ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

¨Σχεδιασμός και ανάπτυξη 2D video game σε Unity Engine με χρήση C#¨

Ονοματεπώνυμο: Βασιλόπουλος Στέφανος

Αριθμός μητρώου: 1884

Επιβλέπων καθηγητής: Ασημακόπουλος Γεώργιος

ΠΑΤΡΑ 2021

Εγκρίθηκε από τριμελή εξεταστική επιτροπή
Πάτρα 2021

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
Γεώργιος Ασημακόπουλος

2. Μέλος επιτροπής

3. Μέλος επιτροπής

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου και την ευγνωμοσύνη μου στον καθηγητή μου Γεώργιο Ασημακόπουλο για την ανάθεση του θέματος. Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για την στήριξή τους όλα αυτά τα χρόνια.

Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύεται την κατασκευή ενός παιχνιδιού 2 διαστάσεων (2D video game). Για την υλοποίηση του παιχνιδιού σαφώς χρειάστηκε η ‘μηχανή’ Unity, ένα γραφικό περιβάλλον με τεράστιες δυνατότητες τόσο σε 2D μοντέλα όσο και σε 3D, με αρκετά πλεονεκτήματα αλλά και μειονεκτήματα. Για τον σχεδιασμό και την ανάπτυξη του προγράμματος ώστε να δημιουργηθεί ένα πλήρως λειτουργικό παιχνίδι χρειάστηκε η γλώσσα προγραμματισμού C#. Μια γλώσσα προγραμματισμού που χρησιμοποιείται αρκετά στα βιντεοπαιχνίδια και όχι μόνο. Με την βοήθεια της C# δημιουργούνται scripts ώστε να μπορέσουμε να δώσουμε με μεγαλύτερη λεπτομέρεια και ακρίβεια διάφορα χαρακτηριστικά που θέλουμε να εισάγουμε. Η δημιουργία των scripts επιτυγχάνεται με το πρόγραμμα Visual Studio καθώς σε συνδυασμό με την Unity υπάρχουν πολλές βιβλιοθήκες. Τέλος για την ολοκλήρωση του παιχνιδιού χρησιμοποιήθηκαν προγράμματα όπως το Audacity για την επεξεργασία του ήχου, Photoshop για την κατασκευή μερικών sprites προσαρμοσμένα ειδικά για τις ανάγκες του παιχνίδι.

Abstract

The present dissertation deals with the construction of a 2D video game. For the implementation of the game clearly needed the ‘engine’ of Unity, a graphical environment with huge potential in both 2D and 3D models, with several advantages but also disadvantages. For the design and development of the program to create a fully functional game, C# was the needed programming language. A programming language that is widely used in video games and more. With the help of C#, scripts are created so that we can give more details and accuracy for various features we want to import. The creation of scripts is achieved with the program Visual Studio, combined with Unity can provide us many libraries. Finally, programs such as Audacity were used to complete the game for editing the sound and Photoshop to create some sprites specially adapted to the needs of the current video game.

Πίνακας Περιεχομένων

Κεφάλαιο 1 – Εισαγωγή

1.1 Πρόλογος

Κατά τους προηγούμενους αιώνες τα μυθιστορήματα, το θέατρο και αργότερα ο κινηματογράφος, αποτελούσαν τον κυρίαρχο τρόπο διασκέδασης. Από την δεκαετία του ’60 και την εφεύρεση του ηλεκτρονικού υπολογιστή, ο τρόπος διασκέδασης φαίνεται να άλλαξε μια για πάντα. Ο Eric Zimmerman , ένας σχεδιαστής video games, σε ένα τολμηρό μανιφέστο δηλώνει ότι αυτός ο αιώνας θα οριστεί από τα παιχνίδια. Όλο και περισσότερο, οι τρόποι με τους οποίους οι άνθρωποι θα περνούν το ελεύθερο τους χρόνο θα είναι τα παιχνίδια ή εμπειρίες που θα μοιάζουν με παιχνίδια. Από το Poker στο Pac-Man μέχρι και το Warcraft τα παιχνίδια είναι μηχανές εισροών και εκροών που κατοικούνται, χειραγωγούνται και διερευνώνται. Όταν παίζουμε, σκεφτόμαστε και μαθαίνουμε να ενεργούμε με νέους τρόπους.

Τα παιχνίδια υπήρχαν από την αρχαιότητα, παιδιά αλλά και ενήλικες στην προσπάθειά τους να διασκεδάσουν και να περάσουν ευχάριστα τον ελεύθερο τους χρόνο, εφεύρισκαν ομαδικά αλλά και ατομικά παιχνίδια, κάτι που καθιστά την αναπαραγωγή τους , μέρος της ζωής του ανθρώπου. Το παιχνίδι ικανοποιεί πολλές ανθρώπινες ανάγκες, όπως η εξερεύνηση, η μάθηση, η κοινωνικοποίηση , ο ενθουσιασμός, η χαρά της νίκης, αλλά και την προσπάθεια για την επίτευξη ενός στόχου. Το παιχνίδι δεν αποτελεί λοιπόν μόνο έναν τρόπο ψυχαγωγίας, αλλά μαθαίνει στον χρήστη πως να λύνει προβλήματα με ταχύτητα και αποτελεσματικότητα, να εξασκεί το μυαλό του, και να αποκτά δεξιότητες που θα του φανούν χρήσιμες στην ζωή του.

Τα παιδιά στις μέρες μας, δεν έχουν την ευκαιρία να παίξουν και να εξερευνήσουν τον κόσμο με την ελευθερία που είχαν κάποτε, μιας και τα κτίρια και οι δρόμοι το καθιστούν σχεδόν αδύνατο. Τα video games αποτελούν το μέσο για να ζήσουν όλες αυτές τις εμπειρίες που αδυνατούν να έχουν στην πραγματική ζωή. Προσφέρουν την εμπειρία του ρίσκου, της μάχης, του κινδύνου και της προσπάθειας, χωρίς όμως την φυσική καταπόνηση. Ο κόσμος πιστεύει ότι στα videogames υπάρχει μια άλλη πραγματικότητα. Η αλήθεια είναι ότι δεν είναι μόνο μία, αλλά πολλές. Μας δίνουν την δυνατότητα να διαλέξουμε όποια διαδρομή, και όποιο μονοπάτι θέλουμε, επηρεάζοντας με αυτό τον τρόπο, την εξέλιξη της ιστορίας. Δεν είναι μόνο μέσω διασκέδασης, αλλά και μέσω εκμάθησης. Όλο και περισσότερες εταιρείες κολοσσοί ανά τον κόσμο,

εκπαιδεύουν το προσωπικό τους μέσα από διαδραστικά παιχνίδια. Ακόμα και στα σχολεία, έχουν αρχίσει να τα εντάσσουν στο πρόγραμμα σπουδών.

Στα video games , όπως και στα παραδοσιακά παιχνίδια, ο παίκτης μπορεί να παίξει μαζί με άλλους παίκτες, είτε αντίπαλος, είτε συνεργατικά με σκοπό να πετύχουν ένα κοινό στόχο, είτε με αντίπαλο τον υπολογιστή- τεχνητή νοημοσύνη. Ακόμα υπάρχουν παιχνίδια, στα οποία δεν υπάρχει αρχή, μέση , και τέλος, αλλά ο παίκτης καλείται να μαζέψει όσους περισσότερους πόντους μπορεί ,πριν εμφανιστεί στην οθόνη του το μήνυμα game over. Ένα τέτοιου είδους παιχνίδι θα παρουσιαστεί σε αυτή την εργασία. Για τον σχεδιασμό ενός video game, απαιτούνται προγραμματιστές, γραφίστες, κειμενογράφοι, μουσικοί κ.α. πράγμα που το καθιστά προϊόν τέχνης και τεχνολογίας. Τα ηλεκτρονικά παιχνίδια αποτελούν την 8^{η} τέχνη (Manovich 2001) The Language of New Media, MIT Press, .Εκείνο λοιπόν που σκεφτόμαστε παρατηρώντας τα είναι ότι αποτελούν ένα είδος τέχνης το οποίο δημιουργεί κόσμους μέσα από την αναδημιουργία πολιτισμών.

Κεφάλαιο 2

2.1 Ορισμός βιντεοπαιχνιδιών

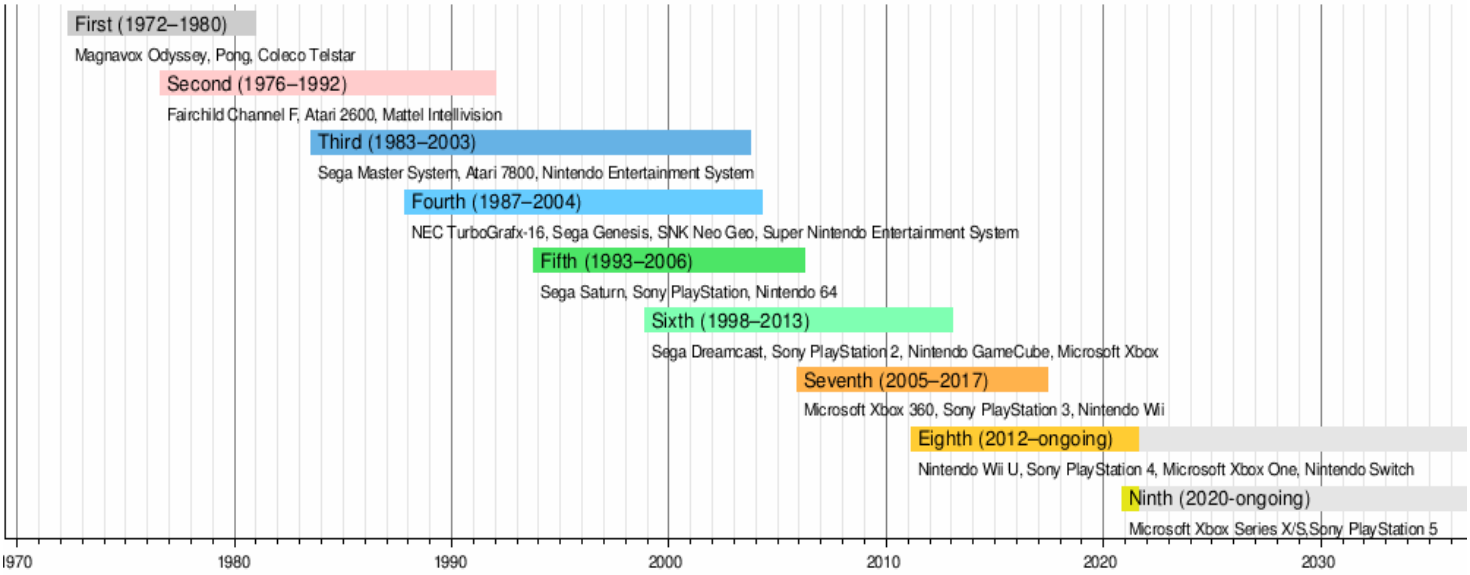
Με το όρο βιντεοπαιχνίδι (video game) εννοείται κάθε παιχνίδι που πραγματοποιείται με την χρήση οποιασδήποτε ηλεκτρονικής συσκευής, είτε κονσόλας, είτε ηλεκτρονικού υπολογιστή, είτε κινητού τηλεφώνου. Όλα τα ηλεκτρονικά παιχνίδια έχουν κάποια συσκευή εισόδου δεδομένων με την μορφή πληκτρολογίου, joystick, οθόνη αφής κ.α. , αλλά και μία μορφή εξόδου η οποία ικανοποιεί τις αισθήσεις του παίκτη, όπως οθόνη ή ηχεία. Περιλαμβάνουν αλληλεπίδραση με την διεπαφή του χρήστη για την παραγωγή οπτικής ανάδρασης. Αναφέρεται δηλαδή στον κάθε τύπου συσκευής που μπορεί να αναπαραστήσει δισδιάστατα ή τρισδιάστατα γραφικά.

Οι Salen και Zimmerman (2004) στην προσπάθειά τους να δώσουν τον κατάλληλο ορισμό στην περίπλοκη αλλά και πολυδιάστατη έννοια του ηλεκτρονικού παιχνιδιού, κάνουν μία ιστορική βιβλιογραφική ανασκόπηση με οκτώ διαφορετικές έννοιες του παιχνιδιού. Έτσι καταλήγουν στο ότι

¨*το παιχνίδι είναι ένα σύστημα όπου οι παίκτες συμμετέχουν σε μια τεχνητή σύγκρουση, η οποία οριοθετείται από κανόνες, και καταλήγει σε ένα μετρήσιμο αποτέλεσμα*¨ (σελ. 80)

Το σύστημα αυτό αποτελείται από αντικείμενα, ιδιότητες, εσωτερικές σχέσεις ανάμεσα στα αντικείμενα και τέλος το περιβάλλον. Οι παίκτες, είναι τα άτομα που παίρνουν μέρος στο παιχνίδι και το βιώνουν. Ο όρος τεχνητή χρησιμοποιείται για να απομονώσει τα γεγονότα του παιχνιδιού από αυτά της πραγματικής ζωής. Ως σύγκρουση, καλούμε την διαδικασία στην οποία ο ένας παίκτης προσπαθεί να υπερισχύσει έναντι των αντιπάλων του. Με τον όρο κανόνες είναι το πιο σημαντικό στοιχείο του παιχνιδιού, οι οποίοι οριοθετούν το παιχνίδι δίνοντας με αυτόν τον τρόπο στον παίκτη τα όρια μέσα στα οποία κινείται. Τέλος, ως μετρήσιμο αποτέλεσμα καλούμε τον τρόπο με τον οποίο αξιολογείται η προσπάθεια του παίκτη στο τέλος του παιχνιδιού, με τρόπο ξεκάθαρο και σαφή.

2.2 Ιστορία και εξέλιξη



Εικόνα 1 : «Ιστορική εξέλιξη των βιντεοπαιχνιδιών»

Η ιστορία των βιντεοπαιχνιδιών άρχισε κατά τα τέλη της δεκαετίας του 1950 στην Αμερική, την ίδια περίοδο που άρχισαν να μπαίνουν και οι ηλεκτρονικοί υπολογιστές στην καθημερινή ζωή των ανθρώπων. Στην αρχή τα παιχνίδια αυτά μπορούσαν να χρησιμοποιηθούν από το κοινό μόνο σε ειδικά καταστήματα, αλλά στην πορεία έγινε δυνατή και η χρήση τους εντός των σπιτιών. Τα πρώτα βιντεοπαιχνίδια έκαναν την εμφάνισή τους σε κονσόλες, σε φλίπερ, σε υπολογιστές αλλά και σε φορητές κονσόλες. Χωρίζονται σε 8 γενιές παιχνιδιών, ανάλογα με τις κονσόλες που κυκλοφορούσαν και την καινοτομία που είχε το κάθε παιχνίδι όπως θα δούμε παρακάτω.

Το πρώτο παιχνίδι για υπολογιστή σχεδιάστηκε το 1952 από τον Alexander Douglas, ως μέρος της διπλωματικής του διατριβής που αφορούσε την αλληλεπίδραση ανθρώπου – υπολογιστή. Το παιχνίδι αυτό λεγόταν oxo και είναι η γνωστή σε όλους τρίλιζα. Το 1958 , ο William Higinbotham δημιούργησε το Tennis For Two , ένας εξομοιωτής τένις ή Πινγκ πονγκ. Σκοπός του παιχνιδιού ήταν να περάσει το μπαλάκι από την μία μεριά του φιλέ στην άλλη. Ήταν το πρώτο ηλεκτρονικό παιχνίδι που σχεδιάστηκε για εμπορικούς σκοπούς. Τρεις φοιτητές από το MIT δημιούργησαν ένα διαστημικό παιχνίδι, το spacewar το 1961 το οποίο είχε διάδραση ανάμεσα σε δύο παίκτες. Ο παίκτες προσπαθούσαν να καταστρέψουν ο ένας το διαστημόπλοιο του άλλου ενώ ταυτόχρονα έπρεπε να αποφύγουν διάφορα εμπόδια.

Η πρώτη γενιά βιντεοπαιχνιδιών αφορούσε τα έτη 1972 έως και 1975. Η πρώτη εμπορική κονσόλα που βγήκε στην αγορά ήταν το Magnavox Odyssey, το έτος 1972, όπου μετέτρεπε την τηλεόραση, από μία παθητική οθόνη σε μία διαδραστική συσκευή. Ήταν ένα επαναστατικό βήμα που άλλαξε την οικιακή ψυχαγωγία για πάντα. Ο χειρισμός γινόταν με δύο ροδέλες της οποίες ο παίκτης καλούταν να γυρίσει, μία για την κίνηση πάνω-κάτω και την άλλη για την κίνηση αριστερά-δεξιά . Μερικά από τα παιχνίδια που κυκλοφόρησαν εκείνη την εποχή ήταν τα εξής: Analogic, Baseball, Brain Wave, Cat & Mouse, Hockey, Tennis.

Εικόνα 2 : «Η κονσόλα Magnavox Odyssey»

Η δεύτερη γενιά ξεκίνησε το έτος 1976 όπου η Fairchild κυκλοφόρησε το Fairchild Video Entertainment System (VES) όπου είχε την ίδια λειτουργία με κτύπημα διακόπτη όπως το Odyssey. Το έτος 1980 η Atari κυκλοφόρησε μία παραλλαγή του Arcade, το Space Invaders που γνώρισε τεράστια επιτυχία. Πολλοί καταναλωτές αγόραζαν την κονσόλα Atari 2600 μόνο και μόνο για να παίξουν αυτό το παιχνίδι.

Εικόνα 3 : «Το παιχνίδι Space Invaders στην κονσόλα Atari 2600»

Ωστόσο το έτος 1983, η βιομηχανία των βιντεοπαιχνιδιών ήρθε αντιμέτωπη με νέες πιο φθηνές κονσόλες που παρείχαν παιχνίδια χαμηλής ποιότητας. Έτσι η Atari που ήταν μέχρι τότε ηγέτης στον χώρο, έχασε την αξιοπιστία της μιας και το κοινό στράφηκε προς τα Home Computers. Πολλές εταιρείες κήρυξαν πτώχευση ή μετακινήθηκαν σε άλλες βιομηχανικές δραστηριότητες.

Το 1983, η Nintendo κυκλοφορεί το Family Computer στην Ιαπωνία, που σηματοδοτεί την έναρξη της τρίτης γενιάς κονσόλων. Υποστήριζε sprites υψηλής ανάλυσης και το υπόβαθρο είχε μεγαλύτερη ποικιλία χρωμάτων πράγμα που καθιστούσε την κονσόλα αυτή ικανή να έχει καλύτερα και πιο λεπτομερή γραφικά. Το 1985 κάνει και την εμφάνισή του στην Αμερική με την μορφή του Nintendo Entertainment System (NES). Χρησιμοποιούσε μία μπροστινή υποδοχή για κασέτες και στην συσκευασία περιείχε το παιχνίδι Super Mario Brothers καταφέρνοντας να καθιερωθεί στην αγορά των Ηνωμένων Πολιτειών .

Εικόνα 4 : «Το παιχνίδι Super Mario Brothers (1985)»

Η τέταρτη γενιά βιντεοπαιχνιδιών ξεκινάει τον Οκτώβριο του 1988 όπου μία νέα εταιρεία, η Sega, κάνει το ντεμπούτο της λανσάροντας το Mega Drive/Genesis. Δύο χρόνια αργότερα η Nintendo κυκλοφορεί μία ανανεωμένη έκδοση της υπάρχουσας κονσόλας που παρασκευάζει, την Super Nintendo Entertainment System (SNES). Άλλες σημαντικές κονσόλες που ανήκουν κι αυτές με την σειρά τους στην 4η γενιά είναι το TurboGrafx-16 της NEC και η σειρά Neo Geo της SNK.

Εικόνα 5: «Η κονσόλα Super Nintendo»

Εικόνα 6: «Η κονσόλα TurboGrafx-16»

Μεγάλη καινοτομία αποτέλεσε το Game Boy της εταιρείας Nintendo. Το μικρό του μέγεθος και η δυνατότητα που παρείχε στον χρήστη να μπορεί να το μεταφέρει και να το χρησιμοποιεί και εκτός σπιτιού κατέστησαν αυτή την παιχνιδομηχανή ορόσημο για την εποχή της. Για πρώτη φορά μία κονσόλα δεν χρειαζόταν ρεύμα για να λειτουργήσει, αλλά μπαταρίες και το χειριστήριο καθώς και η οθόνη ήταν ενσωματωμένα πάνω στην συσκευή.

Μία από τις πρώτες κονσόλες της πέμπτης γενιάς, ήταν το Atari Jaguar, μία παιχνιδομηχανή αρκετά πιο ισχυρή από τους προκάτοχούς της. Είχε καλύτερη απόδοση πολυγώνων και αρκετά μεγαλύτερη χρωματική παλέτα, δεν μπόρεσε ποτέ όμως να ανταγωνιστεί την Nintendo και την Sega Drive. Για πολλούς η Πέμπτη γενιά ξεκίνησε με την κυκλοφορία των Sega Saturn, Sony PlayStation και Nintendo 64. Το Sega Saturn και το PlayStation χρησιμοποιούσαν CD, κάτι που ήταν πρωτοφανές για τα μέχρι τότε δεδομένα. Μερικά παιχνίδια ορόσημο αυτής της γενιάς ήταν το Super Mario 64, το Mario Kart, The Legend of Zelda, Donkey Kong και Tekken.

Το έτος 1999 η Sega , δεν είναι ικανοποιημένη από την πορεία του Saturn και ρίχνει στην αγορά το Dreamcast. Ένα χρόνο αργότερα η Sony βγάζει το PlayStation 2 που γίνεται η πιο εμπορική κονσόλα της έκτης γενιάς. Η Nintendo βγάζει το Game Cube ενώ για πρώτη φορά στον χώρο των ηλεκτρονικών παιχνιδιών , κάνει την εμφάνισή της η Microsoft με το Xbox. Η γενιά αυτή διαφέρει κατά πολύ από τις υπόλοιπες, γιατί πλέον έχει αποκτήσει έναν πολύ δυναμικό ανταγωνιστή, τους ηλεκτρονικούς υπολογιστές και έτσι η εξέλιξή τους αλλάζει κατά πολύ ως προς την δομή του λειτουργικού, τα μέσα αποθήκευσης αρχείων αλλά και την πρόσβασή τους στο διαδίκτυο.

Η έβδομη γενιά ξεκινάει το 2005, όταν η Microsoft λανσάρει το Xbox 360, η πρώτη κονσόλα με ασύρματα χειριστήρια και μια σαφώς καλύτερη απευθείας σύνδεση στο διαδίκτυο. Ακόμα, επιτρέπει στους χρήστες να εισάγουν μουσική από MP3 Player ή και άλλες πηγές. Ένα χρόνο αργότερα, η Sony κυκλοφορεί το PlayStation 3 που παρέχει όλα τα προνόμια του Xbox αλλά έχει και την δυνατότητα προβολής ταινιών. Το 2005 η Nintendo καινοτομεί για μία ακόμα φορά, λανσάροντας το Nintendo Wii, με ασύρματο χειριστήριο και ενσωματωμένο αισθητήρα κίνησης, που δίνει την επιλογή στο χρήστη να ελέγξει το παιχνίδι απλά κουνώντας και άκρα του.

Το 2011 κυκλοφορεί στην αγορά στο PlayStation Vita και το PlayStation 4 και δύο χρόνια αργότερα το Xbox One και το Nintendo Switch. Πλέον , διανύουμε την 9^{η} γενιά κονσόλων με το

PlayStation 5 να βγάνει σε κυκλοφορία κατά τα τέλη του 2019 μετά από αρκετά χρόνια αναμονής των θαυμαστών του. Δίνει την δυνατότητα στον χρήστη όπως και το PlayStation 4 αλλά και οι ηλεκτρονικοί υπολογιστές αντί να τροφοδοτούν τον χρήστη με τις πληροφορίες και τις εικόνες του παιχνιδιού μέσα από μια οθόνη, να τις δίνουν μέσα από γυαλιά εικονικής πραγματικότητας,(VR) κάνοντας την εμπειρία του παιχνιδιού πιο ζωντανή από ποτέ.

Εικόνα 7: «Oculus Rift S VR headset (Οπτικοακουστικό σετ εικονικής πραγματικότητας)»

Εικόνα 8: «Nintendo Wii με αισθητήρες κίνησης»

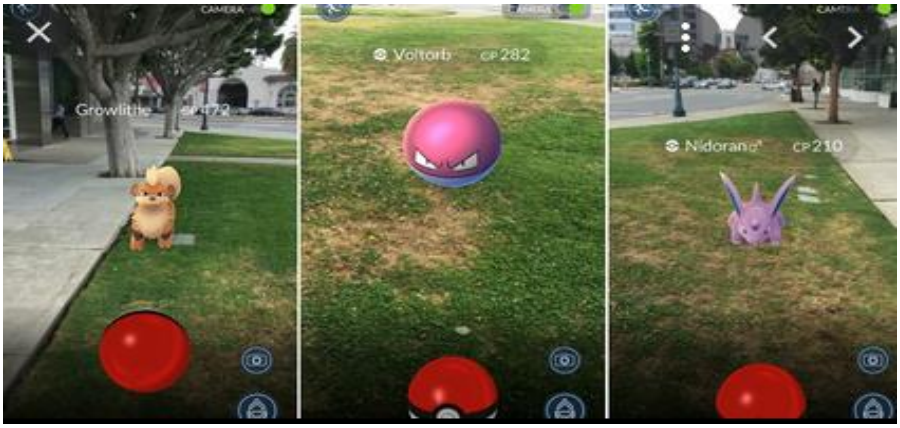
2.3 Τεχνολογίες που επηρέασαν την εξέλιξη των Videogames

Οι τεχνολογικές εξελίξεις συμβαίνουν με ραγδαίους ρυθμούς τα τελευταία χρόνια και έχουν διαμορφώσει νέα δεδομένα στον χώρο των βιντεοπαιχνιδιών. Οι παράγοντες που επηρέασαν αλλά και συνεχίσουν να επηρεάζουν την εξέλιξη των βιντεοπαιχνιδιών είναι οι παρακάτω:

- Η σύνδεση στο διαδίκτυο: Οι μεγάλες ταχύτητες κατέστησαν δυνατή την ταυτόχρονη συμμετοχή πολλών παικτών σε ένα βιντεοπαιχνίδι πράγμα που το καθιστά σαν ένα μέσο κοινωνικής δικτύωσης
- Η τεχνητή νοημοσύνη: μπορεί να παρατηρεί ελάχιστα ένα παιχνίδι, και να προβλέπει τις μελλοντικές κινήσεις ώστε να καταφέρει να δημιουργήσει τρόπους να μπορεί ο κάθε παίκτης να κερδίσει την κάθε πίστα. Έτσι τα παιχνίδια κατάφεραν να γίνουν πιο απρόβλεπτα με την προσθήκη καινούργιων μηχανισμών βελτιώνοντας κατά πολύ την εμπειρία του παιχνιδιού.
- Νέες τεχνολογίες ήχου: μπορούν να δημιουργήσουν την κατάλληλη ηχητική επένδυση. Οι βελτιωμένες τεχνολογίες του πολυκαναλικού ήχου (π.χ. Dolby Atmos) και τα αξεσουάρ Headset κάνουν τους ήχους ρεαλιστικούς και ατμοσφαιρικούς. Οι παίκτες μπορούν να

ενταχθούν πλήρως στο παιχνίδι, να απομονωθούν από τον πραγματικό κόσμο και αντιλαμβάνονται με την ακοή που βρίσκεται ο αντίπαλος ή και ο συμπαίκτης του.

- Φορητότητα: σε κονσόλες ή και έξυπνα κινητά ο καθένας μπορεί να κατεβάσει είτε επι πληρωμή είτε δωρεάν πολλά αξιόλογα παιχνίδια
- Η Επαυξημένη πραγματικότητα: μπορεί να κάνει δύσκολο στο να ξεχωρίσει ο παίκτης τα όρια του πραγματικού και του εικονικού. Πρωτοστάτης σε αυτή την τεχνολογία ήταν το Pokemon Go (2016) όπου εκατομμύρια παίκτες ξεχύθηκαν στους δρόμους με μία κινητή συσκευή.



Εικόνα 9: «Χρήση επαυξημένης πραγματικότητας στο Pokemon Go»

- 3D Κάμερες: μπορούν να σαρώνουν διαφορετικά σημεία του προσώπου ώστε να γίνονται πιο αληθοφανή τα συναισθήματα, οι αντιδράσεις του παίκτη, η κίνησή του και η ανατομία του.

Εικόνα 10: «Η εξέλιξη του χαρακτήρα Lara Croft (Google Pictures)»

- Virtual Reality Headset: (συσκευές εικονικής πραγματικότητας) έχουν καταφέρει να κάνουν τους παίκτες να ενσωματωθούν πλήρως στο περιβάλλον του παιχνιδιού και να βρίσκονται στο κέντρο της δράσης. Με αυτό τον τρόπο είναι πολύ δύσκολο για τον χρήστη να ξεχωρίσει τα όρια του πραγματικού από το ψηφιακό.

2.4 Κατηγορίες βιντεοπαιχνιδιών

Υπάρχουν πολλοί και διαφορετικοί τύποι βιντεοπαιχνιδιών όπου η κατηγοριοποίησή τους γίνεται συνήθως με βάση τα χαρακτηριστικά τους ή των υποκείμενων στόχων που καλείται να πιάσει ο χρήστης, και όχι από τον τύπο του παιχνιδιού τον οποίο περιέχουν. Η κάθε κατηγορία μπορεί επίσης να αποτελείται και από άλλες υποκατηγορίες. Αυτές είναι οι εξής:

1. Παιχνίδια Δράσης (Action Games)
2. Παιχνίδια Δράσης – Περιπέτειας (Action-Adventure Games)
3. Παιχνίδια Περιπέτειας (Adventure Games)
4. Παιχνίδια Ρόλων (Role-Playing Games)

5. Παιχνίδια Προσομοίωσης (Simulation Games)
6. Παιχνίδια Στρατηγικής (Strategy Games)
7. Παιχνίδια Αθλητισμού (Sports Games)
8. Παιχνίδι με γρίφους (Puzzle Games)
9. Παιχνίδια Αδράνειας (Idle Games)

1. Ως παιχνίδια δράσης, καλούμε αυτά στα οποία ο παίκτης έχει όλο τον έλεγχο και βρίσκεται στο επίκεντρο της δράσης. Καλείται να πετύχει στόχους, να ανταπεξέλθει σε προκλήσεις με σκοπό να ξεκλειδώσει επόμενες αποστολές, πάντα προσέχοντας την μπάρα ζωής του χαρακτήρα του. Σε αυτή την κατηγορία ανήκουν και τα παιχνίδια μάχης (Fighting games) καθώς και τα Shooting Games. Μερικά από τα πιο γνωστά και δημοφιλή παιχνίδια αυτής της κατηγορίας είναι το Mortal Kombat, Fortnite, Call of Duty, Halo, God of War και το Dishonored.

Εικόνα 11: «Mortal Combat»

Εικόνα 12: «Call of Duty»

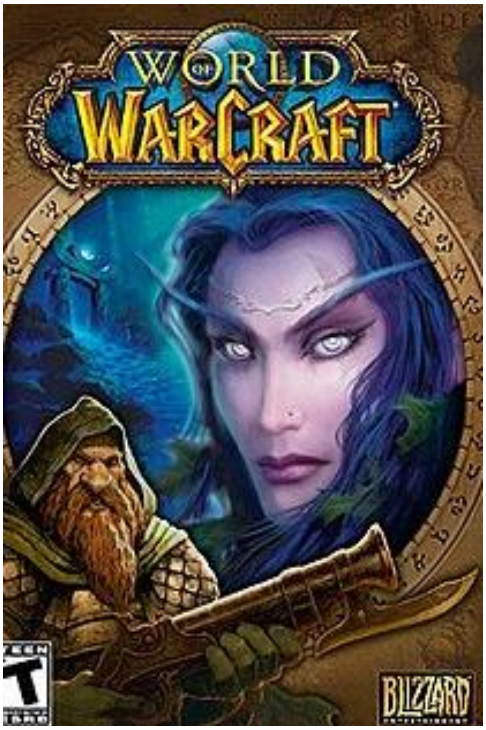
Εικόνα 13: «Halo»

2. Παιχνίδια Περιπέτειας - Δράσης είναι αυτά που απαιτούν μηχανισμούς και από τις δύο κατηγορίες με αποστολές, εμπόδια και στόχους που πρέπει να επιτύχει ο παίκτης. Τέτοια παιχνίδια είναι το The Legend of Zelda και το Link. Σε αυτή την κατηγορία ανήκουν και τα παιχνίδια επιβίωσης (Survival) όπως και τα παιχνίδια τρόμου (Horror games)

3. Τα παιχνίδια δράσης που ανήκουν σε αυτή την κατηγορία αφορούν στο στυλ παιχνιδιού και όχι την ιστορία ή το περιεχόμενο. Συνήθως ο παίκτης μπορεί να αντιδράσει με το περιβάλλον του ή και με άλλους παίκτες. Είναι μία κατηγορία που έχει εξελιχθεί ελάχιστα και δεν είναι ιδιαίτερα αρεστή από τους φίλους του gaming.
4. Η δεύτερη πιο δημοφιλής κατηγορία παιχνιδιών είναι τα παιχνίδια ρόλων (Role Playing). Είναι συνήθως παιχνίδια φαντασίας ή μεσαιωνικά παιχνίδια όπου ο παίκτης διαλέγει με ποιον χαρακτήρα θα παίξει και ποιες ικανότητες θα έχει ανάμεσα σε μία λίστα πιθανών επιλογών. Τέτοιου είδους παιχνίδια είναι κυρίως 3D όπως το Final Fantasy, World of Warcraft , Lineage που γνωρίζουν μεγάλη επιτυχία από την ημέρα κυκλοφορίας τους μέχρι και σήμερα.

Εικόνα 14 : «Lineage II»

Εικόνα 15 : «Final Fantasy»



Εικόνα 16 : «World of Warcraft»

5. Σαν παιχνίδια προσομοίωσης αναφέρονται όλα όσα ο παίκτης μπορεί να βιώσει πραγματικές καταστάσεις την καθημερινότητάς του. Σε αυτή την κατηγορία ανήκουν και πολλά εκπαιδευτικά παιχνίδια όπως προσομοίωση πτήσης για πιλότους ή χειρουργεία για ιατρούς. Το πιο γνωστό παιχνίδι σε αυτή την κατηγορία είναι το Sims.
6. Τα παιχνίδια στρατηγικής είναι αυτά που βασίζονται στα παραδοσιακά επιτραπέζια παιχνίδια στρατηγικής. Απαιτούν προσεκτικές και στρατηγικές κινήσεις ώστε να

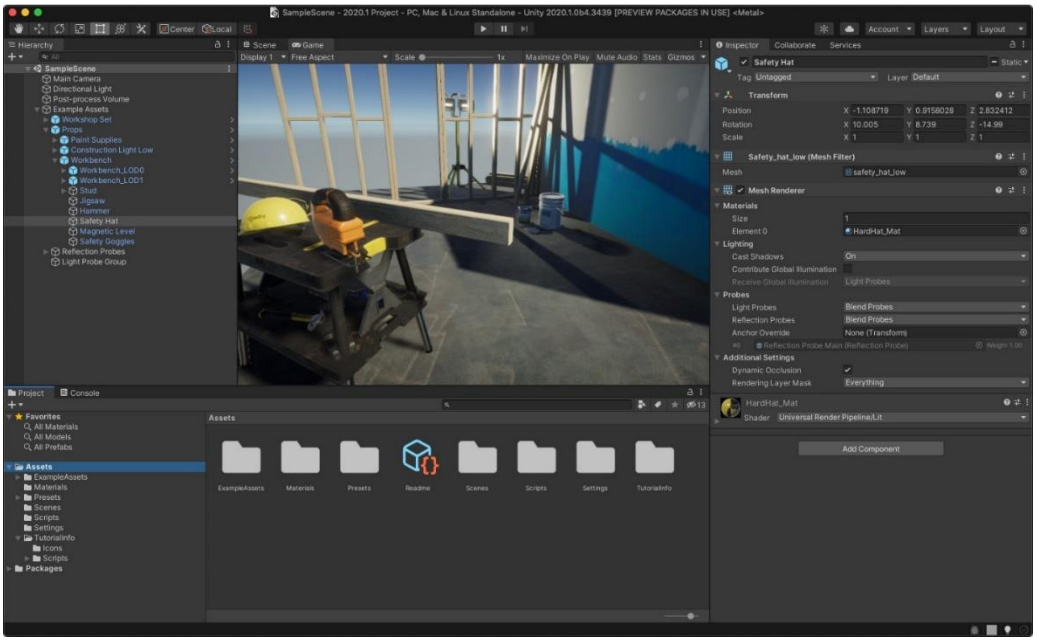
ανταποκριθεί ο χρήστης στις προκλήσεις. Άλλοτε παίζει ο κάθε παίκτης εναλλάξ και άλλοτε παίζουν και οι δύο ταυτόχρονα σε πραγματικό χρόνο. Τα καλύτερα παιχνίδια σε αυτό το είδος είναι το Age of Empires και το Command and Conquer.

7. Τα Sport Games είναι παιχνίδια που προσομοιώνουν αθλήματα όπως το ποδόσφαιρο, το μπάσκετ, το γκολφ ή και αγώνες ταχύτητας με αυτοκίνητα, φόρμουλα ή μηχανές.
8. Στην κατηγορία παιχνιδιών με γρίφους ή παιχνίδια λογικής ανήκουν όλα όσα απαιτούν από τον χρήστη να λύσει κάποιο πρόβλημα. Ένα από αυτά τα παιχνίδια είναι και το Tetris.
9. Παιχνίδια αδράνειας καλούμε αυτά τα οποία απαιτούν την λιγότερη ενασχόληση από τον χρήστη οποίος καλείτε ανά κάποιες ώρες να πετύχει έναν γρήγορο στόχο ώστε να ανταμειφθεί γι' αυτόν.

Κεφάλαιο 3 – Unity

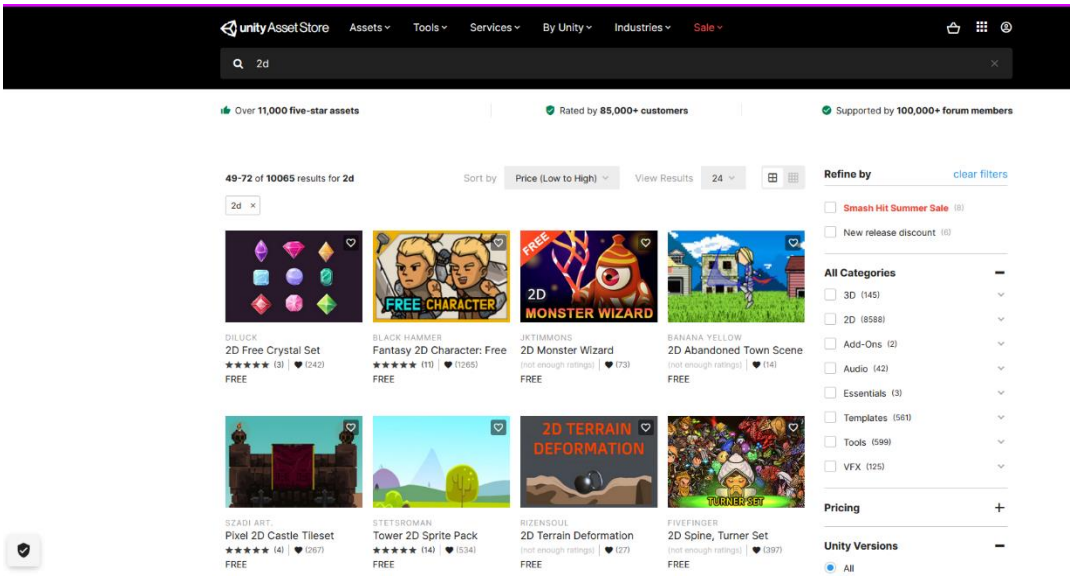
3.1 Περιγραφή του προγράμματος Unity

Η Unity είναι μια μηχανή-πλατφόρμα παιχνιδιών 3D/2D και VR (virtual reality) με ολοκληρωμένο περιβάλλον ανάπτυξης IDE (Integrated Development Environment) πολλαπλών πλατφόρμων (cross platform) για προγραμματιστές.Η Unity Software Inc είναι μια αμερικάνικη εταιρεία με έδρα το San Fransisco. Ιδρύθηκε στην Δανία το 2004 με ονομασία Over the Edge Entertainment και μετονομάστηκε έπειτα σε Unity το 2007. Ως μηχανή παιχνιδιών, η Unity είναι σε θέση να παρέχει πολλές από τις σημαντικές ενσωματωμένες λειτουργίες που χρειάζεται ένα παιχνίδι για να είναι πλήρως λειτουργικό. Ενσωματωμένες λειτουργίες όπως physics, 3D rendering και collision detection. Από την οπτική πλευρά του προγραμματιστή, αυτό σημαίνει ότι δεν χρειάζεται να ανακαλύψουμε ξανά τον τροχό και να ξεκινήσουμε από το μηδέν για την έναρξη ενός νέου project.



Εικόνα 17 : «Περιβάλλον Unity»

Αυτό που κάνει την Unity μια δυνατή μηχανή παιχνιδιών είναι πως περιέχει το “Asset Store”. Το ‘Asset Store’ είναι ένας ιστότοπος όπου προγραμματιστές και graphic designers (γραφίστες) μπορούν να ανεβάσουν (upload) τις δημιουργίες τους ώστε να είναι άμεσα διαθέσιμες στο κοινό (community). Για παράδειγμα ένα έτοιμο γραφικό 3D περιβάλλον ή ακόμα και sprites 2D (μοντέλα). Όλα αυτά σημαίνουν ότι ο προγραμματιστής του παιχνιδιού είναι ελεύθερος να επικεντρωθεί σε αυτό που έχει ουσιαστική σημασία στο να δημιουργήσει μια μοναδική και διασκεδαστική εμπειρία ενώ παράλληλα προγραμματίζει (coding) και δίνει ‘ζωή’ στο όραμα του.



Εικόνα 18 : «Unity Asset Store»

3.2 Τα πλεονεκτήματα της Unity

Τα πλεονεκτήματα της Unity παρουσιάζονται παρακάτω:

- ✓ Platform Support Flexible deployment

Η μηχανή της Unity προτιμάται ιδιαίτερα για την εκτεταμένη υποστήριξη της σε πολλαπλές πλατφόρμες. Η εφαρμογή που αναπτύχθηκε μέσω της Unity μπορεί πολύ εύκολα να τρέξει σε PC (ηλεκτρονικό υπολογιστή), στο διαδίκτυο, κονσόλες ακόμα και στο κινητό.

- ✓ Unity Environment easy to use

Το περιβάλλον δημιουργίας και ανάπτυξης είναι αρκετά εύκολο και προσιτό και μπορεί κανείς εύκολα να μάθει και να πειραματιστεί καθώς οι δυνατότητες που παρέχει είναι απεριόριστες.

- ✓ Documentation

Παρέχει ένα τεράστιο όγκο πληροφοριών για όλους τους προγραμματιστές από αρχάριους έως και επαγγελματίες περιλαμβάνοντας την λεπτομερή τεκμηρίωση και εξήγηση κάθε θέματος.

- ✓ Σχεδιασμός 3D & 2D

Παρέχει πολλά εργαλεία για κατασκευή παιχνιδιών τόσο σε 3D περιβάλλον όσο και σε 2D. Θεωρείται αν όχι η καλύτερη πλατφόρμα μία από τις καλύτερες πλατφόρμες

3.3 Τα μειονεκτήματα της Unity

Τα μειονεκτήματα της είναι τα εξής:

- × Old Graphics

Παρόλο που η Unity είναι μια πολύ καλή μηχανή παιχνιδιών λειτουργεί σε μια παλαιότερη μηχανή γραφικών όσον αναφορά την οπτικοποίηση σε σύγκριση με τους ανταγωνιστές της όπως για παράδειγμα η Unreal Engine. Αυτό έχει ως αποτέλεσμα καλλιτέχνες και game developers που θέλουν μεγάλη ποιότητα στα αντικείμενα τους να μην την επιλέγουν.

- × Memory Issues

Τα παιχνίδια που αναπτύσσονται στη Unity καταναλώνουν περισσότερη μνήμη, η οποία με την σειρά της δημιουργεί σφάλματα OOM (out of memory) και διάφορα προβλήματα εντοπισμού σφαλμάτων στις εφαρμογές.

- License Cost

Οι προγραμματιστές της χρειάζονται ειδική άδεια για τα την καλύτερη απόδοση του προγράμματος και για καλύτερα γραφικά η οποία είναι αρκετά ακριβή στην αγορά της.

- Terrain engine and Movie Textures

Ο μηχανισμός εδάφους και τα movie textures δεν είναι αρκετά καλά και χρειάζονται πολύ περισσότερη προσπάθεια συγκριτικά με τους ανταγωνιστές της όπως Unreal Engine.

- Lags on world editing

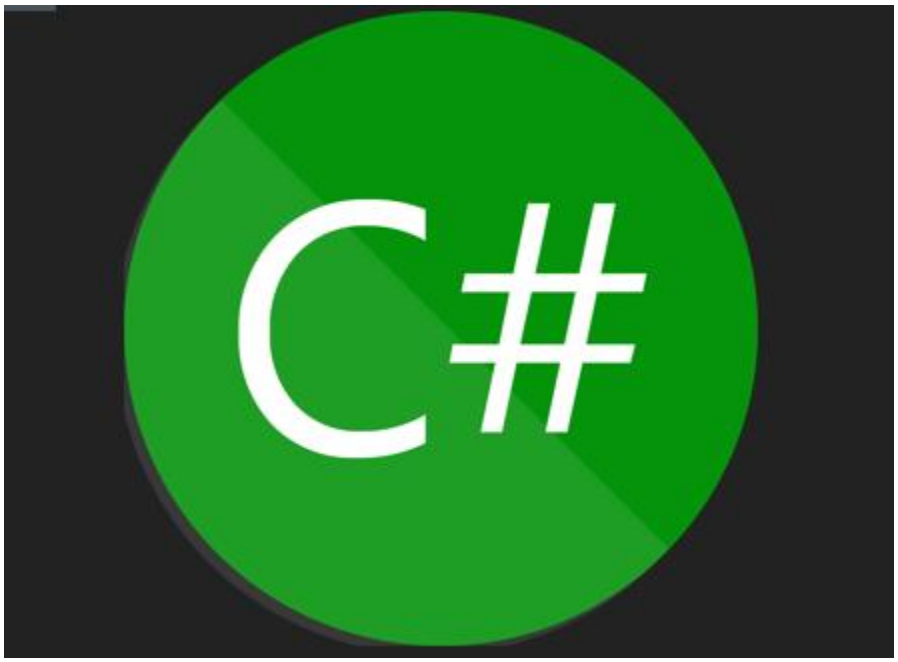
Μερικές φορές το πρόγραμμα της Unity δεν ανταποκρίνεται πολύ καλά στην κατασκευή και δημιουργία ενός πολύ μεγάλου ανοιχτού κόσμου καθώς αντιμετωπίζει προβλήματα rendering στο terrain, δηλαδή απαιτεί πολύ χρόνο μέχρι να φορτώσουν όλα τα γραφικά στοιχεία του παιχνιδιού.

Κεφάλαιο 4 – Εισαγωγή στην C# και ο ρόλος της στην Unity

4.1 Εισαγωγή στην C#

Η C# είναι μια απλή μοντέρνα αντικειμενοστραφής και ασφαλής τύπου γλώσσα προγραμματισμού. Η C# έχει τις ρίζες της στην οικογένεια γλωσσών της C με αποτέλεσμα να είναι αρκετά οικείες με την C, C++ και την Java. Η C# δημιουργήθηκε και αναπτύχθηκε το 2000 από την Microsoft ως μέρος της .NET initiative και αργότερα εγκρίθηκε ως διεθνές πρότυπο του

ECMA (European Computer Manufacturers Association) το 2002 και της ISO (International Organization for Standardization) το 2003. Ο διοικητής της ομάδα που διαχειρίζεται την γλώσσα ονομάζεται Άντερς Χάιλσμπεργκ. Στις 15 Αυγούστου 2012 κυκλοφόρησε η έκδοση 5.0 η οποία είναι η πιο πρόσφατη μέχρι σήμερα.



Εικόνα 19 : «Unity Logo»

4.2 C# σε Unity Engine

Η Unity υποστηρίζει την γλώσσα προγραμματισμού C# και μάλιστα θεωρείται η καλύτερη για την ανάπτυξη βιντεοπαιχνιδιων. Έτσι λοιπόν με την βοήθεια της C# ο χρήστης δημιουργεί ένα σύνολο από scripts τα οποία έχουν την δυνατότητα να αλληλοεπιδρούν μεταξύ τους. Αυτό έχει ως αποτέλεσμα να προγραμματίζει ο χρήστης μια σειρά από εντολές που εκτελούνται όταν και όποτε αυτές καλεστούν. Ουσιαστικά τα scripts λένε στα GameObjects να συμπεριφερθούν με όποιον τρόπο ορίσει ο χρήστης.

Ενα script είναι της μορφής:

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

public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Κατά την εισαγωγή ενός script δημιουργούνται αυτόματα δύο συναρτήσεις όπως φαίνεται πάνω η `void Start()` και η `void Update()` καθώς και τρεις βιβλιοθήκες.

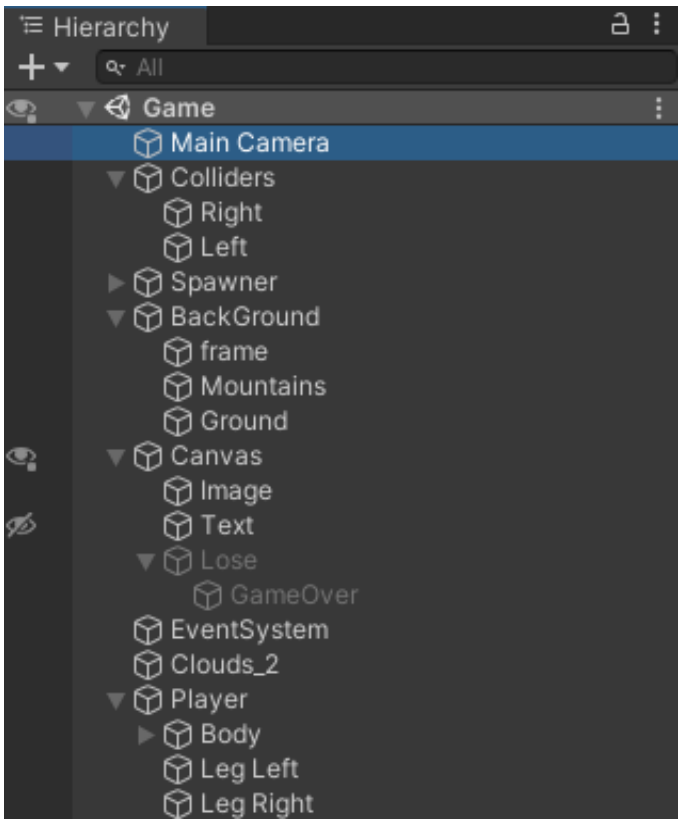
Η συνάρτηση `void Start()` καλείται μόνο μία φορά κατά την εκκίνηση του script όταν δηλαδή καλείται για πρώτη φορά.

Η συνάρτηση `void Update()` καλείται συνέχεια για κάθε frame per second (εικόνα το δευτερόλεπτο) και εκεί βρίσκεται το μεγαλύτερο ποσοστό του κώδικα καθώς εκεί αλληλοεπιδρούν σε πραγματικό χρόνο τα αντικείμενα μεταξύ τους, όσο το παιχνίδι είναι σε λειτουργία.

Κεφάλαιο 5 - Σχεδιασμός και ανάπτυξη του video game

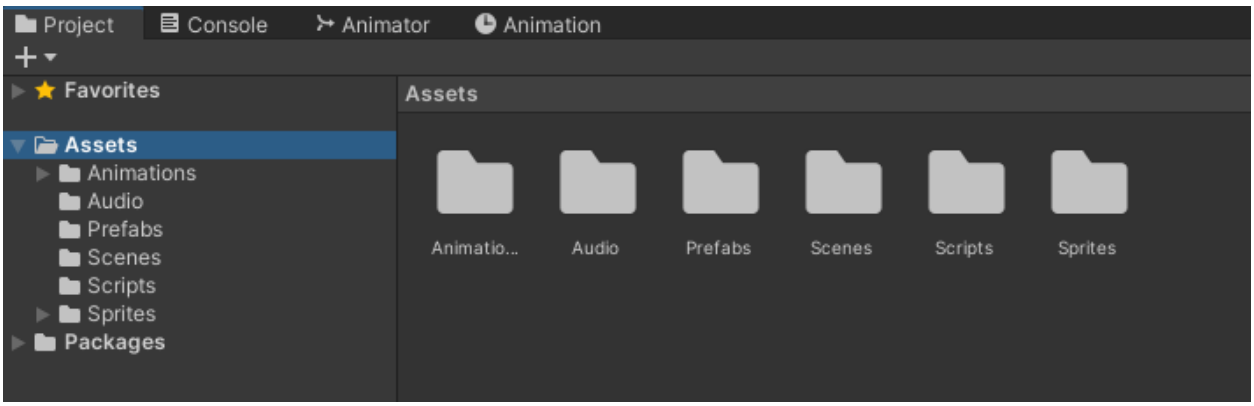
5.1 Εργαλεία και Interface of Unity

Καθώς έχει γίνει ήδη η εγκατάσταση της Unity ανοίγουμε το πρόγραμμα επιλέγοντας 2D project. Με αυτό τον τρόπο θα γίνουνε import κατάλληλα templates και εργαλεία που θα βοηθήσουν στον σχεδιασμό του παιχνιδιού. Το interface της Unity αποτελείται από πολλά και διάφορα εργαλεία-παράθυρα(tools-windows) τα οποία είναι αναγκαία για την σύνθεση και ανάπτυξη του παιχνιδιού. Ένα πολύ σημαντικό εργαλείο που βοηθάει στην διαχείριση των αντικειμένων που βρίσκονται πάνω στην σκηνή (scene*) είναι το **Hierarchy**. Το συγκεκριμένο εργαλείο απεικονίζει το σύνολο των αντικειμένων που έχουν ήδη εισαχθεί για παράδειγμα sprites, backgrounds, ήχοι, events, colliders και πολλά ακόμα.



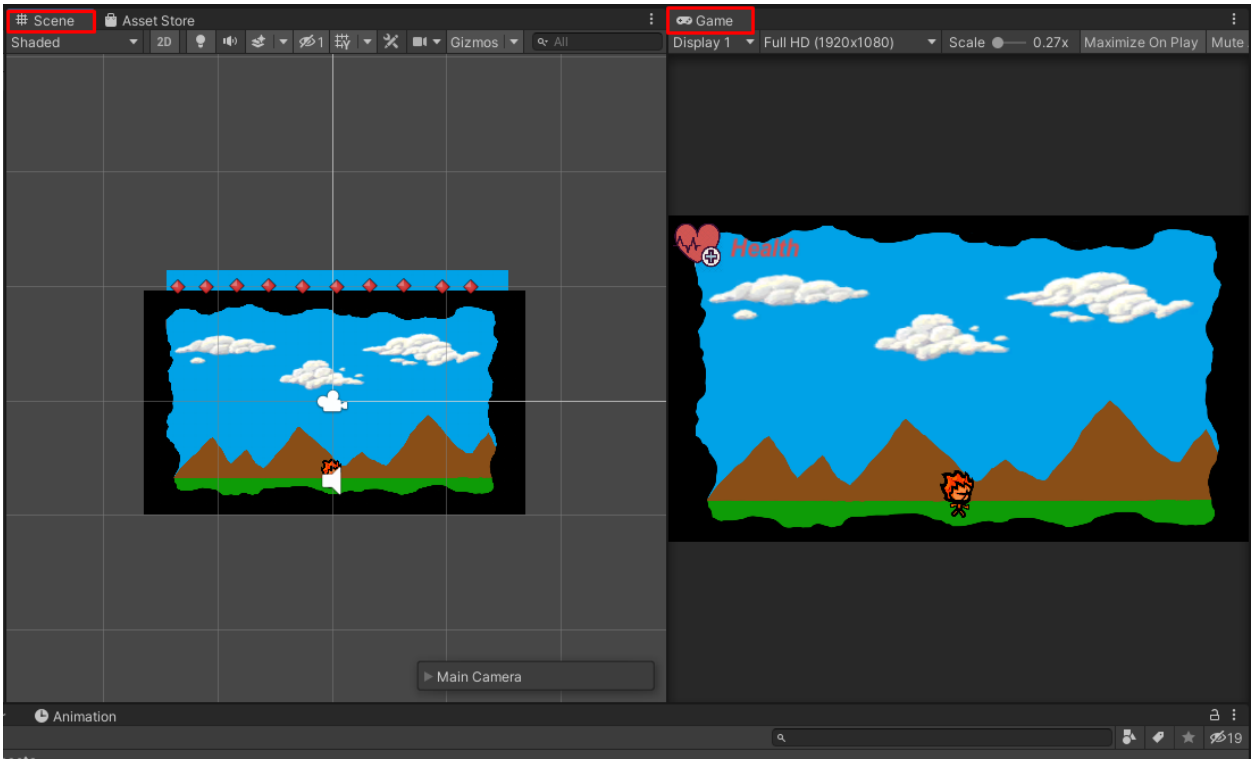
Εικόνα 20 : «Hierarchy Window»

Το μέρος που αποθηκεύονται όλα τα assets για να χρησιμοποιηθούν αργότερα στην ανάπτυξη του παιχνιδιού ονομάζεται project window. Στο συγκεκριμένο παράθυρο ο χρήστης δημιουργεί φακέλους και εισάγει γραφικά/ήχους/εικόνες/scripts και άλλα.



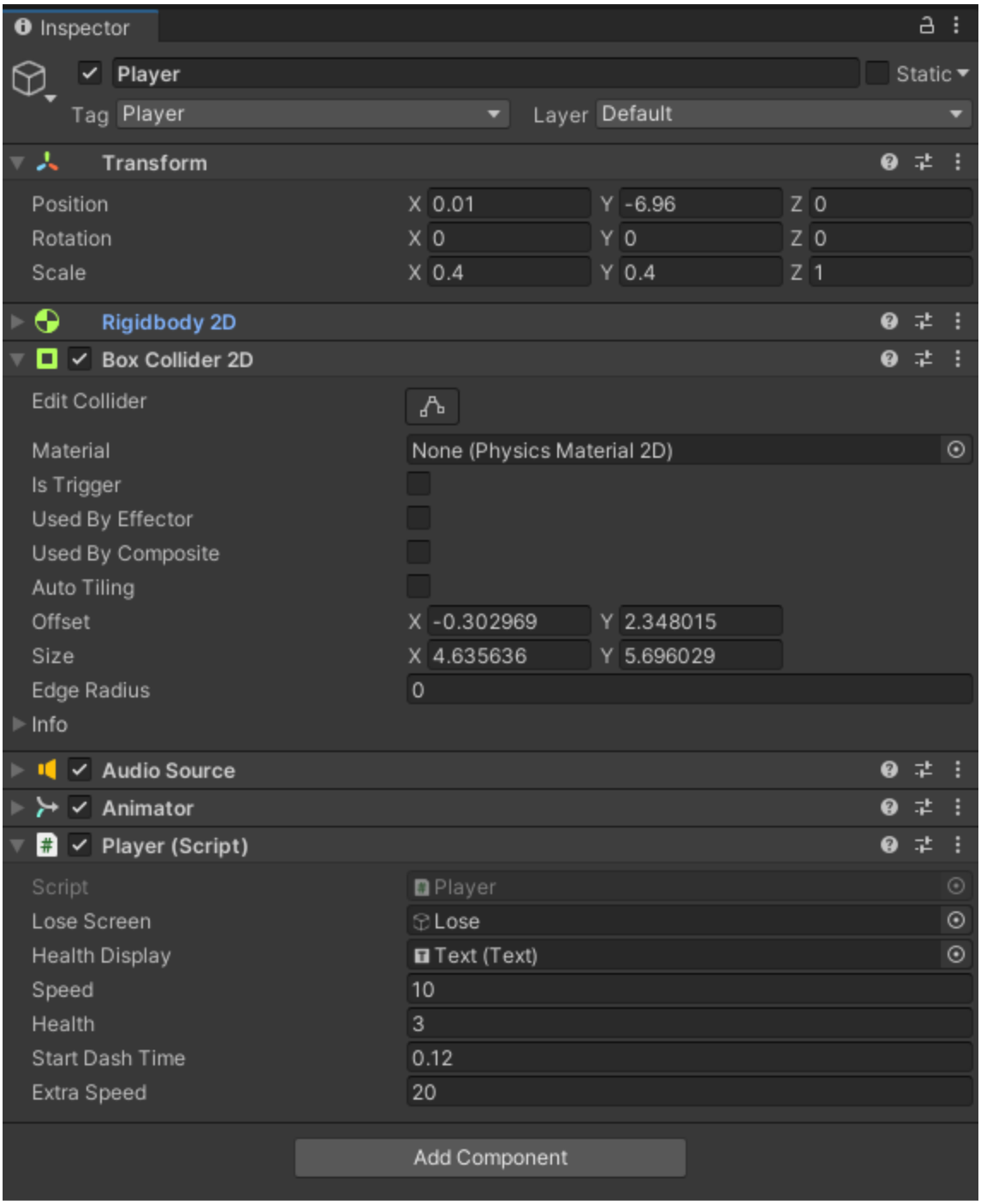
Εικόνα 21 : «Project Window»

Για να γίνει η οπτικοποίηση των εισαχθέντων αρχείων στο παιχνίδι εδώ έρχεται το **Scene window** και το **Game Preview**. Τα συγκεκριμένα παράθυρα είναι ο άξονας που ο προγραμματιστής ή ο καλλιτέχνης δουλεύει και φέρνει σε 'ζωή' τα γραφικά/αντικείμενα του. Ο χώρος είναι κενός μόλις ξεκινήσει ένα project καθώς περιέχει μόνο την κάμερα η οποία καθορίζει τι θα μας εμφανίσει και είναι χωρισμένος σε άξονες για να βοηθήσει τον χρήστη στην εισαγωγή αντικειμένων. Το Scene window δεν είναι μοναδικό καθώς ο χρήστης μπορεί να δημιουργήσει όσα scenes θέλει. Αυτό συνήθως γίνεται για να δώσει νέα levels, pause menu η ένα εισαγωγικό μενού πριν ξεκινήσει το παιχνίδι. Στο συγκεκριμένο παιχνίδι έχουν δημιουργηθεί δυο scenes, ένα αρχικό main menu με την επιλογή play και ένα δεύτερο που ξεκινάει το παιχνιδι. Το Game Preview είναι η τελική απεικόνιση που θα τρέξει το παιχνίδι.



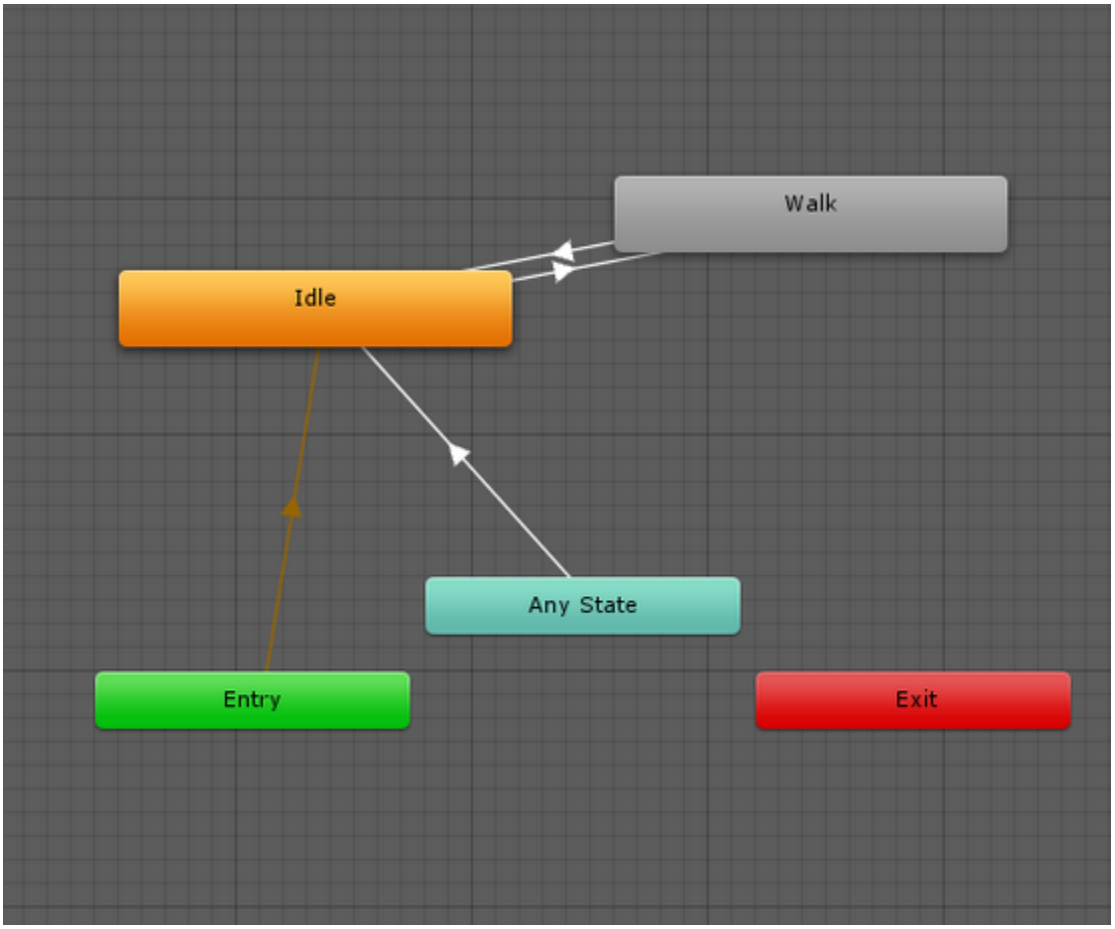
Εικόνα 22 : «Scene Manager, Game Preview»

Το παράθυρο το οποίο δείχνει όλα τα components του προεπιλεγμένου αντικειμένου είναι ο **Inspector**. Εδώ εισάγουμε στοιχεία και ιδιότητες που θέλουμε να ενσωματώσουμε στο εκάστοτε αντικείμενο. Ο Inspector εμφανίζει όλες τις λεπτομέρειες/πληροφορίες σχετικά με το επιλεγμένο GameObject (αντικείμενο) συμπεριλαμβανομένων όλων των συνημμένων στοιχείων και των ιδιοτήτων τους.



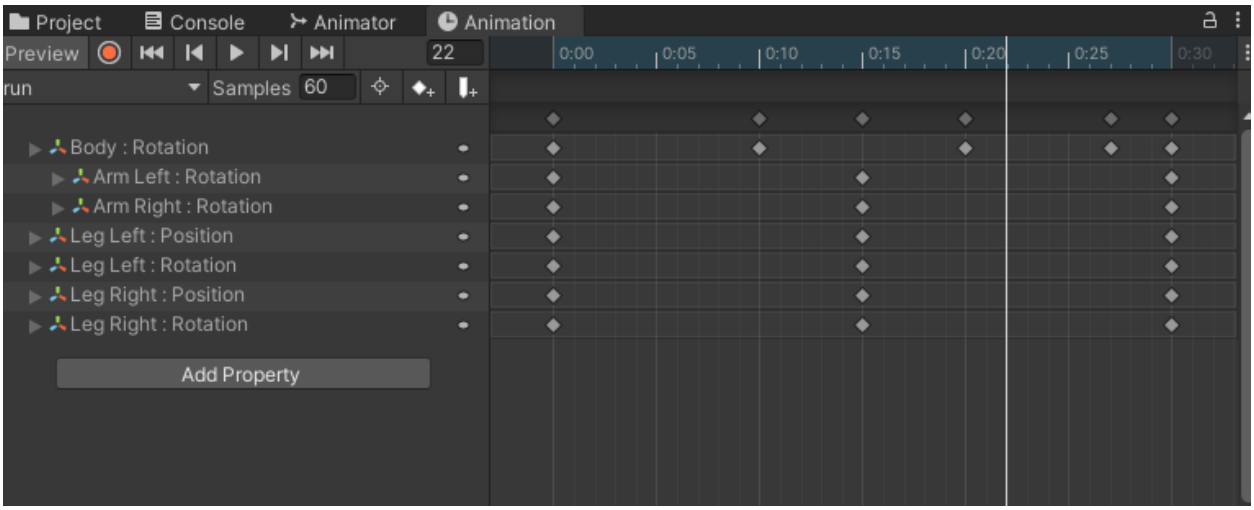
Εικόνα 23 : «Inspector window»

Το εργαλείο που περιγράφει καταστάσεις και η μετάβαση (transitions) από την μια στην άλλη λέγεται **Animator.**



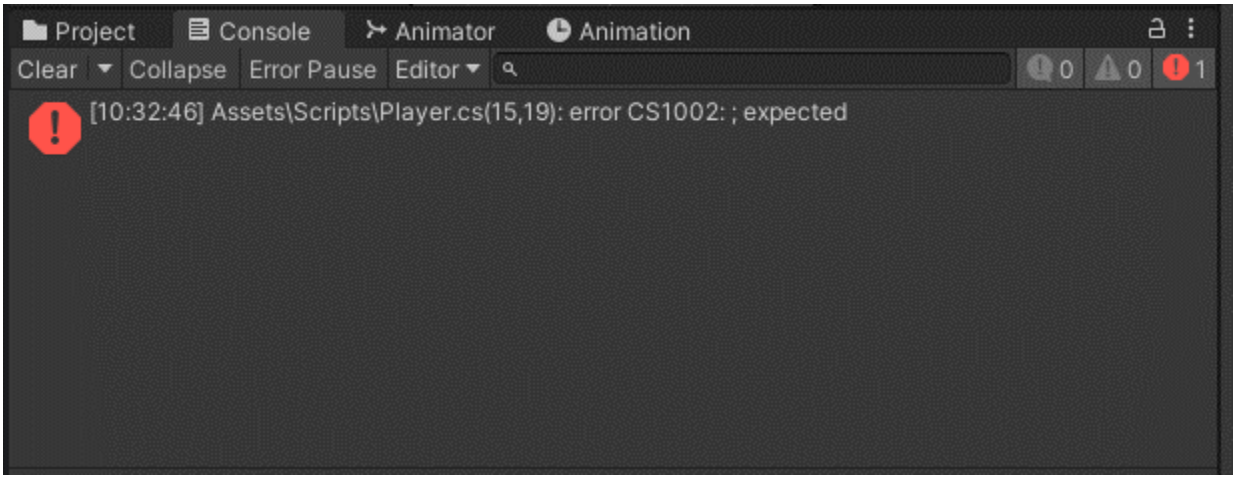
Εικόνα 24 : «Σχεδιασμός κατάστασης παίκτη στο Animator»

Με βάση τον Animator που περιγράφει καταστάσεις, το **Animation** tool κατασκευάζει και επεξεργάζεται τα animations (κινήσεις) καρέ καρέ ενός αντικειμένου. Εδώ ο χρήστης αφιερώνει αρκετό χρόνο όπως για παράδειγμα μια κίνηση για ένα sprite καθώς τρέχει, πηδάει ή ακόμα και όταν είναι ακίνητο.



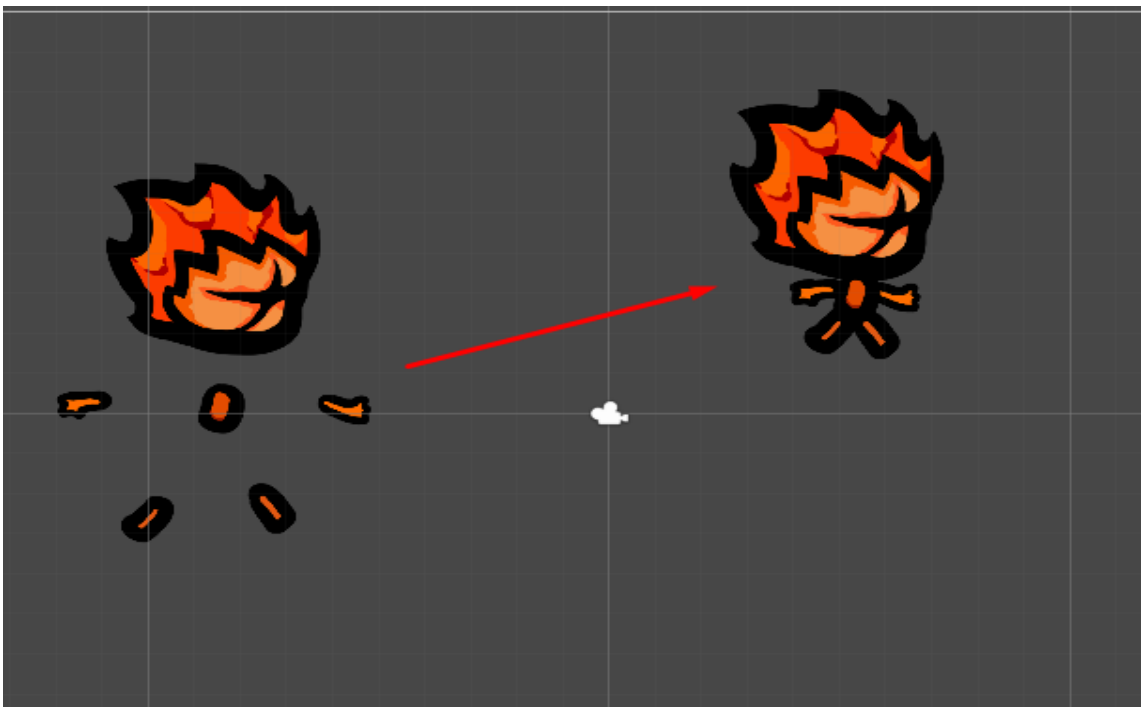
Εικόνα 25 : «Animation window»

Βοηθητικό εργαλείο κατά την υλοποίηση του video game είναι ο **Debugger-Console.** Το συγκεκριμένο παράθυρο είναι κυρίως για τα scripts τα οποία έχουμε δημιουργήσει στο παιχνίδι. Εδώ ο χρήστης εντοπίζει όλα τα errors και warnings που έχουν προκύψει, μπορεί επίσης να δει ποιο script και σε ποια σειρά υπάρχει λάθος ή πρόβλημα στον κώδικα C#.



Εικόνα 26 : «Console window»

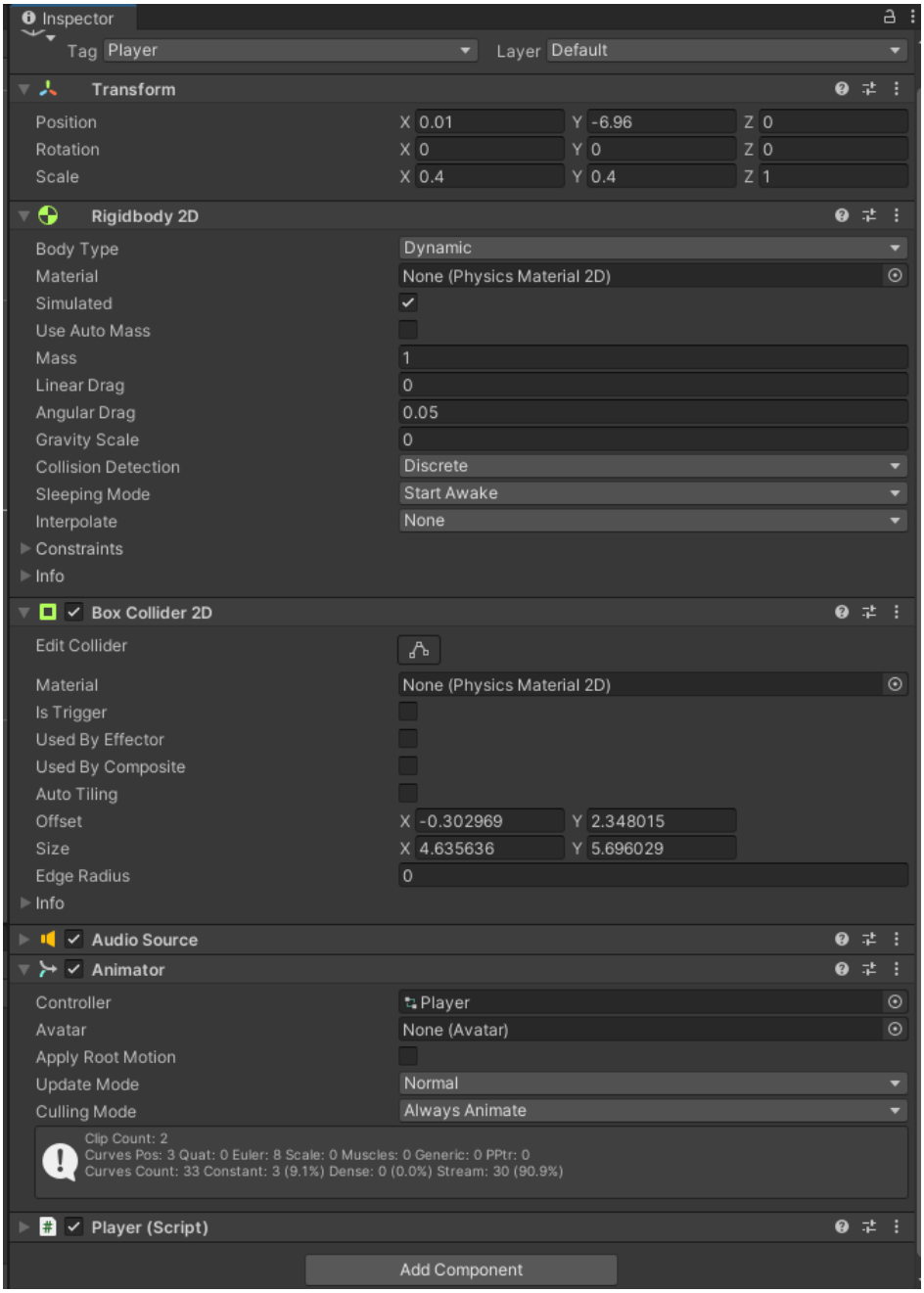
Στο αρχικό στάδιο του σχεδιασμού και ανάπτυξης του παιχνιδιού, πρώτος στόχος ήταν η δημιουργία του παίκτη εφόσον θα είναι ο κύριος πρωταγωνιστής του παιχνιδιού. Στο σημείο αυτό έγινε η πλήρης σύνθεση του παίκτη (κεφάλι, σώμα, αριστερό δεξί πόδι και χέρι).



Εικόνα 27 : «Σύνθεση του παίκτη»

Κατά την διάρκεια της σύνθεσης των assets χρειάστηκε να οριστεί ο κεντρικός άξονας ξεχωριστά του κάθε σημείου του σώματος του παίκτη. Δηλαδή με ποιον τρόπο θα περιστρέφει το κεφάλι ο παίχτης, που στην περίπτωση αυτή θέλουμε να περιστρέφεται το κεφάλι από τον λαιμό και όχι από το κέντρο του κεφαλιού του. Με την ίδια φιλοσοφία έγιναν τα χέρια το σώμα και τέλος τα πόδια. Έπειτα στο παράθυρο Hierarchy δημιουργήθηκε ένα νέο object (αντικείμενο) ώστε να συμπεριλαμβάνει όλα τα σημεία του σώματος σαν ένα ενιαίο αντικείμενο. Αυτή η μέθοδος ονομάζεται parenting, το νέο object γίνεται ‘γονέας’ των άλλων game objects. Όταν για παράδειγμα ένα game object έχει γονέα, θα εκτελέσει όλες τις αλλαγές μετατροπής του σε σχέση με ένα άλλο GameObject αντί για τον κόσμο του παιχνιδιού.

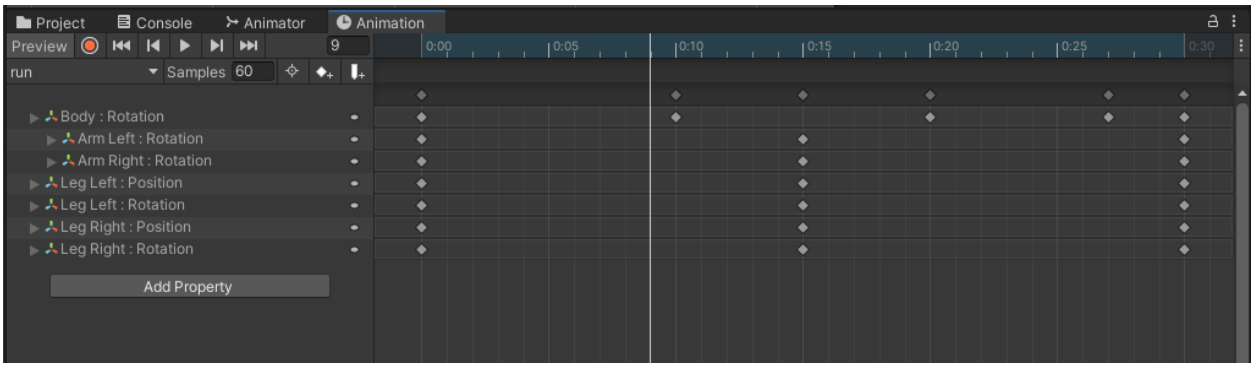
Στην συνέχεια στο παράθυρο του *Inspector* δόθηκαν τα απαραίτητα στοιχεία και ιδιότητες(components) για τον παίκτη που μόλις δημιουργήθηκε. Αρχικά επιλέχθηκαν οι σωστές διαστάσεις για την αναπαράσταση του παίκτη και επίσης η θέση του σε άξονες (x,y). Στο σημείο αυτό δόθηκε το πρώτο στοιχείο/ιδιότητα στον παίκτη κάνοντας add component και προστέθηκε το **Rigidbody2D.** Το συγκεκριμένο component συμβολίζει την φυσική υπόσταση του παίκτη. Ουσιαστικά του δίνει μάζα ακολουθεί μια σειρά κινήσεων βάση φυσικών νόμων. Έπειτα προσθέσαμε ένα νέο component το **Box Collider 2D**. Το συγκεκριμένο component δίνει την δυνατότητα να ορίσουμε τον χώρο τον οποίο ο παίκτης καταλαμβάνει στον παιχνίδι και είναι πλήρως customizable. Στο επόμενο βήμα προσθήκης νέων components στον παίκτη δημιουργήθηκαν το **Audio Source** και το **Animator**. Το Audio Source χρειάστηκε για να αναπαράγει ήχο ο παίχτης και το Animator, για τις κινήσεις(animations) του παίκτη που θα κατασκευαστούν σε δεύτερο χρόνο. Τέλος προστέθηκε ένα τελευταίο component το **Script**. Το script είναι το αρχείο που περιέχει κώδικα C# που ορίζει την συμπεριφορά του.



Εικόνα 28 : «Προσθήκη Components στον παίκτη»

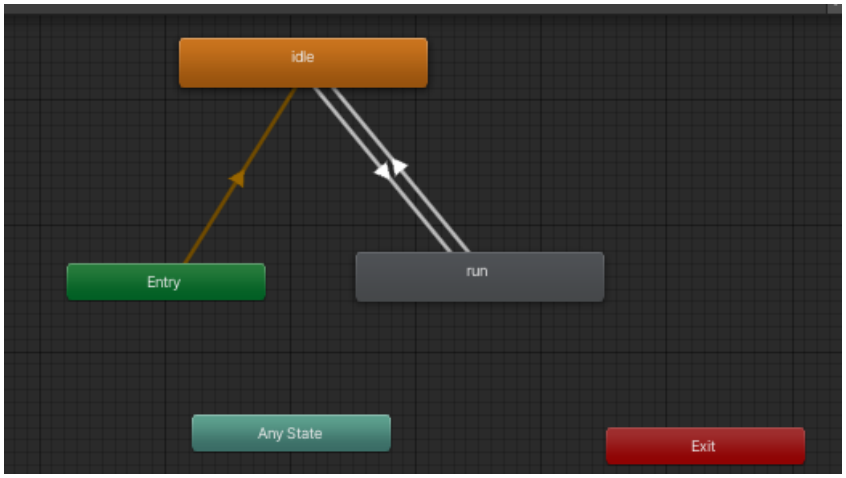
5.2 Animations και Transitions

Στο σημείο αυτό δόθηκε κίνηση στον παίκτη με το Animation Window. Προστέθηκαν δυο κινήσεις για τις ανάγκες του παιχνιδιού δύο παράμετροι συνθηκών που διαχειρίζονται μέσω των scripts C#. Η πρώτη κίνηση είναι idle δηλαδή όταν ο παίκτης είναι αδρανής και η δεύτερη running όταν δηλαδή ο παίκτης τρέχει. Σχεδιάστηκε λεπτομερώς στο παράθυρο αυτό σε συνάρτηση με τον χρόνο καρέ καρέ (frame per second) μία ομαλή κίνηση του παίχτη όταν τρέχει στον χώρο και μία στάσιμη όταν βρίσκεται αδρανής.



Εικόνα 29 : «Κατασκευή ομαλών κινήσεων σε συνάρτηση με το χρόνο για τον παίκτη»

Στο Animator Window δημιουργήθηκαν οι δυο μεταβολές(transitions) που μπορεί να υποβληθεί ο παίκτης run, idle. Εισάχθηκε μια Boolean παράμετρος (isRunning)* που με την βοήθεια της C# αργότερα θα αντιλαμβάνεται animator την μεταβολή της κίνησης από την μια κατάσταση στην άλλη.



Εικόνα 30 : «Διάγραμμα μετάβασης κατάσταση του παίκτη»

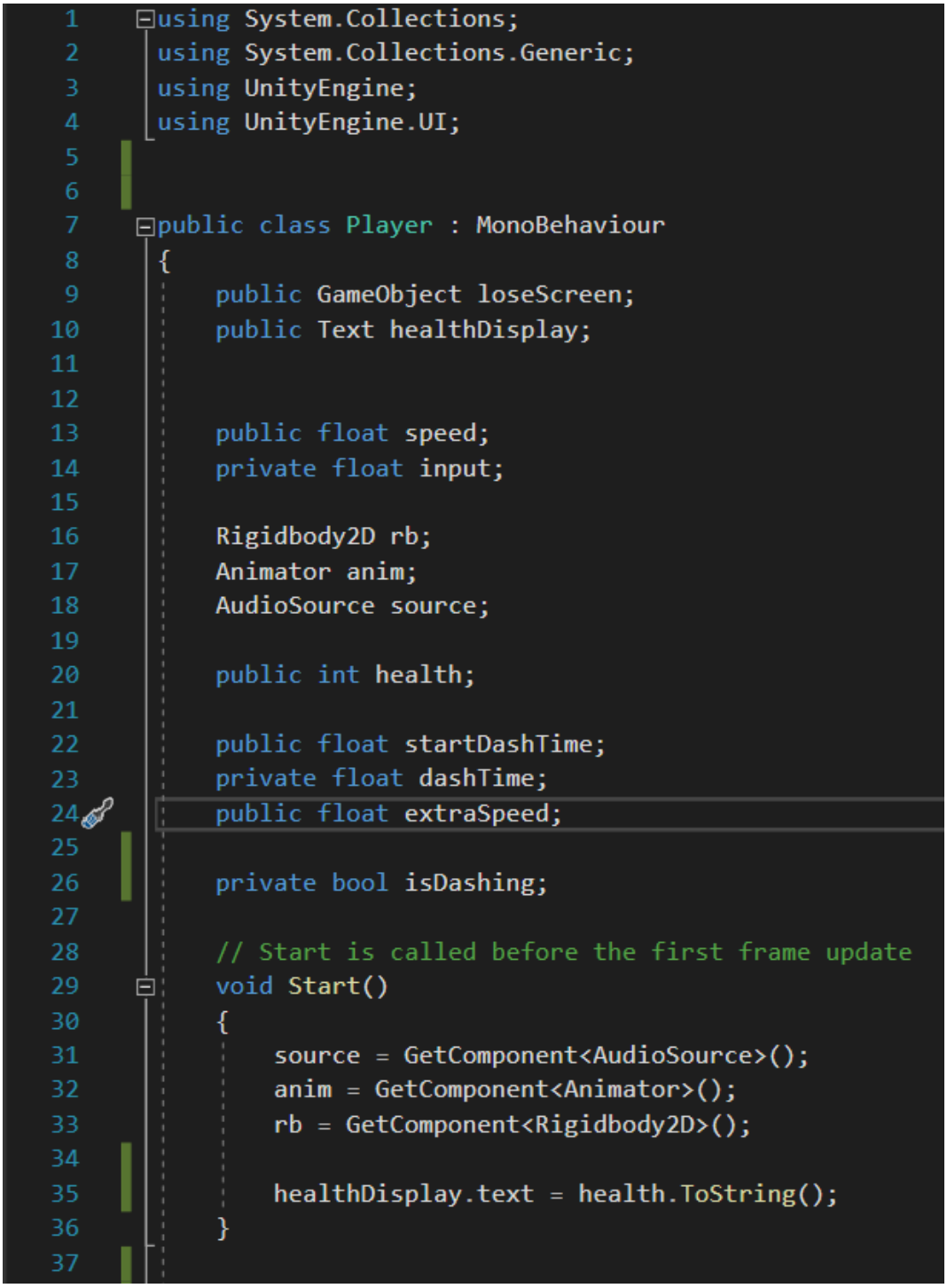
5.3 Script του παίκτη

Με την υλοποίηση για το script του παίκτη, όλα όσα αναπτύχθηκαν παραπάνω θα είναι πλήρως λειτουργικά.

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


public class Player : MonoBehaviour
{
    public GameObject loseScreen;
    public Text healthDisplay;


    public float speed;
    private float input;

    Rigidbody2D rb;
    Animator anim;
    AudioSource source;

    public int health;

    public float startDashTime;
```

```
private float dashTime;
public float extraSpeed;

private bool isDashing;

// Start is called before the first frame update
void Start()
{
    source = GetComponent<AudioSource>();
    anim = GetComponent<Animator>();
    rb = GetComponent<Rigidbody2D>();

    healthDisplay.text = health.ToString();
}

private void Update()

{
    if (input != 0) {
        anim.SetBool("isRunning", true);
    }
    else
    {
        anim.SetBool("isRunning", false);
    }
    if (input > 0)
    {
        transform.eulerAngles = new Vector3(0, 0, 0);
   }
   else if (input < 0)
   {
      transform.eulerAngles = new Vector3(0, 180, 0);
    }

if (Input.GetKeyDown(KeyCode.Space) && isDashing == false)
    {
        speed += extraSpeed;
        isDashing = true;
        dashTime = startDashTime;
    }
    if (dashTime <= 0 && isDashing == true)
    {
        isDashing = false;
        speed -= extraSpeed;
    }
    else
    {
        dashTime -= Time.deltaTime;
    }
}

// Update is called once per frame
void FixedUpdate()
{
    //Storing player's input
```

```
        input = Input.GetAxisRaw("Horizontal");

        //How to move the player
        rb.velocity = new Vector2(input * speed, rb.velocity.y);
    }
    public void TakeDamage(int damagegAmount) {
        source.Play();
        health -= damagegAmount;
        healthDisplay.text = health.ToString();
        if (health <= 0) {
            loseScreen.SetActive(true);
            Destroy(gameObject);
        }
    }
}
```



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


public class Player : MonoBehaviour
{
    public GameObject loseScreen;
    public Text healthDisplay;


    public float speed;
    private float input;

    Rigidbody2D rb;
    Animator anim;
    AudioSource source;

    public int health;

    public float startDashTime;
    private float dashTime;
    public float extraSpeed;

    private bool isDashing;

    // Start is called before the first frame update
    void Start()
    {
        source = GetComponent<AudioSource>();
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody2D>();

        healthDisplay.text = health.ToString();
    }

```

Εικόνα 31 : «Script του παίκτη 1/3»

Γραμμές 1-4 : Είναι οι βασικές βιβλιοθήκες της Unity καθώς έχουμε προσθέσει στην τέταρτη γραμμή μία ακόμη που θα βοηθήσει για το UI του παιχνιδιού.

Γραμμή 7 : Εδώ γίνεται η δήλωση των μεταβλητών που θα χρησιμοποιηθούν στο script.

Γραμμές 8-27 : Δήλωση μεταβλητών της κλάσης

Γραμμή 31 : Χρησιμοποιεί το στοιχείο AudioSource

Γραμμή 32: Χρησιμοποιείται το στοιχείο Animator και το αποθηκεύει στην μεταβλητή anim

Γραμμή 33 : Χρησιμοποιείται το στοιχείο Rigidbody2D και το αντιστοιχεί στην μεταβλητή rb

Γραμμή 35 :Απεικονίζει την ζωή του παίχτη κατά την εκκίνηση του παιχνιδιού.

```
38      private void Update()
39      
40      {
41          if (input != 0) {
42              anim.SetBool("isRunning", true);
43          }
44          else
45          {
46              anim.SetBool("isRunning", false);
47          }
48          if (input > 0)
49          {
50              transform.eulerAngles = new Vector3(0, 0, 0);
51          }
52          else if (input < 0)
53          {
54              transform.eulerAngles = new Vector3(0, 180, 0);
55          }
56      
57      if (Input.GetKeyDown(KeyCode.Space) && isDashing == false)
58          {
59              speed += extraSpeed;
60              isDashing = true;
61              dashTime = startDashTime;
62          }
63          if (dashTime <= 0 && isDashing == true)
64          {
65              isDashing = false;
66              speed -= extraSpeed;
67          }
68          else
69          {
70              dashTime -= Time.deltaTime;
71          }
72      }
```

Εικόνα 32 : «Script του παίκτη 2/3»

Γραμμή 40-47: Είναι η συνθήκη που ορίζει πότε ο παίκτης τρέχει ή όχι ώστε ο δώσει το σωστό animation.

Γραμμή 48-56: Συνθήκη όπου ορίζει την κίνηση του παίκτη με το ανάλογο input από τον χρήστη. Ουσιαστικά αυτό που κάνει είναι να περιστρέφει στον άξονα y τον παίκτη κατά 180 μοίρες ώστε να κοιτάζει στην κατεύθυνση στην οποία του δίνει σαν εντολή ο χρήστης.

Γραμμή 57-72: Εδώ δόθηκε στον παίκτη με το πάτημα του κουμπιού space μία μικρή επιτάχυνση προς τα μπροστά όπου στην συνέχεια αφαιρείται και επανέρχεται στην αρχική του ταχύτητα.

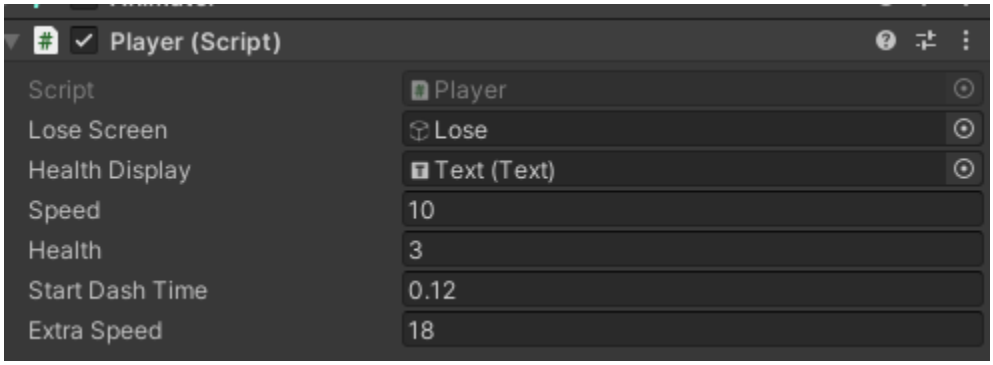
```
74      // Update is called once per frame
75      void FixedUpdate()
76      {
77          //Storing player's input
78          input = Input.GetAxisRaw("Horizontal");
79
80          //How to move the player
81          rb.velocity = new Vector2(input * speed, rb.velocity.y);
82      }
83      public void TakeDamage(int damagegAmount) {
84          source.Play();
85          health -= damagegAmount;
86          healthDisplay.text = health.ToString();
87          if (health <= 0) {
88              loseScreen.SetActive(true);
89              Destroy(gameObject);
90          }
91      }
92  }
93
```

Εικόνα 33 : «Script του παίκτη 3/3»

Γραμμή 76-81: Ορίστηκε η πραγματική κίνηση του παίκτη. Ο Vector δηλώνει διάνυσμα με κατεύθυνση κίνησης. Το input παίρνει τιμές -1 για κίνηση αριστερά και 1 για κίνηση προς τα δεξιά.

Γραμμή 83-91: Εδώ ορίζεται η ζημιά που παθαίνει ο παίκτης, να βγάζει ήχο και να μειώνεται η ζωή του ανάλογα με την ζημιά και να αναγράφεται η νέα ζωή που του έχει απομείνει. Έπειτα όταν δεν έχει άλλες ζωές να εμφανίζεται νέο screen(game over) καθώς ο παίχτης δεν έχει άλλες ζωές και να καταστρέφεται.

Όπως παρατηρούμε πολλές μεταβλητές δηλώθηκαν ως public στο script του παίκτη. Αυτό έγινε για να μπορέσει ο προγραμματιστής να ορίσει τις τιμές από τον Inspector χωρίς να χρειάζεται κάθε φορά να ανοίγει το script.



Εικόνα 34 : «Μεταβλητές τιμές του παίκτη»

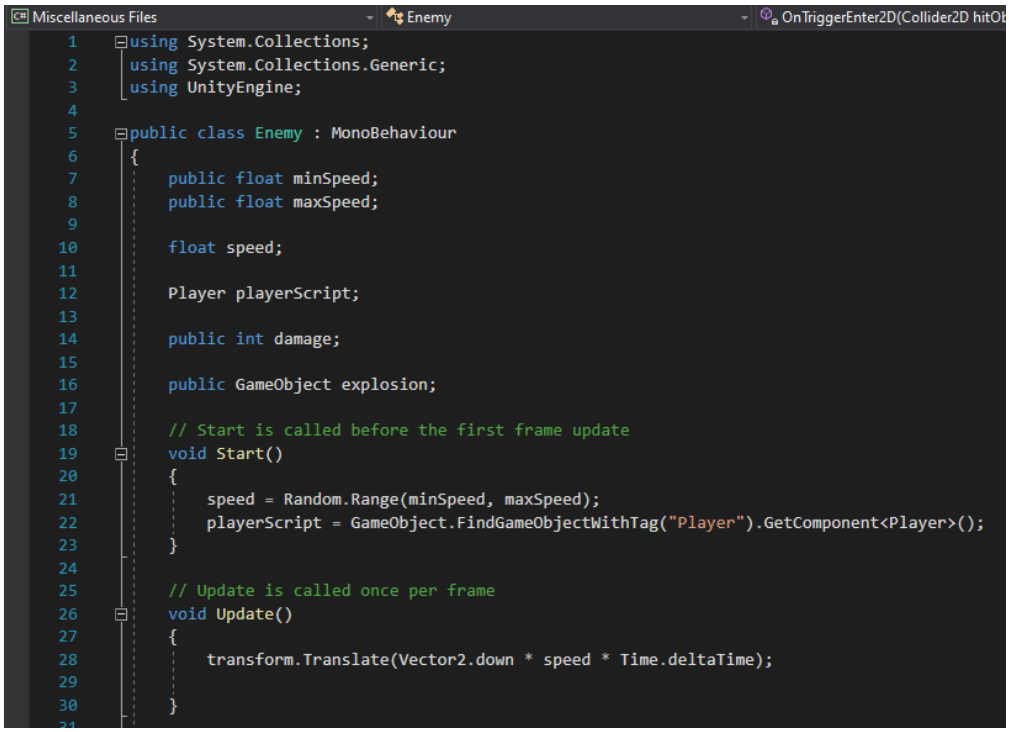
5.4 Δημιουργία των εχθρών και Scripts

Στο σημείο αυτό πρέπει να αναλυθεί η έννοια **Prefab**. Το prefab system επιτρέπει να δημιουργούμε, να διαμορφώνουμε και να αποθηκεύουμε ένα GameObject με όλα τα στοιχεία, τιμές ως επαναχρησιμοποιήσιμο Asset. Το prefab λειτουργεί ως πρότυπο από το οποίο μπορούμε να δημιουργήσουμε καινούρια. Στην ουσία είναι το καλούπι ενός αντικειμένου. Σκοπός είναι να προσθέσουμε το ίδιο αντικείμενο στην σκηνή πολλές φορές με custom ρυθμίσεις κάθε φορά χωρίς να χρειαστεί να το κατασκευάζουμε συνέχεια από την αρχή.

Με αυτόν τον τρόπο λοιπόν δημιουργούνται τα prefabs για τους τρεις εχθρούς. Ορίζουμε με τον ίδιο ακριβώς τρόπο τις διαστάσεις όπως έγινε και με τον παίκτη. Προσθέτουμε και στα τρία prefabs το στοιχείο Box Collider 2D για να οριοθετήσουμε τον χώρο τον εχθρών. Τέλος εισάγουμε και στους τρεις εχθρούς ένα script.

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

public class Enemy : MonoBehaviour
{
    public float minSpeed;
    public float maxSpeed;

    float speed;

    Player playerScript;

    public int damage;

    public GameObject explosion;

    // Start is called before the first frame update
    void Start()
    {
        speed = Random.Range(minSpeed, maxSpeed);
        playerScript =
GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
```

```
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector2.down * speed * Time.deltaTime);

    }

    void OnTriggerEnter2D(Collider2D hitObject)
    {
        if(hitObject.tag == "Player"){

            playerScript.TakeDamage(damage);
            Instantiate(explosion, transform.position, Quaternion.identity);
            Destroy(gameObject);
        }
        if (hitObject.tag == "Ground")
        {
            Instantiate(explosion, transform.position, Quaternion.identity);
            Destroy(gameObject);
        }
    }
}
```



Εικόνα 35 : «Script του εχθρού 1/2»

Γραμμή 1-3: Οι default βιβλιοθήκες.

Γραμμή 7-18: Δήλωση των μεταβλητών που θα καλεστούν αργότερα στο script. Εδώ καλείται στην γραμμή 12 ο “player” από το script του παίκτη.

Γραμμή 19-21: Οι συγκεκριμένες γραμμές εκτελούνται μια φορά. Ορίζεται η ταχύτητα με τυχαίο (random) min & max. Ψάχνει από το script του παίκτη το tag με όνομα player για να ξεκινήσουν να έρχονται οι εχθροί. Σε περίπτωση που δεν βρει τον παίκτη το script τελειώνει εκεί.

Γραμμή 28: Η κίνηση των εχθρών.

```
31
32      void OnTriggerEnter2D(Collider2D hitObject)
33      {
34          if(hitObject.tag == "Player"){
35
36              playerScript.TakeDamage(damage);
37              Instantiate(explosion, transform.position, Quaternion.identity);
38              Destroy(gameObject);
39          }
40          if (hitObject.tag == "Ground")
41          {
42              Instantiate(explosion, transform.position, Quaternion.identity);
43              Destroy(gameObject);
44          }
45      }
46  }
47
```

Εικόνα 36 : «Script του παίκτη 2/2»

Γραμμή 32-46: Όταν έρχεται σε σύγκρουση με τον παίκτη, ο παίκτης να χάνει ζωή και να γίνεται μία έκρηξή και να εξαφανίζεται αυτόματα ο εχθρός. Αν δεν έρθει σε σύγκρουση με τον παίκτη τότε όταν ακουμπάει στο έδαφος να κάνει την ανάλογη έκρηξη και να καταστρέφεται.

Οι μεταβλητές που δόθηκαν και δηλώθηκαν ως public έγιναν με αυτό τον τρόπο ώστε να μπορούμε να αλλάξουμε τις τιμές όποτε εμείς θελήσουμε χωρίς να χρειάζεται κάθε φορά να ανοίγουμε το script.

Εικόνα 37 : «Μεταβλητές τιμές του εχθρού»

5.5 Spawn των εχθρών και Explosion Effect

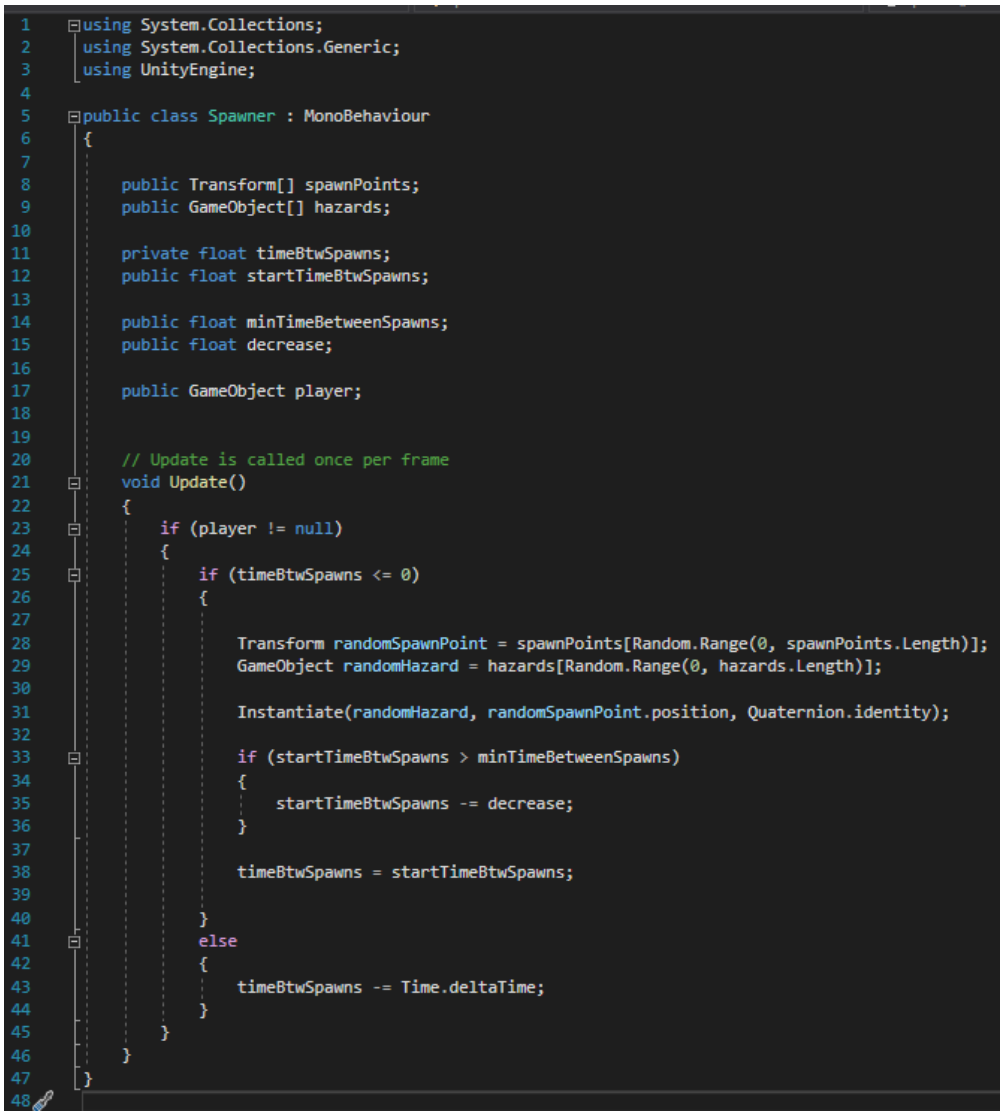
Στο προηγούμενο στάδιο έγινε ο σχεδιασμός και το script των εχθρών αλλά δεν ορίστηκαν τα spawn points (σημεία εκκίνησης). Τα spawn points καθορίζουν το σημείο έναρξης όπου από εκεί θα εμφανιστεί ο εχθρός. Στη περίπτωση του παιχνιδιού έχουν τοποθετηθεί 10 διαφορετικά spawn points ώστε να μην είναι προβλέψιμο το σημείο εκκίνησης του εχθρού.

Εικόνα 38 : «Δημιουργία Spawn Points»

Εφόσον τοποθετήθηκαν σε συγκεκριμένα σημεία κατά μήκος ώστε να καλύπτει όλη την οθόνη ενοποιήθηκαν σε ένα νέο GameObject(Spawner) και δόθηκε από τον Inspector ένα script.

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

public class Spawner : MonoBehaviour
{

    public Transform[] spawnPoints;
```

```
    public GameObject[] hazards;

    private float timeBtwSpawns;
    public float startTimeBtwSpawns;

    public float minTimeBetweenSpawns;
    public float decrease;

    public GameObject player;


    // Update is called once per frame
    void Update()
    {
        if (player != null)
        {
            if (timeBtwSpawns <= 0)
            {

                Transform randomSpawnPoint = spawnPoints[Random.Range(0,
spawnPoints.Length)];
                GameObject randomHazard = hazards[Random.Range(0, hazards.Length)];

                Instantiate(randomHazard, randomSpawnPoint.position,
Quaternion.identity);

                if (startTimeBtwSpawns > minTimeBetweenSpawns)
                {
                    startTimeBtwSpawns -= decrease;
                }

                timeBtwSpawns = startTimeBtwSpawns;

            }
            else
            {
                timeBtwSpawns -= Time.deltaTime;
            }
        }
    }
}
```



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

public class Spawner : MonoBehaviour
{

    public Transform[] spawnPoints;
    public GameObject[] hazards;

    private float timeBtwSpawns;
    public float startTimeBtwSpawns;

    public float minTimeBetweenSpawns;
    public float decrease;

    public GameObject player;


    // Update is called once per frame
    void Update()
    {
        if (player != null)
        {
            if (timeBtwSpawns <= 0)
            {

                Transform randomSpawnPoint = spawnPoints[Random.Range(0, spawnPoints.Length)];
                GameObject randomHazard = hazards[Random.Range(0, hazards.Length)];

                Instantiate(randomHazard, randomSpawnPoint.position, Quaternion.identity);

                if (startTimeBtwSpawns > minTimeBetweenSpawns)
                {
                    startTimeBtwSpawns -= decrease;
                }

                timeBtwSpawns = startTimeBtwSpawns;


            }
            else
            {
                timeBtwSpawns -= Time.deltaTime;
            }
        }
    }
}
```

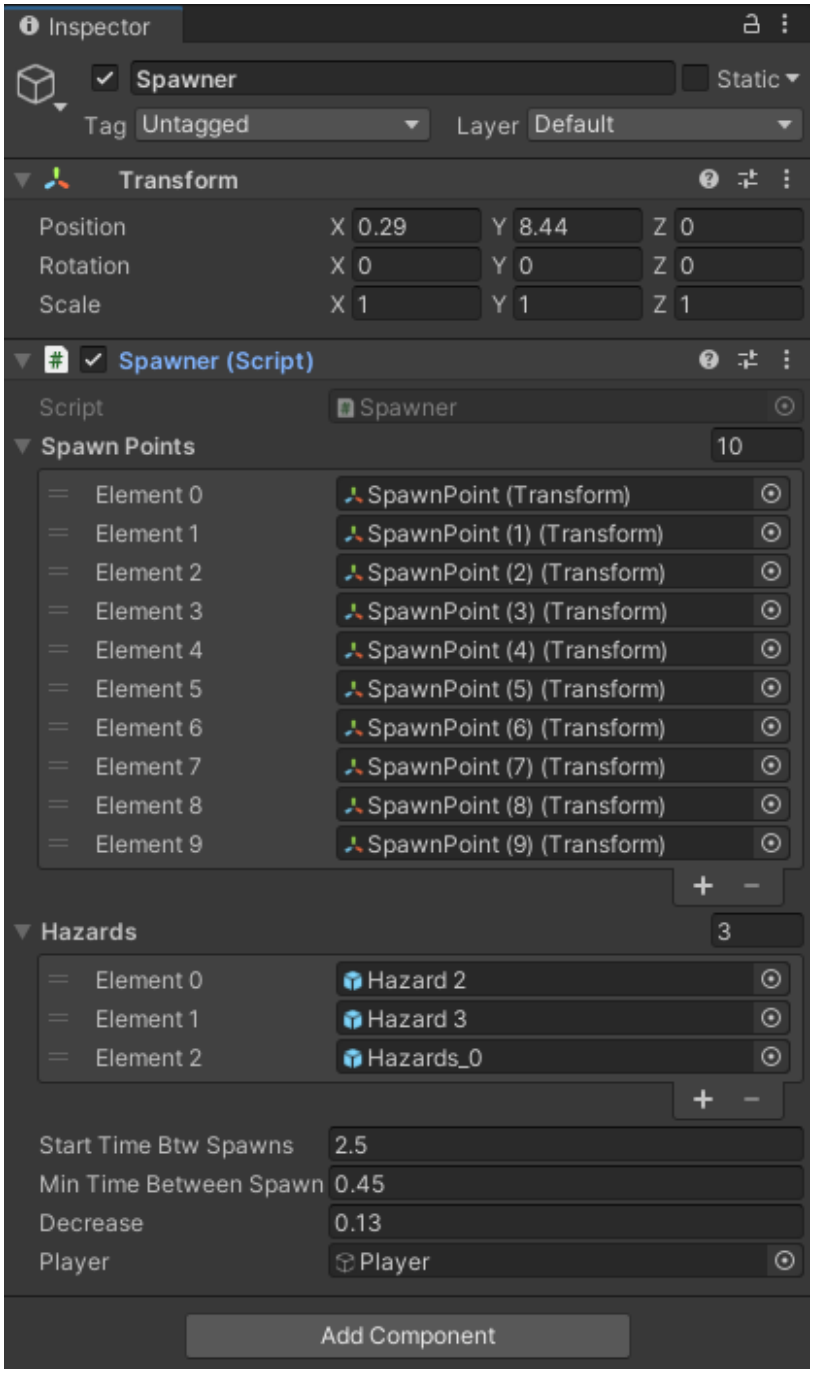
Εικόνα 39 : «Script δημιουργίας εχθρών»

Γραμμή 1-3: Default Libraries

Γραμμή 6-20: Δήλωση μεταβλητών(δήλωση χρόνου μεταξύ spawn, ελάχιστος χρόνος, χρόνος εκκίνησης και μια μείωση(decrease) που θα δώσει βαθμό δυσκολίας)

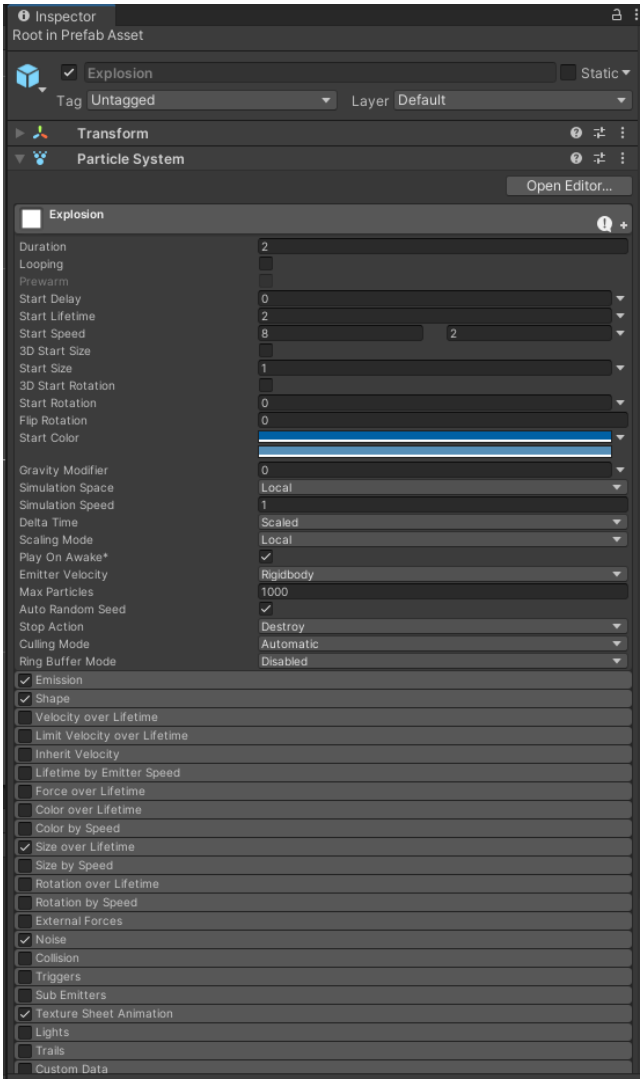
Γραμμή 21-48: Οι συνθήκες στο βήμα αυτό ελέγχει αρχικά ότι ο παίχτης υπάρχει ακόμα δηλαδή είναι ζωντανός, έπειτα ορίζει ένα τυχαίο spawn point[10] με επίσης τυχαία επιλογή εχθρού[3]. Τέλος μετά από την εμφάνιση και την εκκίνηση του εχθρού μειώνει τον χρόνο spawn του επόμενου εχθρού για να δώσει μια δυσκολία.

Οι μεταβλητές που δόθηκαν και δηλώθηκαν ως public έγιναν με αυτό τον τρόπο ώστε να μπορούμε να αλλάξουμε τις τιμές όποτε εμείς θελήσουμε χωρίς να χρειάζεται κάθε φορά να ανοίγουμε το script.



Εικόνα 40 : «Spawner Inspector window»

Επίσης σχεδιάστηκε στο σημείο αυτό ένα νέο object effect particle system (σύστημα σωματιδίων επίδρασης) για να δημιουργηθεί η αίσθηση της έκρηξης. Η έκρηξη εισάχθηκε στο παιχνίδι με την ίδια λογική των εχθρών δηλαδή σαν Prefab. Σχεδιάστηκε στον Inspector για να ταιριάζει στο παιχνίδι.



Εικόνα 41 : «Inspector window of Explosion»


Εικόνα 42 : «Particle Effects στο παιχνίδι»

5.6 User Interface

Το UI είναι απαραίτητο σε όλα τα παιχνίδια διότι απεικονίζει πληροφορίες χρήσιμες για τον χρήστη για την εξέλιξη του παιχνιδιού και όχι μόνο. Έτσι λοιπόν δεν θα έλειπε και από το συγκεκριμένο παιχνίδι ένα UI το οποίο να απεικονίζει τις εναπομείναντες ζωές του παίκτη και μία σκηνή όταν ο παίκτης δεν έχει άλλες ζωές (lose screen) καθώς και ένα Main Menu. Προσθέτουμε λοιπόν ένα UI image το οποίο να απεικονίζει μία καρδιά και ένα UI text στο οποίο να γράφει πόσες ζωές έχει ο παίκτης.

Εικόνα 43 : «Απεικόνιση ζωής του παίκτη»

Προσθέσαμε στο script του παίκτη την εντολή `healthDisplay.text = health.ToString();` για να βλέπει ο χρήστης την ζωή να ανανεώνετε κάθε φορά που τρέχει το script.

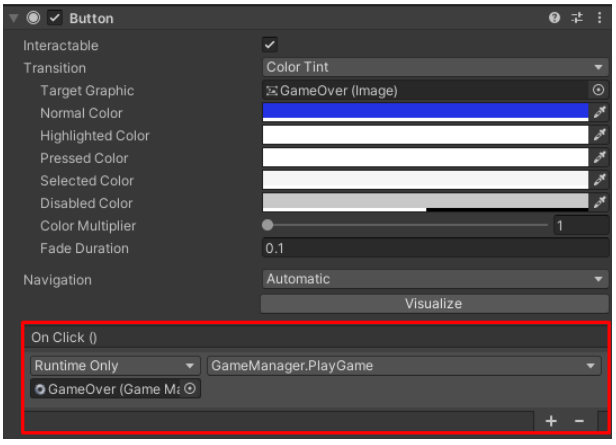
Στην συνέχεια δόθηκε ένα τέλος στο παιχνίδι σε ένα Lose Screen με button Game Over ώστε να μπορεί να το πατήσει ο χρήστης και να ξεκινήσει το παιχνίδι πάλι από την αρχή.

Εικόνα 44 : «Game Over Scene»

Στον Inspector έγινε το Lose Screen όταν χάνει ο παίκτης να αλλάζει χρώμα η σκηνή. Έπειτα τοποθετήθηκε το κουμπί Game Over. Επιλέχθηκε η ανάλογη φωτογραφία για το κουμπί και δημιουργήσαμε ένα script τύπου Game Manager ώστε όταν το πατήσει ο χρήστης το παιχνίδι να ξεκινήσει από την αρχή.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class GameManager : MonoBehaviour
7   {
8       public void PlayGame()
9       {
10          SceneManager.LoadScene("Game");
11      }
12  }
```

Εικόνα 45 : «Script διαχείρισης σκηνής»



Εικόνα 46 : «Inspector Game Over Scene»

Έπειτα δημιουργήθηκε ένα νέο scene στο παιχνίδι για να δοθεί ένα Starting Menu με την δυνατότητα ο χρήστης να πατήσει Play. Με τον ίδιο ακριβώς τρόπο που φτιάχτηκε το Lose Screen έγινε και το Starting Menu. Τοποθετήθηκε το logo του παιχνιδιού σαν image και ένα button Play ώστε όταν πατηθεί να ξεκινάει το παιχνίδι από την αρχή ίδιας φιλοσοφίας με το game over του lose screen. Έτσι τοποθετήθηκε το ίδιο script για να κάνει την μετάβαση από το scene του αρχικού μενού στο scene της δράσης.

5.7 Μουσική και ηχητικά εφέ scripts

Τελικό στάδιο του παιχνιδιού ήταν να δημιουργηθεί μια ωραία ατμόσφαιρα με την βοήθεια μουσικής και ηχητικών εφέ. Ξεκινώντας από το αρχικό μενού δημιουργήσαμε ένα script Music Manager και τοποθετήθηκε από τον Inspector σε looping (να αναπαράγετε ο ήχος επαναλαμβανόμενα).

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class MusicManager : MonoBehaviour
6   {
7       private static MusicManager instance;
8
9       private void Awake()
10      {
11          if (instance == null)
12          {
13              instance = this;
14              DontDestroyOnLoad(instance);
15          }
16          else
17          {
18              Destroy(gameObject);
19          }
20      }
21  }
22
```

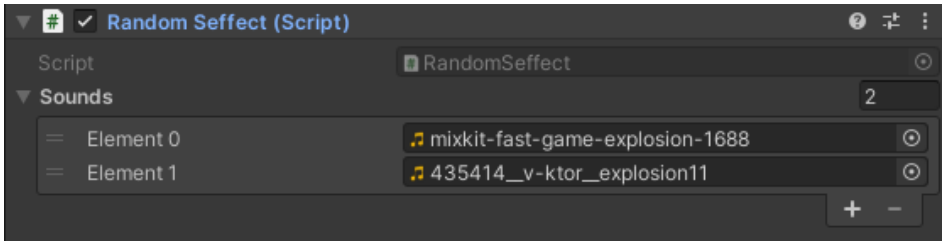
Εικόνα 47 : «Script Music Manager»

Έπειτα στην σκηνή του παιχνιδιού στο Prefab της έκρηξης προστέθηκε ένα script ώστε να ακούγεται διαφορετικός ήχος τυχαία κάθε φορά κατά την έκρηξη.

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

public class RandomSeffect : MonoBehaviour
{

    private AudioSource source;
    public AudioClip[] sounds;

    void Start()
    {
        source = GetComponent<AudioSource>();
        int randSound = Random.Range(0, sounds.Length);
        source.clip = sounds[randSound];
        source.Play();
    }

    void Update()
    {

    }
}
```

Εικόνα 48 : «Script τυχαίας δημιουργίας ήχου»

Στο συγκεκριμένο script δώσαμε την δυνατότητα να εισάγουμε όποτε θελήσουμε παραπάνω ήχους με την βοήθεια του Inspector.



Εικόνα 49 : «Inspector Random sound effect»

Βιβλιογραφία

1. Anurag (2013) *13 Pros & Cons to Know Before Choosing Unity 3D* διαθέσιμο στη διεύθυνση: https://www.newgenapps.com/blogs/unity-3d-pros-cons-analysis-choose-unity/ [Ημερομηνία πρόσβασης 30 Μαρτίου 2018]
2. BBC UK *The 8 Generations of Video Game Consoles* διαθέσιμο στη διεύθυνση: https://www.bbc.co.uk/archive/the-8-generations-of-video-game-consoles/zvcjkty
3. Bycer J. (2020) *Pros & Cons of Unity – What is the best Game engine for you?* διαθέσιμο στη διεύθυνση: https://game-wisdom.com/general/pros-cons-unity-best-game-engine [Ημερομηνία πρόσβασης 9 Ιουλίου 2020]
4. Vince (2018) *The Many Different Types of Video Games & Their Subgenres* διαθέσιμο στη διεύθυνση:https://www.idtech.com/blog/different-types-of-video-game-genres [Ημερομηνία πρόσβασης 12 Απριλίου 2018]
5. Wikipedia *Video game console* διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Video_game_console

Βοηθητικά Tutorials

1. Documentation C# https://docs.microsoft.com/en-us/dotnet/csharp/
2. Manual Unity https://docs.unity3d.com/Manual/index.html
3. Youtube Channel Brackeys https://www.youtube.com/c/Brackeys
4. Youtube Channel Code Monkey https://www.youtube.com/channel/UCFK6NCbuCIVzA6Yj1G_ZqCg
5. Youtube Channel freeCodeCamp.org https://www.youtube.com/channel/UC8butISFwT-Wl7EV0hUK0BQ