέση*, μας βοηθάει στο να οριοθετήσουμε και να τοποθετήσουμε ένα αντικείμενο μέσα στο χώρο, με βάση τη θέση ενός άλλου αντικειμένου στο σκηνικό μας. Με λίγα λόγια, με το συγκεκριμένο εργαλείο, ομαδοποιούμε δύο ή περισσότερα αντικείμενα με στη σκηνή μας και ορίζουμε τις θέσεις τους με βάσει τα υπόλοιπα αντικείμενα τα οποία έχουμε επιλέξει.

2.6.7 Το μενού έναρξης – παύσης – διακοπής του παιχνιδιού μας

Εικόνα 21: PrintScreen από το περιβάλλον του Unity

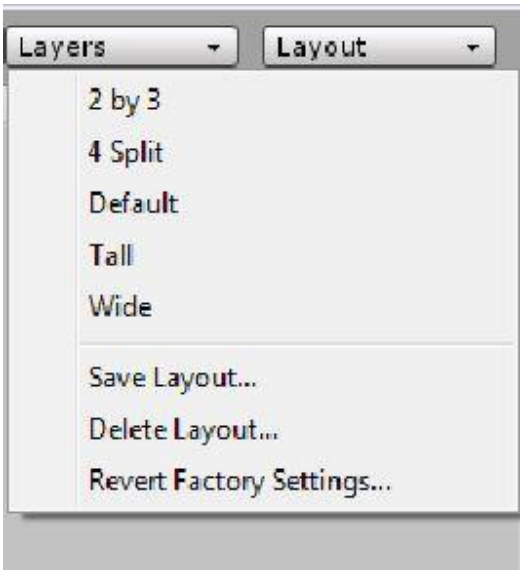
Ένα τρίτο και τελευταίο μενού επιλογών το οποίο υπάρχει μέσα στο Unity, είναι το μενού με το οποίο μπορούμε να ξεκινήσουμε το παιχνίδι μας, να το σταματήσουμε ή και να το διακόψουμε τελείως.

Πατώντας την πρώτη επιλογή με *το βελάκι*, το παιχνίδι μας ξεκινάει να τρέχει εσωτερικά του Unity, δίνοντας μας τη δυνατότητα να παίξουμε και να εξετάσουμε τις ιδιότητες και της

παραμέτρους τις οποίες έχουμε θέσει. Πατώντας το δεύτερη φορά, το παιχνίδι σταματάει τελείως.

Η δεύτερη επιλογή του *pause*, παγώνει το παιχνίδι στο σημείο το οποίο εμείς το πατάμε, ενώ κάνοντας ένα δεύτερο κλικ πάνω στο ίδιο κουμπί, το παιχνίδι μας συνεχίζει από το σημείο το οποίο αφήσαμε.

Το τρίτο κουμπί, σταματάει το παιχνίδι στο σημείο το οποίο βρισκόμαστε και μετά με μία από τις δύο προηγούμενες επιλογές, αποφασίζουμε αν θα το τερματίσουμε σε εκείνο το σημείο ή θα συνεχίσουμε παρακάτω.



Εικόνα 22: PrintScreen από το περιβάλλον του Unity

2.6.8 Τα layout του Unity3d

Ανάλογα με το project πάνω στο οποίο εργάζεται κάποιος προγραμματιστής του Unity3d, πιθανότατα να χρειαστεί διαφορετικός τρόπος τοποθέτησης των διάφορων παραθύρων του προγράμματος, προκειμένου να διευκολυνθεί η εργασία του. Με απλά λόγια, ίσως χρειαστεί διαφορετικό *layout*. Ως layout ορίζεται ο τρόπος με τον οποίο πέφτει η πληροφορία, είτε σε κάποιο χαρτί, είτε στην οθόνη του υπολογιστή μας ή της κινητής μας συσκευής κ.ο.κ. Στην συγκεκριμένη περίπτωση μιλάμε για τον τρόπο προβολής των παραθύρων του προγράμματος.

Την επιλογή layout, μπορούμε να τη βρούμε πάνω από το inspector window και να επιλέξουμε την προβολή των παραθύρων, ανάλογα με τις επιλογές τις οποίες μας δίνονται.

Πέρα από την τοποθέτηση των παραθύρων που κάνει αυτόματα το πρόγραμμα ανάλογα με την επιλογή που θα επιλέξουμε, ο χρήστης έχει τη δυνατότητα να σύρει τα παράθυρα και να τα τοποθετήσει όπου θέλει, διαμορφώνοντας έτσι το layout των παραθύρων του, όπως εκείνος επιθυμεί.

2.7 Ο code editor του Unity3d

Εικόνα 23: PrintScreen από το περιβάλλον του Unity

Το Unity έχει ενσωματωμένο code editor, για την ανάπτυξη κώδικα και τη δημιουργία της διαδραστικότητας του παιχνιδιού. Για να δημιουργήσουμε ένα script κώδικα, σε ένα από τα Object τα οποία έχουμε εισάγει στο σκηνικό μας, μέσω της επιλογής, Add Component, εισάγουμε ένα script και μέσα σε αυτό αναπτύσσουμε κώδικα για να δημιουργήσουμε τη διαδραστικότητα στο συγκεκριμένο αντικείμενο. Όπως προαναφέραμε, ο κώδικας συγγραφής είναι ή C# ή Javascript.

2.8 Εξαγωγή εφαρμογής Android από το Unity

Όπως αναφέραμε και παραπάνω, το Unity είναι διαθέσιμο για πάνω από 20 πλατφόρμες, οι οποίες όμως δεν περιλαμβάνουν το ίδιο λογισμικό.

Για να εξάγουμε ένα παιχνίδι, ή μία εφαρμογή από το Unity για συσκευές οι οποίες τρέχουν το λογισμικό του Android θα χρειαστούμε, τα εξής,

1. Το Java SE Development Kit, το οποίο είναι διαθέσιμο στο επίσημο site της Oracle - http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html και από εκεί, όπως φαίνεται και στην παρακάτω εικόνα, κατεβάζουμε,

ανάλογα με το λειτουργικό του υπολογιστή μας, την ανάλογη έκδοση λογισμικού και την εγκαθιστούμε.

Java SE Development Kit 7u79

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	130.4 MB	jdk-7u79-linux-i586.rpm
Linux x86	147.6 MB	jdk-7u79-linux-i586.tar.gz
Linux x64	131.69 MB	jdk-7u79-linux-x64.rpm
Linux x64	146.4 MB	jdk-7u79-linux-x64.tar.gz
Mac OS X x64	196.89 MB	jdk-7u79-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.79 MB	jdk-7u79-solaris-i586.tar.Z
Solaris x86	96.66 MB	jdk-7u79-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.67 MB	jdk-7u79-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u79-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	140 MB	jdk-7u79-solaris-sparc.tar.Z
Solaris SPARC	99.4 MB	jdk-7u79-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	24 MB	jdk-7u79-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.4 MB	jdk-7u79-solaris-sparcv9.tar.gz
Windows x86	138.31 MB	jdk-7u79-windows-i586.exe
Windows x64	140.06 MB	jdk-7u79-windows-x64.exe

Εικόνα 24: PrintScreen από www.oracle.com

2. Κατεβάζουμε και εγκαθιστούμε το Android SDK από τον επίσημο ιστότοπο του Android - http://developer.android.com/sdk/installing/index.html?pkg=studio , και ακολουθούμε τα παρακάτω βήματα, όπως φαίνονται στην εικόνα.

To set up Android Studio on Windows: [Show instructions for all platforms]

1. Launch the `.exe` file you just downloaded.
2. Follow the setup wizard to install Android Studio and any necessary SDK tools.

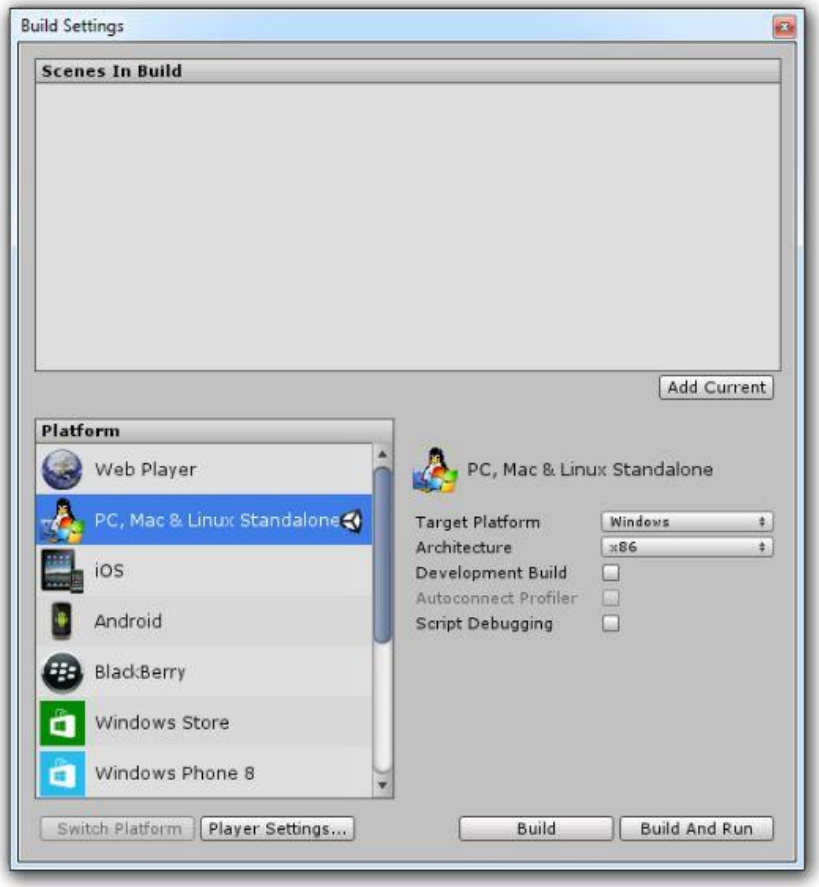
   On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

   Select **Start menu > Computer > System Properties > Advanced System Properties**. Then open **Advanced tab > Environment Variables** and add a new system variable `JAVA_HOME` that points to your JDK folder, for example `C:\Program Files\Java\jdk1.7.0_21` .

The individual tools and other SDK packages are saved outside the Android Studio application directory. If you need to access the tools directly, use a terminal to navigate to the location where they are installed. For example:

```
\Users\<user>\sdk\
```

Εικόνα 25: PrintScreen από developer.android.com



Εικόνα 26: PrintScreen από το περιβάλλον του Unity

Αφού ολοκληρώσουμε τις εγκατάσταση των δύο παραπάνω, από το βασικό μενού του Unity και την επιλογή *File*, επιλέγου το *Built Setting* ή χρησιμοποιούμε την συντόμευση Ctrl+Shift+B και μας ανοίγει το παρακάτω παράθυρο.

Από το παράθυρο *Platform* στα αριστερά, επιλέγουμε το Android. ΄Υστερα πατάμε το κουμπί *Player Settings…* με το οποίο ανοίγουμε ένα παράθυρο επιλογών στο Inspector Window του Unity. Από εκείνο το παράθυρο μπορούμε να διαμορφώσουμε το export της εφαρμογής μας, ως προς *το εικονίδιο* που θα έχει, *την ποιότητα των γραφικών* και άλλου τέτοιου είδους ρυθμίσεις.

Αφού ολοκληρώσουμε και την παραπάνω ενέργεια, πατάμε το κουμπί *Build* και το Unity, “χτίζει” την εφαρμογή μας.

Κατά τη διάρκεια του Build, το Unity θα μας ζητήσει το *sdk* του Android, το οποίο κάναμε install προηγουμένως. Βρίσκουμε το path και το τοποθετούμε στο Unity προκειμένου να συνεχίσει κανονικά το “χτίσιμο” της εφαρμογής μας. Το path συνήθως είναι της παρακάτω μορφής:

C:\Users\User\AppData\Local\Android\sdk

Το format του αρχείου που έχουμε στον υπολογιστή μας, είναι της μορφής .apk την οποία όμως δε μπορούμε να την τρέξουμε μέσω του υπολογιστή μας, αλλά μέσω κάποια συσκευής smartphone με λογισμικό Android.

Αν περάσουμε το αρχείο μέσα σε μία κινητή συσκευή (είτε κινητό, είτε tablet) με Android OS, θα πρέπει να το εγκαταστήσουμε προκειμένου να τρέξει η εφαρμογή μας. By default, οι Android συσκευές, για λόγους προστασίας, δεν επιτρέπουν στους χρήστες την εγκατάσταση κάποιας εφαρμογής, αν δεν προέρχεται από το Play Store. Για να μπορέσουμε να εγκαταστήσουμε την εφαρμογή μας, από τις ρυθμίσεις της συσκευής μας, θα πρέπει να ενεργοποιήσουμε την επιλογή εγκατάστασης εφαρμογών οι οποίες δεν προέρχονται από το Google Market. Με την ολοκλήρωση και αυτής της διαδικασίας, είμαστε έτοιμοι να εγκαταστήσουμε και να τρέξουμε την εφαρμογή στην κινητή μας συσκευή.

Κεφάλαιο 3 : Η γλώσσα προγραμματισμού C# (sharp)

Εικόνα 27: www.ouedkniss.com

Η *C#* είναι μία multi-paradigm γλώσσα προγραμματισμού η οποία έχει κοινά στοιχεία με την C, καθώς προέρχεται από αυτή. Σχεδιάστηκε και λειτουργεί από την *Microsoft* στο πλαίσιο της πρωτοβουλίας της .NET, και εδώ και 15 χρόνια και συγκεκριμένα από τον Ιούλιο του 2000, εγκρίθηκε ως πρότυπη από την Ecma.

Πρόκειται για μια απλή, σύγχρονη, γενικής χρήσης αντικειμενοστραφής γλώσσα προγραμματισμού. Μερικά από τα στοιχεία τα οποία χαρακτηρίζουν την C# είναι ότι είναι *στατική, δυναμική, ισχυρή, ασφαλής και ονομαστική.* Την ομάδα του προγραμματισμού της C# ηγείται ο *Anders Hejlsberg*. Η τελευταία έκδοση που κυκλοφορεί είναι η C# 6.0, η οποία κυκλοφόρησε στις 20 Ιουλίου 2015.

Η C# είναι σχεδιασμένη για τη δημιουργία λογισμικού σε .Net Framework. Οτιδήποτε εμπεριέχεται στην C# είναι αντικείμενα. Πολύ σημαντικό είναι το γεγονός ότι παρέχει άμεση πρόσβαση σε τεράστιες βιβλιοθήκες κλάσεων του .Net Framework και επίσης υπάρχει μεγάλη ασφάλεια των τύπων που διαθέτει.

3.1 Ιστορία και εξέλιξη της C#

Τα χαρακτηριστικά που κληρονόμησε η C# από την C είναι :

- Η σύνταξη
- Οι λέξεις κλειδιά και
- Οι τελεστές

Η σχέση που έχει η Java με τη C# είναι ότι και οι δύο έχουν:

- Μεταφερτό κώδικα και
- Εκτέλεση προγραμμάτων σε ασφαλές περιβάλλον

Η σχέση της C# με το .NET Framework είναι ότι :

- Το .NET Framework είναι το περιβάλλον χρόνου εκτέλεσης της C#.
- Οι βιβλιοθήκες της γλώσσας ορίζονται από το .NET Framework.

3.1.1 Από την C στη C++

Στα τέλη της δεκαετίας το ’70 το μέγεθος των έργων που υλοποιούσαν οι προγραμματιστές με τη C έφτασε στα όρια του. Έτσι εμφανίστηκε ένας νέος τρόπος προγραμματισμού, ο αντικειμενοστραφής προγραμματισμός (Object Oriented Programming – OOP) και μια νέα γλώσσα, η αντικειμενοστραφής γλώσσα C++.

3.1.2 Από τη C++ στη Java

Λόγω της μεγάλης διάδοσης του internet οι προγραμματιστές επιθυμούσαν να μεταφέρουν τον κώδικα τους σε διαφορετικά περιβάλλοντα. Αυτό δεν μπορούσε να γίνει με τη C++, οπότε δημιουργήθηκε η αντικειμενοστραφής γλώσσα προγραμματισμού Java, η οποία έλυνε 2 βασικά προβλήματα:

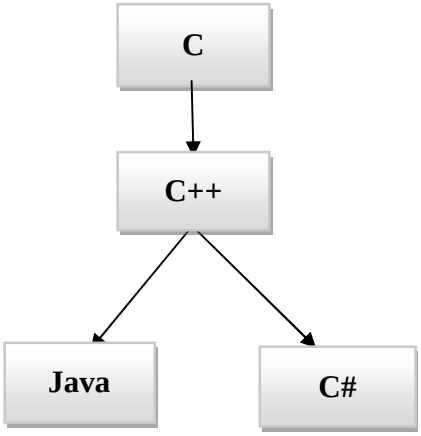
- **Μεταφερσιμότητα**: Μετάφραση του πηγαίου κώδικα στην ενδιάμεση γλώσσα bytecode και στη συνέχεια εκτέλεση αυτού του κώδικα στην εικονική μηχανή της Java (Java Virtual Machine -JVM)
- Ο κώδικας μπορούσε να εκτελεστεί σε οποιοδήποτε περιβάλλον διέθετε εικονική μηχανή Java.
- **Ασφάλεια**: Επειδή ο JVM τρέχει τον κώδικα bytecode, έχει τον πλήρη έλεγχο του προγράμματος έτσι ώστε να μπορεί να αποτρέψει οποιαδήποτε κακόβουλη ενέργεια ενός Java προγράμματος. [2]

3.1.3 Από τη C++ στη C#

Αν και η Java έλυσε 2 βασικά προβλήματα, της λείπουν ακόμη 2 χαρακτηριστικά:

- **Διαγλωσσική διαλειτουργικότητα**: για τη δημιουργία μεγάλων κατανεμημένων συστημάτων λογισμικού χρειάζεται ο κώδικας που παράγεται από μία γλώσσα, να συνεργάζεται με κώδικα που παράγει μία δεύτερη γλώσσα.
- **Πλήρης ενσωμάτωση με τα Windows**: η Java, αν και μπορεί να εκτελεστεί κάτω από τα Windows, δεν συνδέεται στενά με αυτά, σε αντίθεση με τη C#.[2]

3.1.4 Το Δέντρο εξέλιξης



[2]

[2] Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, 2013. Προγραμματισμός σε C Sharp. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.

3.1.5 Τι είναι το .NET Framework

Το .NET Framework είναι ένα περιβάλλον που υποστηρίζει την ανάπτυξη και την εκτέλεση άκρως κατανεμημένων εφαρμογών που βασίζονται σε συστατικά. Επιτρέπει στις γλώσσες προγραμματισμού να συνεργάζονται και παρέχει χαρακτηριστικά ασφάλειας, μεταφερτότητας και ένα κοινό μοντέλο προγραμματισμού για τα Windows. Παρέχει τα εξής:

- Σύστημα Common Language Runtime: διαχειρίζεται την εκτέλεση ενός προγράμματος σε C#.
- Βιβλιοθήκη Κλάσεων: παρέχει στα προγράμματα δυνατότητες πρόσβασης στο περιβάλλον χρόνου εκτέλεσης.

3.2 Απλό παράδειγμα χρήσης της C#

Θα δούμε παρακάτω τη σύνταξη και τη δομή ενός απλού προγράμματος στη C# που εμφανίζει το μήνυμα “Hello World” και θα αναλύσουμε στη συνέχεια γραμμή - γραμμή τα μέρη του.

```
1.    using System;

2.    namespace MyNamespace
3.
4.    {
5.       class Helloworld
6.       {
7.        static void Main()
a.        {
i.          Console.WriteLine("Hello World");
ii.          Console.ReadKey();

b.        }
8.       }
9.      }
```

Εικόνα 28: www.studentguru.gr

3.2.1 Χώροι ονομάτων (Namespaces)

Σε ένα C# πρόγραμμα χρησιμοποιούμε Namespaces για την οργάνωση του κώδικα μας. Στο παραπάνω πρόγραμμα χρησιμοποιήσαμε τον χώρο ονομάτων System (γραμμή 1 του παραδείγματος) μέσο της λέξης κλειδιού using ,το οποίο περιλαμβάνει έτοιμες κλάσεις και μεθόδους του .NET με έτοιμο επαναχρησιμοποιήσιμο κώδικα.

Ένα Namespace μπορεί να περιέχει τους εξής τύπους,

- Namespaces
- Classes
- Delegates
- Enums
- Structs
- Interfaces

Για τη δημιουργία ενός δικού μας Namespace χρησιμοποιούμε τη δεσμευμένη λέξη-κλειδί namespace (γραμμή 2), έτσι η κλάση Helloworld (γραμμή 4) εμπεριέχεται στο MyNamespace που δημιουργήσαμε.

Στη συνέχεια του κώδικα μπορούμε να δούμε πως μπορούμε να εμφανίσουμε στη γραμμή εντολών το μήνυμα "Hello World". Η μέθοδος Main() (γραμμή 6) περιέχει την εντολή Console.WriteLine("Hello World"); ,η οποία εμφανίζει το τελικό μήνυμα.

Προσοχή!

- Η κλάση Console ανήκει στο χώρο ονομάτων System και η WriteLine() είναι μέθοδος της Console. Εάν δεν χρησιμοποιούσαμε την εντολή using System; Η εντολή θα συντασσόταν System.Console.WriteLine("Hello Word");. Επίσης χρησιμοποιούμε την τελεία ανάμεσα από Namespaces, κλάσεις και μεθόδους για να δείξουμε τι εμπεριέχεται σε τι.
- Την εντολή Console.ReadKey(); την χρησιμοποιούμε απλά για να κρατήσουμε το παράθυρο της γραμμής εντολών ανοικτό και να δούμε το μήνυμα που εμφανίζουμε, το πρόγραμμα απλά περιμένει ο χρήστης να πατήσει κάποιο κουμπί και στη συνέχεια τερματίζει.

3.2.2 Ψευδώνυμα (Aliases)

Μπορούμε επίσης να χρησιμοποιήσουμε ψευδώνυμα για να κάνουμε τον κώδικα μας ακόμα πιο σύντομο και να αποφύγουμε να γράφουμε μακροσκελείς εντολές. Στη συγκεκριμένη περίπτωση το προηγούμενο παράδειγμα θα μπορούσε να γραφτεί και έτσι:

```
using  Alias = System.Console ;

namespace MyNamespace
{
    public class Helloworld
    {
       static void Main()
        {
            Alias.WriteLine("Hello World");
            Console.ReadKey();
        }
     }
 }
```

Εικόνα 29: www.studentguru.gr

3.2.3. Σχόλια στον Κώδικα

Μπορούμε να χρησιμοποιήσουμε σχόλια στον κώδικα μας κυρίως για να εξηγούμε κάποια σημεία του, όπως την λειτουργία μιας κλάσης/μεθόδου που δημιουργήσαμε, έτσι ώστε να γίνεται ο κώδικας εύκολος στην ανάγνωση τόσο για μας, αν θελήσουμε κάποια στιγμή να ανατρέξουμε σε αυτόν και να δούμε την λειτουργία του, αλλά και για κάποιο τρίτο πρόσωπο που θα χρειαστεί να τον δει.

Τα σχόλια συντάσσονται χρησιμοποιώντας αυτόν τον συμβολισμό (//), εάν πρόκειται για μία γραμμή σχολίου.

//αυτό είναι ένα σχόλιο

ή εάν το σχόλιο απαιτεί παραπάνω από μια γραμμή μπορούμε να το δηλώσουμε με τον παρακάτω τρόπο:

/* αυτό είναι

Ένα σχόλιο */

Με τον τρόπο αυτό ο Compiler προσπερνάει οτιδήποτε βρίσκεται μέσα στα εισαγωγικά και δεν επηρεάζει το τελικό αποτέλεσμα του προγράμματος.

3.2.4 Δήλωση Μεταβλητών, Ανάθεση-Σύγκριση τιμών & Εμφάνιση αποτελεσμάτων στην οθόνη

Σε ένα C# πρόγραμμα μπορούμε να δηλώσουμε μεταβλητές σε οποιοδήποτε σημείο του προγράμματος, πριν τις χρησιμοποιήσουμε για κάποια πράξη.

- Για να δηλώσουμε μία μεταβλητή ορίζουμε πρώτα τον τύπο της και ύστερα το όνομα της.
  Πχ. Int x; string y5;
- Για να κάνουμε ανάθεση τιμής σε μεταβλητές χρησιμοποιούμε το ίσον (=)
  Πχ. χ=8; ή η ίδια εντολή ως int x=2; εάν την δηλώναμε ταυτόχρονα μαζί με την ανάθεση της τιμής.
- Οι τελεστές που χρησιμοποιούνται είναι:

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ	
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
++	Προσαύξηση κατά 1
--	Μείωση κατά 1

ΣΧΕΣΙΑΚΟΙ ΤΕΛΕΣΤΕΣ	
==	Ίσο
!=	Διάφορο
>	Μεγαλύτερο
<	Μικρότερο
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο

ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ	
&	AND
|	OR
^	XOR
||	Βραχυκυκλωμένο OR
&&	Βραχυκυκλωμένο AND
!	NOT

· Για να εμφανίσουμε το περιεχόμενο μεταβλητών στην οθόνη η C# χρησιμοποιεί τη μέθοδο WriteLine() με δύο τρόπους σύνταξης.

1ος τρόπος

```
using System;

namespace MyNamespace
{
    class Sum_x1_x2
    {
        static void Main()
        {
            // 1ος τρόπος δήλωσης στην αρχή του προγράμματος
            int x1 = 5;
            int x2 = 3;
            int y = x1 + x2;

            // 1ος τρόπος σύνταξης της WriteLine
            Console.WriteLine(" Το αποτέλεσμα είναι : {0} ", y) ;
```

Εικόνα 30: www.studentguru.gr

Στην περίπτωση αυτή, το output είναι: *Το αποτέλεσμα είναι : 8*

Στη θέση του συμβόλου {0} η WriteLine τοποθετεί την τιμή που έχει το y.

Εάν θέλαμε να εμφανίσουμε παραπάνω μεταβλητές, θα γράφαμε το εξής:

Console.WriteLine (“{0} + {1} = {2}”, x1, x2, y)

Και το output εδώ θα ήταν 5+3=8

2ος τρόπος

```
string z = "!!!"; /*2ος τρόπος δήλωσης κατά τη ροή του
                    προγράμματος */

// 2ος τρόπος σύνταξης της WriteLine
Console.WriteLine("Το αποτέλεσμα είναι : {0}" +z ,y);
```

Εικόνα 31: www.studentguru.gr

Γιατί+z??? Στη WriteLine οτιδήποτε είναι μέσα σε εισαγωγικά αποτελεί string. Η μεταβλητή z είναι επίσης string. Βλέπουμε λοιπόν ότι μπορούμε να προσθέσουμε δύο strings μαζί και το αποτέλεσμα να είναι το ένα string να "κολλήσει" πίσω από το άλλο. Το output εδώ θα είναι : *Το αποτέλεσμα είναι : 8 !!!*

Παρατηρήσεις!

- Η C# είναι *Case sensitive*, δηλαδή, για να δηλώσουμε την Main() χρησιμοποιήσαμε κεφαλαίο Μ. Θα ήταν λάθος να λέγαμε static void main(). Ο compiler εξ ορισμού καταλαβαίνει ότι το βασικό πρόγραμμα βρίσκεται στην μέθοδο Main() και όχι στην main().
- **Keywords:** στη C# υπάρχουν κάποιες δεσμευμένες λέξεις-κλειδιά. Γι' αυτό το λόγο δεν μπορούμε να δηλώσουμε πχ. μια μεταβλητή με το όνομα console/int/void/class όπως επίσης δεν μπορούμε να βάλουμε και αριθμό μπροστά από την μεταβλητή.
- **Σταθερές (Constans):** Η δήλωση των σταθερών στο πρόγραμμα όπως πχ. η τιμή του π=3,14 γίνεται με αυτόν τον τρόπο: const double p = 3.14;
- Στη WriteLine() υπάρχουν τρόποι *εμφάνισης μεταβλητών*.
  Για: int x=1;
  int y=2;
  int sum = x+y;
  - **1ος τρόπος (ως συνεχόμενο string)**
    Console.WriteLine (x + "+" + y +"=" + sum); **à** output: 1+2=3
  - **2ος τρόπος (με int μεταβλητές)**
    Console.WriteLine("{0}+{1}={2}",x,y,sum); **à** output: 1+2=3

3.3 Δομές Ελέγχου στη C#

3.3.1 Η Δομή if

Σύνταξη της δομής είναι η εξής:

If (συνθήκη ελέγχου)
{
Ομάδα εντολών 1
}
{
Ομάδα εντολών 2
} [3]

Η δομή λειτουργεί ως εξής:
Πρώτα ελέγχεται η συνθήκη. **Αν** ισούται με **true** τότε εκτελείται η ομάδα εντολών 1 **αλλιώς** εκτελείται η ομάδα εντολών 2.
Η συνθήκη ελέγχου είναι συνήθως μία ισότητα ή ανισότητα. Η ισότητα εκφράζεται με το σύμβολο == και όχι με το απλό = (το οποίο σημαίνει "ανάθεση").

3.3.2 Η Δομή switch

Η σύνταξη της δομής είναι η εξής:

switch (επιλογέας)
{
case τιμή 1: εντολές; **break**;
case τιμή 2: εντολές; **break**;
…
case τιμή n: εντολές; **break**;
default: εντολές; **break**;
} [3]

Η δομή λειτουργεί ως εξής:
Στην αρχή της δομής ελέγχεται η τιμή του **επιλογέα**. Ανάλογα με την τιμή του εκτελούνται οι εντολές του αντίστοιχου case και στη συνέχεια το break (το οποίο και τερματίζει το switch). Αν οι case διαφέρουν από την τιμή του επιλογέα, τότε εκτελούνται τα default.

[3] Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, 2013. Προγραμματισμός σε C Sharp. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.

3.4 Δομές Επανάληψης στη C#

3.4.1 Η δομή while

Η σύνταξη της δομής είναι η εξής:

while (συνθήκη)
{
Ομάδα εντολών
} [4]

Η δομή λειτουργεί ως εξής:
Αρχικά ελέγχεται η συνθήκη. Αν το αποτέλεσμα ισούται με **true**, τότε εκτελείται ο κώδικας που βρίσκεται ανάμεσα στα άγκιστρα και στη συνέχεια ο έλεγχος επανέρχεται στην κορυφή του **while**. Αν το αποτέλεσμα ισούται με **false**, τότε η συνθήκη παρακάμπτεται.

3.4.2 Η δομή do while

Η σύνταξη της δομής είναι η εξής:

do
{
Ομάδα εντολών
}
while (συνθήκη) [4]

Η δομή λειτουργεί ως εξής:
Αρχικά εκτελούνται οι εντολές που βρίσκονται ανάμεσα στα άγκιστρα & στο **τέλος ελέγχεται** η συνθήκη που βρίσκεται μέσα στην παρένθεση. Αν το αποτέλεσμα ισούται με **true** τότε εκτελείται **εκ νέου** η ομάδα εντολών μέσα στα άγκιστρα. Σε διαφορετική περίπτωση το πρόγραμμα συνεχίζει στις **επόμενες** εντολές.

[4] Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, 2013. Προγραμματισμός σε C Sharp. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.

3.4.3 Η δομή της for

Η σύνταξη της δομής είναι η εξής:

for (αρχική τιμή ; έλεγχος ; ανανέωση)
{
…
} [5]

Η δομή λειτουργεί ως εξής:
Σε πρώτη φάση εκτελείται η **αρχική τιμή**. Στην ουσία πρόκειται για μια **αρχικοποίηση ενός μετρητή** πριν την εκτέλεση του κώδικα που βρίσκεται ανάμεσα στα άγκιστρα. Στη συνέχεια ελέγχεται αν είναι true η **συνθήκη ελέγχου**. Αν το αποτέλεσμα είναι αληθές τότε εκτελούνται οι εντολές που βρίσκονται ανάμεσα στα άγκιστρα. Στο τέλος εκτελείται η **έκφραση ανανέωσης** και στη συνέχεια ελέγχεται ξανά η **συνθήκη ελέγχου** και όλη η διαδικασία επαναλαμβάνεται μέχρι η συνθήκη να **αποτύχει**.

3.5 Κλάσεις, Αντικείμενα & Μέθοδοι

3.5.1 Ορισμός Κλάσης

Κλάση είναι ένα πρότυπο που ορίζει την μορφή ενός αντικειμένου. Καθορίζει τα δεδομένα και τον κώδικα που θα δρα πάνω σε αυτά τα δεδομένα. Τα αντικείμενα είναι στιγμιότυπα της κλάσης στην οποία ανήκουν. Ότι ανήκει μέσα σε μία κλάση αποτελεί μέλος της κλάσης.

Οι περισσότερες κλάσεις περιέχουν κώδικα και δεδομένα. Γενικότερα μια κλάση μπορεί να έχει ως μέλη τα εξής: μεταβλητές στιγμιότυπου, στατικές μεταβλητές, σταθερές, μεθόδους, κατασκευαστές, καταστροφείς, δεικτοδότες, συμβάντα, τελεστές και ιδιότητες.

3.5.1.1 Καθοριστής Προσπέλασης Μέλους

Ο καθοριστής προσπέλασης μέλους καθορίζει ποιος μπορεί να χρησιμοποιήσει την μεταβλητή ή τη μέθοδο μιας κλάσης. Υπάρχουν 4 καθοριστές προσπέλασης:

- **public** (δημόσια) : το μέλος μπορεί να προσπελαύνεται από κάθε κώδικα που βρίσκεται μέσα σε ένα πρόγραμμα, φυσικά και εκτός της κλάσης.
- **private** (ιδιωτική) : το μέλος μπορεί να προσπελαστεί μόνο από άλλα μέλη της κλάσης στην οποία βρίσκεται.
- **protected** (προστατευμένη)

[5] Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, 2013. Προγραμματισμός σε C Sharp. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.

· **internal** (εσωτερική)

Ο καθοριστής προσπέλασης είναι προαιρετικός. Αν λείπει, τότε το μέλος είναι private (ιδιωτικό). [6]

3.5.1.2 Παράδειγμα: η κλάση Amaksi

Μια κλάση πρέπει να ορίζει μία και μόνο λογική οντότητα. Αν θέλουμε να έχουμε συγκεντρωμένα στοιχεία σχετικά με οχήματα, οπότε μας ενδιαφέρει να αποθηκεύσουμε πληροφορία σχετικά με την μάρκα, με τον αριθμό επιβατών και την χωρητικότητα σε καύσιμο, τότε μας βολεύει να ορίσουμε μια κλάση με το όνομα Amaksi η οποία θα είναι η εξής:

```
class Amaksi{
        public string marka;
        public int atoma;
        public int kausima;
        public double katanalwsh;
        }
```

Η κλάση αυτή περιέχει μόνο δεδομένα, δηλαδή μεταβλητές. Τέτοιου τύπου μεταβλητές ονομάζονται μεταβλητές στιγμιοτύπου. Για να μπορούμε να χρησιμοποιήσουμε την κλάση θα πρέπει να ορίζουμε ένα στιγμιότυπο, δηλαδή ένα αντικείμενο.

3.5.2 Δημιουργία ενός αντικειμένου

Ένα αντικείμενο της κλάσης Amaksi που είδαμε προηγουμένως μπορεί να δημιουργηθεί ως εξής:

```
Amaksi ix = new Amaksi();
```

3.5.2.1 Λειτουργία ενός αντικειμένου

Αρχικά δηλώνουμε μια μεταβλητή με το όνομα ix τύπου κλάσης Amaksi. Αυτή η μεταβλητή δεν είναι από μόνη της ένα αντικείμενο. Το αντικείμενο δημιουργείται πραγματικά με τον τελεστή new και στο τέλος αναθέτουμε την μεταβλητή ix, μία αναφορά στο αντικείμενο που μόλις δημιουργήθηκε στη μνήμη. Γι' αυτό το λόγο οι κλάσεις ονομάζονται τύποι αναφοράς. Ο τελεστής new κατανέμει δυναμικά μνήμη για ένα αντικείμενο και επιστρέφει μία αναφορά προς αυτό, η οποία αποθηκεύεται σε μία μεταβλητή.

[6] Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, 2013. Προγραμματισμός σε C Sharp. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.

3.5.2.2 Τύποι αναφοράς

Οι διαφορές μεταξύ **τύπων τιμών** και **τύπων αναφορών** είναι μεγάλες. Μία μεταβλητή τύπου τιμής περιέχει η ίδια την τιμή. Για παράδειγμα, αν είχαμε την δήλωση : **int x = 10;** τότε η μεταβλητή x περιέχει πραγματικά την τιμή 10.

Σε αντίθετη περίπτωση, μια μεταβλητή τύπου αναφοράς όπως η μεταβλητή ix στη δήλωση **Amaksi ix = new Amaksi ();** δεν περιέχει το αντικείμενο αλλά **μία αναφορά προς το αντικείμενο.**

Επίσης, σε περίπτωση ανάθεσης, ενώ σε μεταβλητές τύπου τιμής απλά η μεταβλητή στο αριστερό μέρος δέχεται ένα αντίγραφο της τιμής της μεταβλητής στο δεξιό μέρος, στις μεταβλητές τύπου αναφοράς η μεταβλητή στο αριστερό μέρος αναφέρεται στο ίδιο αντικείμενο που αναφέρεται και η μεταβλητή στο δεξί μέρος.

Amaksi car1=new Amaksi();

Amaksi car2=car1;

Αυτό σημαίνει, ότι αν προσπαθήσουμε να αλλάξουμε τιμή σε οποιαδήποτε μεταβλητή-μέλος του αντικειμένου τότε η αλλαγή θα ισχύει και για τις δυο μεταβλητές.

3.5.3 Μέθοδοι

Οι μέθοδοι κλάσης είναι υπορουτίνες που χειρίζονται τα δεδομένα της και σε πολλές περιπτώσεις παρέχουν πρόσβαση σε αυτά. Ουσιαστικά, άλλα μέρη του προγράμματος θα αλληλεπιδρούν με μία κλάση μέσω των μεθόδων της. Τέλος, μία μέθοδος υλοποιεί μια συγκεκριμένη λειτουργία, μπορεί να χρειάζεται ορίσματα για να δουλέψει και μετά το πέρας λειτουργίας της να επιστρέψει κάτι σε αυτόν που την κάλεσε.

3.5.3.1 Σύνταξη της Μεθόδου

Η γενική μορφή μιας μεθόδου είναι η εξής:

Προσπέλαση τύπος_επιστροφής όνομα_μεθόδου (παράμετροι) {
//κώδικας
}

Η προσπέλαση καθορίζει ποιος μπορεί να χρησιμοποιεί αυτή την μέθοδο, ο τύπος_επιστροφής καθορίζει τι τύπου είναι η πληροφορία που επιστρέφει η μέθοδος και εντός των παρενθέσεων υπάρχουν τα ονόματα και οι τύποι των ορισμάτων (χωρισμένα με κόμματα) που χρειάζεται για να δουλέψει.

Κεφάλαιο 4 : Η C# στο Unity3d

Εικόνα 32: κολλάζ εικόνων από www.ouedkniss.com και www.unity3d.com

Στο προηγούμενο κεφάλαιο κάναμε αναφορά στη γλώσσα προγραμματισμού C# και στον τρόπο λειτουργίας της. Σε αυτό το κεφάλαιο θα αναφερθούμε στη C# και τις μεθόδους τις οποίες χρησιμοποιεί στο πρόγραμμα του Unity3d.

Το Unity3d, έχει κάποιες συγκεκριμένες συναρτήσεις και κλάσεις, οι οποίες καλούνται ανάλογα με τις ενέργειες τις οποίες θέλουμε να εκτελέσουμε.

4.1 MonoBehaviour

Η *MonoBehaviour* αποτελεί τη βασική κλάση, από την οποία προέρχεται το κάθε script της C# στο πρόγραμμα του Unity3d. Με τη συγκεκριμένη κλάση, κάνουμε χρήση των παρακάτω συναρτήσεων.

4.1.1 Void Awake ()

Η *Awake*, αποτελεί μία συνάρτηση η οποία καλείται, κατά τη διάρκεια την οποία ένα script φορτώνεται. Χρησιμοποιείται δηλαδή για την αρχικοποίηση των μεταβλητών ή για οποιαδήποτε κατάσταση θέλουμε να εκτελεστεί, πριν το παιχνίδι ξεκινήσει. Την Awake καθώς και τις υπόλοιπες συναρτήσεις που θα αναφέρουμε, μπορούμε να τις καλέσουμε *μόνο μία φορά*.

```
using UnityEngine;
using System.Collections;

public class Example : MonoBehaviour {
        private GameObject target;
        void Awake(){
                target = GameObject.FindWithTag("Gunner");
        }
}
```
[7]

[7] Unity Official Website, 2005. Unity, από http://unity3d.com

4.1.2 Void Start ()

Η *Start*, καλείται όταν ένα σενάριο έχει ενεργοποιηθεί και πριν οποιαδήποτε Update method καλεστεί για πρώτη φορά από το πρόγραμμα.

Χρήση της Start γίνεται με τον ίδιο τρόπο όπως και της Awake, αλλάζοντας απλός το void Awake () σε void Start (), προκειμένου να καταλάβει το πρόγραμμα πότε και πως θα εκτελέσει το εκάστοτε κομμάτι κώδικα μέσα στη συνάρτηση.

4.1.3 Void Update ()

Η *Update*, καλείται σε κάθε frame του προγράμματος μας, εφόσον είναι ενεργοποιημένη η MonoBehaviour. Η Update είναι η πιο συχνά χρησιμοποιούμενη λειτουργία για την εφαρμογή σε κάθε είδους συμπεριφορά του παιχνιδιού.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
        void Update(){
                transform.Translate(0, 0, Time.deltaTime * 10);
        }
}
```
[8]

4.1.4 Void FixedUpdate ()

Η Fixed*Update*, αποτελεί μία συνάρτηση η οποία καλείται σε κάθε fixed framerate frame, όταν η MonoBehaviour είναι ενεργοποιημένη.

Η FixedUpdate πρέπει να χρησιμοποιείται αντί της Update, όταν μέσα στο πρόγραμμα μας υπάρχει η επιλογή Rigidbody (ελέγχει τη θέση ενός αντικειμένου με βάση τους φυσικούς νόμους).

```
using UnityEngine;
```

[8] Unity Official Website, 2005. Unity, από http://unity3d.com

```
using System.Collections;

public class Example : MonoBehaviour {
        public Rigidbody2D rb;
        void Start(){
                rb = GetComponent<Rigidbody2D>();
        }
        void  FixedUpdate(){
                rb.AddForce(Vector3.up);
        }}
```
[9]

4.2 Περαιτέρω χαρακτηριστικά της κλάσης MonoBehaviour

Όπως αναφέραμε παραπάνω, η MonoBehaviour είναι η βασική κλάση την οποία χρησιμοποιεί το development system του Unity. Αν η γλώσσα στην οποία γράφουμε είναι η JavaScript, τότε η MonoBehaviour εισάγεται αυτόματα καθώς υπάρχει σαν προεπιλογή, ενώ αν γράφουμε C#, θα πρέπει να την εισάγουμε στον κώδικα μας, όπως φαίνεται στα παραπάνω παραδείγματα.

Πέρα από τις συναρτήσεις στις οποίες αναφερθήκαμε, οι οποίες στην MonoBehaviour καταγράφονται ως *Messages*, υπάρχουν και άλλου είδους συναρτήσεις στη συγκεκριμένη κλάση, οι οποίες εκτελούν συγκεκριμένες λειτουργίες.

4.2.1 Public functions & variables

Όπως αναφέρει και η λέξη, *public*, οι μεταβλητές και οι συναρτήσεις αυτές είναι δημόσιες. Με τον όρο δημόσια, εννοούμε πως ο χρήστης μπορεί να επέμβει σε αυτές και να τις τροποποιήσει, χωρίς να χρειαστεί να επέμβει στον κώδικα. Δημόσιες μεταβλητές και συναρτήσεις, είναι εκείνες οι οποίες έχουν να κάνουν κυρίως με κάποια χαρακτηριστικά ενός παιχνιδιού, όπως είναι ο χρόνος ή η ταχύτητα, τα οποία θέλουμε να εισάγουμε και που πιθανότατα να χρειαστούν αλλαγή.

Παράδειγμα Public function & variables

Μία χαρακτηριστική *public function* της MonoBehaviour είναι η *GetComponents*, η οποία παίρνει τα χαρακτηριστικά του Component το οποίο έχουμε εισάγει στο αντικείμενο μας και μπορεί να τα ορίσει σε κάποια μεταβλητή ή και να τα τροποποιήσει.

Ως *public variable*, ορίζεται κάθε μεταβλητή, η οποία έχει την παρακάτω μορφή,

public float speed;

[9] Unity Official Website, 2005. Unity, από http://unity3d.com

Όπου με public δηλώνουμε ότι το είδος της μεταβλητής είναι “δημόσια μεταβλητή”, με το float ότι το είδος της μεταβλητής είναι “δεκαδική μεταβλητή”, ενώ το speed είναι το όνομα που δίνουμε στη μεταβλητή μας.

4.2.2 Private functions & variables

Όταν μιλάμε για *private* συναρτήσεις και μεταβλητές, μιλάμε για εκείνες οι οποίες αφορούν κάποια λειτουργία του κώδικα και τις οποίες ο χρήστης δεν μπορεί να τις δει ή να τις τροποποιήσεις αν δεν επέμβει στον κώδικα. Τέτοιου είδους συναρτήσεις και μεταβλητές είναι εκείνες οι οποίες ρυθμίζουν κάποια χαρακτηριστικά του παιχνιδιού μας, όπως είναι η κίνηση που θα δώσουμε σε ένα αντικείμενο ή τον τρόπο με τον οποίο αυτό το αντικείμενο θα μετακινείτε.
Ως *private variable*, ορίζεται κάθε μεταβλητή, η οποία έχει την παρακάτω μορφή,

private int frame;

Όπου με private δηλώνουμε ότι το είδος της μεταβλητής είναι “ιδιωτική μεταβλητή”, με το int ότι το είδος της μεταβλητής είναι “ακέραια μεταβλητή”, ενώ το frame είναι το όνομα που δίνουμε στη μεταβλητή μας.

4.2.3 Χαρακτηριστικά 2D vs 3D

Η δημιουργία ενός παιχνιδιού 2D και ενός 3D, πέρα από τον χώρο και τη διαφορά ανάμεσα στις διαστάσεις που έχουν, διαφέρουν και στον τρόπο χρήσης των μεταβλητών τους. Η MonoBehaviour είναι η βασική κλάση στη δημιουργία και των 2D αλλά και των 3D παιχνιδιών, αλλά οι μεταβλητές και τα components τα οποία χρησιμοποιεί για την υλοποίηση τους είναι διαφορετικά από την διπλή στην τριπλή διάσταση.

Χαρακτηριστικό παράδειγμα αποτελεί το component Rigidbody, το οποίο στον κώδικα το χρησιμοποιούμε και σαν μεταβλητή, όπου στην περίπτωση των δύο διαστάσεων, το εισάγουμε και το χρησιμοποιούμε ως Rigidbody2D, ενώ στην περίπτωση των τριών διαστάσεων, το εισάγουμε και το χρησιμοποιούμε ως Rigidbody. Άλλες μεταβλητές και components που χρησιμοποιούνται με τον ίδιο ακριβώς τρόπο, είναι όλα τα είδη Collider που υπάρχουν (Box στις τρεις διαστάσεις και Circle στις δύο διαστάσεις κ.α.), τα οποία στη μία περίπτωση εισάγονται και χρησιμοποιούνται ως Collider2D και στην άλλη, απλά ως Collider, όλα τα είδη Joint κ.ο.κ.

Κεφάλαιο 5 : Οι εφαρμογές μας

5.1 Εισαγωγή : 2D Car Racing

Εικόνα 33: Δικά μας γραφικά

Το 2D Car Racing, είναι ένα παιχνίδι αγώνων formula σε δύο διαστάσεις, στο οποίο ο χρήστης πρέπει να μαζέψει ένα συγκεκριμένο αριθμό πόντων (200 points), για να μπορέσει να περάσει στο επόμενο επίπεδο.

Ο σχεδιασμός και η ανάπτυξη του παιχνιδιού, στηρίχθηκε σε έτοιμα γραφικά από το Asset Store του Unity, αλλά και σε γραφικά τα οποία τα δημιουργήσαμε εμείς οι ίδιοι για τις ανάγκες του παιχνιδιού.

Το παιχνίδι λειτουργεί με τον εξής τρόπο:

- Το αυτοκίνητο έχει μία συγκεκριμένη ταχύτητα με την οποία τρέχει σταθερά, χωρίς ο χρήστης να πατάει κάποιο κουμπί, προσπαθώντας ταυτόχρονα να προσπεράσει τους αντιπάλους του και να συλλέξει τις ρόδες που θα βρει στο δρόμο του, προκειμένου να μαζέψει παραπάνω πόντους και να φτάσει το επιθυμητό όριο τον 200 πόντων.
- Η formula κινείται μόνη της πάνω στον άξονα y (ευθεία), ενώ ο χρήστης μπορεί να την μετακινήσει πάνω στον άξονα x (δεξιά – αριστερά), περιστρέφοντας την κινητή του συσκευή στην ανάλογη κατεύθυνση.

5.1.2 Υλοποίηση του παιχνιδιού

5.1.2.1 Αρχική Οθόνη

Κάθε παιχνίδι αποτελείται από τη βασική, οθόνη έναρξης, η οποία βοηθάει το χρήστη να πλοηγηθεί μέσα στην εφαρμογή και στις διάφορες επιλογές τις οποίες έχουμε δημιουργήσει.

Η δική μας οθόνη έναρξης αποτελείται από τέσσερα buttons τα οποία βοηθούν τον χρήστη να πλοηγηθεί στις *ρυθμίσεις* του παιχνιδιού, σε κάποιες *πληροφορίες* σχετικά με το παιχνίδι, στην *εκκίνηση* του παιχνιδιού και στην εξαγωγή από την εφαρμογή.

Εικόνα 34: Δικά μας γραφικά

5.1.2.2 Ρυθμίσεις παιχνιδιού

Εικόνα 35: Δικά μας γραφικά

Μέσα από τις ρυθμίσεις του παιχνιδιού, ο χρήστης μπορεί να εκτελέσει τις εξής ενέργειες:

- Μπορεί να επιλέξει αν το παιχνίδι θα έχει ήχο ή όχι, με τη χρήση των δύο πάνω buttons που φαίνονται στην εικόνα αριστερά.
- Μπορεί να επιλέξει με το return button να επιστρέψει στην προηγούμενη οθόνη, που στη συγκεκριμένη περίπτωση είναι η οθόνη έναρξης,
- Μπορεί με το play button να ξεκινήσει αμέσως το παιχνίδι, χωρίς να χρειαστεί να κάνει το περιττό πίσω βήμα και μέσω του race button να ξεκινήσει το παιχνίδι.

5.1.2.3 Οθόνη πληροφοριών

Μέσα από την οθόνη πληροφοριών, ο χρήστης μπορεί να δει κάποιες πληροφορίες οι οποίες αφορούν τον τρόπο λειτουργίας του παιχνιδιού και να διαβάσει το λόγο υλοποίησης του.

Στη συγκεκριμένη οθόνη, διατηρήσαμε το return button και το play button, προκειμένου ο χρήστης να μπορεί να κινηθεί όπως επιθυμεί μέσα στην εφαρμογή χωρίς να έχει περιορισμούς.

Εικόνα 36: Δικά μας γραφικά

5.1.2.4 Κώδικας πλοήγησης και event στις παραπάνω οθόνες

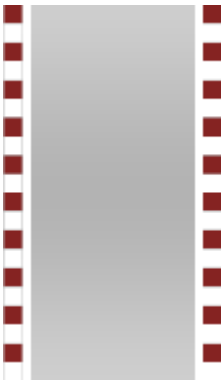
Ο κώδικα υλοποίησης των παραπάνω οθονών είναι πολύ απλός, καθώς με τη δημιουργίας κάποιων event functions, δίνει κάποιες συγκεκριμένες οδηγίες στα παραπάνω κουμπιά, προκειμένου να εκτελέσουν μία ενέργεια.

```
public void AudioPause(){
        AudioListener.volume = 0.0f;
} // Σταματάμε τον ήχο στο παιχνίδι
public void AudioPlay(){
        AudioListener.volume = 1.0f;
} // Ενεργοποιούμε τον ήχο στο παιχνίδι
public void Menu(){
        Application.LoadLevel("menuScene");
} // Φορτώνουμε τη οθόνη του μενού
public void Info(){
        Application.LoadLevel("info");
} // Φορτώνουμε τη οθόνη των πληροφοριών
public void Settings(){
        Application.LoadLevel("Settings");
} // Φορτώνουμε τη οθόνη των ρυθμίσεων
```

5.1.2.5 Η οθόνη του παιχνιδιού

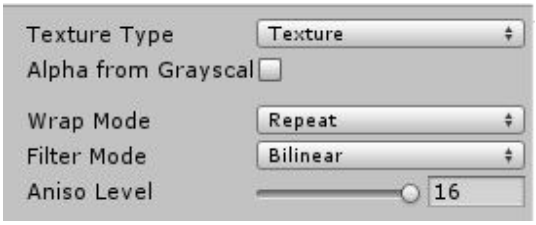
Όπως προαναφέραμε, το παιχνίδι μας, αποτελεί ένα παιχνίδι αγώνων formula, με σκοπό τη συλλογή κάποιων πόντων, προκειμένου να περάσουμε στο επόμενο επίπεδο.

Για να υλοποιηθεί αυτό, θα χρειαστούμε αυτοκίνητα καθώς και μία πίστα πάνω στην οποία θα τρέχουν.



Εικόνα 38: Δικά μας γραφικά

Εικόνα 37: Έτοιμα γραφικά από το Asset Store του Unity



Εικόνα 39: PrintScreen από το περιβάλλον του Unity

Από τα αυτοκίνητα τα οποία τρέχουν πάνω στην πίστα, το κόκκινο αποτελεί το αυτοκίνητο του χρήστη, ενώ τα υπόλοιπα έχουν το ρόλο των enemy cars, τα οποία πρέπει να προσπερνάει ο χρήστης κατά τη διάρκεια του παιχνιδιού.

Το αυτοκίνητο του χρήστη, όπως αναφέραμε παραπάνω, τρέχει με συγκεκριμένη ταχύτητα πάνω στον άξονα y, ενώ πάνω στον άξονα x, μπορεί να μετακινηθεί με την περιστροφή της κινητής συσκευής, δεξιά ή αριστερά.

Τα υπόλοιπα αυτοκίνητα εμφανίζονται σε συγκεκριμένο χρόνο πάνω σε διάφορα σημεία της πίστας, ενώ η ταχύτητα τους είναι σταθερή, πάνω στον άξονα y, χωρίς να έχουν τη δυνατότητα μετακίνησης πάνω στον άξονα x.

Την πίστα την έχουμε ορίσει μέσα από το unity και το inspector window, κατά τη διάρκεια που το παιχνίδι τρέχει, να *επαναλαμβάνεται* (Wrap Mode – Repeat), προκειμένου να δίνεται η αίσθηση της συνέχειας, ενώ η διάρκεια της είναι ορισμένη στο άπειρο, προκειμένου να μπορούν τα αυτοκίνητα να τρέχουν πάνω της συνεχώς.

5.1.2.6 Ο κώδικας του κόκκινου αυτοκινήτου

```
using UnityEngine;
using System.Collections;  // Καλούμε τις βιβλιοθήκες τις οποίες θα χρειαστούμε
using UnityEngine.UI;

public class carController : MonoBehaviour {  // Όπως αναφέραμε στο κεφάλαιο “Η C#
στο Unity3d”                                  //η MonoBehaviour είναι η βασική
κλάση την οποία                               // χρησιμοποιούμε

        public float carSpeed;
        public float maxPos = 2.75f;
        public uiManager ui;          // Ορίζουμε private και public μεταβλητές που θα
        public Rigidbody2D rb;        // χρησιμοποιήσουμε
        private Vector3 acceleration;


        void Start () {
                rb = GetComponent<Rigidbody2D> ();  // Κατά την έναρξη του παιχνιδιού
καλούμε         }                                   // τις φυσικές παραμέτρους τις
οποίες                                              //ορίζουμε στη μεταβλητή
rb
```

```
        void Update(){                                  // Κατά τη διάρκεια του παιχνιδιού
                                                        // εισάγουμε στη
                Vector3 dir = Vector3.zero; // μεταβλητή dir τις ιδιότητες του Vector 3
                dir.x = Input.acceleration.x;// Το αυτοκίνητο (Vector3) μετακινείται δεξιά
                dir.x = -Input.acceleration.x;          // Το αυτοκίνητο (Vector3) μετακινείται
                                                        //αριστερά
                if (dir.sqrMagnitude > 1)
                        dir.Normalize();

                dir *= Time.deltaTime;                  // Το αυτοκίνητο κινείται σε πραγματικό
                                                        // χρόνο,
                transform.Translate(dir * carSpeed); // με συγκεκριμένη ταχύτητα
                rb.velocity = acceleration * carSpeed * Time.deltaTime; // ορίζουμε τον
                                                                        // τρόπο
                                                                        // μετακίνησης
                transform.Translate(rb.velocity * carSpeed * Time.deltaTime);  // Το
                                                                               //αυτοκίνητο
                                                                               // κινείται σε
                // πραγματικό χρόνο, με συγκεκριμένη ταχύτητα, με συγκεκριμένο τρόπο
        }

        void OnCollisionEnter2D(Collision2D col){       // Κατά τη σύγκρουση
                if (col.gameObject.tag == "Enemy") {    // με αντίπαλο αυτοκίνητο
                        Destroy (gameObject);           // το αμάξι μας καταστρέφεται
                        ui.gameOverActivated();     // και το παιχνίδι τερματίζει
                }
                if (col.gameObject.tag == "Box") { // Κατά τη σύγκρουση με τη ρόδα
                        Destroy (col.gameObject);  // η ρόδα καταστρέφεται
                        ui.UpScore();              // και παίρνουμε τους extra πόντους
        }
        }
}
```

5.1.2.7 Ο κώδικας των αντίπαλων αυτοκινήτων

Ο τρόπος κίνησης των αντίπαλων αυτοκινήτων, εξαρτάται από τον παρακάτω κώδικα,

```
using UnityEngine;
using System.Collections;  // Καλούμε τις βιβλιοθήκες τις οποίες θα χρειαστούμε

public class EnemyCarMove: MonoBehaviour {  // Όπως αναφέραμε στο κεφάλαιο
                                            // “Η C# στο Unity3d”
                                                //η MonoBehaviour είναι η βασική κλάση
την οποία                                                   // χρησιμοποιούμε

        public float speed = 8f;  // Ορίζουμε την ταχύτητα μας

        void Start () {

        }

        void Update () {
                transform.Translate (new Vector3 (0, 1, 0) * speed * Time.deltaTime);
                // Τα αυτοκίνητα μετακινούνται συνεχώς στον άξονα y, που έχουμε ορίσει
με 1
                // με την ταχύτητα που έχουμε ορίσει
                // σε πραγματικό χρόνο
        }
}
```

Ο συνεχής πολλαπλασιασμός και η εμφάνιση των αντίπαλων αυτοκινήτων εξαρτάται από τον παρακάτω κώδικα,

```
using UnityEngine;
using System.Collections;  // Καλούμε τις βιβλιοθήκες τις οποίες θα χρειαστούμε

public class carSpawner: MonoBehaviour {  // Όπως αναφέραμε στο κεφάλαιο “Η C# στο
Unity3d”                                                    //η MonoBehaviour είναι η βασική
κλάση την οποία                                                   // χρησιμοποιούμε
```

```
public GameObject[ ] cars; // Ορίζουμε πίνακα για να εισάγουμε τα αυτοκίνητα
int carNo;                 // Ορίζουμε μεταβλητή για τα αυτοκίνητα
public float maxPos = 2.68f;       // Ορίζουμε περιθώρια εμφάνισης τους
public float delayTimer = 2f;      // Ορίζουμε χρόνο εμφάνισης
float timer;               // Ορίζουμε τον χρονομετρητή

void Start () {
        timer = delayTimer;  // Ισοδυναμούμε χρονομετρητή με χρόνο εμφάνισης
}

void Update () {
        timer -= Time.deltaTime;  // Ο χρονομετρητής μειώνει σταδιακά τον χρόνο εμφάνισης
                                  // σε συνάρτηση με τον πραγματικό χρόνο
        if (timer <= 0){              // Όταν ο χρονομετρητής φτάσει στο μηδέν
        Vector3 carPos = new Vector3 (Random.Range (-2.68f, 2.68f),transform.position.y, transform.position.z);          // δημιουργείτε ένα αυτοκίνητο
                carNo = Random.Range (0,4);      // με τυχαία σειρά από τον πίνακα
                Instantiate (cars[carNo], carPos, transform.rotation);
                timer = delayTimer;
        }
}
}
```

5.1.2.8 Συλλογή πόντων στο παιχνίδι

Στο παιχνίδι μας εισήγαμε μία ρόδα, ο ρόλος της οποίας είναι, κατά τη σύγκρουση με το αυτοκίνητο μας, να του δίνει extra πόντους, που θα μας βοηθήσουν να φτάσουμε στο επιθυμητό αποτέλεσμα.

Εικόνα 40: Έτοιμο γραφικό από το Asset Store του Unity

Ο κώδικας τον οποίο χρησιμοποιήσαμε για τη συλλογή των extra πόντων είναι ο εξής,

```
void OnCollisionEnter2D(Collision2D col){  // Κατά τη σύγκρουση
        if (col.gameObject.tag == "Box") {  // με τη ρόδα
                Destroy (col.gameObject); // η ρόδα καταστρέφεται
                        ui.UpScore(); }   // και καλείται η συνάρτηση UpScore ()
        }
}


public void UpScore(){                    // Όταν καλείται αυτή η συνάρτηση
```

```
        if (gameOver == false) {      // και το παιχνίδι ακόμα δεν έχει λήξει
            score += 5;               // το score αυξάνεται κατά 5 πόντους
        }
    }
```

Η ρόδα θέλουμε να εμφανίζεται με την ίδια συχνότητα που εμφανίζονται και τα αντίπαλα αυτοκίνητα. Για το λόγο αυτό εισάγουμε τη ρόδα στον ίδιο πίνακα που έχουμε εισάγει και τα αντίπαλα αυτοκίνητα.

Όταν το σκορ φτάσει στους 200 πόντους όπως ορίζει ο παρακάτω κώδικας, τότε φορτώνεται αυτόματα το επόμενο επίπεδο του παιχνιδιού,

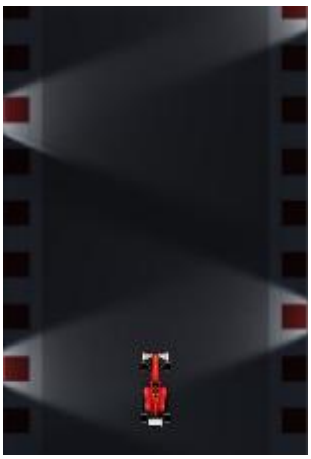
```
void Update () {
        scoreText.text = "Score: " + score; //Εμφανίζουμε το σκορ στην οθόνη

        if (score >= 200) {          // Όταν το σκορ είναι μεγαλύτερο ή ίσο με
200
            Application.LoadLevel("2D Racing_level_2"); // Φορτώνεται το
                                                        // επόμενο level
        }
    }
```

5.1.2.9 Μήνυμα προειδοποίησης

Σκοπός του παιχνιδιού μας, είναι η συλλογή 200 πόντων, προκειμένου να περάσουμε στο επόμενο επίπεδο. Το επόμενο επίπεδο ξεκινάει αμέσως μόλις ο χρήστης συλλέξει τον επιθυμητό αριθμό πόντων. Για το λόγο αυτό, όταν ο χρήστης συλλέξει 180 πόντους, εμφανίζεται στην οθόνη το ακόλουθο μήνυμα, “Twenty more points for next level. The next level starts immediately”, για να είναι προετοιμασμένος.
Το μήνυμα εμφανίζεται με τον κάτωθι κώδικα,

```
    if (score >= 180) {                  // Όταν το σκορ φτάσει τους 180
πόντους
        Message.gameObject.SetActive (true);   // Εμφάνισε το μήνυμα
    }
    if (score >= 190) {                  // Όταν το σκορ φτάσει τους 190
πόντους
        Message.gameObject.SetActive (false);  // Εξαφάνισε το μήνυμα
    }
```

5.1.2.10 Το δεύτερο επίπεδο του παιχνιδιού



Εικόνα 41: : Δικά μας γραφικά

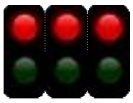
Μόλις ο χρήστης συλλέξει 200 πόντους, περνάει αυτόματα στο επόμενο επίπεδο, το οποίο έχει κάποιες παραπάνω δυσκολίες από το πρώτο επίπεδο.

Στο δεύτερο επίπεδο, η πίστα έχει μειωμένο φωτισμό και για το λόγο αυτό έχουμε προσθέσει φώτα στο δρόμο μας, προκειμένου να φωτίζει τα αντικείμενα τα οποία υπάρχουν πάνω στην πίστα.

Η λογική και στο δεύτερο επίπεδο παραμένει η ίδια. Ο χρήστης πρέπει να συλλέξει πάλι 200 πόντους, ενώ πρέπει να αποφύγει τα αντίπαλα αυτοκίνητα τα οποία υπάρχουν πάνω στην πίστα και να φροντίσει να μείνει εντός των περιθωρίων της πίστας.

5.1.2.11 Ανάπτυξη και μείωση ταχύτητας

Μία ακόμα δυσκολία στο δεύτερο επίπεδο, μετά το μειωμένο φωτισμό, είναι η ύπαρξη κάποιων objects, τα οποία μειώνουν και αυξάνουν την ταχύτητα του αυτοκινήτου μας.



Όταν ο χρήστης πέσει πάνω στα πράσινα φανάρια, η ταχύτητα του αυξάνεται κατά μία φορά, ενώ αν πέσει πάνω στα κόκκινα φώτα, η ταχύτητα του μειώνεται κατά μία φορά.

Εικόνα 42: Έτοιμο γραφικό από το Asset Store του Unity

Αυτό έχει την εξής δυσκολία,

- Η ταχύτητα δεν αυξάνεται στο αυτοκίνητο, καθώς όπως έχουμε αναφέρει, η formula μας κινείται με σταθερή ταχύτητα πάνω στον άξονα y. Η ταχύτητα αυξάνεται κατά τη μετατόπιση του αντικειμένου μας, δεξιά και αριστερά πάνω στον άξονα x, προκειμένου να αποφύγουμε τα αντίπαλα αυτοκίνητα και να συλλέξουμε πόντους.
- Αν η ταχύτητα μας πέσει υπό το μηδέν, τότε αλλάζει το gameplay του παιχνιδιού, μέχρι η ταχύτητα μας να ανέβει πάλι πάνω από το μηδέν. Αυτό σημαίνει πως όταν στρίβουμε το αυτοκίνητο στα αριστερά εκείνο θα κινείται προς τα δεξιά και το ανάποδο.

Ο κώδικας ο οποίος ελέγχει τις παραπάνω λειτουργίες είναι ο εξής,

```
if (col.gameObject.tag == "fast") {  // Όταν έρθουμε σε επαφή με το
                                     // αντικείμενο
    Destroy (col.gameObject);  // αυτό καταστρέφεται
    carSpeed += Time.deltaTime * 600;}        // κι η ταχύτητα αυξάνεται
```

```
if (col.gameObject.tag == "slow") {          // Όταν έρθουμε σε επαφή με το
                                             // αντικείμενο
        Destroy (col.gameObject);  // αυτό καταστρέφεται
        carSpeed -= Time.deltaTime * 600; // και η ταχύτητα μειώνεται
}
```

5.2 Εισαγωγή : Score The Basket

Εικόνα43: www.dreamstime.com

Το Score The Basket, είναι ένα παιχνίδι τύπου Catch Game σε δύο διαστάσεις, στο οποίο ο χρήστης πρέπει να μαζέψει έναν συγκεκριμένο αριθμό πόντων (60 points) προκειμένου να τερματίσει το παιχνίδι.

Ο σχεδιασμός και η ανάπτυξη του παιχνιδιού, στηρίχθηκε εξ ολοκλήρου σε δικά μας γραφικά τα οποία τα δημιουργήσαμε ειδικά για τις ανάγκες του συγκεκριμένου παιχνιδιού.

Το παιχνίδι λειτουργεί με τον εξής τρόπο:

- Ο χρήστης έχει στην διάθεση του μία μπασκέτα την οποία την μετακινεί ανάλογα με τον δείκτη του χεριού του δεξιά και αριστερά. Το παιχνίδι ρίχνει μπάλες και ανάλογα με το τι χρώμα είναι, ο παίκτης λαμβάνει και τα αντίστοιχα bonus ή ποινές ώσπου να φτάσει στο επιθυμητό αποτέλεσμα.
- Αν μέσα στο καλάθι μπει η κανονική μπάλα ο παίκτης παίρνει έναν πόντο. Αν μπει η χρωματιστή ο παίκτης παίρνει δύο πόντους. Για κάθε πράσινη μπάλα, διπλασιάζεται το μέχρι τότε σκορ του. Ενώ για κάθε μπάλα-μπόμπα αφαιρούνται 5 βαθμοί.

5.2.1 Υλοποίηση του παιχνιδιού

5.2.1.1 Αρχική Οθόνη

Όπως είπαμε, σε κάθε παιχνίδι υπάρχει η βασική οθόνη μέσα από την οποία ο χρήστης μπορεί να πλοηγηθεί σε διάφορες επιλογές που του δίνει το παιχνίδι.

Στην δική μας οθόνη, ο χρήστης μπορεί δει κάποιες πληροφορίες για το παιχνίδι, να το ξεκινήσει, αλλά και να βγει από την εφαρμογή εάν το επιθυμεί.

Εικόνα 44: Δικά μας γραφικά

Πιο συγκεκριμένα:

· με το 

ο χρήστης μπορεί να ξεκινήσει να παίζει το παιχνίδι.

· Με 

το ο χρήστης μπορεί να δει τον σκοπό που δημιουργήθηκε το παιχνίδι αλλά και κάποιες οδηγίες γι' αυτό.

· Ενώ με το ο χρήστης μπορεί να τερματίσει την εφαρμογή.

5.2.1.2 Οθόνη Πληροφοριών

Εικόνα 45: Δικά μας γραφικά

Μέσα από την οθόνη πληροφοριών, ο χρήστης μπορεί να δει κάποιες πληροφορίες σχετικά με τον σκοπό του παιχνιδιού αλλά και να διαβάσει και τον λόγο που υλοποιήθηκε το συγκεκριμένο παιχνίδι.

Στη οθόνη αυτή, έχουμε κρατήσει τα βασικά buttons που είχαμε και στην αρχική μας οθόνη, πλην αυτό των πληροφοριών αφού πλέον βρισκόμαστε στην οθόνη των Πληροφοριών.

5.2.1.3 Κώδικας πλοήγησης και event στις παραπάνω οθόνες

Ξεκινάμε το παιχνίδι μας με την προϋπόθεση ότι όλα τα κουμπιά προϋπάρχουν στην οθόνη μας. Έτσι, με αυτόν τον τρόπο και αφού δημιουργήσουμε το κάθε button σαν ένα GameObject γράφουμε και τον παρακάτω κώδικα, ανάλογα κάθε φορά με το τι κουμπιά θέλουμε να κρατήσουμε.

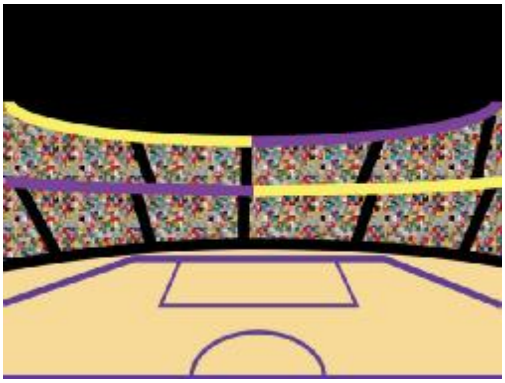
```
using UnityEngine;
using UnityEngine.UI;                    // Καλούμε τις βιβλιοθήκες τις οποίες θα χρησιμοποιήσουμε
using System.Collections;

public class GameController : MonoBehaviour {  //Η MonoBehaviour είναι η μόνη κλάση που χρησιμοποιούμε

public GameObject restartButton;
public GameObject splashScreen;    // Ορίζουμε public μεταβλητές που θα χρειαστούμε
public GameObject startButton;
```

```
public GameObject exitButton;
public GameObject infoScreen;
public GameObject infoButton;
public GameObject congratsText;
public BasketaController basketaController;

public void StartGame() {      // Όταν πατήσουμε να ξεκινήσει το παιχνίδι
                splashScreen.SetActive (false);      // η οθόνη με την μπάλα-animation θα
εξαφανιστεί
                startButton.SetActive (false);        // το button start θα φύγει
                exitButton.SetActive (false);         //το button exit επίσης
                infoScreen.SetActive (false);         // το γραφικό με τις πληροφορίες θα
        φύγει
                infoButton.SetActive (false);         //και το κουμπί info θα εξαφανιστεί.
                basketaController.ToggleControl (true);  // θα μπορούμε να χειριστούμε την
        μπασκέτα
                StartCoroutine (Spawn ());            //και το παιχνίδι θα αρχίσει να πετάει
μπάλες

        }

        public void Exit(){                  // Όταν πατήσουμε το exit
                Application.Quit();       // η εφαρμογή τερματίζεται
        }

        public void InfoGame(){                        // Όταν πατήσουμε το info
                splashScreen.SetActive (false);       // η μπάλα-animation θα εξαφανιστεί
                infoScreen.SetActive (true);          // η οθόνη πληροφοριών θα εμφανιστεί
                infoButton.SetActive (false);         //και το κουμπί info θα φύγει από την
οθόνη μας
        }
```

5.2.1.4 Η οθόνη του παιχνιδιού

Όπως αναφέραμε ήδη, το παιχνίδι μας, είναι ένα παιχνίδι τύπου Catch Game, με σκοπό την συλλογή κάποιων πόντων.

Για να υλοποιηθεί αυτό, θα χρειαστούμε το γήπεδο, την μπασκέτα καθώς και διάφορες μπάλες.



Εικόνα 46: Δικά μας γραφικά

Εικόνα 47: Δικά μας γ

Το παιχνίδι μας αποτελείτε από ένα φόντο που δείχνει ένα γήπεδο μπάσκετ το οποίο το δημιουργήσαμε μόνοι μας, μία μπασκέτα που κινείται δεξιά-αριστερά ανάλογα με το που θα την κουνήσει ο χρήστης με το δάχτυλο του αλλά και από μπάλες τις οποίες έχουμε ορίσει να πέφτουν τυχαία με έναν κώδικα προκειμένου να μην γνωρίζουμε την σειρά που θα πέφτουν.

5.2.1.5 Ο κώδικας για να πέφτουν οι μπάλες

Όπως είπαμε, έχουμε δημιουργήσει έναν κώδικα για να ρίχνει κάθε φορά τυχαία, μία από τις τέσσερις μπάλες που έχει στην διάθεση του προκειμένου να γίνεται αυτή η ενέργεια μέχρι να τελειώσει το παιχνίδι.

Ο κώδικα που χρησιμοποιήσαμε λοιπόν είναι ο εξής:

```
public GameObject[] balls;  // Ορίζουμε έναν πίνακα που θα έχει μέσα τις μπάλες
public float timeLeft;      // μεταβλητή που θα δείχνει τον χρόνο που απομένει
private float maxWidth;  // το μέγεθος που έχει η οθόνη μας
private bool playing;    // μεταβλητή bool που θα είναι αληθής όταν ο χρήστης επιλέξει να παίξει και                          ψευδής αν ο χρήστης δεν παίζει και βρίσκεται σε άλλη οθόνη

IEnumerator Spawn(){  // χρησιμοποιούμε την IEnumerator που είναι για να ξεκινήσουμε μία λούπα

        yield return new WaitForSeconds (2.0f);  // το σύστημα περιμένει 2 δεύτερα
```

```
        playing = true;                         //  ο χρήστης έχει επιλέξει να παίξει και έτσι ξεκινά η διαδικασία
        while (timeLeft > 0) {                  // όσο ο χρόνος που απομένει δεν έχει τελειώσει

                GameObject ball = balls [Random.Range (0, balls.Length)];  //επιλέγει από τον  πίνακα με τις μπάλες που θα έχει μέγεθος από 0 μέχρι όσο επιθυμούμε εμείς
                Vector3 spawnPosition = new Vector3 (       // θα δημιουργείται κάθε φορά ένα vector3 που θα είναι η κάθε μπάλα και θα πέφτει από την spawnPosition που έχουμε ορίσει εμείς (κάπου ακριβώς έξω από την βασική μας εικόνα)

                Random.Range (-maxWidth, maxWidth),     // στον άξονα x θα παίρνει τυχαίες τιμές ανάλογα με το μέγεθος που θα έχει η οθόνη μας
                transform.position.y,           // στον άξονα y υπάρχει η τιμή που έχουμε δώσει για να αρχίζουν να πέφτουν οι μπάλες
                0.0f); // δεν υπάρχει z άξονας άρα βάζουμε 0
                Quaternion spawnRotation = Quaternion.identity;      // όταν θέτουμε το identity σημαίνει ότι δεν βάζουμε καθόλου  rotation
                Instantiate (ball, spawnPosition, spawnRotation);      //ορίζουμε την υπόσταση που θα                    έχει η μπάλα, τυχαία μπάλα, σε τυχαία θέση
                yield return new WaitForSeconds (Random.Range (1.0f, 2.0f));     // περιμένει μεταξύ ενός και δύο δευτερολέπτων για να εκτελέσει την λούπα ξανά από την αρχή                                        }
                }
```

5.2.1.6 Ο κώδικας για την μπασκέτα

```
public class BasketaController : MonoBehaviour {  //η MonoBehaviour είναι η βασική συνάρτηση που χρησιμοποιούμε

public Camera cam;  // ορίζουμε την μεταβλητή cam για την κάμερα μας
private float maxWidth;  //ορίζουμε την μεταβλητή maxWidth για το μέγεθος της οθόνης μας


        void Start () {
                if (cam == null) {      // αν δεν βάλουμε καθόλου κάποια συγκεκριμένη κάμερα
                        cam = Camera.main;   // τότε θα ορίσει την κεντρική μας κάμερα
                }
                Vector3 upperCorner = new Vector3 (Screen.width, Screen.height, 0.0f);  // ορίζουμε την γωνία του αντικειμένου μας σύμφωνα με τις διαστάσεις της οθόνης μας
                Vector3 targetWidth = cam.ScreenToWorldPoint (upperCorner);   //  η θέση του αντικειμένου θα εξαρτηθεί από την upperCorner
```

```
        float basketaWidth = GetComponent<Renderer> ().bounds.extents.x;  // η μετακίνηση της μπασκέτας μας εξαρτάται από το μέγεθος της που έχει να κάνει με τα όρια της εικόνας μας
        maxWidth = targetWidth.x - basketaWidth;   // το μέγεθος της εικόνας εξαρτάται από την θέση του αντικειμένου και το μέγεθος της μπασκέτας μας (έτσι είναι όλη η μπασκέτα στο σκηνικό μας και όχι μόνο η μισή στις άκρες)
    }

    void FixedUpdate () {
        Vector3 rawPosition = cam.ScreenToWorldPoint (Input.mousePosition);   // η θέση του αντικειμένου θα εξαρτηθεί από το που θα πατήσουμε ή από τον υπολογιστή ή από το κινητό μας
        Vector3 targetPosition = new Vector3 (rawPosition.x, -1.88f, 0.0f);      // θα δημιουργηθεί ένα καινούργιο αντικείμενο στο σημείο που πατήσαμε εμείς να πάει το αντικείμενο μας
        float targetWidth = Mathf.Clamp (targetPosition.x, -maxWidth, maxWidth);
        // το αντικείμενο μας θα κινείται μεταξύ των ορίων της οθόνης μας
        targetPosition = new Vector3 (targetWidth, targetPosition.y,
        targetPosition.z);  //η μπασκέτα μας θα έχει όρια αυτά της οθόνης μας και δεν θα βγαίνει εκτός αυτής
        GetComponent<Rigidbody2D> ().MovePosition (targetPosition);   // δημιουργείται η κίνηση του αντικειμένου μας από το προηγούμενο σημείο στο σημείο που αγγίξαμε
            }
```

5.2.1.7 Ο κώδικας για την βαθμολογία του παιχνιδιού

Όπως σε κάθε σχεδόν παιχνίδι έτσι και στο δικό μας σκοπός είναι να μαζέψεις ένα σύνολο πόντων. Για να γίνει αυτό χρησιμοποιούμε τον παρακάτω κώδικα.

```
using UnityEngine;
using UnityEngine.UI;  // Δηλώνουμε τις βιβλιοθήκες που θα χρειαστούμε
using System.Collections;

public class Score : MonoBehaviour {

    public Text scoreText;  // ορίζουμε μεταβλητή με το κείμενο που θα εμφανίζει το σκορ
    public int ballValue;  // μεταβλητή με την αξία της μπάλας
    public int score;  //μεταβλητή που δηλώνει το σκορ

    void Start () {
        score = 0;  //αρχικά το σκορ είναι 0
        UpdateScore ();  // και το εμφανίζουμε στην οθόνη μας
    }
```

```
void OnTriggerEnter2D(Collider2D other) {  //στην συνάρτηση αυτή το αντικείμενο πρέπει να μπει μέσα στο collision της μπασκέτας που έχουμε ορίσει
        if (other.gameObject.tag == "Ball") {  // αν η μπάλα που θα μπει είναι η //κανονική
            score += ballValue;  //τότε στο σκορ θα προστεθεί ένας βαθμός
            UpdateScore ();   //και θα ενημερώσει το σκορ  - εμφανίσει στην // οθόνη
        } if (other.gameObject.tag == "Colour Ball") {   // αν η μπάλα που θα μπει είναι η χρωματιστή
            score += ballValue * 2;   //τότε στο σκορ θα προστεθούν 2 βαθμοί
            UpdateScore ();  //και θα ενημερώσει το σκορ  - εμφανίσει στην // οθόνη
        } if (other.gameObject.tag == "Green_Ball") {  // αν η μπάλα που θα μπει είναι η πράσινη – bonus x2 μπάλα
            score = score * 2;   //τότε το σκορ θα διπλασιαστεί
            UpdateScore ();  //και θα ενημερώσει το σκορ  - εμφανίσει στην οθόνη
        }
    }

    void OnCollisionEnter2D (Collision2D collision){  //αν το αντικείμενο έρθει σε επαφή με το collision της μπασκέτας που έχουμε ορίσει
        if (collision.gameObject.tag == "Ball Bomb") {  //και αυτό το αντικείμενο είναι η μπόμπα-μπάλα
            score -= ballValue * 5;  //τότε από το σκορ θα αφαιρεθούν 5 πόντοι
            UpdateScore ();  // και θα ενημερωθεί το σκορ – εμφανιστεί στην οθόνη
        }
    }


    public void UpdateScore () {  //η συνάρτηση για την ενημέρωση του σκορ και εμφάνιση στην οθόνη μας
        scoreText.text = "Score:\n" + score; //η μεταβλητή scoreText θα είναι το κείμενο μας + το αποτέλεσμα του σκορ που έχουμε
    } }
```

5.2.1.8 Το εφέ της έκρηξης

Όταν η μπάλα-μπόμπα χτυπήσει σε οποιοδήποτε σημείο της μπασκέτας μας γίνεται μία έκρηξη και έτσι αφαιρούνται πέντε πόντοι από το σύνολο μας. Το εφέ της έκρηξης το πήραμε έτοιμο από κάποια effect που υπήρχαν στο asset store του Unity αλλά εμείς το τοποθετήσαμε στο σημείο που θέλαμε να εμφανίζεται με τον παρακάτω κώδικα.

Εικόνα 48: Δικά μας γραφικά

```
using Unity Engine // δηλώνουμε τις βιβλιοθήκες που θα χρειαστούμε
using System.Collections;

public class Explode: MonoBehaviour{

        public GameObject explosion; //ορίζουμε τη μεταβλητή explosion

void OnCollisionEnter2D (Collision2D collision){ // όταν το object έρθει σε επαφή με ένα
                                                 // collision
if(collision.gameObject.tag == "Basketa"){ // και το Collision είναι της μπασκέτας μας
Instantiate (explosion, transform.position, transform.rotation); // γίνεται το εφέ της
έκρηξης που έχουμε ορίσει στο σημείο ακριβώς που γίνεται η επαφή
        Destroy(gameObject); // και καταστρέφεται το αντικείμενο, δηλαδή η μπόμπα
} } }
```

5.2.1.9 Μηνύματα προς τον χρήστη

Εικόνα 49: Δικά μας γραφικά

Όταν ο χρήστης συλλέξει τους 60 πόντους του εμφανίζεται ένα μήνυμα που του λέει Congratulations, τον αφήνει όμως να τελειώσει τον χρόνο του παιχνιδιού και να μάθει τελικά πόσους πόντους μπορεί να μαζέψει πέρα από αυτούς που του ζητάει.

Στο σημείο που έχουμε βάλει να εμφανίζει τους συνολικούς βαθμούς μας προσθέσαμε τον παρακάτω κώδικα:

```
public void UpdateScore () {  // στην συνάρτηση που εμφανίζει το σκορ
                scoreText.text = "Score:\n" + score;  //εμφανίζει το σκορ που εχουμε
                if (score >= 60) {    //αν το σκορ είναι μεγαλύτερο από 60 βαθμούς
```

```
                congrats.gameObject.SetActive(true);  //το congratulations!
εμφανίζεται
        }
        if (score >= 80) {  // όταν το σκορ πάει πάνω από 80 πόντους
                congrats.gameObject.SetActive(false);  //το μήνυμα
εξαφανίζεται
        }
    }
```

Ενώ όταν τα 60 δευτερόλεπτα τελειώσουν εμφανίζει στο χρήστη ότι ο χρόνος του τελείωσε, εμφανίζοντας του το κουμπί restart και την δυνατότητα να βγει από την εφαρμογή.

Εικόνα 50: Δικά μας γραφικά

Στη συνέχεια από τον κώδικα που ρίχνει τυχαία τις μπάλες

προσθέτουμε τον παρακάτω κώδικα.

```
yield return new WaitForSeconds (2.0f);  //η εφαρμογή περιμένει 2 δεύτερα
gameOverText.SetActive (true);  //το μήνυμα ότι ο χρόνος τελείωσε εμφανίζεται
yield return new WaitForSeconds (2.0f);  //η εφαρμογή περιμένει 2 δεύτερα
restartButton.SetActive (true);  //το restart κουμπί εμφανίζεται
exitButton.SetActive (true);   //ταυτόχρονα εμφανίζεται και το exit button
```

Βιβλιογραφία

Ελληνική Βιβλιογραφία

- Ν.Θ.Λιόλιος, Β.Γ.Νεβραντζάς, (2013). *Προγραμματισμός σε C Sharp*. ΤΕΙ Λάρισας: Τμήμα Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών.
- Lauren Darcey, Shane Conder, (2011). *Ανάπτυξη Εφαρμογών με το Android*. Δεύτερη Έκδοση. Απόδοση: Αγαμέμνων Μήλιος. Αθήνα: Μ. Γκιούρδας.
- Ιωάννης Παναγιωτόπουλος, Χρήστος Παναγιωτόπουλος. (2009). *Διαδικτυακός Προγραμματισμός C#.* Αθήνα: Σταμούλης Α.Ε.

Ξενόγλωσση Βιβλιογραφία

- Ed Burnette. (2010). *Hello, Android.* Third Edition. Raleigh, North Carolina, Dallas Texas: The Pragmatic Bookshelf.
- Mark L. Murphy. (2008-2011). *The Busy Coder's Guide to Android Development.* United States of America: CommonsWare.
- Patrick Felicia. (2013). *Getting Started with Unity.* Publishing: Packt Publishing.
- Joseph Hocking. (2015). *Unity IN ACTION.* Shelter Island, New York: Manning Publications Co.
- Wallace B, McClure, Nathan Blevins, John J. Croft IV, Jonathan Dick, Chris Harty. (2012). *Android Programming*. USA: John Wiley & Sons Inc.

Ηλεκτρονική Βιβλιογραφία

- Android Developers. *SDK Archives*, από http://developer.android.com/
- Android Official Website. (2008). *Android*, από https://www.android.com/history
- Unity Official Website. (2005). *Unity*, από http://unity3d.com
- Student guru. *Εισαγωγή στη C#,* από http://studentguru.gr/w/tutorials/01-c