Πανεπιστήμιο
Ιωαννίνων

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πτυχιακή Εργασία

«Ανάπτυξη Εφαρμογής σε Περιβάλλον Unity»

ΝΙΚΟΛΑΚΗΣ ΝΙΚΟΛΑΟΣ

ΑΜ: 620

Επιβλέπων**:** Λιαροκάπης Δημήτριος

Άρτα, Φεβρουάριος 2024

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 2 Φεβρουαρίου 2024

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Λιαροκάπης Δημήτριος,

2. Μέλος επιτροπής

   Μαργαρίτη Σπυριδούλα,

3. Μέλος επιτροπής

   Στεργίου Ελευθέριος,

UNIVERSITY OF IOANNNINA

SCHOOL OF INFORMATICS AND TELECOMMUNICATIONS

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

THESIS

for the Undergraduate Degree

«Application Development in Unity Game Engine Enviroment»

Nikolakis Nikolaos

YD: 620

Supervisor: Liarokapis Dimitrios

Arta February 2024

Approved by Examination Committee

Arta, 2 February 2024

EXAMINATION COMMITTEE

1. Supervisor

   Liarokapis Dimitrios,

2. Member

   Margariti Spyridoula,

3. Member

   Stergiou Eleutherios,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Νικολάκης Νικόλαος

Περίληψη

Η παρούσα πτυχιακή εργασία συνοδεύει εφαρμογή της οποίας η ανάπτυξη έχει γίνει μέσω της μηχανής παιχνιδιού Unity. Σκοπός της συγγραφής της είναι αρχικά να διερευνήσει ιστορικά το είδος του παιχνιδιού που αναπτύχθηκε, το οποίο ανήκει στην κατηγορία των παιχνιδιών πλατφόρμας δύο διαστάσεων, για τα οποία παρουσιάζονται κάποια βασικά στοιχεία, αναφέρονται οι τίτλοι ορόσημο οι οποίο εδραίωσαν το είδος και ο λόγος για τον οποίο επανήλθαν στην αγορά εκ νέου με αρκετές πωλήσεις τα τελευταία χρόνια. Στη συνέχεια υπάρχει η εισαγωγή στο Unity το περιβάλλον στο οποίο έχει αναπτυχθεί το παιχνίδι. Περιγράφεται ο τρόπος με τον οποίο το συγκεκριμένο λογισμικό προσελκύει ένα μεγάλο κοινό δημιουργών και προγραμματιστών καθώς πρόκειται για ένα φιλικό περιβάλλον ανάπτυξης. Συνεχίζοντας με το Unity περιγράφεται η λογική πίσω από τη δημιουργία ενός παιχνιδιού σε αυτό το περιβάλλον καθώς παρατίθενται και επεξηγούνται οι βασικές λειτουργίες και απαιτήσεις για το σχεδιασμό και τον προγραμματισμό ενός παιχνιδιού σαν αυτό. Συγκεκριμένα υπάρχει η τοποθέτηση των δισδιάστατων στοιχείων καθώς και δημιουργία περιβάλλοντος μιας πίστας ενώ εμβαθύνοντας παρουσιάζεται η εισαγωγή του βασικού χαρακτήρα και ο τρόπος με τον οποίο κινείται μέσω του κώδικα και της κατάστασης απεικόνισης καθώς και η παραμετροποίηση της αλληλεπίδρασης του με το περιβάλλον του και άλλους χαρακτήρες όπως τους εχθρούς. Κλείνοντας, το κεφάλαιο αναφέρει τον τρόπο που πυροδοτείται η εναλλαγή των σκηνών δηλαδή οι διαφορετικές πίστες και γίνεται ειδική αναφορά στα στοιχεία συλλογής και την καταμέτρηση τους μέσω ειδικού μετρητή ο οποίος εμφανίζεται στην οθόνη. Στο τελευταίο κεφάλαιο αναδεικνύεται η ανάπτυξη των βιντεοπαιχνιδιών για σκοπούς πέραν της ψυχαγωγίας. Γίνονται αναφορές για τρόπους με τους οποίους τα βιντεοπαιχνίδια έχουν χρησιμεύσει για εκπαιδευτικούς σκοπούς είτε αυτό πρόκειται για σκοπούς σχολικής/ πανεπιστημιακής εκπαίδευσης αλλά και για την απόκτηση δεξιοτήτων και πως διάφορα παραδείγματα του παρελθόντος μπορούν να υλοποιηθούν στο σήμερα με περιβάλλοντα σαν το Unity. Σε αυτό το πλαίσιο γίνεται ειδική αναφορά στις δυνατότητες προσομοίωσης που προσφέρει το λογισμικό.

Λέξεις – Κλειδιά

Βιντεοπαιχνίδια Πλατφόρμας, Unity Game Engine, Εκπαίδευση, Προσομοίωση

Abstract

The present thesis document accompanies a video game developed using the Unity Game Engine. The purpose of its implementation is to historically cover the game genre that was developed, which is the 2D platform game genre. The basic elements of those games are presented, accompanied with references to landmark titles which are responsible for the genre's commercial rise, covering it up to today's new wave of indie 2D platform games and the reason for which they were brought back to the market with commercial success. Subsequently there is an introduction to Unity Game Engine. To begin with, the reasons behind Unity's success among a huge audience of developers and designers are described with the emphasis on it being a friendly software environment. Going further with Unity, the logic behind the creation of a 2D Platform Video Game is described with its basic functions and requirements for designing and programming it, being explained. Specifically, the placement of 2D elements and the design process of a scene or level are presented, accompanied with the introduction of the game object of the Playable Character and its movement with the assistance of the code and the animation states. Player interaction with the environment and Foes are achieved through specific configuration and parameterization. Finishing with this chapter, a reference to the way of triggering a scene or level change is described and the logic behind collectables which are displayed on the screen is described. Heading to the final chapter, ways of video games usage outside entertainment purposes are highlighted. Video Games and their relation to Education is examined weather it is in terms of School/University Education but also how Video Games have been used in the acquisition of skills and how various past examples can be implemented today with the usage of environments like Unity. In this context, special reference is made to the simulation capabilities offered by the software.

Keywords

2D Platform Games, Unity Development, Education, Simulation

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

Κεφάλαιο 1: Ιστορική Αναδρομή των Παιχνιδιών Πλατφόρμας και τα κύρια συστατικά τους

1.1 Έως τα τέλη της δεκαετίας του 90’

Τα ηλεκτρονικά παιχνίδια πλατφόρμας συνιστούν συνήθως μια αλληλουχία από κινήσεις και ενέργειες ενός χαρακτήρα που έχει στο χειρισμό του κάποιος χρήστης. Συνήθως η περάτωση ενός επιπέδου ή όλου του παιχνιδιού έχει να κάνει με την κίνηση του παίχτη από ένα σημείο Α σε ένα σημείο Β. Ιστορικά το πρώτο παιχνίδι του είδους ήταν το “Donkey Kong” το οποίο κυκλοφόρησε από την Nintendo το 1981. [1] Μέσα στα επόμενα έτη από την ίδια εταιρία υπήρξαν άλλες κυκλοφορίες στο είδος όπως τα “Mario Bros” και “Super Mario Bros”. Η δημοφιλής φιγούρα του Mario αποτέλεσε mascot για την εταιρία της Nintendo και έτσι και άλλες εταιρίες αποφάσισαν να ακολουθήσουν το συγκεκριμένο μοντέλο προώθησης δημιουργώντας τις δικές τους mascot. Χαρακτηριστικό παράδειγμα αποτελεί ο Sonic της Sega, από το παιχνίδι του είδους, “Sonic the hedgehog”. [2] Η δημοφιλία των συγκεκριμένων παιχνιδιών φαίνεται πως φτάνει να μετατρέπεται εκείνη την εποχή σε μια τεράστια εμπορική επιτυχία με τις εταιρίες να προχωρούν σε συμβόλαια για παραγωγές ταινιών και άλλων προϊόντων τα οποία δημιουργούν τεράστιο τζίρο.
Από τεχνικής απόψεως τα συγκεκριμένα παιχνίδια μέχρι τα τέλη της δεκαετίας του 80 αλλά και τις αρχές του 90, παραμένουν στις δύο διαστάσεις, ενώ από το 1994 με την κυκλοφορία του Sony PlayStation και του Sega Saturn αλλά και του Nintendo 64 το 1995, δηλαδή συστημάτων με υποστήριξη αρχιτεκτονικής 32 bit, το είδος ανοίγει στις 3 διαστάσεις. Το είδος παραμένει δημοφιλές, ενώ την τελευταία δεκαετία παρατηρείται η τάση να κυκλοφορούν πολλά παιχνίδια στις 2 διαστάσεις από διάφορες μικρότερες ή μη, εταιρίες με μια εστίαση στη retro αισθητική και φιλοσοφία.

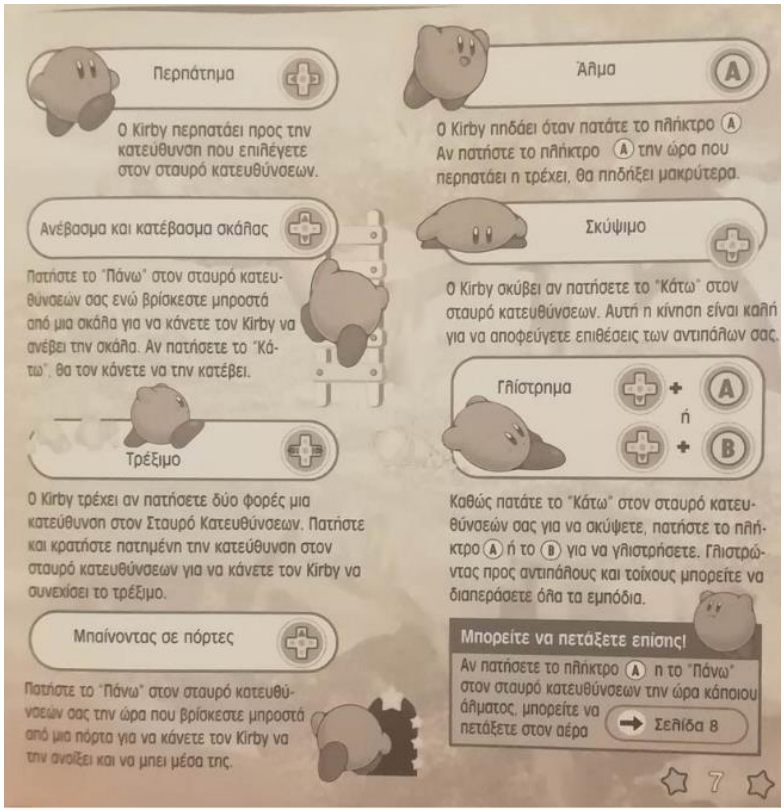
1.2 Indie games development και δισδιάστατα παιχνίδια πλατφόρμας

Σημαντικό ρόλο στη μεγάλη επαναφορά των δισδιάστατων παιχνιδιών πλατφόρμας έπαιξαν οι ανεξάρτητες ομάδες ανάπτυξης παιχνιδιών (indie game developers) [3] Σε μια εποχή που είχε επέλθει κάποιος κορεσμός σχετικά με τα συγκεκριμένα παιχνίδια στην τρισδιάστατη εκδοχή τους και με την επικράτηση των 3d shooters σε κονσόλες και υπολογιστές, οι προγραμματιστές αυτοί σε πολλές περιπτώσεις με χρηματοδότηση

του κοινού αλλά και με τις δικές τους μικρές οικονομικά, συγκριτικά με τους καπιταλιστικούς κολοσσούς του χώρου, δυνάμεις, κατάφεραν να εισέλθουν δυναμικά στη σκηνή. [4] Έτσι το 2008 έχουμε την κυκλοφορία του “Braid” από την “Number None” του Jonathan Blow, μια επαναστατική πρόταση για τους λάτρεις του είδους μιας και αποτελεί ένα συνδυασμό ενός κλασικού παιχνιδιού πλατφόρμας με στοιχεία παζλ και πρωτοποριακές δυνατότητες για το χρήστη, όπως την παρέμβαση στο χρόνο με επαναφορά σε προηγούμενες χρονικές στιγμές με στόχο την επίλυση της εκάστοτε δοκιμασίας. [5] Από τη στιγμή εκείνη μέχρι και σήμερα έχουμε δει πολλά άλλα παραδείγματα και συνάμα μπορούμε να παρατηρήσουμε πως τα περισσότερα από αυτά ήταν κερδοφόρα. Με μια απλή αναζήτηση στον ιστότοπο του games-stats.com μπορεί κάποιος να παρατηρήσει πως τα δέκα πιο δημοφιλή παιχνίδια του είδους στην πλατφόρμα “Steam” ξεκινούν από καθαρό κέρδος που ξεκινά από τα 2.6 εκατομμύρια και φτάνουν στα 18 εκατομμύρια με το παιχνίδι «Celeste”(2018) [6]

1.3 Παραδοσιακό υπόδειγμα χειρισμού

Αν πάρουμε ως παράδειγμα το κλασικό “Super Mario Bros.” μπορούμε απ’ το εγχειρίδιο του να δούμε ένα βασικό τρόπο χειρισμού που είναι αρκετά διαδεδομένος στο είδος. Η κίνηση του παίκτη γίνεται μέσω των βελών της κατεύθυνσης. Υπάρχουν ακόμα 2 πλήκτρα, το ένα χρησιμεύει στο άλμα (A) , ενώ το δεύτερο (B) χρησιμοποιείται για την επιτάχυνση του χαρακτήρα και τη ρίψη βλημάτων έπειτα από ειδική αναβάθμιση με την λήψη κατάλληλου αντικειμένου εντός παιχνιδιού. Στόχος του παίχτη είναι να φτάσει στο τερματικό σημείο της κάθε πίστας η οποία αποτελείται από διάφορα εμπόδια, εχθρούς και αντικείμενα συλλογής. Κάθε πίστα είναι κομμάτι ενός ευρύτερου χάρτη ο οποίος είναι ο κόσμος του παιχνιδιού. Η λογική για την ολοκλήρωση του παιχνιδιού συμβαδίζει με την εξερεύνηση όλων των περιοχών του χάρτη. [7]



Εικόνα 1 - Οδηγίες κίνησης από το παιχνίδι "Kirby – Nightmare In Dreamland"

(Πηγή εγχειρίδιο Kirby – Nightmare in Dreamland)

1.4 Αντικείμενα προς συλλογή και η λογική τους

Μέσα στα παιχνίδια είναι πολύ συχνή η ύπαρξη αντικειμένων προς συλλογή. Συνήθως αυτά χρησιμεύουν είτε με την επιβράβευση του παίκτη με μια επιπλέον «ζωή» έπειτα από τη συμπλήρωση κάποιου συγκεκριμένου αριθμού, είτε με την αναβάθμιση των ικανοτήτων του χαρακτήρα. Για παράδειγμα η συλλογή κάποιου από τα αντικείμενα μπορεί να δίνει στον χαρακτήρα τη δυνατότητα ρίψης βλημάτων, κάτι που βοηθά στην αποτελεσματικότερη αντιμετώπιση των εχθρών που βρίσκονται στο περιβάλλον. Υπάρχουν κι άλλων ειδών αντικείμενα προς συλλογή και σε κάθε περίπτωση δίνουν ένα συγκεκριμένο πλεονέκτημα στον χαρακτήρα αναβαθμίζοντας τις δυνατότητες του ή και σε κάποιες περιπτώσεις δημιουργώντας μειονέκτημα με σκοπό την αύξηση της δυσκολίας. [8]

Κεφάλαιο 2: Το περιβάλλον Unity και η λογική πίσω απ' τη δημιουργία ενός παιχνιδιού πλατφόρμας σε αυτό.

2.2 Η «δημοκρατικοποίηση» της Ανάπτυξης παιχνιδιών

Το Unity είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τη δημιουργία μέσων αλληλεπίδρασης, τα οποία κατά κύριο λόγο είναι παιχνίδια. Η πρώτη έκδοση του λογισμικού κυκλοφόρησε τον Ιούνιο του 2005 και ο στόχος ήταν η πιο εύκολη πρόσβαση στη δημιουργία παιχνιδιών από ερασιτέχνες προγραμματιστές και δημιουργούς σε ένα φιλικό περιβάλλον. Το γεγονός αυτό υπήρξε μια τεράστια ριζοσπαστική εξέλιξη καθώς ο χώρος της ανάπτυξης παιχνιδιών «δημοκρατικοποιείται» και ξαφνικά η υλοποίηση ενός παιχνιδιού εξ αρχής είναι διαθέσιμη σε οποιονδήποτε, ενώ μέσα στα χρόνια παγκοσμίως διοργανώνονται συνέδρια ανοιχτά σε μαθητές, φοιτητές αλλά και indie δημιουργούς. [9] Η πληθώρα δυνατοτήτων που προσφέρει το Unity στο χρήστη, τον καθιστούν ικανό να δουλέψει πάνω σε δισδιάστατα ή τρισδιάστατα περιβάλλοντα και να αξιοποιήσει πόρους προς τη δημιουργία του παιχνιδιού τους. Οι επιλογές στο είδος του παιχνιδιού είναι απεριόριστες και η ισχυρή γκάμα εργαλείων βοηθούν στο σχεδιασμό, τον προγραμματισμό των αλληλουχιών κίνησης και συμβάντων αλλά και στην αλλαγή περιβάλλοντος και την προσθήκη εφέ. Πρόκειται για ένα εργαλείο με cross-platform δυνατότητες για δημιουργία παιχνιδιών συμβατών με διάφορα λειτουργικά συστήματα και κονσόλες. [10]
Παραλλήλως αν και η γλώσσα χρησιμοποιείται εγγενώς από τη μηχανή είναι η C#, προσφέρεται στο χρήστη η δυνατότητα της πολυγλωσσικότητας για χρήση επιπρόσθετων προγραμματιστικών γλωσσών όπως η Uniscript που πρόκειται για παραλλαγή της javascript για τις ανάγκες του Unity, αλλά και η Boo. Βέβαια από το 2017 και μετά η Unity έχει σταματήσει να υποστηρίζει επίσημα τη Uniscript, ενώ το ίδιο είχε κάνει και για τη Boo το 2014, αφαιρώντας τις παράλληλα από τις επίσημες οδηγίες της, εξηγώντας τους λόγους αναλυτικά στο ιστολόγιο της. [11]

2.3 Δισδιάστατα στοιχεία για χρήση στο περιβάλλον Unity (2D assets)

Ένας δημιουργός έχει τη δυνατότητα να χρησιμοποιήσει κάποια δωρεάν στοιχεία ή να δημιουργήσει το δικά του. Μία από τις επιλογές για την εξ ολοκλήρου δημιουργία των

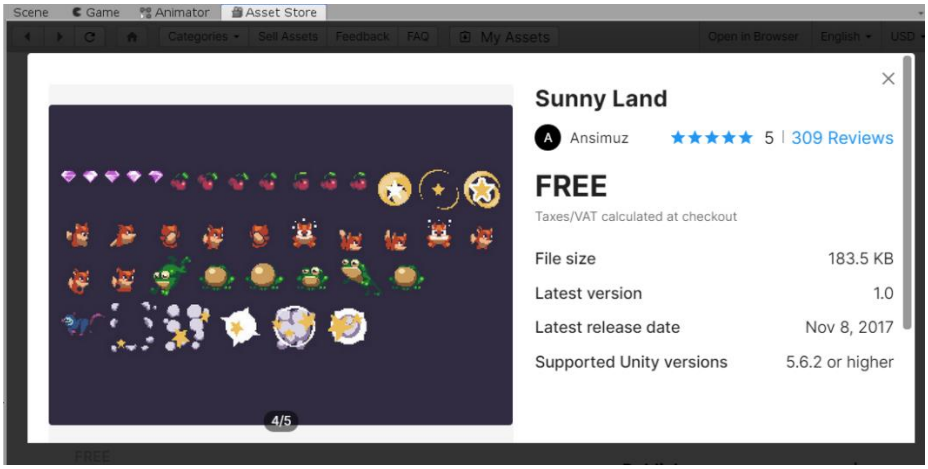
στοιχείων είναι το Aseprite, λογισμικό για τη δημιουργία γραφικών στοιχείων (sprites) [12] τα οποία μπορούν να χρησιμοποιηθούν για τη δημιουργία κινούμενων στοιχείων, το οποίο κατ' επέκταση δίνει δυνατότητες κατασκευής πακέτων πλακιδίων (tilesets) τα κι εκείνα με τη σειρά τους χρησιμοποιούνται για το περιβάλλον της εκάστοτε πίστας ή και τη δημιουργία χαρακτήρων με τη δυνατότητα να ρυθμίζεται η κίνηση τους ανά καρέ.

Εικόνα 2 - Περιβάλλον Aseprite

(Πηγή https://nathanielmorihara.medium.com/*)*

Από την άλλη ένας δημιουργός που θέλει να εστιάσει περισσότερο στο σχεδιασμό και το στήσιμο ενός παιχνιδιού σε Unity υπό το πρίσμα της λειτουργίας και του προγραμματισμού έχει τη δυνατότητα να χρησιμοποιήσει τα έτοιμα στοιχεία του Unity, η κοινότητα του οποίου προσφέρει πολλά τέτοια πακέτα στο Κατάστημα Στοιχείων (Unity Asset Store) και να επιλέξει από μια μεγάλη γκάμα αποφασίζοντας εν τέλει τι θεωρεί πως είναι πιο συμβατό με την ιδέα του. Για τις ανάγκες της παρούσας εργασίας επιλέχθηκε το πακέτο με τίτλο "Sunny Land" το οποίο διατίθεται δωρεάν μέσω του καταστήματος στοιχείων και αποτελεί μια πολύ καλή λύση για κάποιο δημιουργό καθώς δίνει πολλές δυνατότητες για πειραματισμό μιας και αποτελείται από πληθώρα γραφιστικών σχεδίων και κινήσεων για τους χαρακτήρες αλλά και ενδιαφέροντα περιβάλλοντα. Από τη στιγμή που ο δημιουργός θα εισάγει το πακέτο στο περιβάλλον του, μπορεί πλέον να σχηματίσει την πρώτη του σκηνή δίνοντας ένα περιβάλλον στο

παιχνίδι του. Το συγκεκριμένο πακέτο προσφέρει κάποια στοιχεία για το περιβάλλον μιας πίστας αλλά και χαρακτήρες για να χρησιμοποιηθούν από τον δημιουργό. [13]



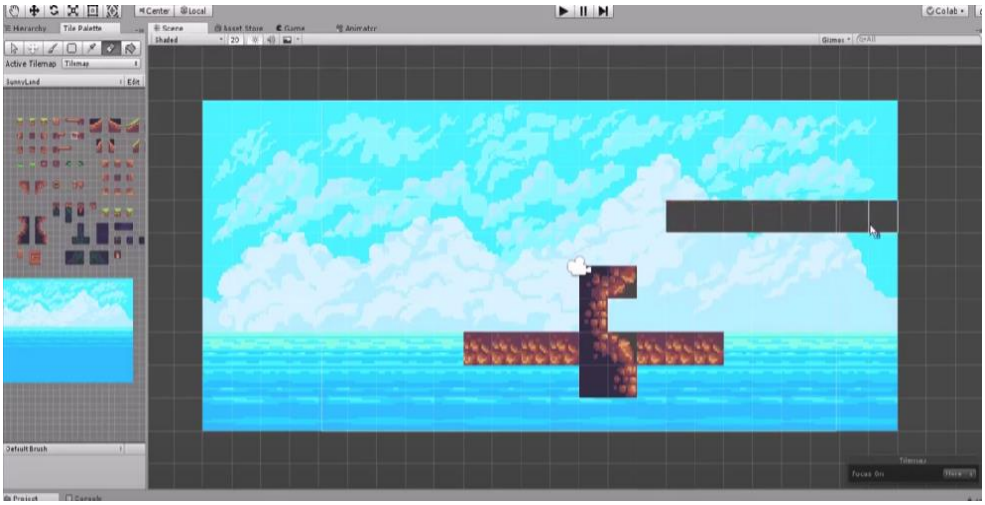
Εικόνα 3 - Unity Asset Store

(Πηγή Unity Hub)

2.4 Σετ Πλακιδίων (Tilesets) και δημιουργία περιβάλλοντος πίστας

Ως πρώτο βήμα για τη δημιουργία της πίστας μας, βασικό είναι ο σωστός τεμαχισμός και επαναμεγέθυνση των στοιχείων μας στο επιθυμητό. Η διαδικασία αυτή γίνεται μέσω του tile editor. Ουσιαστικά ο tile editor είναι το εργαλείο το οποίο βοηθά να προσδιοριστεί η παλέτα με την οποία θα στηθεί το περιβάλλον. Η παλέτα αυτή παίρνει μορφή και είναι το επονομαζόμενο tile palette. Από τη στιγμή που η επιλογή της παλέτας έχει οριστικοποιηθεί ο δημιουργός μπορεί να προχωρήσει στο σχεδιασμό μέσω της χρήσης των στοιχείων που υπάρχουν σε αυτή. Η μέθοδος ουσιαστικά θυμίζει αρκετά χρήση προγράμματος επεξεργασίας εικόνας καθώς ο χρήστης «ζωγραφίζει» τα στοιχεία του πάνω στο πλέγμα και όταν θέλει να χρησιμοποιήσει κάποιο άλλο το επιλέγει και συνεχίζει τη διαδικασία. Σε αυτό το σημείο αξίζει να σημειωθεί ότι απ' την πρώτη επαφή με το περιβάλλον δημιουργίας παιχνιδιών, εντοπίζεται η χρηστικότητα του Unity το οποίο δίνει τη δυνατότητα σε έναν χρήστη που δεν ειδικεύεται στον γραφικό σχεδιασμό να μπορέσει εξ ολοκλήρου να σχεδιάσει ένα βασικό στοιχείο του τελικού προϊόντος το οποίο υπό κανονικές συνθήκες θα έπαιρνε ένα απροσδιόριστα μεγάλο χρονικό κομμάτι ή θα προαπαιτούσε τη βοήθεια κάποιου συνεργάτη με ειδίκευση πάνω στη γραφιστική. Τα παιχνίδια συνήθως αποτελούνται από διαφορετικές πίστες. Με άλλα

λόγια η αλληλουχία και εναλλαγή αυτών μας οδηγεί στην τελευταία πίστα όπου έχουμε και το τέλος του παιχνιδιού. Για τον τρόπο εναλλαγής μεταξύ των πιστών θα δούμε περισσότερα σε επόμενη ενότητα.



Εικόνα 4 - Tile Editor

(Πηγή στιγμιότυπο Unity)

2.5 Εισαγωγή στοιχείου χαρακτήρα και η έννοια των ορίων σύγκρουσης

Η εισαγωγή του χαρακτήρα σα στοιχείο στο περιβάλλον είναι μια βασική κίνηση για να μπορούμε να ξεκινήσουμε να μιλάμε για ένα παιχνίδι τέτοιου τύπου. Αφότου έχει γίνει η εισαγωγή ο σχεδιαστής καλείται να δημιουργήσει κάποια όρια για το στοιχείο αλλά ένας τρόπος αλληλεπίδρασης του ορίου αυτού με τα στοιχεία του περιβάλλοντος. Αυτό που πρέπει να γίνει στις περισσότερες των περιπτώσεων είναι να οριστεί η ιδιότητα κάθε στοιχείου του περιβάλλοντος βάσει ορίων σύγκρουσης και να κατηγοριοποιηθούν ως κομμάτια του προσκηνίου ή του παρασκηνίου. [14] Αφότου γίνει αυτός ο διαχωρισμός τα πράγματα είναι πολύ πιο εύκολα και με την κατανόηση των εννοιών αυτών μπορούμε να προχωρήσουμε στις αντίστοιχες ρυθμίσεις μέσω του πίνακα επιθεώρησης(inspector).

2.6 Κίνηση χαρακτήρα

Το αρχικό βήμα για να ξεκινήσει η παραγωγή κώδικα βάσει το οποίου ο χαρακτήρας του παιχνιδιού θα κινείται, είναι αρχικά να τεθούν κάποια πλήκτρα για την κίνηση μέσω των ρυθμίσεων του project. Αφότου συμβεί αυτό, πρέπει να οριστεί ένας άξονας και να ονομαστεί. Στη συνέχεια, ο δημιουργός – προγραμματιστής έχοντας υπόψιν τα πλήκτρα τα οποία έχουν οριστεί για την κίνηση στον οριζόντιο άξονα, θα πρέπει να δημιουργήσει μια κλάση η οποία θα κινεί τον χαρακτήρα στο οριζόντιο επίπεδο. Αυτό βέβαια έχει να κάνει με την αλλαγή στις τιμές του άξονα κατά τη διάρκεια της μετατόπισης. Καθώς είναι γεγονός πως ένα παιχνίδι μπορεί να διαθέτει διαφορετικά ύψη στο περιβάλλον του, είναι προφανώς βασικό να ληφθεί υπόψιν και ο άξονας y, του οποίου η τιμή θα πρέπει να αλλάζει βάσει των συντεταγμένων που αποκτά συνεχώς ο χαρακτήρας καθώς κινείται στον άξονα x. Με άλλα λόγια μια οριζόντια κίνηση αυτομάτως δεν είναι αποκλειστικά οριζόντια καθώς η θέση του χαρακτήρα ή του εκάστοτε στοιχείου μετατοπίζεται συνεχώς και στους δύο άξονες ακόμα κι αν δεν μιλάμε για άλμα. Αν φανταστεί κάποιος την ανάβαση σε έναν λόφο είναι πολύ απλό να καταλάβει το παραπάνω σκεπτικό. Από κει και πέρα προφανώς αυτό που ενδιαφέρει τον προγραμματιστή είναι αν η κίνηση γίνεται προς τη δεξιά κατεύθυνση ή προς την αριστερή. Χρήσιμο είναι να χρησιμοποιηθεί μια μεταβλητή για την κατεύθυνση η οποία θα τραβάει μια τιμή από τον άξονα που χρησιμοποιείται και ανάλογα με το πρόσημο του άξονα X, θα υπάρχει η αντίστοιχη κατεύθυνση κίνησης. [15] Αν δηλαδή η τιμή είναι μικρότερη του μηδενός θα έχουμε κίνηση προς τα αριστερά και εάν είναι μεγαλύτερη, τότε η κίνηση θα είναι προς τα δεξιά.

```
private void SetVelocityState()
{

  float hDirection= Input.GetAxis("Horizontal");

    if (hDirection<0)
    {
       rb.velocity = new Vector2(-speed, rb.velocity.y);
       rb.transform.localScale= new Vector2(-1,1);

    }

    else if (hDirection>0)
    {
       rb.velocity = new Vector2(speed, rb.velocity.y);
       rb.transform.localScale= new Vector2(1,1);

    }
```

Εικόνα 5 - κώδικας κίνησης

(Πηγή στιγμιότυπο κώδικα)

Κάτι αντίστοιχο βέβαια ισχύει και για το άλμα, το οποίο θα οριστεί με μια νέα μέθοδο η οποία θα αλλάζει την τεταγμένη και όχι την τετμημένη όπως στο παράδειγμα της απλής κίνησης δεξιά ή αριστερά. Το άλμα συντελείται όχι μόνο όταν επιλέγει ο παίκτης να το πραγματοποιήσει χρησιμοποιώντας το αντίστοιχο πλήκτρο πίεσης, αλλά και όταν υπάρχει εξουδετέρωση ενός εχθρού με άλμα επάνω του. Με άλλα λόγια η μέθοδος αυτή πρέπει να πραγματοποιείται υπό κάποιες συγκεκριμένες συνθήκες οι οποίες βέβαια εξαρτώνται και από την κατάσταση απεικόνισης (animation state) που βρίσκεται ο χαρακτήρας μας. Για να γίνει κάτι τέτοιο πιο κατανοητό η έννοια της κατάστασης θα παρουσιαστεί στη συνέχεια.

2.7 Κατάσταση Απεικόνισης (Animation State)

Προφανώς και ένα σταθερό σώμα χωρίς αλλαγή στη γραφική του απεικόνιση που απλά μετατοπίζεται θα φάνταζε μη τελικό ή σε πειραματικό επίπεδο. Βασικό για την ολοκλήρωση ενός χαρακτήρα που κινείται είναι και η αλλαγή στην κατάσταση απεικόνισης η οποία με τη συνεχή εναλλαγή της δίνει μια πιο φυσική αίσθηση κίνησης. Αν μιλάμε για μια κατάσταση στην οποία ένας χαρακτήρας προχωράει τότε σίγουρα ο σχεδιαστής χρειάζεται να έχει στην κατοχή του ή να δημιουργήσει τουλάχιστον 2-3 ή και περισσότερα διαφορετικά στοιχεία στα οποία αλλάζει η θέση των ποδιών του χαρακτήρα και η κίνηση στο σώμα του. Αυτό είναι απαραίτητο για τη δημιουργίας κίνησης ή οποία αποτελείται από πολλά διαφορετικά στοιχεία. Η διαδικασία αυτή ολοκληρώνεται μέσω της εισαγωγής των στοιχείων αυτών αλλά και της σωστής χρόνισης τους ως animation του περιβάλλοντος Unity. Ολοκληρώνοντας ένα τέτοιο βήμα για κάθε κίνηση που έχει σκεφτεί να σχεδιάσει ο δημιουργός τότε είναι έτοιμος να θέσει και το πλαίσιο για την εναλλαγή αυτών των κινήσεων. [16]

2.8 Εναλλαγή πίστας (σκηνής)

Για την εναλλαγή πίστας υπάρχουν διάφοροι μέθοδοι στα ηλεκτρονικά παιχνίδια. Στην περίπτωση των παιχνιδιών πλατφόρμας, ένας πολύ κλασικός τρόπος για αυτή τη λειτουργία είναι η επαφή του χαρακτήρα με κάποιο στοιχείο ή σημείο της πίστας το οποίο μέσω του αντίστοιχου κώδικα θα δημιουργεί την εναλλαγή αυτή. Ένα απλό παράδειγμα μιας τέτοιας περίπτωσης είναι η επαφή του χαρακτήρα με μια ξύλινη πινακίδα η οποία αποτελεί αντικείμενο που έχει δημιουργηθεί εντός του Unity και έχει επιλεχθεί να βρίσκεται στο σημείο που είναι το τέλος της εκάστοτε πίστας. Μέσω των ρυθμίσεων δημιουργίας (build settings) έχει τεθεί μια συγκεκριμένη σειρά στις σκηνές, κι έτσι έπειτα δημιουργείται ο αντίστοιχος κώδικας ο οποίος πυροδοτεί την μετάβαση στην επόμενη πίστα, όταν ο χρήστης μας υπερβεί τα όρια σύγκρουσης του αντικειμένου που έχει επιλεχθεί. Στις περισσότερες περιπτώσεις κάθε πίστα ενός παιχνιδιού έχει παρόμοιο τρόπο μετάβασης στην επόμενη, δε λείπουν όμως και τα παραδείγματα στα οποία η μετάβαση γίνεται μόνο υπό συγκεκριμένες προϋποθέσεις που θέτει ο δημιουργός. Ένα τέτοιο παράδειγμα σε ένα παιχνίδι πλατφόρμας θα ήταν η αδυναμία μετάβασης σε επόμενη πίστα σε περίπτωση που ο χρήστης δεν έχει καταφέρει να συλλέξει έναν συγκεκριμένο αριθμό στοιχείων συλλογής ή ένα συγκεκριμένο στοιχείο, όπως για παράδειγμα ένα κλειδί το οποίο είναι απαραίτητο για την απασφάλιση κάποιας κλειδαριάς που φράζει το δρόμο του χαρακτήρα. Υπάρχουν πολλές δυνατότητες και όλα αυτά μπορούν να οριστούν μέσω του κώδικα. [17]

2.9 Στοιχεία συλλογής και η καταμέτρησή τους

Παραδοσιακά τα παιχνίδια πλατφόρμας διαθέτουν ένα σύστημα συλλογής αντικειμένων. Σε πολλές περιπτώσεις αυτά είναι κάποιου είδους νομίσματα, φαγητό ή κρύσταλλοι. Ο χαρακτήρας συναντά αυτά τα αντικείμενα μέσα σε κάθε επίπεδο του παιχνιδιού σε διαφορετικά σημεία όπως είναι ορισμένα από τον σχεδιαστή – προγραμματιστή. Σε πολλές περιπτώσεις η συλλογή ενός συγκεκριμένου αριθμού από αυτά τα στοιχεία δίνει κάποια επιβράβευση στον παίχτη, ενώ σε άλλες πρόκειται για απλή καταμέτρηση με στόχο την μέγιστη δυνατή συλλογή. Τα στοιχεία αυτά σε ένα στήσιμο και σχεδιασμό/ προγραμματισμό παιχνιδιού με Unity δημιουργούνται ως απλά αντικείμενα παιχνιδιού. Κι έπειτα αντιγράφονται μέσα στην πίστα και τοποθετούνται στα ανάλογα σημεία. Αυτό που μας ενδιαφέρει στον κώδικα που θα επισυνάπτεται σε αυτά είναι να εξαφανίζονται με την επαφή του παίχτη με τα όρια σύγκρουσής τους και ταυτόχρονα ένας μετρητής να αυξάνεται κατά την εκάστοτε επαφή του χαρακτήρα με αυτά. [18] Από κει και έπειτα αυτός ο μετρητής είναι χρήσιμο να εμφανίζεται σε σημείο της οθόνης, το οποίο ενημερώνει τον χρήστη για τον αριθμό των στοιχείων συλλογής που έχει καταφέρει να αποκτήσει μέχρι στιγμής.

2.10 Εχθροί

Στα παιχνίδια πλατφόρμας συνηθίζεται, εκτός από τα φυσικά εμπόδια που ο χρήστης καλείται να ξεπεράσει, να πρέπει να έρθει αντιμέτωπος και με κάποιους εχθρούς. Σκοπός της ύπαρξής τους είναι η δημιουργία μεγαλύτερης δυσκολίας στην ολοκλήρωση του εκάστοτε επιπέδου. Οι εχθροί εισάγονται στο παιχνίδι ως αντικείμενα παιχνιδιού και η διαδικασία του εμπλουτισμού τους με γραφικά είναι αντίστοιχη με αυτή του χαρακτήρα που χειρίζεται ο παίκτης. Οι εχθροί έχουν τα δικά τους όρια σύγκρουσης, τις δικές τους καταστάσεις απεικόνισης αλλά και ένα κώδικα που συνεισφέρει στην αυτόνομη κίνηση τους. Υπό περιπτώσεις μπορεί να χρησιμοποιείται και τεχνητή νοημοσύνη για διάφορες επιπλέον δυνατότητες. Ένα τέτοιο παράδειγμα θα ήταν η συνεχής τροποποίηση του μοτίβου κίνησης ενός εχθρού βάσει των κινήσεων του παίχτη ή η εκτόξευση βλημάτων ανάλογα με την απόσταση του από τον παίχτη. Σε πιο απλά παραδείγματα οι εχθροί είναι προγραμματισμένοι να καλύπτουν ένα συγκεκριμένο εύρος πάνω στην πίστα. Η εξουδετέρωση των εχθρών είναι κάτι το οποίο προγραμματίζεται βάσει των ορίων σύγκρουσής τους. Ανάλογα με την προτίμηση του σχεδιαστή – προγραμματιστή το συμβάν αυτό επιτελείται κάθε φορά που υπάρχει κάποια συγκεκριμένη αλληλεπίδραση με τα όρια σύγκρουσης τους. [18]

Κεφάλαιο 3: Στάδια ανάπτυξης λογισμικού

3.1 Ανάλυση και προδιαγραφή των απαιτήσεων

Η εφαρμογή που αναπτύχθηκε αρχικά από την πλευρά του σχεδιαστή είχε ως βασικό στόχο και σκοπό τη δημιουργία και την ανάπτυξη ενός παιχνιδιού το οποίο θα προσέδιδε την απαραίτητη εξοικείωση με το συγκεκριμένο λογισμικό δηλαδή το Unity. Γίνεται κατανοητό με άλλα λόγια, πως οι σκοποί της ανάπτυξης του λογισμικού αυτού ήταν εκπαιδευτικοί εκ πρώτης όψεως με την απόκτηση ενός συγκεκριμένου πακέτου δεξιοτήτων το οποίο βέβαια μπορεί να αποτελέσει γνώση η οποία θα έχει και επαγγελματικές προεκτάσεις μιας και οποιαδήποτε επιπλέον εκπαίδευση σε προγραμματισμό λογισμικού, σχεδιασμό και εκτέλεση μπορούν να χρησιμεύσουν ως επιπρόσθετα προσόντα για έναν εκκολαπτόμενο προγραμματιστή που αναζητά εργασία στο συγκεκριμένο τομέα.

Κατά δεύτερον η εμπειρία αυτή προφανώς και εμπεριείχε την πρόκληση της ανάπτυξης λογισμικού – προϊόντος το οποίο θα μπορούσε να ανταποκριθεί στις απαιτήσεις της σύγχρονης αγοράς μέσω της ακολουθίας βημάτων που μια σύγχρονη εταιρία θα μπορούσε να ακολουθήσει έτσι ώστε να κυκλοφορήσει το λογισμικό της. Μιας και η παρούσα εργασία είναι ατομική, είναι σαφές πως το κομμάτι αυτό προσομοιώνεται και εξατομικεύεται στα μέτρα των δεδομένων και στους περιορισμούς μιας ατομικής δουλειάς χωρίς όμως να αγνοεί τα βήματα τα οποία ακολουθούνται από μεγάλες ομάδες ανάπτυξης.

Στα πλαίσια αυτά προφανώς και λαμβάνεται υπόψιν το κομμάτι των λειτουργικών απαιτήσεων από την πλευρά ενός χρήστη. Και εξετάζεται η εμπειρία του χρήστη και οι λειτουργικές και μη λειτουργικές απαιτήσεις του λογισμικού. Στο παρόν σημείο είναι απαραίτητο να αναφερθεί πως για το συγκεκριμένο κεφάλαιο βιβλιογραφικά έχει μελετηθεί και χρησιμοποιηθεί πρότυπο που προτείνεται στο εγχειρίδιο σημειώσεων του μαθήματος Τεχνολογία Λογισμικού της σχολής Μηχανικών Πληροφορικής του Πανεπιστημίου Ιωαννίνων του Μάριου Μ. Μάντακα και συγκεκριμένα η έκδοση 1.6. [19]

3.2 Σκοπός και εύρος της εφαρμογής

Το λογισμικό έχει ως στόχο την απόκτηση εμπειρίας στο περιβάλλον Unity. Μέσω αυτής της διαδικασίας δημιουργείται ένα παιχνίδι πλατφόρμας δύο διαστάσεων το οποίο φιλοδοξεί να ανταπεξέλθει στις απαιτήσεις της αγοράς και να κρατήσει έναν χρήστη ικανοποιημένο από την ψυχαγωγική εμπειρία αυτή. Για την υλοποίηση του απαραίτητου κώδικα γίνεται χρήση της αντικειμενοστραφούς γλώσσας C# με κλάσεις οι οποίες είναι συμβατές με τη διεπαφή του περιβάλλοντος Unity.

3.3 Γενική Περιγραφή

Το λογισμικό πρόκειται για ένα παιχνίδι πλατφόρμας δύο διαστάσεων.
Επιτρέπει στον χρήστη να ξεκινήσει το παιχνίδι, να πάρει τον έλεγχο του βασικού χαρακτήρα και να επιχειρήσει να ολοκληρώσει τα διαθέσιμα επίπεδα για να φτάσει στο τέλος.

Το λογισμικό αποτελείται από τα εξής:

- Βασικό περιβάλλον διαφορετικών επιπέδων – σκηνών που αποτελείται από γραφικά στοιχεία
- Χαρακτήρες, οι οποίοι κινούνται μέσω κώδικα και κινούμενων γραφικών στοιχείων που αλληλοεπιδρούν με αυτόν.
- Μετρητής στοιχείων συλλογής
- Στοιχείο κάμερας που είναι προγραμματισμένο να ακολουθεί τον βασικό χαρακτήρα μέσα στην εκάστοτε σκηνή
- Τερματικά σημεία που οδηγούν σε επόμενα επίπεδα

3.4 Αναφορές εγγράφου απαιτήσεων

1) Μάριος Μάντακας (2015) «Τεχνολογία Λογισμικού, Σημειώσεις» , Έκδοση 1.6

3.5 Χρήστες

Το λογισμικό πρόκειται για ένα παιχνίδι πλατφόρμας δύο διαστάσεων.
Επιτρέπει στον χρήστη να ξεκινήσει το παιχνίδι, να πάρει τον έλεγχο του βασικού χαρακτήρα και να επιχειρήσει να ολοκληρώσει τα διαθέσιμα επίπεδα για να φτάσει στο τέλος.

Ο βασικός ρόλος Χρήστη ονομάζεται παίχτης ή απλά χρήστης

Καταστάσεις Χρήστη:

- Εντός Περιβάλλοντος παιχνιδιού
- Εκτός Περιβάλλοντος παιχνιδιού

3.6 Μη λειτουργικές απαιτήσεις

Δυνατότητα cross platform	Το λογισμικό θα επιτρέπει την αναπαραγωγή σε διαφορετικές πλατφόρμες μέσω διαφορετικών builds
Συμβατότητα με παλαιά συστήματα	Το λογισμικό θα μπορεί να τρέχει σε παλαιά συστήματα.
Βάση δεδομένων	Το λογισμικό δε θα κάνει χρήση βάσης δεδομένων.
Λογαριασμός χρήστη	Το λογισμικό δε θα απαιτεί δημιουργία λογαριασμού χρήστη.
Διεπαφή χρήστη	Το λογισμικό κατά την εκκίνηση του θα διαθέτει γραφική διεπαφή χρήστη μέσω μενού επιλογών για την είσοδο στο περιβάλλον παιχνιδιού ή για την έξοδο από την εφαρμογή.

3.7 Λειτουργικές απαιτήσεις

Εκκίνηση	Το λογισμικό θα επιτρέπει στο χρήστη να εισέρχεται στο περιβάλλον του παιχνιδιού μέσω επιλογής του μενού.
Εγκατάλειψη	Το λογισμικό θα επιτρέπει στο χρήστη να εγκαταλείπει το περιβάλλον του μενού μέσω επιλογής του μενού.
Παίκτης. Βασική Κίνηση παίχτη	Το λογισμικό θα επιτρέπει στο χρήστη να ελέγχει την κίνηση του χαρακτήρα μέσω των πλήκτρων του πληκτρολογίου.
Μετρητής αντικειμένων συλλογής	Το λογισμικό θα επιτρέπει στον χρήστη να βλέπει τον αριθμό των αντικειμένων που έχει συλλέξει μέσω γραφικής αναπαράστασης στο πάνω και αριστερά μέρος της οθόνης.
Αλλαγή σκηνής	Το λογισμικό θα επιτρέπει στο χρήστη να πραγματοποιεί μετάβαση στην επόμενη σκηνή.

3.8 Σχεδιασμός

Ως κλασικό παιχνίδι πλατφόρμας δύο διαστάσεων, τα βασικά λειτουργικά στοιχεία στην δομή του από πλευράς του σχεδιασμού, είναι τα εξής:

- Επίπεδα παιχνιδιού
- Χαρακτήρας
- Χειρισμός
- Δομή και λειτουργία επιπέδου
- Αντικείμενα συλλογής
- Αντικείμενα εμποδίου
- Κεράσια προς συλλογή
- Εχθροί
- Πινακίδες αλλαγής πίστας και τερματισμού

3.9 Προκλήσεις Υλοποίησης

Έχει αρκετά μεγάλο ενδιαφέρον να αναφερθούν διάφορες προκλήσεις που προέκυψαν κατά με την υλοποίηση του παιχνιδιού. Αρχικά μια από αυτές ήταν η βασική κατασκευή του εκάστοτε περιβάλλοντος πίστας. Το πακέτο γραφικών στοιχείων αποτελούνταν από κάποια βασικά πλακίδια τα οποία θα ήταν το βασικό υλικό για την οπτική υλοποίηση. Η πρόκληση στο συγκεκριμένο σημείο έχει να κάνει περισσότερο με το σχηματισμό ενός αρκετά σωστά υπολογισμένου περιβάλλοντος, το οποίο εξ αρχής θα δημιουργούσε μια λειτουργική βάση για την ομαλή εκτέλεση όλων των δυνατοτήτων του παιχνιδιού. Στο συγκεκριμένο σημείο ήταν απαραίτητο να υπάρχει διορατικότητα σχετικά με την τοποθέτηση των ορίων και την εμπειρική προδιαγραφή σχετικά με τις αποστάσεις και το μήκος των περιοχών στις οποίες απαιτείται άλμα από τον χαρακτήρα. Εάν ο σχεδιαστής πάρει υπόψιν του όλα αυτά τα δεδομένα, τότε η δουλειά των δοκιμαστών γίνεται ευκολότερη καθώς δεν απαιτούνται μεγάλες αλλαγές κατά τον δοκιμαστικό έλεγχο όπως θα δούμε στο παρακάτω κεφάλαιο. Ένας σχεδιαστής με μικρότερη εμπειρία πάνω στον τομέα είναι δεδομένο πως θα χρειαστεί να κάνει περισσότερες δοκιμές για να καταλήξει σε ένα ασφαλές περιβάλλον για τον παίχτη χωρίς τεχνικής φύσεως προβλήματα. Βέβαια σε ένα ατομικό project όπου καλείται να δουλέψει ένας προγραμματιστής υλοποιώντας όλους τους ρόλους μια ομάδας, κάποια δεδομένα είναι καλό να ελέγχονται ταυτόχρονα. Έτσι στο συγκεκριμένο πρόβλημα που παρουσιάζεται παραπάνω η υλοποίηση απαιτούσε να υπάρχει συνεχής έλεγχος του κώδικα για την κίνηση του χαρακτήρα παράλληλα με τη δημιουργία κάποιας πίστας. Αυτό συμβαίνει διότι υπάρχουν συγκεκριμένες μεταβλητές οι οποίες ορίζουν το διάνυσμα της ταχύτητας αλλά και το ύψος του άλματος. Σε ορισμένες περιπτώσεις και ιδιαίτερα στις τελευταίες πίστες του παιχνιδιού όπου χρονικά από άποψη σχεδιασμού, οι τιμές αυτές είχαν ήδη οριστεί, ήταν απαραίτητο να γίνεται έλεγχος σχετικά με το αν διαφορετικά σημεία της πίστας ήταν προσιτά. Ένα τέτοιο σφάλμα θα καθιστούσε το παιχνίδι να είναι αδύνατο να τερματιστεί, και έτσι μια βασική πρόκληση ήταν το να μπορεί κάθε τμήμα του να είναι πλήρως λειτουργικό. Από εκεί και πέρα μια επιπλέον πρόκληση η οποία βέβαια βασίζεται και στο παραπάνω πρόβλημα ήταν η κίνηση των εχθρών πάνω σε αυτά τα περιβάλλοντα. Στο συγκεκριμένο παράδειγμα έχουμε την αυτόνομη κίνηση ενός βατράχου που πρόκειται

για εχθρό. Μια κίνηση που γίνεται με βάση τον κώδικα που ορίζεται από τον προγραμματιστή μπορεί να δημιουργεί πρόβλημα ανά συνθήκες και να καθιστά το παιχνίδι μη λειτουργικό καθώς ενδέχεται να προκύπτουν συγκεκριμένες περιπτώσεις, όπως παραδείγματος χάρη ένα μόνιμο μπλοκάρισμα μιας βασικής περιοχής η οποία απαιτείται να προσπελαστεί από τον παίχτη για να φτάσει σε τερματικό σημείο. Ένα σφάλμα τέτοιας φύσης μπορεί να θέσει το λογισμικό ως προβληματικό, οπότε ο κατάλληλος προγραμματισμός του κώδικα για αυτόνομη κίνηση των εχθρών σε συνδυασμό με τον σχεδιασμό της πίστας ως αλληλένδετες διαδικασίες ήταν μια βασική πρόκληση. Σε άλλη μια περίπτωση ένα βασικό ζήτημα που προέκυψε αλλά πάλι σχετίζεται με το σχεδιασμό των επιπέδων-πιστών είναι η λειτουργία της κάμερας καθώς παρότι η κάμερα προγραμματίζεται και παραμετροποιείται για να ακολουθεί το βασικό χαρακτήρα υπάρχουν σημεία τα οποία μπορεί να μην είναι πλήρως ορατός. Βέβαια εδώ είναι απαραίτητο να τονιστεί πως διάφορα εμπορικά παιχνίδια έχουν αφήσει ως έχει τέτοια συμβάντα εντός παιχνιδιού χωρίς όμως να επηρεάζεται η δυνατότητα τερματισμού του παιχνιδιού ή να δημιουργείται πλήρης σύγχυση σε κάποιο παίχτη. Εδώ αξίζει να σημειωθεί πως διάφορα παιχνίδια όπως το κλασικό Super Mario Bros αξιοποιούν μια τέτοια «παραφωνία» προσθέτοντας κάποια μυστική περιοχή η οποία δεν είναι προσβάσιμη υπό φυσιολογικές συνθήκες. Παρ’ όλα αυτά το να μη δημιουργεί η κίνηση της κάμερας σύγχυση στο χρήστη ήταν μια βασική πρόκληση για να καθίσταται πλήρως λειτουργικό το συγκεκριμένο παιχνίδι.

3.10 Δοκιμαστικός Έλεγχος

Η διαδικασία του δοκιμαστικού ελέγχου πρόκειται για ένα απαραίτητο βήμα πριν την έκδοση ενός λογισμικού και το ίδιο προφανώς ισχύει και για κάποιο παιχνίδι. Η ομαλή λειτουργία του παιχνιδιού είναι ο στόχος για την ομάδα. Σε γενικά πλαίσια ο σκοπός είναι η παραγωγή ενός παιχνιδιού που μπορεί να τερματιστεί χωρίς προβλήματα ενώ ταυτόχρονα η εμπειρία για τους χρήστες να είναι ικανοποιητική, χωρίς σφάλματα και να προσφέρει ότι υπόσχεται για να είναι ελκυστικό στους χρήστες. Παρακάτω περιγράφεται ο δοκιμαστικός

έλεγχος που προέκυψε στην παρούσα εργασία έτσι ώστε το παιχνίδι να φτάσει σε ένα τελικό στάδιο.

Κίνηση και έλεγχος του παίκτη: Ο βασικός έλεγχος που έχει να κάνει με την κίνηση του παίκτη αρχικά σχετίζεται με την εμπρός και πίσω μετατόπιση του χαρακτήρα. Αυτό μπορεί να ελεγχθεί με τα πλήκτρα πίεσης που έχουν οριστεί για αυτή τη λειτουργία. Το ίδιο ισχύει και για το άλμα. Οπότε πιέζοντας τα αντίστοιχα πλήκτρα πρέπει να ελεγχθεί αν ο χαρακτήρας ανταποκρίνεται στις συγκεκριμένες εντολές.

Έλεγχος των σημείων σύγκρουσης και του εδάφους: Σε αυτό το σημείο ο έλεγχος έχει να κάνει αρχικά με το έδαφος, με άλλα λόγια περπατώντας πάνω στην πίστα πρέπει να ελεγχθεί εάν τα σημεία του εδάφους ανταποκρίνονται ως τέτοια και δεν υπάρχει κάποια απροσδόκητη πτώση. Το ίδιο ισχύει και για όλα τα αντικείμενα που εμφανίζονται εντός της πίστας. Σε σημεία που υπάρχουν αιωρούμενες πλατφόρμες θα πρέπει ο χαρακτήρας να στέκεται σε αυτές. Εάν αυτό δε συμβαίνει, τότε υπάρχει πρόβλημα με τα σημεία σύγκρουσης των στοιχείων αυτών τα οποία είτε δε θα έχουν τοποθετηθεί καθόλου, είτε δε θα έχουν τοποθετηθεί σωστά.

Στοιχεία Συλλογής: Πρέπει να ελεγχθεί εάν τα στοιχεία συλλογής είναι προσβάσιμα στον παίκτη και αν καταμετρούνται σωστά.

Αλληλεπίδραση με εχθρούς: Οι δοκιμαστικές συγκρούσεις με τους εχθρούς πρέπει να μας δείχνουν εάν αρχικά ο παίκτης τραυματίζεται έπειτα από επαφή στο πλάι με τους εχθρούς και επίσης αν τους καταστρέφει όταν πατά πάνω του.

Συμπεριφορά των εχθρών: Παρατηρώντας την κίνηση των εχθρών μπορεί να ελεγχθεί εάν αυτή είναι φυσιολογική. Εάν τα άλματα των βατράχων είναι σωστά και εάν οι αετοί αιωρούνται σταθερά έτσι ώστε να μην εμποδίζουν την κίνηση άλλων στοιχείων και να μπορούν να καταστραφούν από τον παίχτη.

Έλεγχος φυσικής άλματος και ταχύτητας κίνησης: Κατά την κίνηση του παίκτη πρέπει να ελεγχθεί αν οι πορείες που διανύει είναι ρεαλιστικές ή αν υπάρχει κάποιο σφάλμα το οποίο δημιουργεί κάποια ακραία περίσταση.

Πρόοδος στα επίπεδα: Πρόκειται για έλεγχο των σημείων αλλαγής επιπέδου-πίστας. Αυτά τα σημεία είναι ταμπέλες οι οποίες βρίσκονται στο τέλος κάθε πίστας. Ακουμπώντας αυτά τα σημεία πρέπει να ελεγχθεί αν οι μεταβάσεις πραγματοποιούνται και αν είναι σωστές.

Ακραίες περιπτώσεις: Είναι σημαντικό να ελεγχθούν κάποιες περιπτώσεις οι οποίες δεν είναι άμεσα εμφανείς στο παιχνίδι. Παραδείγματος χάρη πρέπει να γίνουν διάφορες συγκρούσεις υπό πιο περίεργες γωνίες που δεν θα χαρακτηρίζονταν συχνές με τους εχθρούς. Επίσης μπορεί κάποιο πλήκτρο που δεν είναι σχεδιασμένο για παρατεταμένη πίεση να πιεστεί παρατεταμένα για να διαπιστωθεί αν υπάρχει κάποιο περίεργο αποτέλεσμα. Δε θα ήταν σωστό να αιωρείται συνεχώς και να βγαίνει εκτός πίστας ο παίκτης με ένα παρατεταμένο πάτημα του πλήκτρου άλματος, επομένως μια τέτοια περίπτωση θα πρέπει να ελεγχθεί.

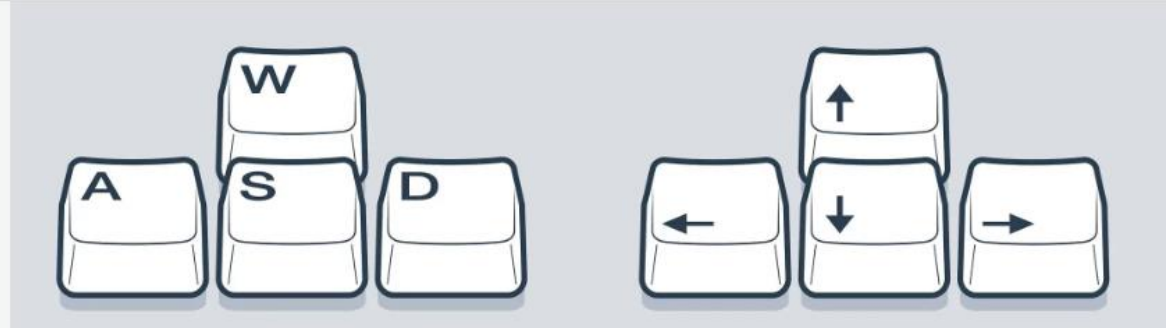
Διεπαφή χρήστη: Τα πλήκτρα της διεπαφής αλλά και ο έλεγχος των στοιχείων που υπάρχουν στην οθόνη πρέπει να ελέγχονται για να δίνουν το αναμενόμενο αποτέλεσμα.

Κεφάλαιο 4: Εγχειρίδιο παιχνιδιού

4.1 Περιγραφή της τελικής εφαρμογής

Εικόνα 6 – Αρχικό Menu

Μέσω του αρχικού Menu δίνονται 2 επιλογές η επιλογή “PLAY” με την οποία ξεκινά το παιχνίδι και η επιλογή “Quit” μέσω της οποίας γίνεται έξοδος από το λογισμικό.



Εικόνα 7 – Πλήκτρα κίνησης

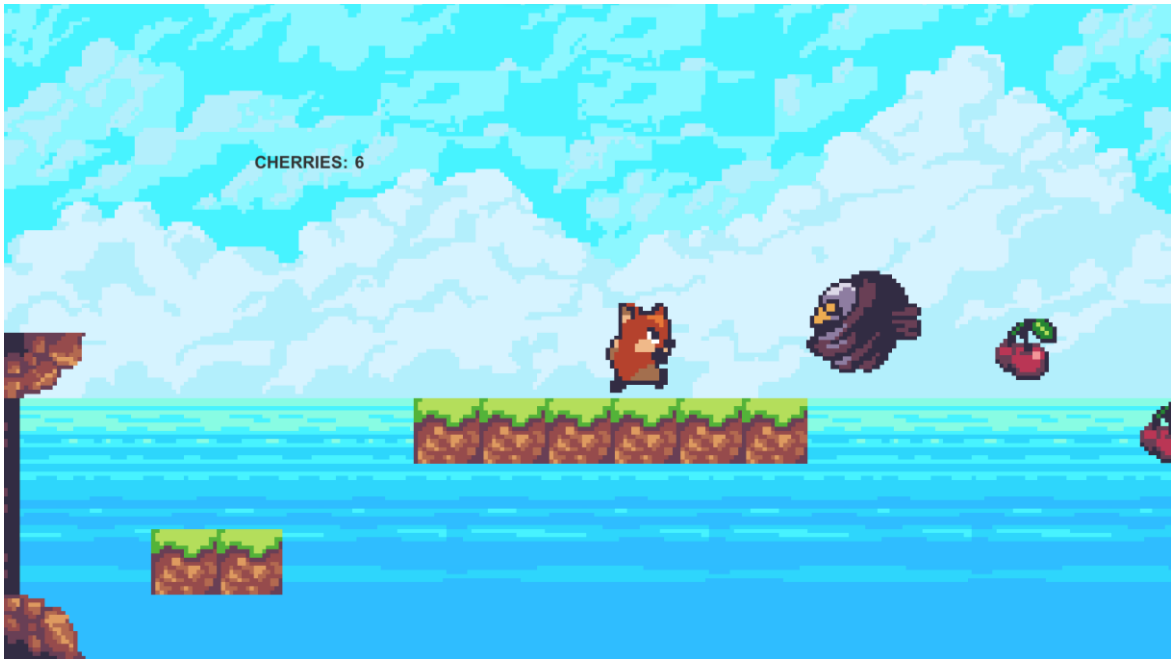
Τα βασικά πλήκτρα της κίνησης του χαρακτήρα του παιχνιδιού είναι τα βέλη του πληκτρολογίου μας ή εναλλακτικά τα πλήκτρα W,A,S,D. Γενικά είναι ένα σύνηθες μοντέλο το οποίο χρησιμοποιείται για την κίνηση. Ο χαρακτήρας εκτός από την κίνηση του δεξιά και αριστερά διαθέτει και τη δυνατότητα να πραγματοποιεί άλματα. Τα άλματα αυτά έχουν ως πλήκτρο πίεσης το spacebar του πληκτρολογίου.

Εικόνα 8 – Πλήκτρο άλματος

Ο στόχος του παίχτη είναι να οδηγήσει τον χαρακτήρα στο τέλος της κάθε πίστας, συλλέγοντας όσο δυνατόν περισσότερα κεράσια και καταστρέφοντας ή αποφεύγοντας εχθρούς έχοντας ταυτόχρονα την αποφυγή εμποδίων ή γκρεμών ως επιπλέον δοκιμασία. Τα επίπεδα είναι έξι σε αριθμό και όταν ο παίχτης καταφέρει να τα περάσει, να φτάσει στο τελευταίο επίπεδο και να καταφέρει να βρεθεί στο σπίτι του χαρακτήρα, εκεί τελειώνει και το παιχνίδι.

Εικόνα 9 – Επίπεδο 1

Το πρώτο επίπεδο έχει διδακτικό χαρακτήρα και βοηθάει τον παίχτη να εξοικειωθεί με τη λειτουργία του παιχνιδιού. Όπως βλέπουμε στην εικόνα 9 γίνεται η πρώτη επαφή με εμπόδια ή οριακά σημεία, όπως γκρεμούς που πρέπει να προσπεραστούν για την ολοκλήρωση του επιπέδου. Τα κεράσια είναι εκεί προς συλλογή. Κάθε πίστα τελειώνει όταν ο παίχτης φτάνει στο σημείο πινακίδας που οδηγεί στην επόμενη πίστα.



Εικόνα 10 – Προσέγγιση εχθρού

Η προσέγγιση ενός εχθρού είναι ένα στοιχείο το οποίο συναντάται σε όλα τα επίπεδα του παιχνιδιού. Υπάρχουν δύο τρόποι αντιμετώπισης του κινδύνου αυτού για τον παίχτη. Η αποφυγή ή η καταστροφή του ανάλογα με την περίπτωση και το περιβάλλον. Για να επιτευχθεί μια καταστροφή θα πρέπει ο παίχτης να πραγματοποιήσει ένα άλμα επάνω στον εχθρό. Σε αντίθετη περίπτωση ο εχθρός μπορεί να προσπεραστεί με μια εναλλακτική πορεία και να μην υπάρχει καμία αλληλεπίδραση με αυτόν. Εάν το άλμα δε γίνει σωστά ή ο εχθρός ακουμπήσει πρώτος τότε ο χαρακτήρας αναχαιτίζεται και δέχεται πλήγμα όπως βλέπουμε στην εικόνα 11. Οπότε σε κάθε περίπτωση πρέπει να υπάρχει η κατάλληλη τακτική ώστε να ξεπερνιούνται ή να καταστρέφονται οι εχθροί.

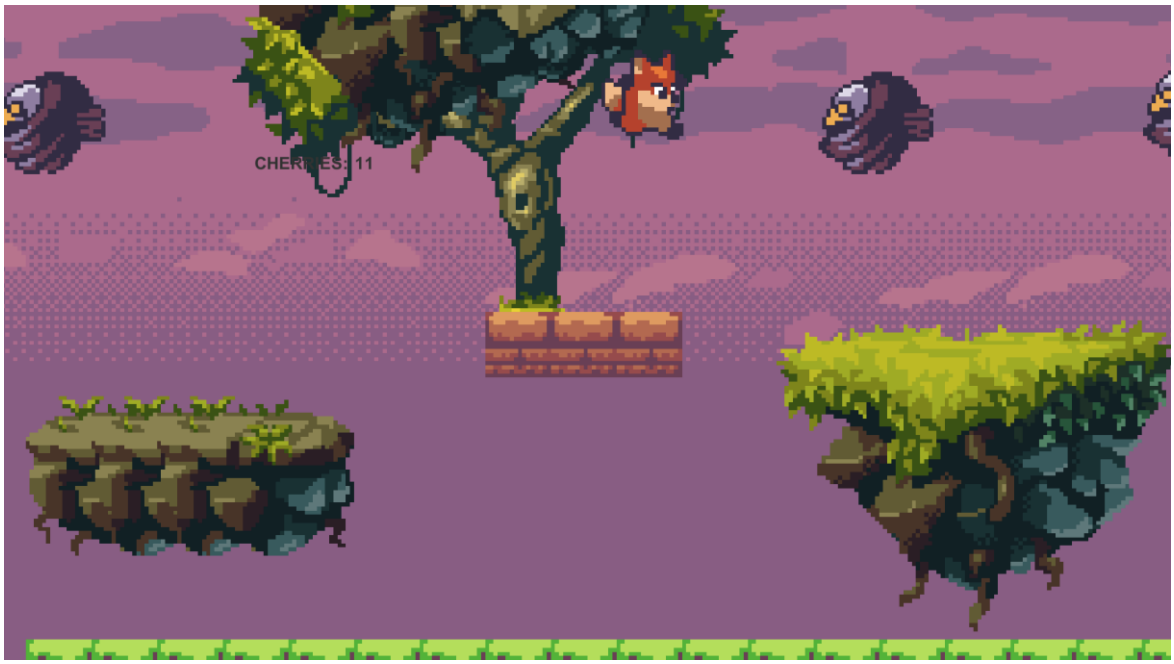
Εικόνα 11 – Ο εχθρός αναχαιτίζει το χαρακτήρα

Διαφορετικοί συνδυασμοί του περιβάλλοντος απαιτούν και τις αντίστοιχες κινήσεις για την ομαλή ολοκλήρωση του κάθε επιπέδου. Μερικές φορές μια συγκεκριμένη εναλλαγή κίνησης και αλμάτων είναι απαραίτητη για να ξεπεραστούν οριακά σημεία γκρεμών και εχθροί το ένα μετά το άλλο. Όπως βλέπουμε στην εικόνα 12 όπου έχουμε ένα στιγμιότυπο της 2^{ης} πίστας.

Εικόνα 12 – Πίστα 2: Οριακό σημείο με εχθρό

Οι πίστες 3 και 4 διαδραματίζονται σε νυχτερινό τοπίο οπότε βλέπουμε ένα διαφορετικό χρωματισμό στο περιβάλλον όπως βλέπουμε στις εικόνες 13 και 14. Επιπλέον προστίθενται περισσότερα οριακά σημεία και εμπόδια αλλά και περισσότεροι εχθροί, κάτι που κάνουν το επίπεδο δυσκολίας σε σημεία να ανεβαίνει από τη στιγμή που ο παίχτης έχει ήδη ένα καλό επίπεδο εξοικείωσης με το παιχνίδι από τα 2 προηγούμενα επίπεδα.

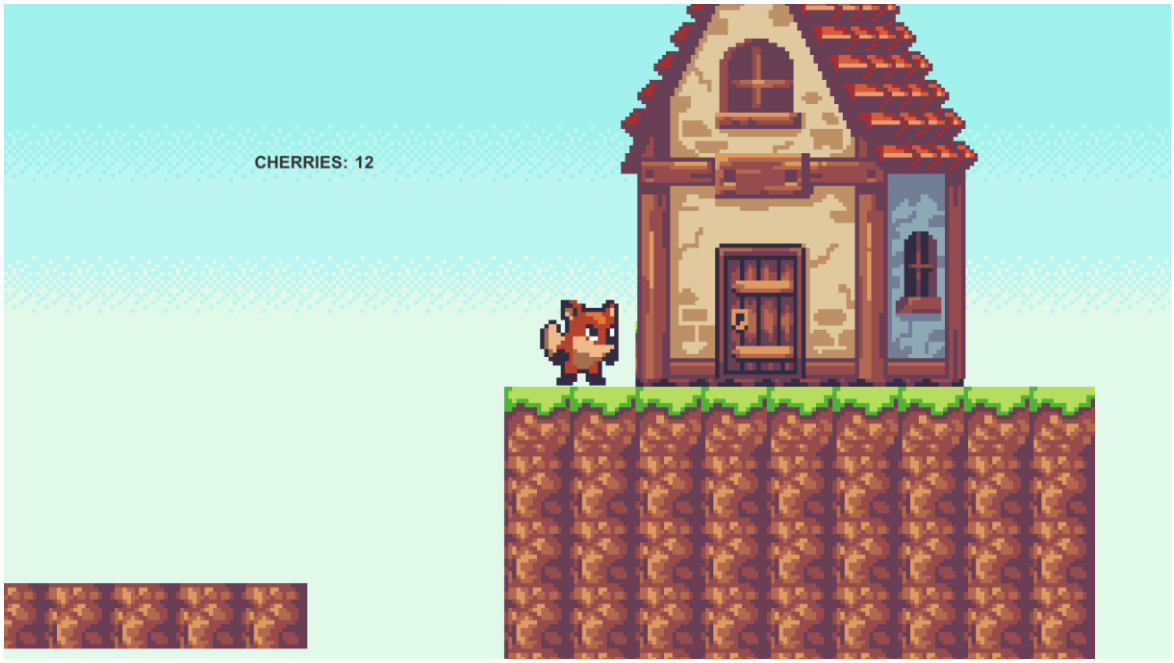
Εικόνα 13 – Πίστα 3



Εικόνα 14 – Πίστα 4

Εικόνα 15 – Προσέγγιση σε τέλος πίστας

Το τερματικό σημείο κάθε πίστας γίνεται μέσω της προσέγγισης της ειδικής ταμπέλας που βλέπουμε στην εικόνα 15. Όταν ο παίχτης ακουμπήσει αυτό το σημείο γίνεται η μετάβαση στο επόμενο επίπεδο. Όταν φτάσουμε στον τελικό μας στόχο δηλαδή το σπίτι του χαρακτήρα μας στο τελευταίο επίπεδο τότε έχουμε την ολοκλήρωση του παιχνιδιού όπως φαίνεται στην εικόνα 16.



Εικόνα 16 – Τερματισμός Παιχνιδιού!

4.2 Κώδικας

Στο παρών τμήμα παρατίθεται ο κώδικας ο οποίος χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής. Η γλώσσα είναι η C#.

Το πρώτο κομμάτι κώδικα είναι αυτό που χρησιμοποιείται για όλες τις λειτουργίες που αφορούν τον παίχτη. Η κίνηση, το άλμα, αλλά και οι αλληλεπιδράσεις με τους εχθρούς και τα αντικείμενα συλλογής υλοποιούνται στο συγκεκριμένο κομμάτι κώδικα μαζί με την ανάθεση συγκεκριμένων καταστάσεων απεικόνισης .

```
private void SetVelocityState()
    {

      float hDirection= Input.GetAxis("Horizontal");

        if (hDirection<0)
        {
           rb.velocity = new Vector2(-speed, rb.velocity.y);
           rb.transform.localScale= new Vector2(-1,1);

        }

        else if (hDirection>0)
        {
           rb.velocity = new Vector2(speed, rb.velocity.y);
           rb.transform.localScale= new Vector2(1,1);

        }


         if (Input.GetButtonDown("Jump")&&
coll.IsTouchingLayers(ground))
        {
            jump();
        }
```

Κώδικας 1

Στο παραπάνω κομμάτι του κώδικα βλέπουμε τη μέθοδο setVelocityState η οποία επιτρέπει την αλλαγή της κατεύθυνσης του παίχτη στον άξονα, επίσης ελέγχει την περίπτωση στην οποία πραγματοποιείται από την πλευρά του παίχτη η μέθοδος του άλματος. Εδώ αξίζει να σημειωθεί πως αυτός ο έλεγχος αποτελείται από 2 συνθήκες, η μια προφανώς και έχει να κάνει με την πίεση του πλήκτρου που έχει ανατεθεί στο άλμα αλλά ταυτόχρονα θα πρέπει και ο χαρακτήρας να πατάει στο έδαφος. Εάν δεν είχε πραγματοποιηθεί αυτού του είδους ο

έλεγχος τότε ο χαρακτήρας θα μπορούσε πραγματοποιεί άλματα ακόμα και στον αέρα κάτι το οποίο δε θέλουμε στη συγκεκριμένη υλοποίηση.

```
private void OnTriggerEnter2D(Collider2D collision)
   {
         if (collision.tag=="Collectable")
         {
            Destroy(collision.gameObject);
            cherries+=1;


            cherryText.text=cherries.ToString();
         }

  }
```

Κώδικας 2

Η μέθοδος `onTriggerEnter2D` στην προκειμένη περίπτωση όπως βλέπουμε στον κώδικα 2 ελέγχει αν ο παίκτης έχει σύγκρουση με ένα αντικείμενο συλλογής. Εάν αυτή η συνθήκη ισχύει, τότε καταστρέφεται το αντικείμενο αυτό μέσω της `Destroy` έτσι ώστε να μη φαίνεται στην οθόνη ένα αντικείμενο το οποίο έχει ήδη συλλεχθεί, και στη συνέχεια αυξάνει τον αριθμό των κερασιών και αποδίδει την τιμή τους σε μεταβλητή η οποία με τη σειρά της αποδίδει στην οθόνη τον μετρητή αυτό.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Enemy : MonoBehaviour
{
    protected Animator anim;
    protected Rigidbody2D rb;
    protected virtual void Start()
    {
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody2D>();
    }
  public void JumpedOn() // methodos gia thanato exthrou
   {
       anim.SetTrigger("Death");
   }
   private void Death()
   {
       Destroy(this.gameObject);
   }
}
```

Κώδικας 3

Στο κώδικα του πλαισίου 3 βλέπουμε την βασική κλάση για τους εχθρούς με το όνομα Enemy, η συγκεκριμένη κλάση κληρονομείται από τις κλάσεις Frog και Eagle όπως θα δούμε στη συνέχεια. Περιλαμβάνει τις μεθόδους της καταστροφής JumpedOn και Death οι οποίες θέτουν την κατάσταση απεικόνισης του θανάτου των εχθρών και καταστρέφουν το αντικείμενο του εχθρού όταν αυτός ηττηθεί.

```
// enallagi apo alma se ptwsi
    if (anim.GetBool("Jumping"))
    {
       if(rb.velocity.y < .1)
       {
        anim.SetBool("Falling",true);
        anim.SetBool("Jumping",false);
   }
  }


       // enallagi apo ptosi se idle
       if (coll.IsTouchingLayers(ground) && anim.GetBool("Falling"))
       {
        anim.SetBool("Falling",false);
}
}



    }
```

Κώδικας 4

Στην περίπτωση των εχθρών υπάρχει η ανάγκη να έχουμε εναλλαγή στις καταστάσεις απεικόνισης για ένα στοιχείο το οποίο δεν ελέγχεται από το χρήστη αλλά έχει προγραμματιστεί να κάνει κάποιες προκαθορισμένες κινήσεις. Στον κώδικα 4 έχουμε ένα κομμάτι από τον κώδικα κίνησης ενός εχθρού, και συγκεκριμένα του βατράχου.

Το συγκεκριμένο κομμάτι κώδικα απαιτείται καθώς ελέγχει σε ποια κατάσταση βρίσκεται ο βάτραχος και αντιστοίχως επιλέγει την κατάλληλη κατάσταση απεικόνισης στην οθόνη.

```
public class Frog : Enemy
```

Κώδικας 5

Όπως βλέπουμε στον κώδικα 5, εκτός από τη δήλωση της κλάσης του βατράχου έχουμε και την κληρονομικότητα μεταξύ της κλάσης αυτής αλλά και της κλάσης του εχθρού όπως αναφέρθηκε παραπάνω. Το ίδιο βέβαια ισχύει για την κλάση Eagle όπως βλέπουμε στον κώδικα 6. Η κληρονομικότητα είναι σημαντική στην ανάπτυξη του παιχνιδιού αυτού καθώς είναι χρήσιμο εργαλείο στα πλαίσια της οργάνωσης και της σωστής επαναχρησιμοποίησης του κώδικα. Όλοι οι εχθροί διαθέτουν κάποια κοινά γνωρίσματα – μεθόδους οπότε είναι χρήσιμο να διατηρούνται και να χρησιμοποιούνται ως κλάσεις – παιδιά της κλάσης Enemy.

```
public class Eagle : Enemy
```

Κώδικας 6

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class CameraController : MonoBehaviour

{

    public Transform Player;

    private void Update()

    {

        transform.position=new
Vector3(Player.position.x,Player.position.y,transform.position.z);

    }

}
```

Κώδικας 7

Στον κώδικα 7 έχουμε τον προγραμματισμό της κάμερας του παιχνιδιού. Είναι δεδομένο πως αυτό είναι απαραίτητο για να ακολουθείται ο παίκτης όσο εξερευνάει τα όρια του κάθε επιπέδου. Σε περίπτωση που δεν υπήρχε ένα τέτοιο κομμάτι κώδικα θα βλέπαμε μόνο την αρχική οθόνη του επιπέδου με τον χαρακτήρα να χάνεται και να μην είναι ορατός από τους χρήστες καθώς ξεπερνά τα όρια αυτά. Πιο συγκεκριμένα, για την κάμερα δημιουργείται ένας νέος πίνακας ο οποίος καταγράφει την κίνηση του παίχτη στους άξονες X και Y και την ακολουθεί έτσι ώστε όσο κινείται ο παίχτης να κινείται και η κάμερα βάσει αυτού.

```
public class Next : MonoBehaviour

{

    [SerializeField] private string scenetoLoad;


     private void OnTriggerEnter2D(Collider2D collision) {


     {

        if (collision.gameObject.tag == "Player")

        {

            SceneManager.LoadScene(scenetoLoad);


        }


     }

}
```

Κώδικας 8

Για την εναλλαγή των σκηνών του παιχνιδιού είναι απαραίτητο να υπάρχει μια μέθοδος μετάβασης στο επόμενο επίπεδο. Αυτό γίνεται μέσω της σύγκρουσης με το αντικείμενο πινακίδας που υπάρχει στο τέλος κάθε επιπέδου. Εάν η σύγκρουση συμβεί, τότε η μετάβαση στο επόμενο επίπεδο πραγματοποιείται βάσει της αντίστοιχης ρύθμισης για το κάθε επίπεδο μέσω του SceneManager στον Inspector του Unity.

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


public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene("SampleScene");
    }


    public void QuitGame()
    {
        Application.Quit();
    }
```

Κώδικας 9

Στον κώδικα 9 έχουμε τις μεθόδους που χρησιμοποιούνται στο menu χρήστη. Οι μέθοδος PlayGame ξεκινά το παιχνίδι από την πρώτη σκηνή, ενώ η QuitGame τερματίζει τη λειτουργία του λογισμικού.

Κεφάλαιο 5: Η εκπαιδευτική χρήση των βιντεοπαιχνιδιών

5.1 Εκπαίδευση

Σημαντικός είναι ο ρόλος των ηλεκτρονικών παιχνιδιών στην εκπαίδευση τις τελευταίες δεκαετίες και η χρήση τους σε πολλές περιπτώσεις αποτελεί μια λειτουργική μέθοδο που παράγει αποτελέσματα. Είτε μιλάμε για σχολική/ πανεπιστημιακή εκπαίδευση, είτε για χρήση προσομοιώσεων για την απόκτηση δεξιοτήτων τα παραδείγματα είναι πολλά. [20] Κάνοντας μια αρχική αναφορά στα παιχνίδια εντός σχολικού περιβάλλοντος εντοπίζεται η δυνατότητα χρήσης εκπαιδευτικού λογισμικού σε μορφή κουίζ γνώσεων. Με τον τρόπο αυτό ο μαθητευόμενος αποκτά το στοιχείο της διάδρασης καθώς καλείται να ελέγξει τις γνώσεις του μέσω διάφορων επιλογών. Μπορεί να επιλέξει από ένα πλήθος απαντήσεων ή να εισάγει μια δική του απάντηση ανάλογα με τον τρόπο που είναι σχεδιασμένο το εκάστοτε κουίζ. Χαρακτηριστικό παράδειγμα σε αυτή την κατηγορία είναι το National Geographic Challenge, παιχνίδι ιστορικών γνώσεων, γεγονότων και γεωγραφίας. [21]
Φυσικά το στήσιμο ενός βιντεοπαιχνιδιού κουίζ είναι μια διαδικασία η οποία μπορεί να γίνει μέσω του περιβάλλοντος Unity. Ένα υπόδειγμα στησίματος θα αποτελούνταν από πλήκτρα πίεσης τα οποία θα βοηθούσαν την επικύρωση της απάντησης του χρήστη στην εκάστοτε ερώτηση. Η εγκυρότητα της απάντησης θα ήταν εφικτό να ελεγχθεί μέσω κώδικα. Αν θεωρήσουμε πως οι ερωτήσεις είναι στοιχεία μιας λίστας τότε στην απλουστευμένη εκδοχή ενός τέτοιου παιχνιδιού κατά την οποία ζητείται μια απάντηση μεταξύ 2 επιλογών και συγκεκριμένα ψευδούς και αληθούς, η εκχώρηση Boolean τιμών σε αυτές θα βοηθούσε την υλοποίηση. Οι ερωτήσεις θα ήταν εφικτό να αλλάζουν ως διαφορετικές σκηνές. [22]
Εάν αναλύσουμε περεταίρω το παραπάνω παιχνίδι μπορούμε να το φανταστούμε ακόμα και ως μέρος μικρού παιχνιδιού(mini-game) εντός ενός άλλου παιχνιδιού. Βεβαίως και υπάρχουν τέτοια παραδείγματα στην αγορά, εάν ανατρέξουμε στο Canis Canem Edit (2006) μπορούμε να εντοπίσουμε πως ο πρωταγωνιστής ως μαθητής Λυκείου παρακολουθεί διάφορα μαθήματα. Ένα από αυτά είναι το μάθημα των Αγγλικών. Κατά το μάθημα αυτό παρουσιάζονται στον παίχτη διάφορα γράμματα τα οποία βρίσκονται μπερδεμένα στην οθόνη και αυτός καλείται να τα βάλει στη σωστή σειρά. Κάθε φορά που η επιλογή των κατάλληλων χαρακτήρων τον οδηγεί στο σχηματισμό μιας λέξης τότε ένας δείχτης αναπαριστά την πρόοδό του. Το όριο με το οποίο μπορεί να πάρει προβιβάσιμο βαθμό διαφέρει ανάλογα με την πρόοδο του παίχτη. Με την επιτυχία σε κάθε μάθημα εντός του μικρού παιχνιδιού ο παίχτης ανταμείβεται με αντικείμενα τα οποία τον βοηθούν στην εξέλιξη της ολοκλήρωσης του κυρίως παιχνιδιού. Η ύπαρξη μικρών παιχνιδιών εντός των παιχνιδιών μπορεί σε πολλές περιπτώσεις να έχει

εκπαιδευτική χρήση ακόμα κι αν ο κύριος σκοπός του παιχνιδιού δεν είναι απαραίτητα εκπαιδευτικός. [23]
Σε γενικές γραμμές και λαμβάνοντας υπ' όψη και τα παραπάνω παραδείγματα, η ενασχόληση με τα βιντεοπαιχνίδια αποτελεί έναν από τους παράγοντες που μπορούν να επηρεάσουν τις γνωστικές δεξιότητες αλλά και να βελτιώσουν την ικανότητα διαχείρισης χρόνου και την λήψη αποφάσεων υπό πίεση. Τα παραπάνω αποτελούν χαρακτηριστικά τα οποία μπορούν να συνδράμουν στη βελτίωση της εκπαιδευτικής διαδικασίας και στον τρόπο με τον οποίο κάποιος μαθητής καταφέρνει να αφομοιώσει ευκολότερα τη γνώση και να αποκτά την ικανότητα να είναι αποδοτικότερος. [24]

5.2 Προσομοίωση και απόκτηση δεξιοτήτων

Προσομοίωση και απόκτηση δεξιοτήτων
Η προσομοίωση είναι άλλος ένας από τους σκοπούς για τους οποίους χρησιμοποιούνται τα βιντεοπαιχνίδια και προσδίδουν όφελος μέσω της ανάπτυξης διάφορων δεξιοτήτων. Εστιάζοντας ξανά στο Unity, το λογισμικό έχει αναβαθμίσει τις δυνατότητες ανάπτυξης λογισμικού προσομοίωσης δίνοντας νέες δυνατότητες από το 2021. Αυτό είχε ως αποτέλεσμα τη συνεργασίας της εταιρίας με τη Volvo η οποία πραγματοποιεί ελέγχους για το λογισμικό του συστήματος αυτόνομης οδήγησης. [25]

Παραμένοντας στον τομέα του ελέγχου ενός οχήματος, μπορούμε να μεταφερθούμε στις πτήσεις. Ένα γνωστό παιχνίδι το οποίο προσομοιώνει την πτήση σε αεροσκάφη είναι το Flight Simulator της Microsoft. Η ρεαλιστική προσέγγιση του συγκεκριμένου τίτλου αποδεικνύεται από το ποσοστό του 88% των πιλότων που απάντησαν θετικά στο ότι βρίσκουν την συγκεκριμένη προσομοίωση βοηθητική όταν χρειάζεται να προσεγγίσουν αεροδρόμια τα οποία είναι για εκείνους άγνωστα. [26]
Σε άλλο ένα παράδειγμα παρατηρούμε την ύπαρξη δυνατότητας για προσομοίωση στον τομέα των αγροτικών εργασιών. Τα βιντεοπαιχνίδια μπορούν να βοηθήσουν στην απόκτηση δεξιοτήτων μέσω προσομοιώσεων πρακτικών εργασιών. Χαρακτηριστικό παράδειγμα αποτελεί το «Lawn Mowing Simulator» που κυκλοφόρησε το 2021 και αποτελεί προσομοίωση εργασίας με χορτοκοπτικό μηχάνημα. Το παιχνίδι διαθέτει αρκετά πραγματικά χορτοκοπτικά μηχανήματα υπό την άδεια χρήσης των εταιριών και ο παίκτης καλείται να διαχειριστεί την αυτή καθ' αυτή εργασία χωρίς όμως να ξεχνά τη διαχείριση μπαταρίας ή καυσίμου της συσκευής και τη σωστή επιλογή εξαρτημάτων ανάλογα με το έδαφος και τον τύπο του γρασιδιού. Σίγουρα βάσει αυτού του

παραδείγματος πολλά άλλα αντίστοιχα παιχνίδια μπορούν να αναπτυχθούν με εστίαση την αποτελεσματικότερη εργασία στον αγροτικό, καλλωπιστικό τομέα ή στη διαλογή καρπών. Το unity δίνει άπειρες δυνατότητες για την υλοποίηση αντίστοιχων ιδεών.

Συμπεράσματα

Τα συμπεράσματα που παράγονται από την υλοποίηση της παρούσας εργασίας, έπειτα από την υλοποίηση της εφαρμογής του παιχνιδιού πλατφόρμας δύο διαστάσεων αλλά και την συγγραφή του παρόντος κειμένου είναι χρήσιμα και μπορούν να βοηθήσουν οποιοδήποτε προγραμματιστή θα ήθελε να ασχοληθεί με το συγκεκριμένο τομέα ερευνητικά ή επαγγελματικά.

Αρχικά κατανοούμε πως το συγκεκριμένο είδος πρόκειται για ένα από τα προσοδοφόρα της βιομηχανίας ακόμα και το παρόν έτος, και παράλληλα γίνεται κατανοητό πως η αναβίωση του βοήθησε τα εναλλακτικά στούντιο παραγωγής και τις αντίστοιχες εταιρίες να εδραιωθούν στην αγορά. Το φιλικό προς το χρήστη περιβάλλον του Unity μας δείχνει πως οι λύσεις που προσφέρονται είναι ιδανικές για τους ερασιτέχνες προγραμματιστές που θέλουν να κάνουν το βήμα παρακάτω στη δουλειά τους και να στήσουν εξ’ ολοκλήρου ένα έργο το οποίο μπορεί να στηριχθεί οικονομικά και με μεθόδους χρηματοδότησης του κοινού σε περιπτώσεις έλλειψης κεφαλαίου κατά την παραγωγή. Στο επίπεδο του στησίματος, της ανάπτυξης και του προγραμματισμού το Unity αποδεικνύεται ως ένα αξιόπιστο λογισμικό το οποίο μπορεί να χρησιμοποιηθεί με άνεση ακόμα και από δημιουργούς που διαθέτουν ακόμα και μηδενική εμπειρία στην ανάπτυξη παιχνιδιών. Ένα εκτενές επίσημο έγγραφο οδηγιών καθώς και εκατοντάδες αντίστοιχα μαθήματα που είναι δωρεάν διαθέσιμα στο διαδίκτυο είναι αρκετά ως εφόδια για να παραχθεί ένα ολοκληρωμένο έργο.

Στον τομέα της εκπαίδευσης βλέπουμε πως κατά το παρελθόν έχουν αναπτυχθεί βιντεοπαιχνίδια τα οποία έχουν βοηθήσει την εκπαιδευτική διαδικασία. Αυτό μας οδηγεί στο συμπέρασμα πως το Unity πρόκειται για ένα εργαλείο το οποίο μπορεί προσφέρει προσαρμογή ενός περιβάλλοντος παιχνιδιού σε συγκεκριμένα μαθησιακά πρότυπα και να δώσει πολλές δυνατότητες σε εκπαιδευτικούς και εκπαιδευτές δεξιοτήτων. Επίσης η δυνατότητα ανάπτυξης λογισμικών προσομοίωσης είναι σίγουρα κάτι το οποίο βελτιώνει τις δεξιότητες σε επαγγελματικούς τομείς οι οποίοι απαιτούν μια σταδιακή μετάβαση από το εικονικό στο πραγματικό πεδίο.

Βάσει των παραπάνω, το Unity πρόκειται για ένα πανίσχυρο εργαλείο στη φαρέτρα των ανερχόμενων προγραμματιστών, όπως είναι οι απόφοιτοι του τμήματος μας, εργαλείο που μπορεί να τους εφοδιάσει με γνώσεις που ανοίγουν νέα μονοπάτια στον τομέα της ανάπτυξης λογισμικού και όχι μόνο.

Παράρτημα Α: Εκδόσεις Λογισμικών

Το λογισμικό που χρησιμοποιήθηκε για την ανάπτυξη του παιχνιδιού είναι το Unity στην έκδοση 2018.4.32f1

Η γλώσσα με την οποία γράφτηκε ο απαραίτητος κώδικας είναι η C#

Αναφορές

[1] W. L. Hosch, «electronic platform game,» Encyclopedia Britannica, 2018.

[2] A. Michaelas, «A Cultural Biography of Sonic the Hedgehog,» academia.edu, 2017.

[3] M. Byrd, «How the 2D Platformer Survived the 3D Revolution,» denofgeek.com, 2017.

[4] K. C. Georgina Collins, «For Shovelry! Register, Puns and Pitfalls in the Localisation of Yacht Club Games's 'Shovel Knight',» 2016.

[5] F. B. -. T. East, «Time for Braid: A Philosophical View of a Digital,» Students' Philosophical Society, 2015.

[6] games-stats.com, «Steam Marketing Tool,» games-stats.com, 2023.

[7] Nintendo, «SUPER MARIO BROS manual,» Nintendo, 1985.

[8] A. Gustafsson, «An Analysis of Platform Game Design: Implementation Categories and Complexity Measurements,» Linaeus University Sweden, 2014.

[9] Unity, «Unity Drives the Democratization of Development in 2016 With Eight Unite Conferences Globally,» Melissa Sheridan, 2016.

[10] S. Ahamed, «STUDY OF AN APPLICATION DEVELOPMENT ENVIRONMENT BASED ON UNITY GAME ENGINE,» International Journal of Computer Science & Information Technology (IJCSIT), 2020.

[11] R. Fine, «UnityScript's long ride off into the sunset,» Unity Blog, 2017.

[12] F. S. Lauren S.Ferro, (UNITY) unity 2d game development projects, Bermingham: Packt Publishing, 2018.

[13] Ansimuz, «Sunny Land,» ansimuz.itch.io , 2019.

[14] Unity, «Box Collider 2D,» Unity manual, 2016.

[15] Unity, «Creating a 2D game,» 2021.

[16] Unity, «Animation State,» Unity Manual, 2021.

[17] Unity, «Standard Events,» Unity Manual, 2021.

[18] A. Roe, «Unity 2018 - Platformer Tutorial 22: Enemy,» Youtube, 2018.

[19] Μ. Μ. Μάντακας, «Τεχνολογία Λογισμικού,» Τμήμα Μηχανικών Πληρφορικής , Άρτα, 2015.

[20] K. Squire, «Video games in education,» Int. J. Intell. Games & Simulation, Cambridge, MA. 02139, 2003.

[21] S. Brown, «Games That Kids Will Actually Like,» cnet.com, 2022.

[22] Brackeys, «How to make a Quiz Game in Unity (E03 CORE GAME) - Tutorial,» YouTube, 2016.

[23] C. Bradford, «Playing at bullying: The postmodern ethic of Bully (Canis Canem Edit),» Deakin University, Victoria, 2009.

[24] R. C. H. H. Charles Reynaldo, «Using Video Games to Improve Capabilities in Decision Making and Cognitive Skill: A Literature Review,» Procedia Computer Science, 2021.

[25] S. Chu, «Unity Launches Next Level Simulation Technology With Unity Simulation Pro And Unity SystemGraph,» San Francisco, 2021.

[26] W. Beckman, «Pilot perspective on the microsoft flight simulator for instrument training and proficiency,» International Journal of Applied Aviation Studies, 2009.