Πτυχιακή εργασία τμήματος εφαρμοσμένης πληροφορικής & πολυμέσων

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Ανάπτυξη Εκπαιδευτικού Παιχνιδιού για συσκευές android

Τζαγάκη Ελένη (ΑΜ:2698)

Επιβλέπων καθηγητής: Παπαδάκης Νικόλαος

Ημερομηνία παρουσίασης: 30/3/2016

Abstract

In our days as usage of touch devices is more and more common we can grab the chance and create fun but educational software for students. This thesis ”Educational game for android devices” is a representation of a game development in unity engine exclusively for android devices. Android is selected as it has become the most used free source operation system. Also the fact that each application can be uploaded and downloaded freely from the play store makes it a good educational mean.

The purpose is to create an educational game about the periodic table in which the user at first can focus on learning the elements one by one and later to check what he’s learned through a quiz game. In this thesis it is explained how this game was developed in unity 3d, creating the character using mixamo fuse and how it works.

Περίληψη

Σε αυτή τη πτυχιακή εργασία θα αναλυθεί η ανάπτυξη και υλοποίηση παιχνιδιου σε game engine και πιο συγκεκριμένα στο Unity. Το παιχνίδι αυτό είναι εκπαιδευτικού χαρακτήρα και απευθυνεται σε μαθητές για την εκμάθηση του περιοδικού πίνακα μέσω συσκευών αφής με λειτουργικό android. Για το σκοπό της πτυχιακής αυτής χρησιμοποιήθηκε επίσης το πρόγραμμα mixamo fuse για την δημιουργία του κεντρικού χαρακτήρα που συνοδεύει το gameplay.

Contents

Πίνακας Εικόνων

1.Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτύξουμε το σκοπό και τη χρησιμότητα της πτυχιακής αυτής ρίχνοντας μια πρώτη ματιά στο unity game engine και στο λειτουργικό android.

1.1 Σκοπός

Σκοπός αυτής της πτυχιακής εργασίας είναι η εξοικίωση με την ανάπτυξη παιχνιδιών κάνοντας χρήση του unity και συνεπώς την ανάπτυξη γραφικών στοιχείων σε αυτό και τη διασύνδεση τους προγραμματίζοντας τα με τη γλώσσα προγραμματισμού c# όπως επίσης και τη χρήση xml αρχείων για την ανάκτηση πληροφοριών. Το παιχνίδι αυτό σχεδιάστηκε για να λειτουργεί σε συσκευές android. Η επιλογή για τη χρήση android συσκευών δεν είναι τυχαία καθώς η χρήση του λειτουργικού αυτού είναι συνεχώς αυξανόμενη με τους χρήστες του λειτουργικού αυτού να φτάνουν το 1 δισεκατομμύριο ενεργούς χρήστες μηνιαία οπου και κατέχει το μεγαλύτερο μερίδιο στην αγορά. Επίσης το κυριότερο πλεονέκτημα του είναι το ότι η ελεύθερη διακίνηση εφαρμογών από προγραμματιστές προς χρήστες και η μεγάλη χρήση από μαθητές και ανήλικους τα οποία το κάνουν ένα μέσο μάθησης.

1.2 Στόχος - χρησιμότητα

Έχοντας ως στόχο την δημιουργία ενός εκπαιδευτικού παιχνιδιού που απευθύνεται κυρίως σε μαθητές για την εκμάθηση του περιοδικού πίνακα, ο χρήστης έχει τη δυνατότητα αρχικά να επεξεργαστεί ένα προς ένα τα στοιχεία του περιοδικού πίνακα παίρνοντας πληροφορίες για αυτά και μετεπειτα μπορούν να παίξουν ένα παιχνίδι τύπου quiz πάνω σε αυτά που έμαθαν προηγουμένως.

1.3 Τι είναι το android

Το android είναι ένα λειτουργικό σύστημα ανεπτυγμένο αρχικά από τη google και βασισμένο σε linux kernel. Το λειτουργικό αυτό προορίζεται για συσκευές αφής όπως smartphones και tablets. Τα τελευταία όμως χρόνια έχουν αναπτυχθεί εκδόσεις για smartwatches, android tv και android auto. Η διεπαφή του χρήστη επιτυγχάνεται με πολύ απλό τρόπο απλά πατώντας πάνω στην οθόνη η οποία "διαβάζει" το σημείο που πάτησε ο χρήστης και με ποιόν τρόπο. Απο τον Ιούλιο του 2015 έχουν δημοσιευτεί πάνω απο 1 εκατομμύριο εφαρμογές android στο Google play store και πάνω απο 50 δισεκατομμύρια downloads απο τους χρήστες. Απο έρευνες που έχουν διεξαχθεί έχει βρεθεί πως το 71% των developers έχουν αναπτύξει 1 ή περισσότερες εφαρμογές για android και το 40% των επαγγελματιών του είδους έχουν ώς προτεραιότητα τη πλατφόρμα android. Ο κωδικας του android παρέχεται απο την ίδια την google με άδειες χρήσης ανοιχτού κώδικα.

1.4 Τι είναι τα game engines

Είναι προγράμματα τα οποία σχεδιάστηκαν για να παρέχουν εύκολη δημιουργία και ανάπτυξη 2D και 3D παιχνιδιών που προορίζονται για διάφορες συσκευές και κονσόλες όπως windows, playstation, xbox, android, iOS κα. Κατά βάση περιέχουν έναν renderer για τη δημιουργία των γραφικών, ήχο, animation, scripting, μια μηχανή φυσικής και collision detection. Κάποια game engines επίσης μπορεί να διαθέτουν και threading, streaming, networking, localization κα.

1.5 Δομή της εργασίας

Σε αυτό το κεφάλαιο θα αναπτυχθουν ο σκοπός και η χρησιμότητα της πτυχιακής αυτής καθως και θα αναλυθουν έννοιες οι οποίες είναι σημαντικό να επεξηγηθούν προτού μπούμε στο Κεφάλαιο 2 όπου αναλύονται παρόμοιες τεχνολογίες με αυτές του unity όπως η unreal engine. Έπειτα στο 3^{ο} κεφάλαιο θα περάσουμε στην υλοποιηση του παιχνιδιου όπου θα παρουσιαστούν screenshots και ο κώδικας σε C# απο την ανάπτυξη του στο unity, η δημιουργία του χαρακτήρα στο mixamo fuse και η σύνταξη του xml για την λήψη πληροφορίας. Στο κεφάλαιο 5 θα παρουσιαστούν τα συμπεράσματα που βγήκαν και η εμπειρία που αποκτήθηκε μέσα απο τη συγγραφή αυτής της πτυχιακής.

1.6 Η εκπαίδευση στη τεχνολογία

Υπάρχουν αρκετές εφαρμογές οι οποίες στηριζόμενες στην διασκεδαστική πλευρά των μέσων όπως smartphones και tablets αναπτύχθηκαν με στόχο την μάθηση ή την βοήθεια σε συγκεκριμένους επιστημονικούς τομείς. Υπάρχει πληθώρα εφαρμογών, παρόμοιες με αυτή που θα παρουσιαστεί σε αυτή τη πτυχιακή οι οποίες σχετίζονται με χημεία, βιολογία, μαθηματικά κα.

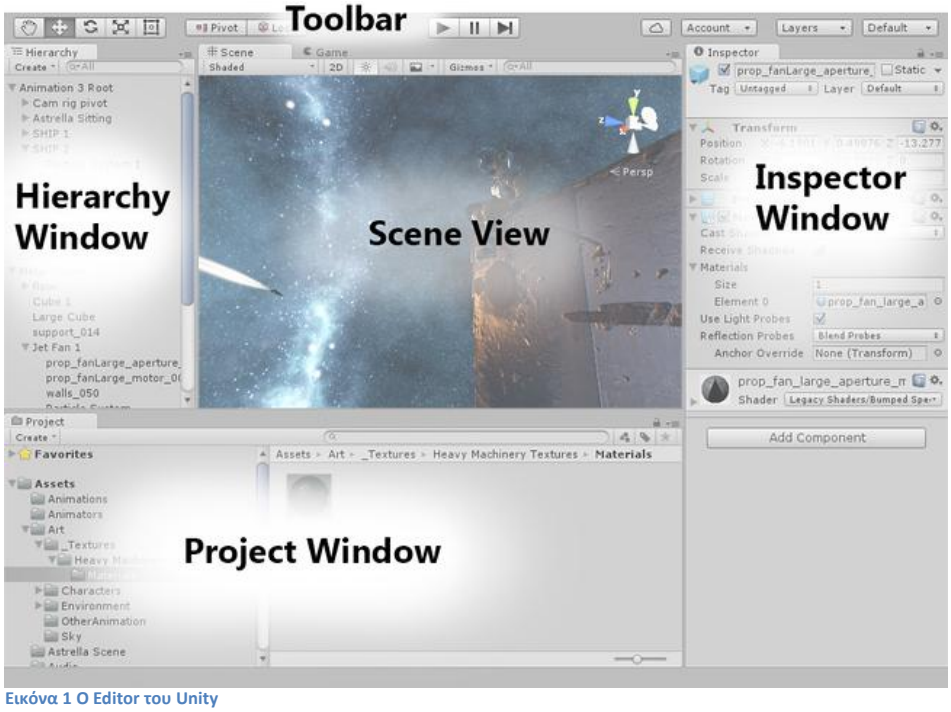
2.Παρόμοιες Τεχνολογίες

Σε αυτό το κεφάλαιο θα δούμε εισαγωγικά το Unity και άλλα game engines όπως την Unreal engine.

2.1 Το Unity

Το Unity είναι ένα game engine το οποίο αναπτύχθηκε από την Unity Technologies, ανακοινώθηκε η κυκλοφορία του το 2005 όπου και απευθυνόταν μόνο για OSX. Σήμερα καλύπτει περισσότερες από 15 πλατφόρμες και αποτελεί το προεπιλεγμένο Software Development Kit(SDK) για το Wii U. Η τρέχων έκδοση του Unity όπου και χρησιμοποιήθηκε για αυτήν τη πτυχιακή είναιι η 5.

2.1.1 Ο Editor



Εικόνα 1 Ο Editor του Unity

Κατά βάση ο editor του Unity αποτελείτε από 5 βασικά μέρη όπως φαίνεται και στη παραπάνω εικόνα 1. Πιο αναλυτικά αυτά είναι:

Project Window

Η δουλειά του είναι να εμφανίζει τα στοιχεία που περιλαμβάνονται στο project όπως γραφικά, animations, scripts, shaders κα.

Scene View Window

Μέσω αυτού δύνεται η δυνατότητα στον χρήστη να δει μια επισκόπιση της σκηνής που δημιουργεί είτε είναι 2D ή 3D αλλα και να την επεξεργαστεί.

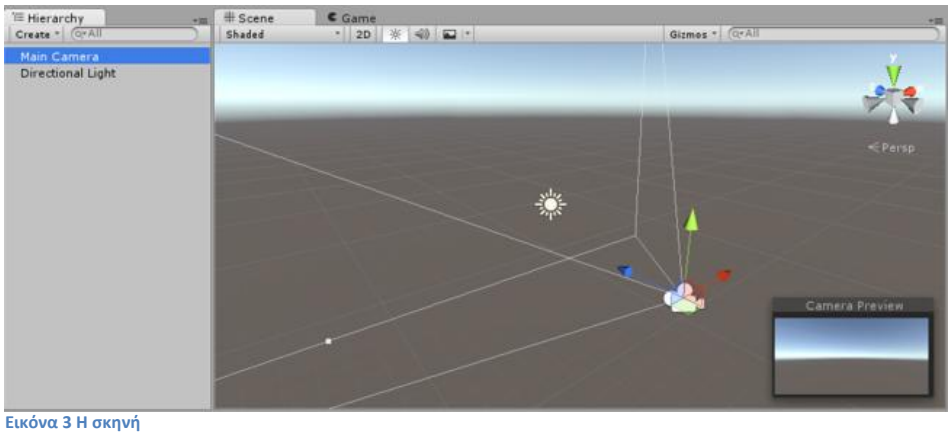
Toolbar

Παρέχει την πρόσβαση προς όλα τα απαραίτητα εργαλεία και χαρακτηρηστικά. Αριστερά υπάρχουν τα κουμπιά τα οποία ελέγχουν το πώς βλέπουμε τη σκηνή όπως και τα αντικείμενα μέσα σε αυτή. Στη μέση υπάρχουν τα κουμπιά «play», «pause» και «step». Στα αριστερά είναι τα κουμπιά πρόσβασης στο unity cloud services, στον προσωπικο λογαριασμό κα.

Εικόνα 2 Το Toolbar

2.1.2 Η σκηνή

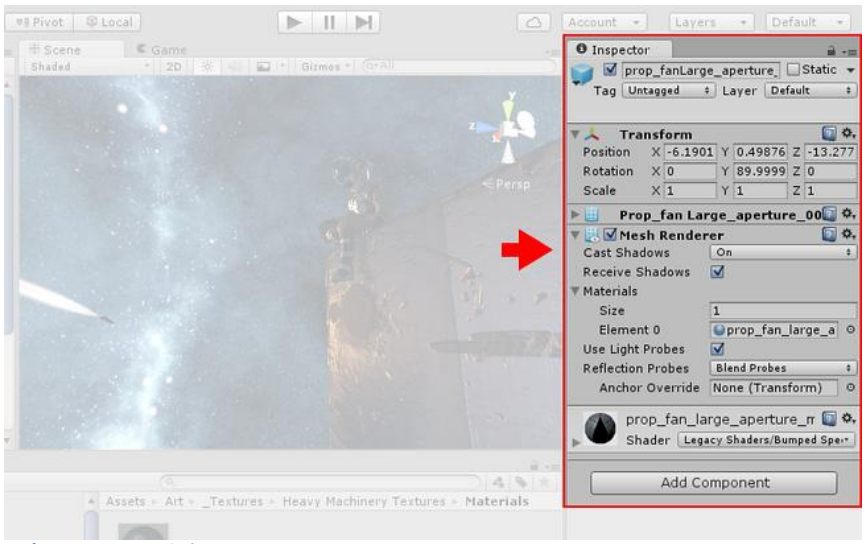
Ένα απο το κυριότερο μέρος της δημιουργίας ενός παιχνιδιού είναι η σκηνή. Εδώ τοποθετούνται ή κατά περιπτώσεις δημιουργούνται τα αντικείμενα, ο φωτεισμός , οι κάμερες κα. Χρησιμοποιείται για τη δημιουργία των μενού, τον επιπέδων και γενικότερα ότι χρειάζεται οπτικοποίηση. Στην εικόνα 3 βλέπουμε μια σκηνή με μία κάμερα και ένα φώς. Στο κάτω δεξιά μέρος έχουμε τη δυνατότητα να δούμε μία προεπισκόπηση του πως φαίνεται η σκηνή μέσα από τη κάμερα.



Εικόνα 3 Η σκηνή

2.1.3 Inspector Window

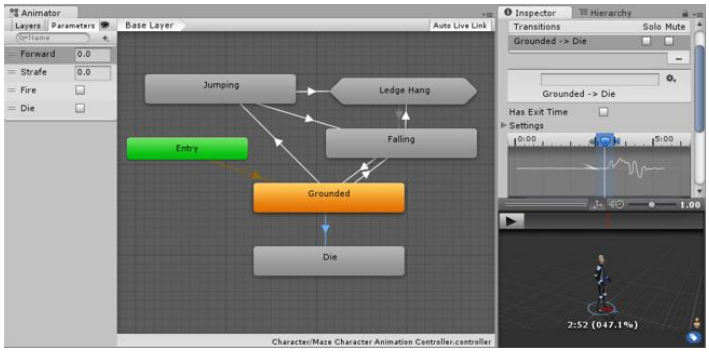
Επιτρέπει την εμφάνιση και την επεξεργασία των ιδιοτήτων του επιλεγμένου αντικειμένου. Το παράθυρο αυτό διαφέρει ανάλογα τον τύπο του αντικειμένου. Μπορούμε να ρυθμίσουμε το μέγεθος ενός αντικειμένου, τις συντεταγμένες του στη σκηνή, τις ιδιότητες των υφών και υλικών που αποτελούν το επιλεγμένο αντικείμενο. Επίσης από εδώ γίνεται η σύνδεση τους με τα scripts και τους animation controllers. (Εικόνα 4)



Εικόνα 4 Inspector Window

2.1.4 Mecanim

Το mecanim είναι το σύστημα animation του unity. Με αυτό μπορούμε να ρυθμίσουμε τις κινήσεις ενός αντικειμένου ώστε να φτιάξουμε κλιπς κίνησης , δηλαδή τη θέση του, τη περιστροφή του, το χρονικό σημείο αυτών των αλλαγών κα. Ένα σημαντικό κομμάτι του mecanim το οποίο και χρησιμοποιήθηκε για αυτή τη πτυχιακή είναι ο animation controller. Χρησιμοποιείται όταν θέλουμε να εισάγουμε έτοιμα κλιπς κίνησης και να τα ενώσουμε ώστε να γίνεται σωστά η εναλλαγή των κινήσεων όπως ακριβώς θέλουμε .



2.1.5 Scripting

Το πιο σημαντικό μέρος της ανάπτυξης ενός παιχνιδιού αποτελεί ίσως το προγραμματιστικό κομμάτι. Μέσω των scripts μπορούμε να επιτύχουμε αλληλεπίδραση του χρήστη και του παιχνιδιού, να ελέγχουμε τη ροη των γεγονότων που συμβαίνουν και γενικά για να ελέγξουμε την συμπεριφορά των αντικειμένων στη σκηνή (π.χ. πότε και πως ξεκινάει ένα animation, ένας ήχος, πως ελέγχουμε το κεντρικό μενού κλπ) .Το unity υποστηρίζει δύο γλώσσες προγραμματισμού:

- C# - Αντικειμενοστραφής γλώσσα προγραμματισμού
- UnityScript – Ειδικά φτιαγμένη για το Unity με προτυπο τη javascript.

Για τη πτυχιακή αυτή χρησιμοποιήθηκε η C#.

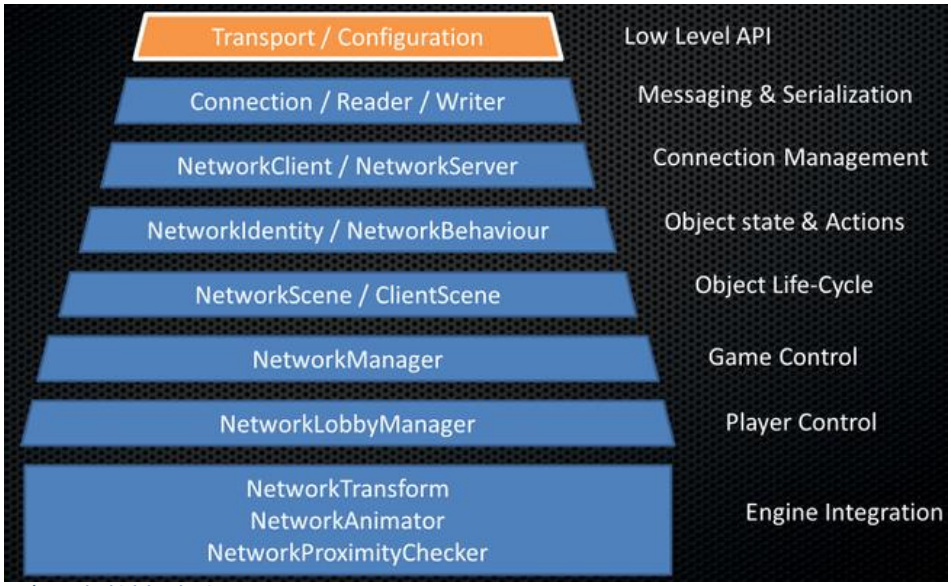
Εικόνα 5 Σύνδεση script μέσω του Inspector

2.1.6 Physics

Όλα τα αντικείμενα σε κάθε σκηνή θα πρέπει να συμβαδίζουν με τους νόμους της φυσικής (κατά βάση) και να επηρεάζονται απο τη βαρύτητα, συγκρούσεις και οποιαδήποτε άλλη δύναμη μπορεί να ασκηθεί πάνω τους.

Το unity διαθέτει μια μηχανή φυσικής μέσω της οποίας μπορούμε να προσδώσουμε τα χαρακτηριστικά σε κάθε αντικείμενο ώστε να συμορφώνεται στους νόμους της φυσικής που θέλουμε. Υπάρχουν δύο ξεχωριστές μηχανές φυσικής στο unity μία για 3D και μία για 2D. Οι βασικές αρχές που τις περιβάλουν είναι ίδιες απλά αλλάζει ο τρόπος που υλοποιούνται. Για παράδειγμα υπάρχει ένα rigidbody για 3D και ένα για 2D.

2.1.7 The High Level API



Εικόνα 6 The high level api

Το high level api είναι ένα σύστημα μέσω του οποίου μπορούν να οικοδομηθούν στοιχεία για multiplayer παιχνίδια. Είναι χτισμένο πάνω απο το χαμηλότερο επίπεδο του στρώματος μεταφοράς και χειρίζεται πολλές απο τις συνήθεις εργασίες που απαιτούνται για multiplayer παιχνίδια. Παρέχει τη δυνατότητα ένας συμμετέχοντας να είναι και πελάτης και διακομιστής ταυτόχρονα και έτσι δεν απαιτείται κάποια ειδική διεργασία διακομιστή. Αυτό σημαίνει πως με κάποια ελάχιστη δουλειά απο μέρος του προγραμματιστή και σε συνεργασία με τις υπηρεσίες

διαδικτύου επιτρέπεται το multiplayer mode.

2.1.8 Γνωστά παιχνίδια

Κάποια γνωστά παιχνίδια τα οποία έχουν αναπτυχθεί στο Unity για android είναι:

- Angry Birds 2
- The Room Two
- Tesla Effect: A Tex Murphy Adventure
- Hearthstone: Heroes of Warcraft
- Scrolls

Εικόνα 7 Το hearthstone το οποίο αναπτύχθηκε σε unity

2.2 To Unreal Engine

Στην ίδια λογική του unity υπάρχει και το unreal engine το οποίο αναπτύχθηκε από την epic games και το 1998 κυκλοφόρησε το first person shooter παιχνίδι Unreal. Είναι γραμμένο σε C++ και διαθέτει υψηλό βαθμό φορητότητας. Η σημερινή του έκδοση είναι η 4 και είναι σχεδιασμένη για Microsoft DirectX11 και 12 για Windows και Xbox, OpenGL για OS X, Linux, Playstation, iOS, Android και JavaScript/WebGL για HTML5 για προγράμματα περιήγησης. Η Unreal engine αποτελείται από διάφορους editors κάποιοι από αυτούς περιγράφονται παρακάτω:

2.2.1 *Level Editor*

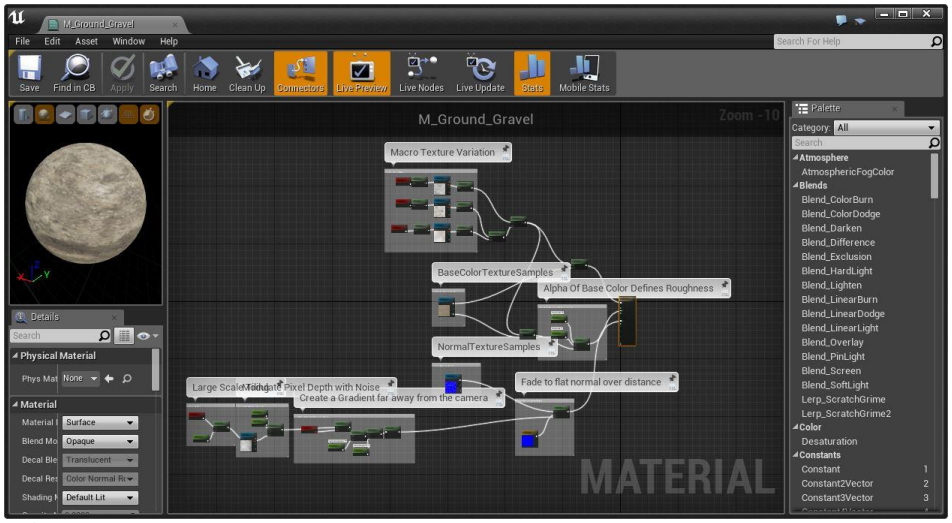
Εικόνα 8 Level Editor

Είναι ο βασικός editor της Unreal, χρησιμοποιείται για την κατασκευή των επιπέδων του παιχνιδιού, σε αυτόν προσθέτουμε αντικείμενα, χαρακτήρες, material και γενικά ότι χρειάζεται το κάθε επίπεδο. Αποτελείται από 7 βασικά μέρη:

- Tab Bar και Menu Bar (Τα βασικά κουμπιά όπως σε κάθε παράθυρο)
- Toolbar (Προσφέρει γρήγορη πρόσβαση σε εργαλεία που χρησιμοποιούνται συχνά)
- Modes
- Content Browser
- Viewports (Περιέχει διαφορετικούς τρόπους να δούμε τους κόσμους που εχουμε δημιουργήσει)
- World Outliner
- Details (Εδώ μπορούμε να επεξεργαστούμε και να δούμε τις ιδιότητες του επιλεγμένου αντικειμένου)

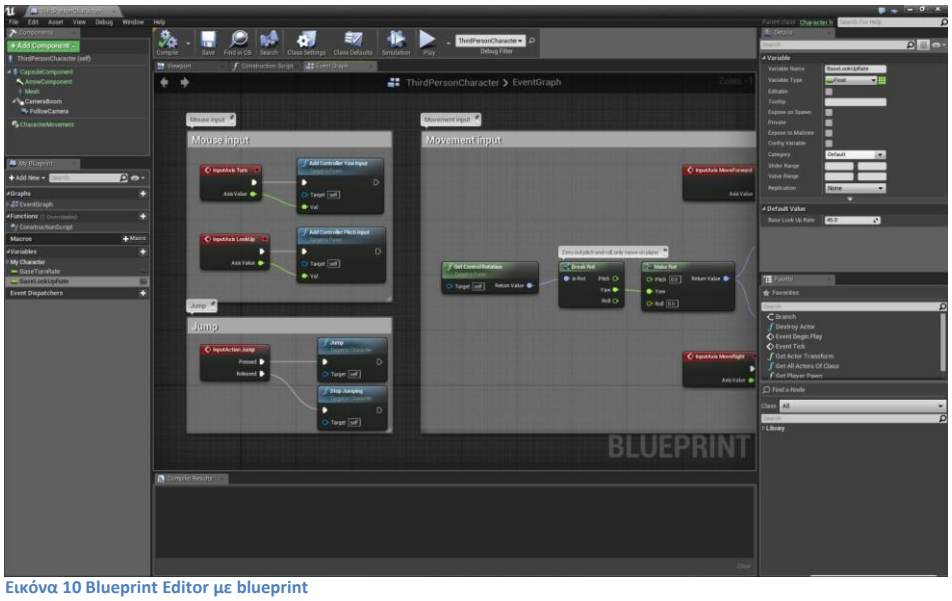
2.2.2 Material Editor

Μέσα σε αυτόν δημιουργούμε ή επεξεργαζόμαστε τα materials τα οποία αποτελούν μέρη που εφαρμόζονται στα meshes για την ρύθμιση του οπτικού τους μέρους. Για παράδειγμα μπορούμε να δημιουργήσουμε ένα material ψηφιδωτής πέτρας και να το βάλουμε στον τοίχο ώστε να φαίνεται σαν πετρόχτιστος.



Εικόνα 9 Material Editor

2.2.3 Blueprint Editor

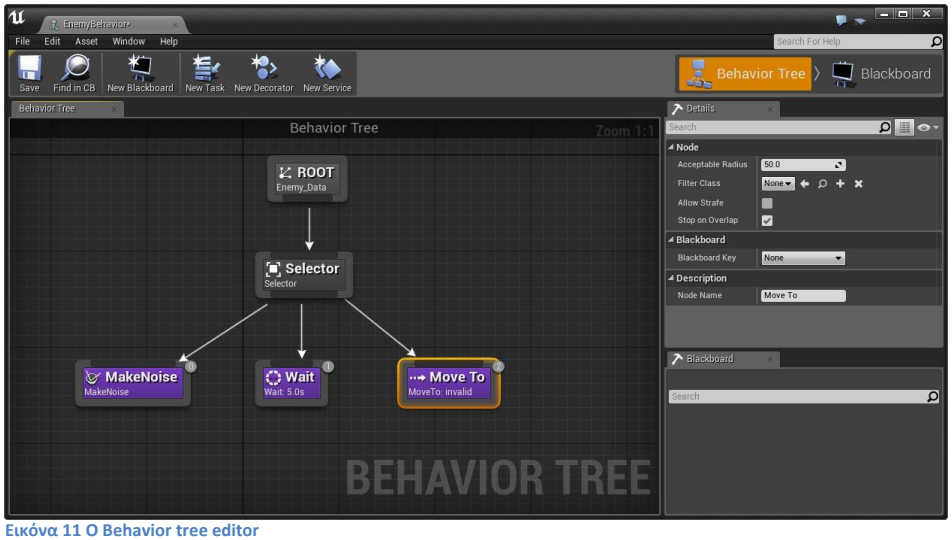


Εικόνα 10 Blueprint Editor με blueprint

Εδώ γίνεται η μορφοποίηση των Blueprints. Τα Blueprints είναι ειδικά μέρη που παρέχουν διεπαφή βασισμένη σε κόμβους μέσω των οποίων μπορούν να δημιουργηθούν καινούριοι τύποι actors, και script events χωρις την χρηση κώδικα. Ο editor αυτός περιέχει διάφορα εργαλεία που βοηθούν στη δημιουργία μεταβλητών, πινάκων, συναρτήσεων. Επίσης διαθέτει εργαλεία για εύκολο debugging. Ένα άλλο χαρακτηριστικό του Blueprint editor είναι ότι παρέχει ποικιλία μορφών ανάλογα με τον τύπο του Blueprint.

2.2.3Behavior Tree Editor

Στον Behavior Tree Editor γίνεται η ανάπτυξη τεχνιτής νοημοσύνης μέσα απο ένα οπτικό σύστημα βασισμένο σε κόμβους για τους actors.



Εικόνα 11 Ο Behavior tree editor

2.2.4 Παιχνίδια που αναπτύχθηκαν στην Unreal Engine

- Batman: Arkham City Lockdown
- Batman: Arkham Origins
- Mortal Kombat X
- Jacob Jones and the Bigfoot Mystery

2.3 Το XML

Το xml είναι μια γλώσσα σήμανσης η οποία ορίζει ένα σύνολο κανόνων με σκοπό την κωδικοποίση εγγράφων έτσι ώστε να είναι κατανοητό και απο μηχανές αλλα και απο ανθρώπους. Βοηθάει στην μορφοποίηση δεδομένων. Σε αυτή την πτυχιακή χρησιμοποιήθηκε για την αποθήκευση δεδομένων με σκοπό την μετέπειτα εξαγωγή τους για την δημιουργία των ερωταπαντήσεων του παιχνιδιού.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Εικόνα 12 Μια απλή μορφή xml

2.4 Η Γλώσσα C#

Η γλώσσα C# χρησιμοποιήθηκε για την ανάπτυξη του παιχνιδιού. Είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που αναπτύχθηκε απο τη Microsoft με στόχο την ανάπτυξη λογισμικού σε .NET.
Σκοπός της **C#** είναι:

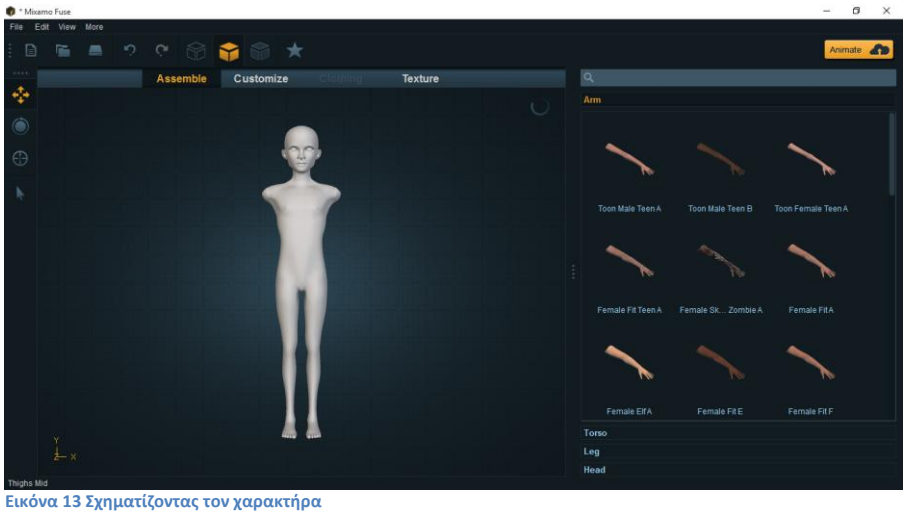
- Να είναι απλή, σύγχρονη και γενικής χρήσης αντικειμενοστραφής γλώσσα προγραμματισμού
- Η γλώσσα και οι εφαρμογές της θα πρέπει να παρέχουν υποστήριξη για τις αρχές λογισμικού όπως type checking, array bounds checking, έλεγχο για την προσπάθεια χρήσης μεταβλητών που δεν τους έχει δοθεί τιμή και αυτόματη συλλογή απορριμάτων.
- Η αξιοπιστία και η παραγωγικότητα του προγραμματιστή είναι πολύ σημαντικές.
- Η γλώσσα αυτή έχει ως σκοπό τη χρήση στην ανάπτυξη στοιχείων λογισμικού κατάλληλα για εγκατάσταση σε κατανεμημένα περιβάλλοντα.
- Η φορητότητα είναι εξίσου σημαντική.
- Όπως επίσης και η υποστήριξη για internationalization.
- Προορίζεται να είναι κατάλληλη για την ανάπτυξη εφαρμογών σε hosted αλλά και embedded συστήματα.
- Αν και οι εφαρμογές που αναπτύσονται σε C# θα έπρεπε να είναι οικονομικές σε μνήμη και επεξεργαστική ισχύ, η γλώσσα αρχικά δεν είχε τον στόχο την ανταγωνιστηκότητα σε αυτά συγκριτικά με την C ή την assembly.

3.Υλοποίηση

3.1 Ο κεντρικός χαρακτήρας

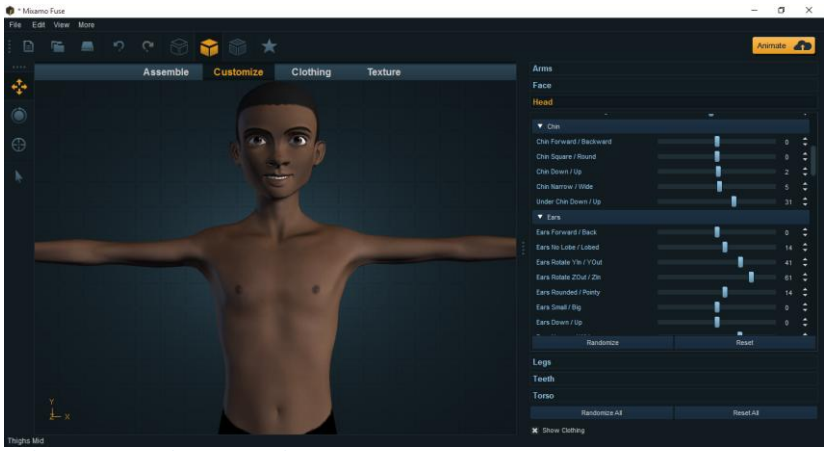
Το 1ο στάδιο για την ανάπτυξη του παιχνιδιού είναι η σχεδίαση του κεντρικού μας χαρακτήρα. Για την δημιουργία του χαρακτήρα που συνοδεύει αυτό το παιχνίδι προτιμήθηκε το πρόγραμμα mixamo fuse.

Είναι ένα απλό πρόγραμμα στη χρήση όπου πολύ εύκολα μπορούμε να δημιουργήσουμε τον χαρακτήρα που θέλουμε και έπειτα να δημιουργήσουμε animation.



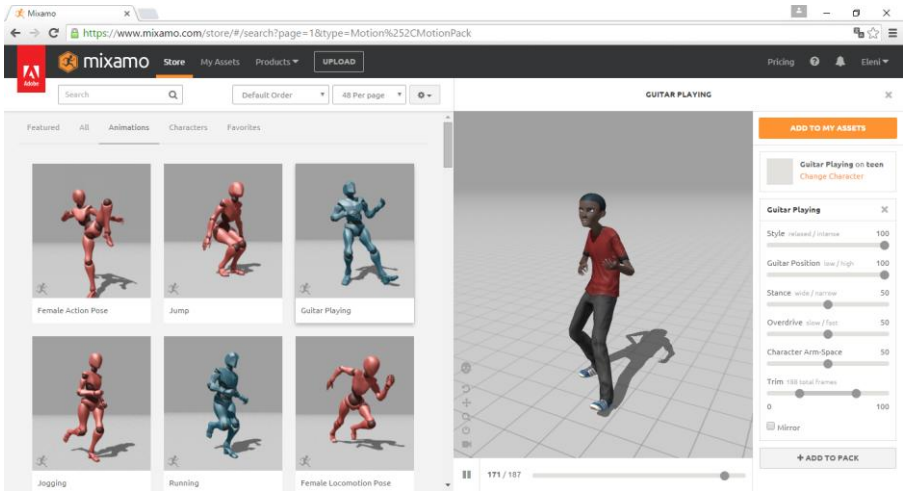
Εικόνα 13 Σχηματίζοντας τον χαρακτήρα

Επιλέγοντας τμηματικά μέρη του σώματος, έπειτα μπορούμε να τα επεξεργαστούμε ώστε να δημιουργήσουμε ακριβώς αυτό που θέλουμε. Μεταβάλλοντας τις τιμές που φαίνονται παρακάτω για όλα τα μέρη του σώματος επιτυγχάνεται αυξομείωση κάποιων μεγεθών (πχ τη θέση των ματιών, πόσο μέσα θα είναι, το χρώμα τους).



Εικόνα 14 Επεξεργασία του χαρακτήρα

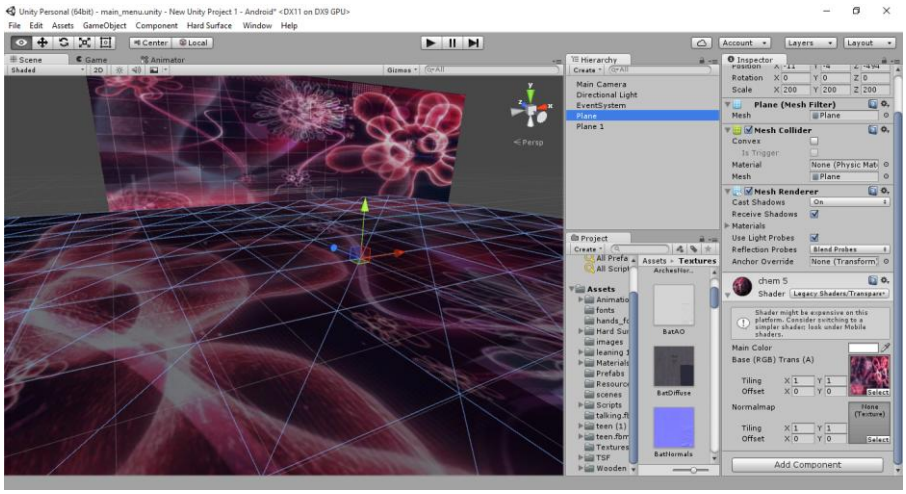
Αφού ολοκληρωθεί η επεξεργασία του χαρακτήρα μας, πατόντας το κουμπί “animate” πάνω δεξιά μεταφερόμαστε στην online πλατφορμα όπου έχει ανέβει ο χαρακτήρας και γίνεται rigged, δηλαδή δημιουργούνται οι «αρθρώσεις» με σκοπό την δημιουργία animation. Έπειτα μπορούμε να βρούμε τα animation που θέλουμε και να τα προσαρμόσουμε στον χαρακτήρα μας.



Εικόνα 15 Προσαρμογή του animation στον χαρακτήρα

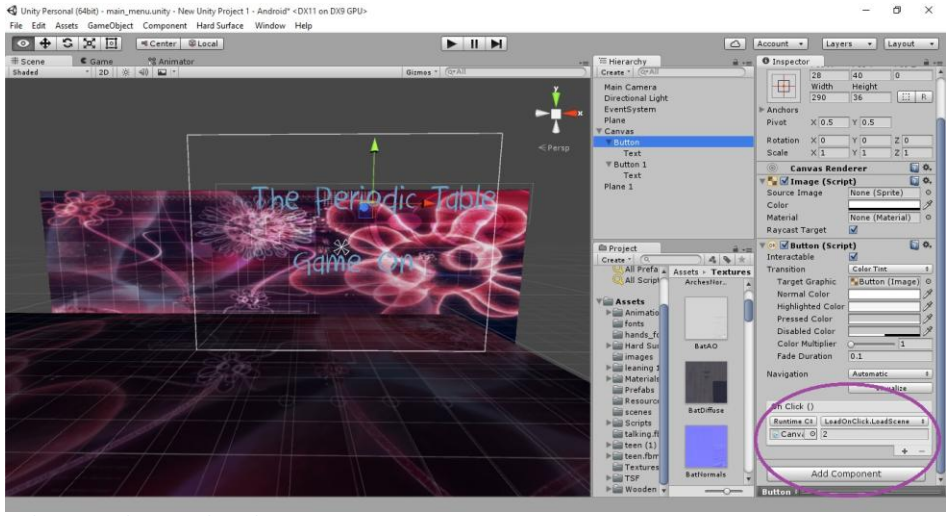
3.2 Το μενού επιλογής

Για την δημιουργία του μενού δημιουργούμε νέο πρότζεκτ και έπειτα καινούρια σκηνή. Μέσα στη σκηνή προσθέτουμε μια κάμερα, ένα directional light, 2 planes τα οποία θα είναι κάτι σαν τον τοίχο και το παάτωμα μας και μέσα σε αυτά προσθέτουμε texture στους shaders.



Εικόνα 16 Δημιουργία του Menu

Στη συνέχεια προσθέτουμε το canvas το οποίο επιλέγουμε να περιέχει 2 κουμπιά το κάθε ‘ενα απο τα οποία θα πηγαίνει στα αντίστοιχα levels. Αυτό γίνεται δηλώνοντας στο panel να τρέχει ένα script και μετά «λεμε» στα buttons να καλούν μια μέθοδο όπως και φαίνεται στην παρακάτω εικόνα.

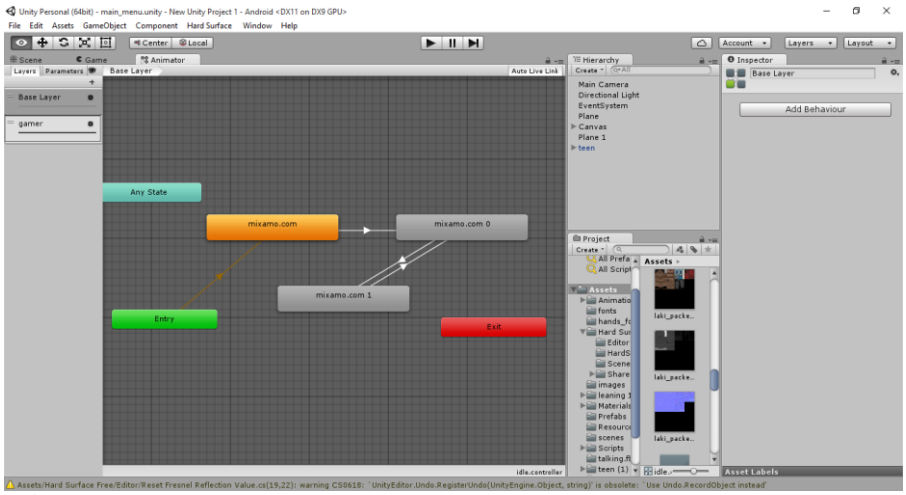


Εικόνα 17 Πως ένα κουμπί καλεί το script

Ο κώδικας που καλείται για την παραπομπή σε άλλο level όταν πατηθεί ένα απο τα 2 κουμπιά είναι:

```
public class LoadOnClick : MonoBehaviour {

        public GameObject loadingImage;

        public void LoadScene (int level) {
         Debug.Log("button clicked", gameObject);
         Application.LoadLevel(level);
}
}
```

Στη συνέχεια προσθέτουμε τον χαρακτήρα μας(Lakis) ο οποίος αποτελείται απο 7 μέρη (body, bottoms, lashes, eyes, hips, shoes, tops) σε κάθε ένα απο τα απoία προσθέτουμε τον αντίστοιχο texture. Για να δώσουμε ρεαλιστική κίνηση στον χαρακτήρα μας δημιουργούμε έναν animator. Ο animator είναι ο editor στον οποίο προσθέτουμε τα animations ώστε να δημιουργήσουμε μια συνεχή και ομαλή ροή κίνησης μέσα απο τα states.



Εικόνα 18 Animator για το μενου

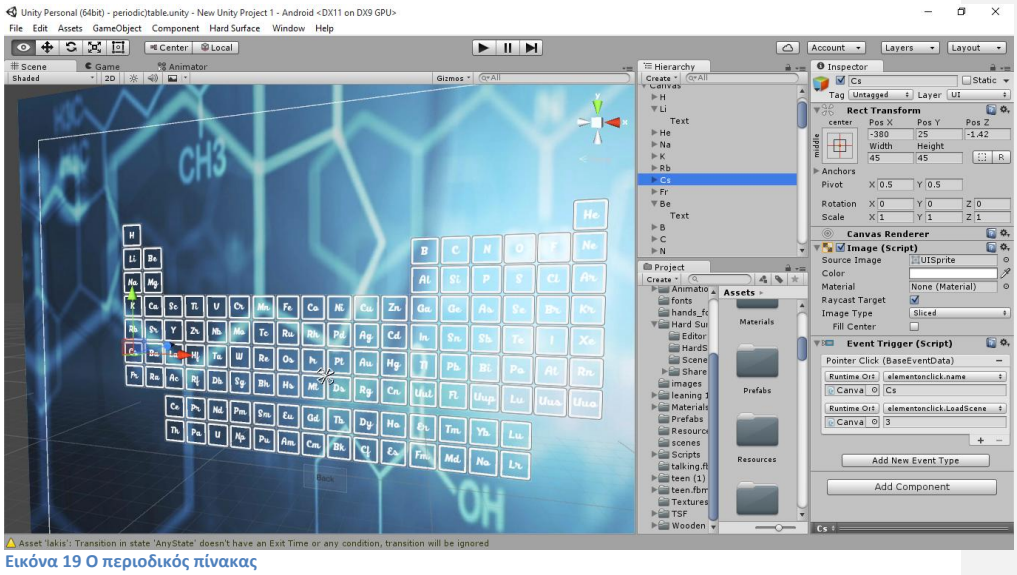
Το αρχικό state είναι το Entry το οποίο αναφέρεται στο ξεκίνημα του level. Τα επόμενα 3 states είναι αυτά που προστέθηκαν σύμφωνα με την κίνηση που θέλουμε να εφαρμόσουμε. Υπάρχουν και άλλα 3 states το «Any state» το οποίο αναφέρεται σε οποιαδήποτε φάση βρίσκεται το αντικείμενο που θέλουμε να δώσουμε animation και το exit.

Μορφοποιήθηκε: Ελληνικά (Ελλάδας)

3.3 Ο Περιοδικός πίνακας

Στο επόμενο level δημιουργήθηκε ο περιοδικός πίνακας, Για την κατασκευή του έβαλα ένα plane με ένα texture και τη κάμερα. Στη συνέχεια πρόσθεσα ένα Canvas το οποίο αποτελείτε απο 118 κουμπιά ένα για κάθε χημικό στοιχείο. Στα κουμπιά αυτά υπάρχει ένα text το οποίο γράφει το όνομα του χημικού στοιχείου. Το κάθε button παραπέμπει σε επόμενη σκηνή η οποία περιέχει αναλυτικές πληροφορίες για κάθε στοιχείο. Η παραπομπή αυτή γίνεται όπως και στο κεντρικό μενού. Στο κάτω μέρος υπάρχει ένα ακόμα button το «back» το οποίο πατώντας το πηγαίνουμε πίσω στο αρχικό μενού.

Μορφοποιήθηκε: Γραμματοσειρά: (Προεπιλεγμένη) +Επικεφαλίδες, 14 pt, Έντονα, Χρώμα γραμματοσειράς: Έμφαση 1, Ελληνικά (Ελλάδας)



Εικόνα 19 Ο περιοδικός πίνακας

3.4 Πληροφοριες για τα στοιχεια

Σε αυτή τη σκηνή έχουμε και πάλι ένα plane με texture, μια κάμερα και ένα directional light. Επίσης υπάρχει ένα back button το οποίο μας πηγαίνει πίσω στον περιοδικό πίνακα. Στο level αυτό η γενική λογική είναι η εξαγωγή δεδομένων απο ένα xml βάση του ονόματος του κουμπιού του οποίου πατήθηκε. Στο xml περιέχονται όλες οι πληροφορίες που θέλουμε να εμφανίζονται στην οθόνη.

```
<elements>
	<element id="H">
		<name>Hydrogen</name>

	<url>https://upload.wikimedia.org/wikipedia/commons/thumb/8/83/Hydrogen_dischar
ge_tube.jpg/1920px-Hydrogen_discharge_tube.jpg</url>
			<number>1</number>
			<property>A colourless, odourless gas.</property>
			<details>Hydrogen is an essential element for life.
It is present in water and in almost all the molecules in living things.
However, hydrogen itself does not play a particularly active role.
It remains bonded to carbon and oxygen atoms,
while the chemistry of life takes place at the more active sites involving,
for example, oxygen, nitrogen and phosphorus.</details>
			<melt>−259.16°C</melt>
			<boil>−252.879°C</boil>
			<state>Gas </state>
			<density>0.000082</density>
		</element>
</elements>
```

Εικόνα 20 Μέρος του xml

Αρχικά στον κώδικα πρέπει να φορτώσουμε το xml το οποίο είναι αποθηκευμένο τοπικά.

```
    void Awake()
    {
        fileName = "elemdetail";
        Players = new List<Element>(); // initalize player list
    }
    void Start()
    {
        loadXMLFromAssest();
        readXml();
    }
    private void loadXMLFromAssest()
    {
        xmlDoc = new XmlDocument();
        if (System.IO.File.Exists(getPath()))
        {
            xmlDoc.LoadXml(System.IO.File.ReadAllText(getPath()));
        }
        else
        {
            textXml = (TextAsset)Resources.Load(fileName, typeof(TextAsset));
            xmlDoc.LoadXml(textXml.text);
        }
    }
   private void createElement()
    {
        Element tempPlayer = new Element();

        tempPlayer.name = "name" + tempPlayer.id;
```

```
        Players.Add(tempPlayer);

        XmlNode parentNode = xmlDoc.SelectSingleNode("elements");
        XmlElement element = xmlDoc.CreateElement("element");
        element.SetAttribute("id", tempPlayer.id.ToString());
        element.AppendChild(createNodeByName("name", tempPlayer.name));
        element.AppendChild(createNodeByName("property", tempPlayer.property));
        element.AppendChild(createNodeByName("details", tempPlayer.details));
        parentNode.AppendChild(element);
        xmlDoc.Save(getPath() + ".xml");
    }
    private XmlNode createNodeByName(string name, string innerText)
    {
        XmlNode node = xmlDoc.CreateElement(name);
        node.InnerText = innerText;
        return node;
    }
```

Εικόνα 21 Φορτώνουμε το xml

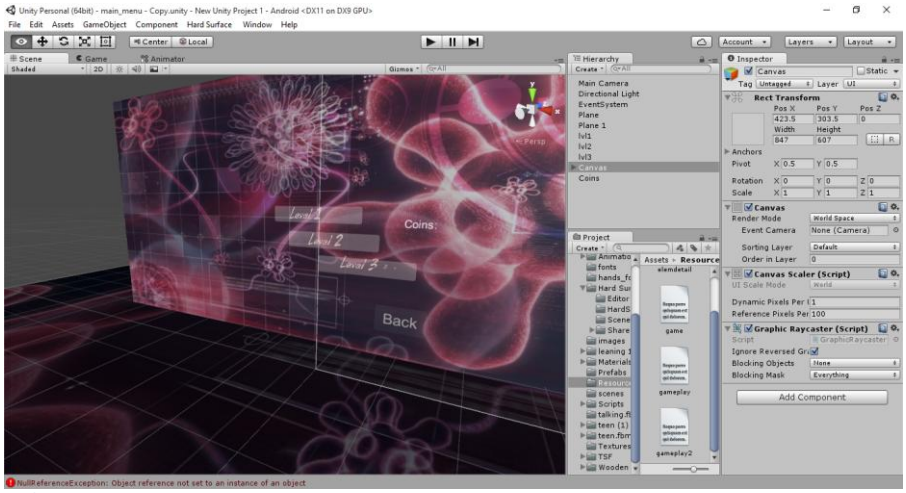
Έπειτα διαβάζουμε το xml και για κάθε περιεχόμενο των tag που ζητάμε αποθηκεύουμε το string που βγαίνει και το τοποθετούμε στο αντίστοιχο TextMesh της σκηνής μας ώστε να εμγανίζεται στην οθόνη. Η μέθοδος IEnumerator παίρνει το string απο το tag url και το μετατρέπει σε image. Για να εμφανιστεί το image αυτό καλείται η OnGUI() και δημιουργεί ένα πλαίσιο για την εμφάνιση της εικόνας.

```
private void readXml()
    {
        foreach (XmlElement node in xmlDoc.SelectNodes("elements/element"))
        {
            Element tempPlayer = new Element();
            tempPlayer.id = node.GetAttribute("id");
            tempPlayer.name = node.SelectSingleNode("name").InnerText;
            strname = tempPlayer.name;
            tempPlayer.property = node.SelectSingleNode("property").InnerText;
            strpro = tempPlayer.property;
            tempPlayer.details = node.SelectSingleNode("details").InnerText;
            strdeta = tempPlayer.details;
            tempPlayer.urlim = node.SelectSingleNode("url").InnerText;
            tempPlayer.number = int.Parse(node.SelectSingleNode("number").InnerText);
            strnum = tempPlayer.number;
            tempPlayer.melt = node.SelectSingleNode("melt").InnerText;
            strme = tempPlayer.melt;
            tempPlayer.boil = node.SelectSingleNode("boil").InnerText;
            strbo = tempPlayer.boil;
             tempPlayer.state = node.SelectSingleNode("state").InnerText;
            strst = tempPlayer.state;
            tempPlayer.density = node.SelectSingleNode("density").InnerText;
            strde = tempPlayer.density;
            url = tempPlayer.urlim;
            if (tempPlayer.id == btnckl)
            {
                tname = (TextMesh)GameObject.Find("name").GetComponent<TextMesh>();
                tname.text = strname;
                tm = (TextMesh)GameObject.Find("melt").GetComponent<TextMesh>();
                tm.text = "melting point:" + strme;
                tpro = (TextMesh)GameObject.Find("property").GetComponent<TextMesh>();
                tpro.text = strpro;
                tb = (TextMesh)GameObject.Find("boil").GetComponent<TextMesh>();
                tb.text = "boiling point:" + strbo;
                ts = (TextMesh)GameObject.Find("state").GetComponent<TextMesh>();
                ts.text = "state:" + strst;
```

```
                td = (TextMesh)GameObject.Find("density").GetComponent<TextMesh>();
                td.text = "density:" + strde;
                tdetail =
(TextMesh)GameObject.Find("details").GetComponent<TextMesh>();
                tdetail.text = strdeta;
                break;
            }
        }
        StartCoroutine(loadImage(url));
    }
    private IEnumerator loadImage(string url)
    {
        yield return 0;
        www = new WWW(url);
        yield return www;
        r = www.texture;

    }
    void OnGUI()
    {

        GUILayout.Label(r, GUILayout.Width(250), GUILayout.Height(250));
    }
```

Εικόνα 22 Η εμφάνιση των πληροφοριών στην οθόνη

3.5 Level Selection Menu



Εικόνα 23 Level selection Menu

Το level αυτό σχεδιάστηκε στην ίδια λογική της πρώτης σκηνής (Κεντρικό Menu). Περιέχει buttons, planes, ένα image αλλα περιέχει ακόμα και ένα TextMesh το οποίο δείχνει τους πόντους που έχουν μαζευτεί απο το παιχνίδι (Το επόμενο level).

Στο script βλέπουμε οτι καλείται η μέθοδος Start() η οποία καλείται κάθε φορά που ξεκινάει να τρέχει ο κώδικας. Μέσα στη μέθοδο αυτή καλείται η μέθοδος getCoins απο την κλάση Coins και ο ακέραιος που επιστρέφεται μετατρέπεται σε string και μπαίνει στο TextMesh. Στη συνέχεια για να γίνει ένας διαχωρισμός για τα level απενεργοποιούμε όλα τα buttons πλην του 1^{ου} και επανενεργοποιούνται ένα ένα κάθε φορά που αυξάνεται ο coins κατα 20. Τέλος καλείται η μέθοδος η οποία μας μεταφέρει στην επόμενη σκηνή του παιχνιδιού και στο level που θέλουμε.

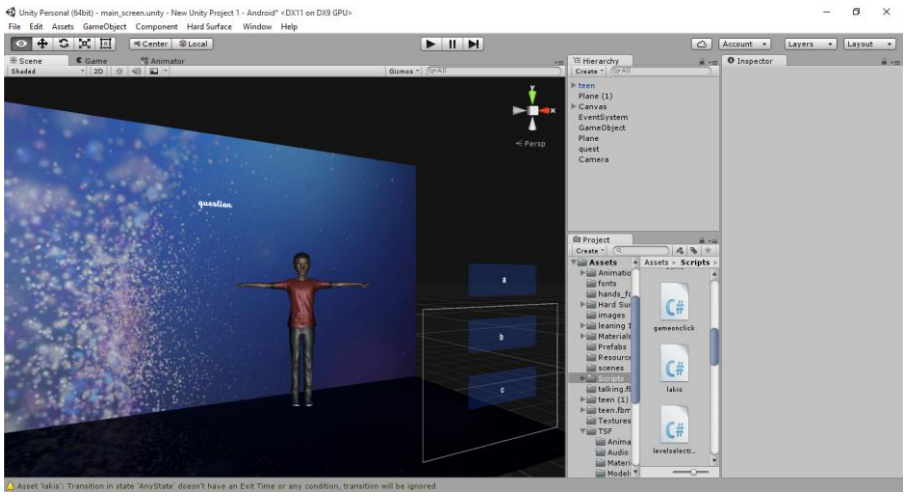
```
void Start()
    {
        coins = Coins.getCoins();

        tm = (TextMesh)GameObject.Find("Coins").GetComponent<TextMesh>();
        tm.text = (coins.ToString());
        btn = GameObject.Find("Button1");
        btn.SetActive(false);
        btn = GameObject.Find("Button2");
        btn.SetActive(false);
        if (coins>20){
            btn.SetActive(true);
        }
        else if (coins > 40)
        {
            btn1.SetActive(false);
        }
        else if (coins > 60)
        {
            btn2.SetActive(false);
        }
        else if (coins > 80)
```

```
        {
            btn2.SetActive(false);
        }
      }

    public void LoadScene(int level)
    {
        Debug.Log(name);
        Application.LoadLevel(level);
        reader.levelselected(name);
    }
```

Εικόνα 19 Ο κώδικας του level selection

3.6 Το παιχνίδι

Αυτή η σκηνή αποτελεί και το κύριο μέρος αυτής της πτυχιακής.
Εδώ βλέπουμε να υπάρχει ο χαρακτήρας μας που θα κινείται, ένα textmesh στο οποίο θα γράφεται η ερώτηση, και ενα canvas με 3 buttons όπου θα φαίνονται οι 3 εναλλακτικές απαντήσεις.



Εικόνα 24 Δημιουργία της σκηνής του παιχνιδιού

Αρχικά θέλουμε να δώσουμε κίνηση στο χαρακτήρα μας. Έχουμε 3 διαφορετικά animation. Ένα για όταν δεν υπάρχει input απο τον χρήστη, ένα για όταν υπάρχει σωστή απάντηση και ένα για λανθασμένη. Αυτό επιτυγχάνεται με την απόδοση script κατευθείαν στον χαρακτήρα.

Εικόνα 25 Απόδοση Animation σε χαρακτήρα

Πρώτα πρέπει να δηλώσουμε στις ιδιότητες του χαρακτήρα μας ποια animation πρέπει να εκτελέσει. Αυτό γίνεται όπως δείχνει η παραπάνω εικόνα 20.

```
public Animation anim;
  public static int mv;
  public static float timeleft = 5;
  public static void mov(int i)
  {
      mv = i;
      Debug.Log("movement" + mv);
  }
  // Use this for initialization
  void Start()
  {
      anim = GetComponent<Animation>();

      GetComponent<Animation>().Play("happy_idle");
  }

  // Update is called once per frame
  void Update()
  {
      timeleft = Time.deltaTime;
      if (mv == 1)
      {
          anim.Stop("happy_idle");
          anim.Play("fist_pump");
          if (timeleft < 5)
          {
              mv = 0;
          }
      }
      else if (mv == 2)
      {
          anim.Stop("happy_idle");
          anim.Play("cocky_head");
          if (timeleft < 5)
          {
              mv = 0;
          }
      }
      if (mv == 0)
      {
          anim.PlayQueued("happy_idle");

      }
  }
```

Εικόνα 26 Script για την κινηση του χαρακτήρα

Στο script αυτό της εικόνας 21 παίρνουμε μια τιμή μέσα απο τη μέθοδο mov (η τιμή αυτή δίνεται απο την κλάση του canvas που δείχνει ποιό button έχει πατηθεί). Στην μέθοδο Start() δηλώνουμε ώς αρχικό animation το «happy idle» αυτό είναι και το default animation όπως και έχει δηλωθεί για κάθε φορά που τελειώνει κάποιο άλλο (δηλαδή μόλις περάσουν 5 δεύτερα) , στις άλλες περιπτώσεις, όταν δηλαδή δωθεί σωστή ή λανθασμένη απάντηση, σταματάει το default animation και ξεκινάει το αντίστοιχο που θέλουμε για 5 δευτερόλεπτα. Το playqueued βοηθάει στην ομαλή εναλλαγή των animation, σε αντίθεση με το play ή stop που «κόβουν» απότομα το animation.

Στη συνέχεια για τις ερωτησεις και τις απαντήσεις χρησιμοποιήθηκε και πάλι xml όπου έχουμε ένα για κάθε level και περιλαμβάνει τον αριθμό της ερώτησης, τον αριθμό του level, την

ερώτηση, 2 λάθος απαντήσεις και τη σωστή.

```
<quiz>
       <question id="1">
       <level>1</level>
              <questionText>An odourless colourless gas with an atomic number
1</questionText>
              <element>H</element>
                     <correct>hydrogen</correct>
                     <wrong1>beryllium</wrong1>
                     <wrong2>francium</wrong2>
              </question>
</quiz>
```

Εικόνα 27 Μέρος του xml του παιχνιδιού

Το parsing του xml γίνεται τοπικά όπως και περιγράφηκε στο 3.4. Αυτή τη φορά όμως πρεπει να γίνεται μια κλήση για το διάβασμα του xml βάση του επιλεγμένου level

```
public static void levelselected(string lvl)
    {
        lvlid = System.Int32.Parse(lvl);
        Debug.Log("lvlid " + lvlid);
        fileName = "gameplay";

        if (lvlid == 2)
        {
            fileName = "gameplay2";
            loadXMLFromAssest();
            readXml(i);
        }
    }
```

Εικόνα 28 'Ανοιγμα xml βάση επιλεγμένου level

Αφού λοιπόν ανοίξει το xml και διαβάσει τους κόμβους του, καλείται η μέθοδος randquest(), η οποία παίρνει τα string απο τους κόμβους και τα τοποθετεί στα textmeshes με τυχαία σειρά.

```
   public static void randquest()
    {
        y = UnityEngine.Random.Range(1, 4);
  //      Debug.Log("y=" + y);

        if (y == 1) {

            s1 = strcorrect;
            s2 = strwr1;
            s3 = strwr2;
            t1 = (TextMesh)GameObject.Find("q1").GetComponent<TextMesh>();
            t1.text = s1 ;
            t2 = (TextMesh)GameObject.Find("q2").GetComponent<TextMesh>();
            t2.text = s2;
            t3 = (TextMesh)GameObject.Find("q3").GetComponent<TextMesh>();
            t3.text = s3;

        }
        if (y == 2)
        {
            s1 = strwr1;
            s2 = strcorrect;
```

```
            s3 = strwr2;
            t1 = (TextMesh)GameObject.Find("q1").GetComponent<TextMesh>();
            t1.text = s1;
            t2 = (TextMesh)GameObject.Find("q2").GetComponent<TextMesh>();
            t2.text = s2;
            t3 = (TextMesh)GameObject.Find("q3").GetComponent<TextMesh>();
            t3.text = s3;

        }
        if (y == 3)
        {
            s1 = strwr2;
            s2 = strwr1;
            s3 = strcorrect;
            t1 = (TextMesh)GameObject.Find("q1").GetComponent<TextMesh>();
            t1.text = s1;
            t2 = (TextMesh)GameObject.Find("q2").GetComponent<TextMesh>();
            t2.text = s2;
            t3 = (TextMesh)GameObject.Find("q3").GetComponent<TextMesh>();
            t3.text = s3;

        }

    }
```

Εικόνα 29 Απο το xml σε textmesh

Όταν γίνει και αυτό πρέπει να διαβάζουμε το input του χρήστη ώστε για κάθε κουμπί που πατάει να ξέρουμε ποιό είναι και αν αυτό είναι η σωστή απάντηση.

Για να γίνει αυτό, καλούμε την εξής μέθοδο:

```
    public void OnClicked(Button button)
    {
        bn = button.name;
        btntx = button;

        sayit();
    }
```

Εικόνα 30 Μέθοδος που βλέπει ποιο button πατήθηκε

Και έπειτα εξακριβώνουμε εαν αυτό που πατήθηκε είναι το σωστό. Εαν το button που πατήθηκε είναι η σωστή απάντηση, τότε ο counter coin αυξάνεται κατά 5 και καλείται το διάβασμα του επόμενου κόμβου του xml και η κλάση για το animation του χαρακτήρα. Διαφορετικά, αν είναι λάθος ο counter loose μειώνεται κατά 1 και καλείται η κλάση για το αντίστοιχο animation.

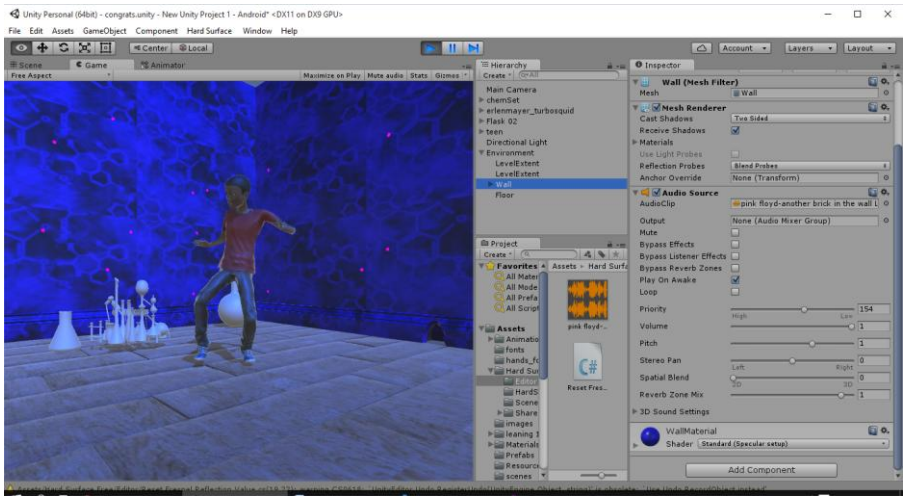
```
    private void sayit()
    {
        if (bn != null)
        {

            if (y == 1)
            {
                if (bn == "q1")
                {

                    ani = 1;
                    coin += 5;
                    Debug.Log("correct!" + xc++);
                    lakis.mov(ani);
                    Coins.setcoins(coin);
                    readXml(++i);
```

```
                    bn = null;
                }
                else if (bn == "q2")
                {

                    loose--;
                    Coins.setcoins(loose);
                    bn = null;
                }
                else if (bn == "q3")
                {

                    loose--;
                    Coins.setcoins(loose);
                    bn = null;
                }

            }
            if (y == 2)
            {
                if (bn == "q2")
                {
                    ani = 1;
                    coin += 5;
                    Coins.setcoins(coin);
                    Debug.Log("correct!");
                    lakis.mov(ani);
                    readXml(++i);

                }

                else if (bn == "q1")
                {

                    loose--;
                    Coins.setcoins(loose);
                    lakis.mov(2);
                    bn = null;
                }
                else if (bn == "q3")
                {

                    loose--;
                    Coins.setcoins(loose);
                    lakis.mov(2);
                    bn = null;
                }
            }
            if (y == 3)
            {
                if (bn == "q3")
                {
                    ani = 1;
                    coin += 5;
                    Coins.setcoins(coin);
                    Debug.Log("correct!");
                    lakis.mov(ani);
                    readXml(++i);
                    bn = null;

                }
```

```
                else if (bn == "q2")
                {

                    loose--;
                    Coins.setcoins(loose);
                    lakis.mov(2);
                    bn = null;
                }
                else if (bn == "q1")
                {


                    loose--;
                    Coins.setcoins(loose);
                    lakis.mov(2);
                    bn = null;
                }
            }
            if (coin >= 20)
            {
                levelselected("2");
            }

        }
        if (coin >= 180)
        {
            Application.LoadLevel(5);
        }
       // Coins.setcoins(score);
        PlayerPrefs.SetInt("level", lvlid);
        PlayerPrefs.SetInt("coins", coin);
        PlayerPrefs.Save();
    }
```

3.7 Η τελευταία σκηνή

Όταν ο χρήστης έχει πλέον συγκετρώσει αρκετούς πόντους, εδώ τελειώνει το παιχνίδι και ξεκινάει η σκηνή όπου βλέπουμε τον χαρακτήρα μας να πανηγυρίζει μέσα σε ένα δωμάτιο. Αυτή η σκηνή περιλαμβάνει έναν animator controller για το animation του χαρακτήρα μας και ένα audio source για την αναπαραγωγή ενός clip ήχου.



Για την προσαρμογή του παιχνιδιού σε συσκεύες android χρειάστηκε και ένα script ώστε να μεταφράζει ακριβώς τα gestures του χρήστη.

```
public class pinchzoom : MonoBehaviour
{
    public float perspectiveZoomSpeed = 0.5f;
    public float orthoZoomSpeed = 0.5f;


    void Update()
    {
        if (Input.touchCount == 2)
        {
            Touch touchZero = Input.GetTouch(0);
            Touch touchOne = Input.GetTouch(1);

            Vector2 touchZeroPrevPos = touchZero.position - touchZero.deltaPosition;
            Vector2 touchOnePrevPos = touchOne.position - touchOne.deltaPosition;

            float prevTouchDeltaMag = (touchZeroPrevPos - touchOnePrevPos).magnitude;
            float touchDeltaMag = (touchZero.position - touchOne.position).magnitude;

            float deltaMagnitudeDiff = prevTouchDeltaMag - touchDeltaMag;

            if (GetComponent<Camera>().orthographic)
            {
                GetComponent<Camera>().orthographicSize += deltaMagnitudeDiff *
orthoZoomSpeed;

                GetComponent<Camera>().orthographicSize =
```

```
Mathf.Max(GetComponent<Camera>().orthographicSize, 0.1f);
            }
            else
            {
                GetComponent<Camera>().fieldOfView += deltaMagnitudeDiff *
perspectiveZoomSpeed;

                GetComponent<Camera>().fieldOfView =
Mathf.Clamp(GetComponent<Camera>().fieldOfView, 0.1f, 179.9f);
            }
        }
    }
}
```

Εικόνα 31 Touch gestures χρήστη

Τέλος υπάρχει ένα script ώστε να αποθηκεύεται το state του παιχνιδιού στο σημείο που το άφησε ο χρήστης.

```
public class SaveLoad : MonoBehaviour {
    public static List<Game> savedGames = new List<Game>();
    public static void Save()
    {
        Debug.Log("saved");
        savedGames.Add(Game.current);
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Create(Application.persistentDataPath +
"/savedGames.gd");
        bf.Serialize(file, SaveLoad.savedGames);
        file.Close();
    }
    public static void Load()
    {
        if (File.Exists(Application.persistentDataPath + "/savedGames.gd"))
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream file = File.Open(Application.persistentDataPath +
"/savedGames.gd", FileMode.Open);
            SaveLoad.savedGames = (List<Game>)bf.Deserialize(file);
            file.Close();
        }
    }
    }
```

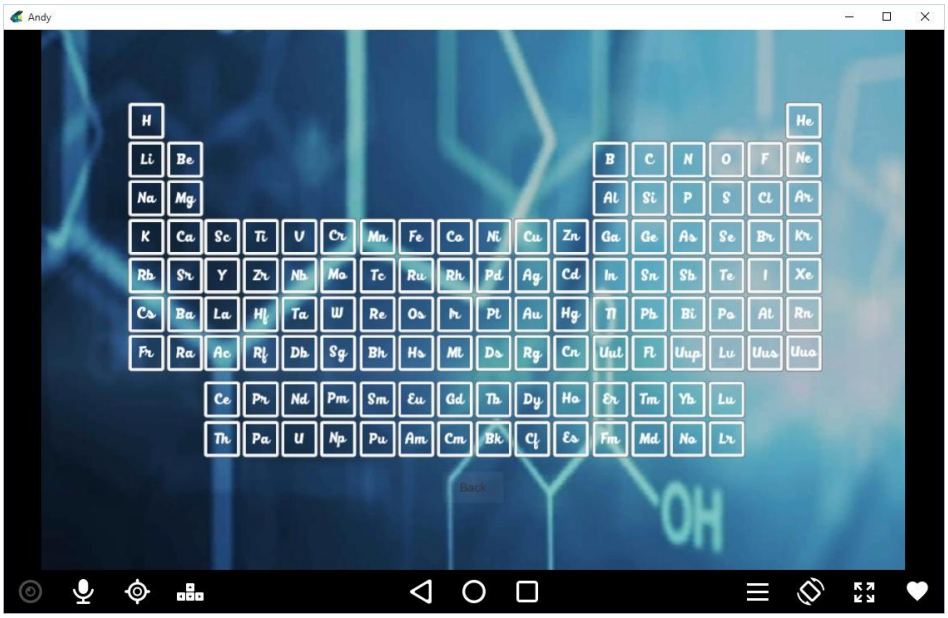
Εικόνα 32 Αποθήκευση του state του παιχνιδιού

3.5 Η Τελική μορφή

Scene 1: Απο εδώ μπορούμε να επιλέξουμε ένα απο τα δύο κουμπιά.

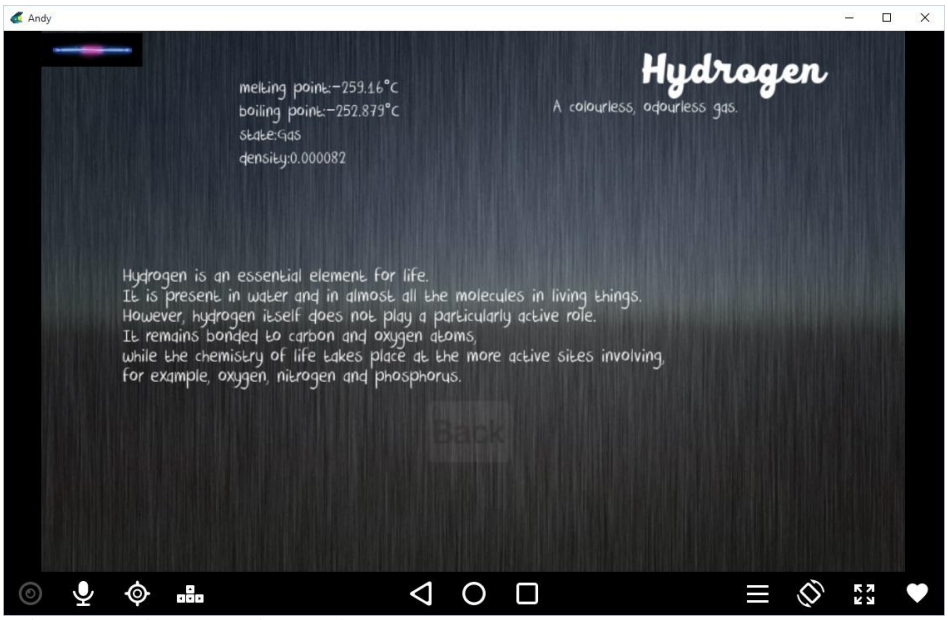
Εικόνα 33 Το μενου

Πατώντας το periodic table βλέπουμε τον περιοδικό πίνακα όπου κάθε στοιχείο λειτουργεί ως button.



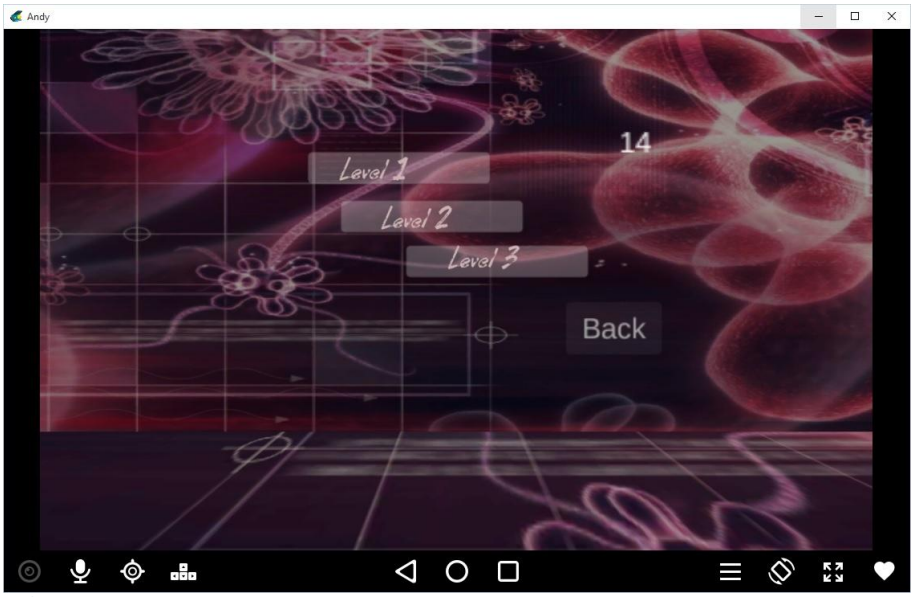
Εικόνα 34 Ο περιοδικός πίνακας

Αφού πατήσουμε σε οποιοδήποτε στοιχείο βλέπουμε αναλυτικές πληροφορίες για αυτό.



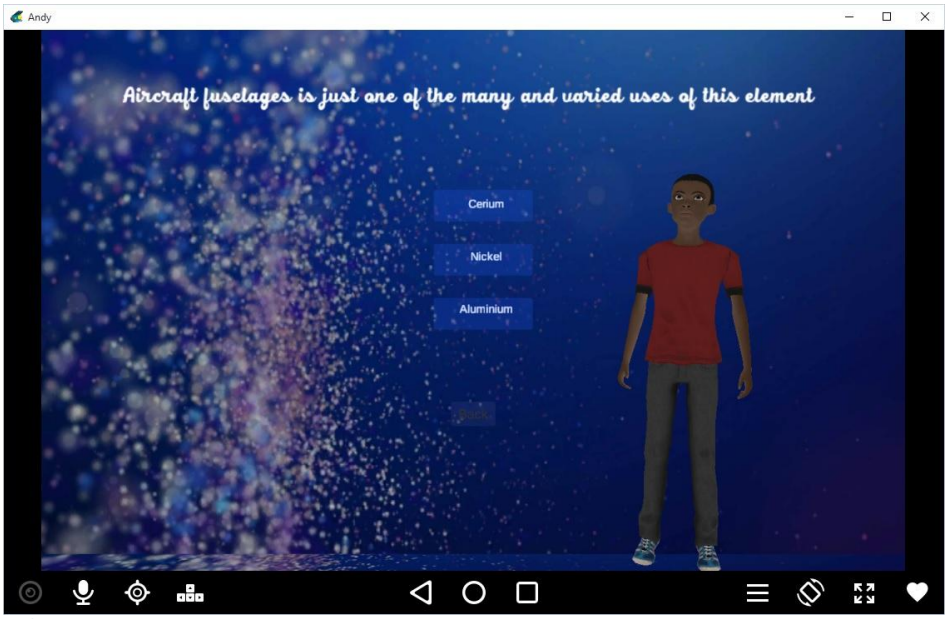
Εικόνα 35 Λεπτομέρειες για το κάθε στοιχείο

Πατώντας το Game On απο το αρχικό μενού μεταφερόμαστε στο level selection menu απο όπου μπορούμε να επιλέξουμε το level που θέλουμε να παίξουμε εφόσον το έχουμε ήδη περάσει και να δούμε συνολικά τους πόντους που έχουμε μαζέψει.



Εικόνα 36 Το level selection

Αυτή η σκηνή μας παρουσιάζει κάποιες ερωτήσεις και πατώντας σε κάποια απο τις 3 απαντήσεις παίρνουμε τους ανάλογους πόντους.



Εικόνα 37 Το quiz game

Όταν πλέον έχουμε μαζέψει τους συνολικούς πόντους που απαιτούνται για το τέλος του παιχνιδιού ο χαρακτήρας εμφανίζεται να χορεύει.

Εικόνα 38 Η νίκη

4. Συμπεράσματα

Παρόλο που η πτυχιακή έφτασε στο τέλος της υπάρχουν ιδέες για μελλοντική βελτίωση της. Μια πιθανή βελτίωση μπορεί να είναι η προσθήκη μιας ακομα σκηνής μετά το τέλους κάθε level για bonus πόντους όπου θα δείχνει τον χαρακτήρα μας να αναμιγνύει χημικά στοιχεία απο δοκιμαστικούς σωλήνες που θα αναγράφουν το στοιχείο και με την καθοδήγηση του χρήστη να γίνεται είτε η σωστη ανάμιξη ή μια μικρή έκρηξη. Μια ακόμα πιθανή σκηνή μπορεί να είναι η αναπαράσταση των μορίων για κάθε στοιχείο του περιοδικού πίνακα.

Η γενική διαδικασία της ανάπτυξης αυτής της πτυχιακής ήταν ιδιαίτερα χρονοβόρα όσον αφορά την έρευνα για την χρήση του σωστού game engine και μετέπειτα την ανάπτυξη του σωστού παιχνιδιού καθώς η αρχική ιδέα ήταν να ασχοληθώ με την αρχαία αγορά, πράγμα όμως που ήδη είχε πάρει αρκετό χρόνο όσο ήταν ακόμα στην αρχή. Το android αποτελεί μεγάλο και σημαντικό μέρος της ζωής μου μιας και ήδη το android development είναι κάτι που κατέχω και εξασκώ. Η πτυχιακή αυτή μου έδωσε την ευκαιρία να ασχοληθώ και με το «fun» μέρος του development αλλά και να έρθω σε επαφή με την ανάπτυξη γραφικών, πράγμα που για εμένα αποτέλεσε το πιο δύσκολο κομμάτι. Τέλος θα ήθελα να εκφράσω τη χαρά μου και την ικανοποίηση μου για το τελικό αποτέλεσμα και για αυτά που έμαθα.

5. Βιβλιογραφία

[1] http://www.w3schools.com/xml/ Πληροφορίες και οδηγίες για xml
[2] https://unity3d.com/learn/tutorials Tutorials και παραδείγματα για το unity
[3] http://www.tutorialspoint.com/csharp/ Tutorials για τη c#
[4] https://en.wikipedia.org/wiki/Android_(operating_system) Γενικές πληροφορίες για το android
[5] https://en.wikipedia.org/wiki/Game_engine Γενικά για τα game engine
[6] http://docs.unity3d.com/Manual/index.html Το documentation του unity
[7] http://docs.unity3d.com/ScriptReference/index.html Το scripting api του unity
[8] https://docs.unrealengine.com/latest/INT/ Το documentation της unreal
[9] https://www.youtube.com/watch?v=fRED_-LvJKQ Ενδιαφέροντα video tutorials για την ανάπτυξη παιχνιδιων στο unity.