Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: Παιχνίδι τύπου racing στην Unity

Παλάσσης Δημήτριος (ΑΜ: 4004)

Επιβλέπων Καθηγητής: κ. Παχουλάκης Ιωάννης

Επιτροπή Αξιολόγησης:

Ημερομηνία Παρουσίασης:

Abstract

This thesis concerns the development of a vehicular combat videogame using the Unity Game Engine and the programming language c#, which the engine supports.

This thesis features a short reference to Unity, its environment, its history and some of its tools. Details of the game regarding the game development, the player's goal and the way the game is played are also presented. In this game, the player has to drive a car equipped with various weapons through two levels while they defeat enemy vehicles and avoid traps and obstacles. In certain points of the game, the car will transform to a helicopter, so the player can fight enemy aircrafts and avoid flying obstacles. The first level is about battling various enemies, wile the second features a boss fight with the player facing the boss and his troops.

Disclaimer: This game was made for academical purposes. I neither created nor own any of the assets, video or audio clips used for the development of the game, as they belong to their rightful owners. Action Man and Action Man logo belong to Hasbro. The game will not be used for any commercial purposes.

Σύνοψη

Η παρούσα πτυχιακή εργασία αφορά την ανάπτυξη ενός vehicular combat (μάχες με οχήματα) ηλεκτρονικού παιχνιδιού μέσω χρήσης της Unity Game Engine και της γλώσσας προγραμματισμού c# που υποστηρίζεται από αυτήν.

Σε αυτήν την πτυχιακή εργασία πραγματοποιείται μια σύντομη αναφορά στην Unity, στην ιστορία της και σε κάποια από τα εργαλεία της. Ακόμα, παρουσιάζονται λεπτομέρειες για την δημιουργία του παιχνιδιού, τον στόχο του παίκτη και τον τρόπο που παίζεται το παιχνίδι. Σκοπός του παιχνιδιού είναι ο παίκτης να οδηγήσει ένα αυτοκίνητο εξοπλισμένο με διάφορα όπλα μέσα σε δύο πίστες ενώ θα εξουδετερώνει εχθρικά οχήματα και θα αποφεύγει εμπόδια και παγίδες. Σε συγκεκριμένα σημεία το αυτοκίνητο θα μεταμορφώνεται σε ελικόπτερο, ώστε ο παίκτης να αντιμετωπίζει εχθρικά αεροσκάφη και να αποφεύγει ιπτάμενα εμπόδια. Η πρώτη πίστα αφορά την μάχη ενάντια σε διάφορους εχθρούς, ενώ η δεύτερη περιέχει την μάχη ενάντια στον "αρχηγό" της πίστας, με τον παίκτη να αντιμετωπίζει τον "αρχηγό" και τους στρατιώτες του.

Disclaimer: Η ανάπτυξη του παιχνιδιού έγινε για ακαδημαϊκούς σκοπούς. Δεν έφτιαξα ούτε μου ανήκουν τα στοιχεία του παιχνιδιού, τα βίντεο και τα ηχητικά κομμάτια, καθώς ανήκουν στους νόμιμους ιδιοκτήτες τους. Ο Action Man και το λογότυπό του ανήκουν στην Hasbro. Το παιχνίδι δεν θα χρησιμοποιηθεί για εμπορικούς σκοπούς.

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Περίληψη

Η παρούσα πτυχιακή εργασία αναφέρεται στην δημιουργία και κατασκευή ενός ηλεκτρονικού παιχνιδιού μέσω χρήσης της δωρεάν (Personal) έκδοσης της μηχανής και ανάπτυξης βιντεοπαιχνιδιών Unity[1]. Καλύπτονται έννοιες όπως το περιβάλλον και οι δυνατότητες που προσφέρει η Unity στον χρήστη και στο προγραμματιστικό μέρος της εργασίας. Η έκδοση που χρησιμοποιήθηκε για την διεκπεραίωση της εργασίας ήταν η 2017.1.1f1 Personal.

Για την υλοποίηση του παιχνιδιού χρησιμοποιήθηκαν έτοιμα 3D μοντέλα, ειδικά και ηχητικά εφέ, καθώς και μουσικά κομμάτια και ένα βίντεο εισαγωγής που βρέθηκαν στο Asset Store[2] της Unity και αλλού στο διαδίκτυο. Έγινε χρήση 2D οντοτήτων του προγράμματος όπως των Canvas, Slider, Text και Raw Image. Η γλώσσα προγραμματισμού που προτιμήθηκε και χρησιμοποιήθηκε ήταν η γλώσσα αντικειμενοστρέφιας C#.

1.2 Κίνητρο για την ανάπτυξη της εργασίας

Βασικό κίνητρο για την υλοποίηση της παρούσας εργασίας αποτελεί η θέληση και διάθεση για εκμάθηση, εξοικείωση και εξάσκηση στον τομέα δημιουργίας και ανάπτυξης ενός βιντεοπαιχνιδιού. Εκεί συμπεριλαμβάνονται έννοιες όπως την δημιουργία των οχημάτων που ελέγχει ο παίκτης, την κατασκευή κατάλληλου περιβάλλοντος αναπαράστασης του παιχνιδιού, την ανάπτυξη κεντρικού μενού και υπό-μενού, καθώς και την κατασκευή κατάλληλης διεπαφής παίκτη-παιχνιδιού.

Τέλος, η πτυχιακή αυτή εργασία έπαιξε σημαντικό ρόλο στην οργάνωση και υλοποίηση εκτενών projects και στην εξοικείωση με αυτά. Προσέφερε πολύτιμη εμπειρία και γνώσεις που πιθανότατα θα φανούν χρήσιμες στην υλοποίηση μεγαλύτερων και πιο απαιτητικών projects και έργων στο μέλλον.

1.3 Σκοπός και στόχοι εργασίας

Κύριος και βασικός σκοπός της εργασίας είναι η δημιουργία και ανάπτυξη ενός βιντεοπαιχνιδιού μέσω χρήσης της Unity. Στόχος είναι το ηλεκτρονικό παιχνίδι που θα προκύψει να είναι σωστά κατασκευασμένο και δομημένο ώστε να ικανοποιεί τις απαιτήσεις του παίκτη και να επιτύχει στον τομέα της διασκέδασης και της ψυχαγωγίας του.

Παράλληλα, στόχο αποτελεί και η εξάσκηση στην δημιουργία βιντεοπαιχνιδιών, η εκμάθηση διάφορων εννοιών σχετιζόμενων με αυτή και τέλος, η κατάκτηση εμπειρίας και απαραίτητων γνώσεων στον τομέα αυτόν.

1.4 Δομή Εργασίας

Η παρούσα πτυχιακή εργασία ξεκινά με μια ιστορική αναδρομή σχετικά με την Unity και πραγματοποιείται μια σύντομη αναφορά σε δημοφιλή και επιτυχημένα παιχνίδια που έχουν κατασκευαστεί με χρήση της μηχανής αυτής. Αφού αναφερθούν ονομαστικά οι διαφορές των διάφορων εκδόσεων, δίνεται βάση στο περιβάλλον της Unity και σε μερικά από τα εργαλεία που είναι διαθέσιμα στον χρήστη, έτσι ώστε ο αναγνώστης να αποκτήσει μια κεντρική ιδέα για την εφαρμογή και τις δυνατότητές της.

Στο κεφάλαιο που ακολουθεί περιγράφεται το είδος του παιχνιδιού και ο σκοπός του παίκτη. Δίνεται ιδιαίτερη βάση στα μέρη που αποτελούν το παιχνίδι, όπως τα μοντέλα, τις πίστες, τα στατιστικά του παίκτη, τα ηχητικά εφέ, τα στοιχεία διεπαφής, κι άλλα. Παρουσιάζονται και περιγράφονται τα μοντέλα του αυτοκινήτου και του ελικοπτέρου που ελέγχει ο παίκτης, τα μοντέλα

των εχθρικών οχημάτων που πρέπει να αντιμετωπίσει, τα εμπόδια που πρέπει να προσπεράσει, τα αντικείμενα που συλλέγει μέσα από τις πίστες για να αναπληρώσει τις τιμές των στατιστικών του και οι συνθήκες που τον οδηγούν στην ολοκλήρωση της εκάστοτε πίστας. Ακόμα, γίνεται αναφορά και στο κεντρικό μενού του παιχνιδιού, το μενού των ρυθμίσεων και στον τρόπο που αυτές επηρεάζουν το παιχνίδι, στο μενού της παύσης και στο βίντεο εισαγωγής πριν ξεκινήσει το παιχνίδι.

Στο τέταρτο κεφάλαιο παρουσιάζεται η δημιουργία του παιχνιδιού και η φιλοσοφία πίσω από αυτήν. Ιδιαίτερη έμφαση δίνεται στην κατασκευή των πιστών, των οχημάτων του παίκτη και του εχθρού και των βοηθητικών αντικειμένων χρήσει έτοιμων 3D μοντέλων και στην σύνθεση των διαφόρων μενού. Στο τέλος του κεφαλαίου περιγράφονται περιληπτικά τα πιο σημαντικά scripts του παιχνιδιού.

Στο πέμπτο κεφάλαιο περιγράφεται ο κώδικας των scripts που παρουσιάζονται στο προηγούμενο κεφάλαιο, ενώ στο τελευταίο παρουσιάζονται στιγμιότυπα από το τελικό αποτέλεσμα, όπως και τα εμπόδια που παρουσιάστηκαν κατά την διαδικασία της υλοποίησης. Τέλος, προτείνονται διάφορες ιδέες για μελλοντική επέκταση και εξέλιξη του παιχνιδιού.

2 Unity Game Engine

Η Unity αποτελεί μια μηχανή ανάπτυξης 3D και 2D βιντεοπαιχνιδιών (Game Engine) μέσω της οποίας μπορούν να δημιουργηθούν ηλεκτρονικά παιχνίδια για πολλαπλές πλατφόρμες (cross-platform). Κυκλοφόρησε στις 8 Ιουνίου 2005 από την Unity Technologies, και οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την κατασκευή της ήταν οι C++ και C#. Υποστηρίζει τις γλώσσες προγραμματισμού C# και Javascript.

Υπάρχουν 4 διαθέσιμα Licenses για την χρήση της μηχανής Unity, το Personal, Plus, Pro και Enterprise, με τις ομοιότητες και διαφορές τους να παρουσιάζονται στον παρακάτω πίνακα:

Πίνακας 1: Τα Licenses της Unity

Όνομα License	**Πλήρεις λειτουργίες μηχανής και πλατφόρμες**	**Splash Screen**	**Σειρά Cloud Build**	**Multiplayer**	**Χωρητικότητα Εσόδων**	**Αναφορά Επιδόσεων**	**Premium Support**	**Πρόσβαση σε πηγαίο κώδικα**	**Τιμή**
Personal	NAI	“Made With Unity” και προαιρετικό Custom Animation	Στάνταρντ	20 CCUs	100,000$	OXI	OXI	OXI	Δωρεάν
Plus	NAI	Custom Animation και/ή “Made with Unity”	Προτεραιότητα	50 CCUs	200,000$	NAI	OXI	OXI	35$ /Μήνα
Pro	NAI	Custom Animation και/ή “Made with Unity”	Σύγχρονα Builds	200 CCUs	Απεριόριστη	NAI	NAI	OXI	125$ /Μήνα
Enterprise	NAI	Custom Animation και/ή “Made with Unity”	Αφοσιωμένοι Build Agents	Custom Multiplayer	Απεριόριστη	NAI	NAI	NAI	Διαπραγματεύσιμη

CCUs = “Ταυτόχρονοι χρήστες”

Πλατφόρμες που υποστηρίζονται από την Unity για την ανάπτυξη βιντεοπαιχνιδιών:

- Windows
- MacOs
- Linux
- Android
- iOS
- Playstation 4
- Playstation Vita
- Xbox One
- Wii U
- 3DS
- Nintendo Switch
- Oculus Rift
- Playstation VR
- Steam VR
- Tizen
- Universal Windows Platform
- WebGL
- Google Cardboard
- Gear VR
- Windows Mixed reality
- Daydream
- Android TV
- Samsung Smart TV
- tvOS
- FireOS
- Facebook Gameroom

- Apple ARKit
- Google ARCore
- Vuforia

2.1 Δημοφιλή και επιτυχημένα βιντεοπαιχνίδια που αναπτύχθηκαν μέσω της Unity

- **Hearthstone**
  Ένα αρκετά δημοφιλές και επιτυχημένο multiplayer παιχνίδι με κάρτες. Το Hearthstone[3] δημιουργήθηκε και κυκλοφόρησε από την Blizzard Entertainment στις 11 Μαρτίου του 2014 για Windows, Mac και iOS και στις 15 Δεκεμβρίου της ίδιας χρονιάς για Android.

Σχήμα 1: Στιγμιότυπο από gameplay του Hearthstone

- **Pillars of Eternity**
  Ο πνευματικός διάδοχος των αγαπημένων και πολυβραβευμένων παιχνιδιών ρόλων Baldur's Gate και Icewind Dale το CRPG (computer role playing game) με το όνομα Pillars of Eternity[4] που αναπτύχθηκε από την Obsidian Entertainment. Κυκλοφόρησε στις 26 Μαρτίου του 2015 για Windows, OS X και Linux και στις 29 Αυγούστου του 2017 για Playstation 4 και Xbox One. Το 2015 απέσπασε τα βραβεία Best RPG και Spirit of the PC.

Σχήμα 2: Εξώφυλλο του Pillars of Eternity

- **Furi**
  Το Furi[5] αποτελεί ένα action single player παιχνίδι που φημίζεται για την δυσκολία του. Αναπτύχθηκε και κυκλοφόρησε από την The Game Bakers για Windows και Playstation 4 στις 5 Ιουλίου του 2016, για Xbox One στις 2 Δεκεμβρίου του 2016 και για Nintendo Switch στις 11 Ιανουαρίου του 2018.

Σχήμα 3: Στιγμιότυπο από gameplay του Furi

- **Hand of Fate**
  Το Hand of Fate[6] αποτελεί ένα single player action RPG που δημιουργήθηκε και κυκλοφόρησε από την Defiant Development στις 17 Φεβρουαρίου του 2017 για Linux, Windows, OS X, Playstation 4 και Xbox One.

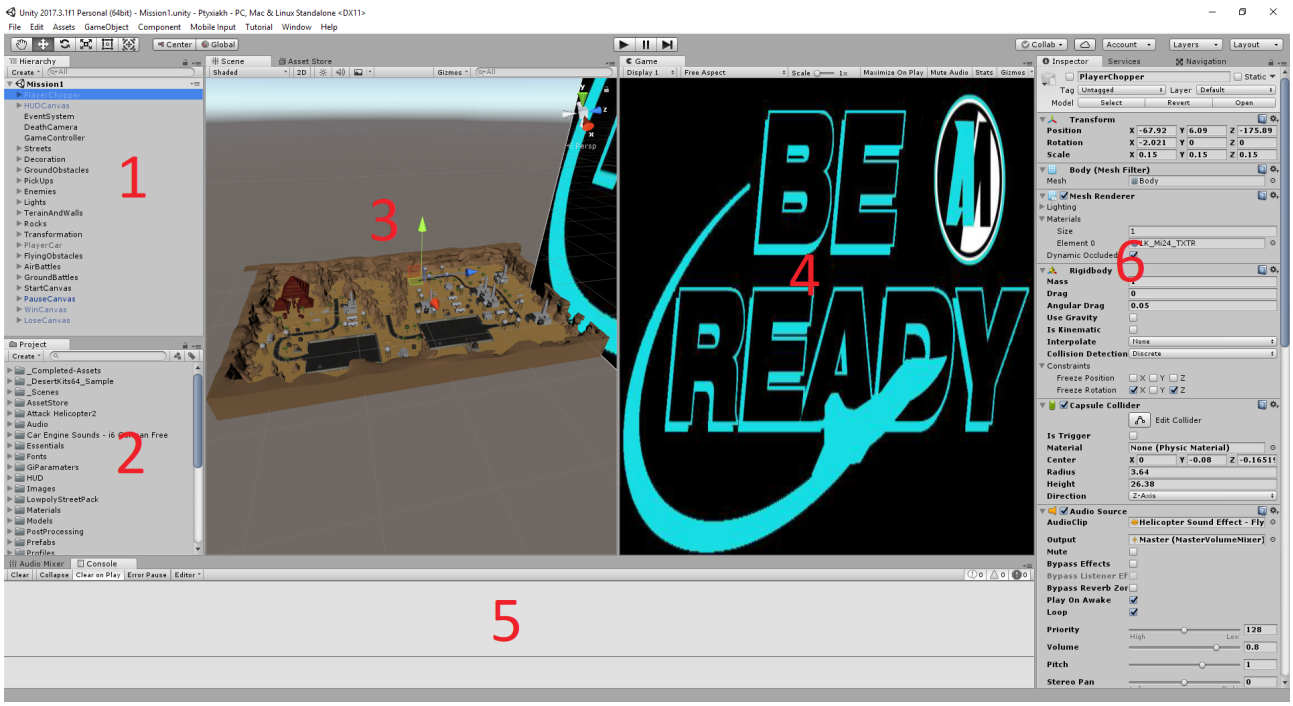
Σχήμα 4: Στιγμιότυπο από gameplay του Hand of Fate

- **Cuphead**
  Εμπνευσμένο από τα cartoons της δεκαετίας του 1930, το Cuphead[7] αναπτύχθηκε από μόλις τέσσερα άτομα και κυκλοφόρησε στις 29 Σεπτεμβρίου του 2017 για Windows και Xbox One. Το εν λόγω 2D βιντεοπαιχνίδι συνδύαζε στοιχεία από τις κατηγορίες side-scrolling shoot'em up και platformer, καθώς παρείχε την δυνατότητα για δύο παίχτες να παίξουν διπλό. Το Cuphead κατάφερε να αποσπάσει 23 βραβεία από διάφορα περιοδικά και gaming sites. Η indie εταιρία που ανέλαβε την ανάπτυξη και την κυκλοφορία του παιχνιδιού είναι η StudioMDHR.

Σχήμα 5: Στιγμιότυπο από gameplay του Cuphead

2.2 Το περιβάλλον της Unity

Στο παρακάτω σχήμα απεικονίζεται το εύχρηστο και φιλικό προς τον χρήστη περιβάλλον της Unity. Το περιβάλλον αυτό αποτελείται από αρκετά υποπαράθυρα με διαφορετικούς ρόλους και λειτουργίες το καθένα, τα οποία αναφέρονται και περιγράφονται παρακάτω.



Σχήμα 6: Το περιβάλλον της Unity

1. Hierarchy Window
2. Project Window
3. Scene Window
4. Game Window
5. Console Window
6. Inspector Window

Hierarchy Window

Στο Hierarchy Window βρίσκονται όλα τα 3D και 2D αντικείμενα που υπάρχουν μέσα στην σκηνή που επεξεργάζεται ο χρήστης. Αυτό το παράθυρο παρέχει η δυνατότητα στον χρήστη να τοποθετήσει τα αντικείμενα του σε μια ιεραρχία, για να κάνει την χρήση και αναζήτησή τους, καθώς και την χρήση της ίδιας της σκηνής πιο εύκολη. Η ιεραρχία των αντικειμένων που αναφέρεται σε αυτό το υποπαράθυρο, είναι του τύπου γονιός-παιδί.

Project Window

Το Project Window περιέχει σε φακέλους όλα τα assets που ανήκουν στο project στο οποίο εργάζεται ο χρήστης. Τα assets που βρίσκονται σε αυτό το υποπαράθυρο μπορεί να είναι από 3D ή 2D αντικείμενα, μέχρι ηχητικά κομμάτια, scripts με κώδικα, καθώς και ολόκληρες σκηνές παιχνιδιού.

Scene Window

Ίσως το υποπαράθυρο που προσφέρει στον χρήστη την πιο άμεση επαφή με το παιχνίδι που δημιουργεί. Το Scene Window περιέχει την τρέχουσα σκηνή παιχνιδιού που επεξεργάζεται ο χρήστης. Εκεί υπάρχει η δυνατότητα να τοποθετηθούν, αφαιρεθούν και να τροποποιηθούν με διάφορους τρόπους τα αντικείμενα που αποτελούν την σκηνή.

Game Window

Μέσω του Game Window ο χρήστης έχει την δυνατότητα να δοκιμάσει και να τεστάρει άμεσα το αποτέλεσμα των ενεργειών του πάνω στο παιχνίδι που δημιουργεί, χωρίς να χρειάζεται να το εξάγει

σε εκτελέσιμη μορφή, πρώτα. Αυτό το υποπαράθυρο προσφέρει μια προσομοίωση του τρέχοντος σταδίου του παιχνιδιού, παρουσιάζοντας το από την οπτική γωνία του παίκτη.

Console Window

Από το Console Window ο χρήστης μπορεί να εντοπίσει τυχόν λάθη στον κώδικά του και στην μεταχείριση των αντικειμένων του, αφού αυτό το υποπαράθυρο του παρέχει μηνύματα σφαλμάτων και προειδοποιήσεων για τις ενέργειές του.

Inspector Window

Με χρήση του Inspector Window ο χρήστης έχει την δυνατότητα να προσθέσει επιπλέον στοιχεία στα αντικείμενα του παιχνιδιού, να τα απενεργοποιήσει και ενεργοποιήσει τροποποιήσει και να τους δώσει παραπάνω ιδιότητες και συμπεριφορές.

2.3 Βασικές έννοιες της Unity

Σε αυτήν την υποενότητα θα γίνει μια σύντομη περιγραφή και επεξήγηση ορισμένων βασικών εννοιών της Unity.

Game Object

Με τον όρο Game Object περιγράφονται όλα τα αντικείμενα που υπάρχουν και χρησιμοποιούνται σε μια σκηνή. Τα Game Objects δεν μπορούν να έχουν ιδιότητες από μόνα τους, για αυτό πρέπει να τις κληρονομούν από τα components που περιέχουν.

Prefab

Prefabs ονομάζονται τα αντικείμενα που υπάρχουν αποθηκευμένα στους φακέλους του παιχνιδιού με σκοπό να δημιουργούνται άμεσα αντίγραφα αυτών μέσα στην σκηνή για να χρησιμοποιηθούν. Μια αλλαγή στο αποθηκευμένο αντικείμενο συνεπάγεται την ίδια αλλαγή και στα αντίγραφά του, όμως τα αντίγραφα αυτά είναι δυνατόν να τροποποιηθούν χωρίς να επηρεάσουν το prefab τους.

Component

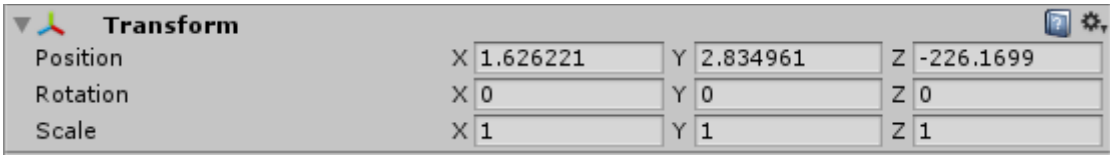
Τα “εργαλεία” τα οποία χρησιμοποιούνται για να προσθέσουν κάποια ιδιότητα και συμπεριφορά σε ένα Game Object ονομάζονται Components. Ο χρήστης έχει την δυνατότητα να προσθέσει όσα Components επιθυμεί σε ένα game Object, ή και κανένα, χωρίς να επιβάλλεται κάποιος περιορισμός.

2.3.1 Σημαντικά Components της Unity

Η Unity προσφέρει στον χρήστη μια μεγάλη γκάμα από Components που παρέχουν αρκετές και διάφορες ιδιότητες στα Game Objects στα οποία προστίθενται μέσω του υποπαράθυρου Inspector Window. Παρακάτω θα αναφερθούν τα σημαντικότερα από αυτά.

Transform

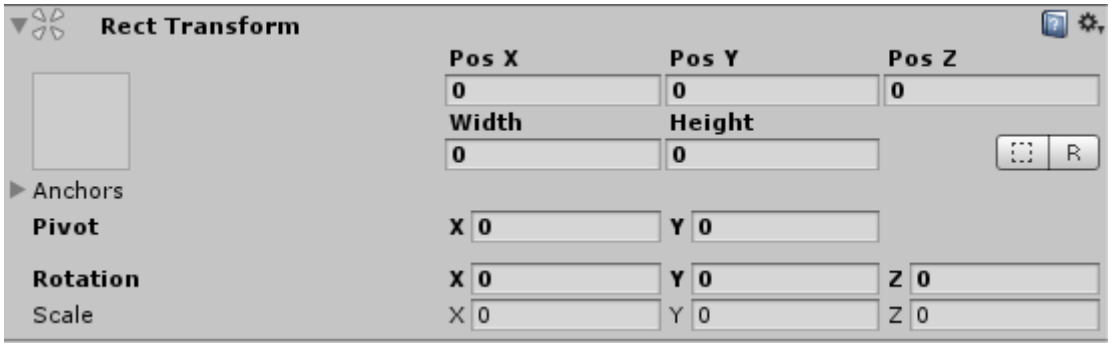
Ένα Component που υπάρχει σε όλα τα Game Objects της Unity. Το Transform είναι υπεύθυνο για την θέση, περιστροφή και μέγεθος του αντικειμένου μέσα στην σκηνή.



Σχήμα 7: Το Transform Component

Rect Transform

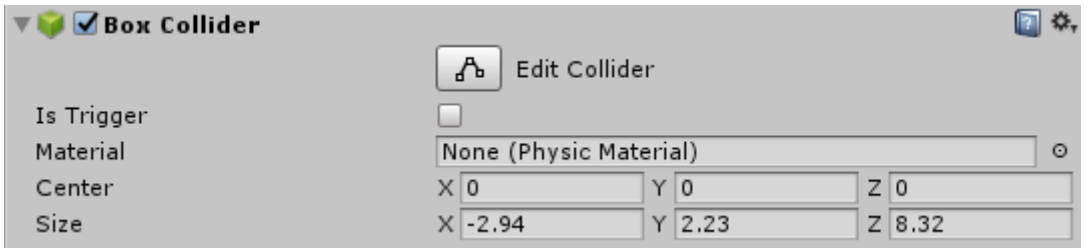
Το Rect Transform αντιπροσωπεύει το αντίστοιχο Component του Transform για UI Elements σε 2D τοποθέτηση και διάταξη.



Σχήμα 8: Το Rect Transform Component

Collider

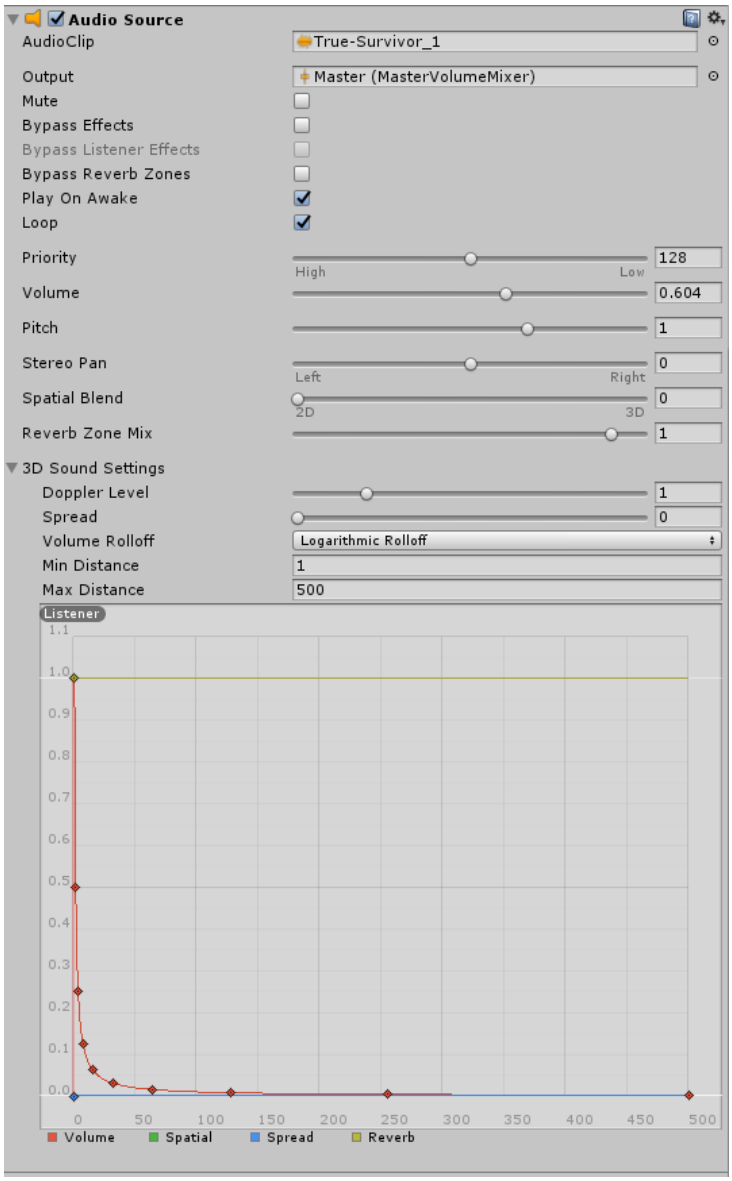
Τα Colliders προστίθενται σε κάποιο Game Object για να προσδιοριστούν οι συγκρούσεις του με άλλα αντικείμενα. Στον 3D χώρο κατατάσσονται οι κατηγορίες Box Colliders, Sphere Colliders, Capsule Colliders, Wheel Colliders και Terrain Colliders, ενώ στον 2D υπάρχουν οι Box Collider 2D, Polygon Collider 2D, Circle Collider 2D και Edge Collider 2D. Η κάθε κατηγορία συνδέεται με το αντίστοιχο σχήμα, και προσθέτει διαφορετικές ιδιότητες στα αντικείμενα. Στα Colliders υπάρχει η επιλογή να μετατραπούν σε Triggers. Αν ένα Collider γίνει Trigger τότε χάνεται το στοιχείο της σύγκρουσης αλλά αν ένα άλλο αντικείμενο έρθει σε επαφή μαζί του, τότε μπορεί να πυροδοτήσει ένα γεγονός, όπως την αναπαραγωγή ενός ήχου, ή την εμφάνιση ενός άλλου αντικειμένου.



Σχήμα 9: Το Box Collider Component

Audio Source

Audio Source ονομάζεται το Component που συνδέεται με την αναπαραγωγή ενός ηχητικού αντικειμένου. Αυτά τα ηχητικά αντικείμενα ονομάζονται Audio Clips. Το Audio Source μπορεί να ρυθμιστεί ποικιλοτρόπως, π.χ. για να αναπαράγει το Audio Clip σε λούπα ή για να το αναπαράγει αμέσως όταν ενεργοποιηθεί στην σκηνή το Game Object στο οποίο έχει προστεθεί.



Σχήμα 10: Το Audio Source Component

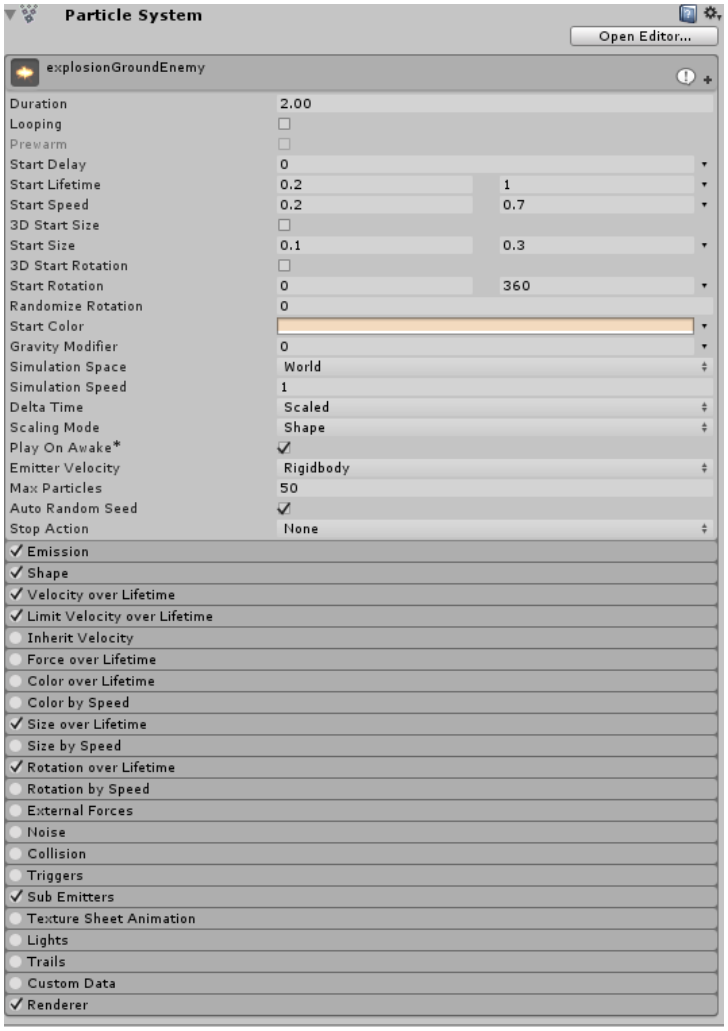
Audio Listener

Τα Audio Listener Components τοποθετούνται συνήθως σε Camera Game Objects και είναι απαραίτητα για την αναπαραγωγή ενός Audio Source. Σκοπός τους είναι να λαμβάνουν τον ήχο που εκπέμπουν τα Audio Sources και να τον αναπαράγουν.

Σχήμα 11: Το Audio Listener Component

Particle System

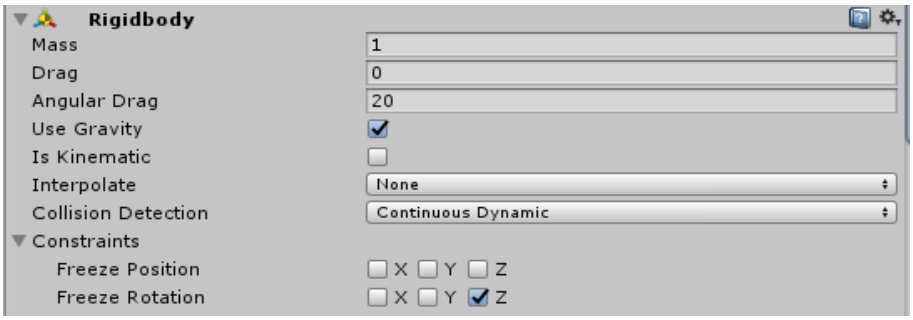
Το Particle System χρησιμοποιείται για την δημιουργία ειδικών εφέ, αφού συνδέεται με την αναπαράσταση μη στερεών αντικειμένων όπως εκρήξεων και καπνού.



Σχήμα 12: Το Particle System Component

Rigidbody

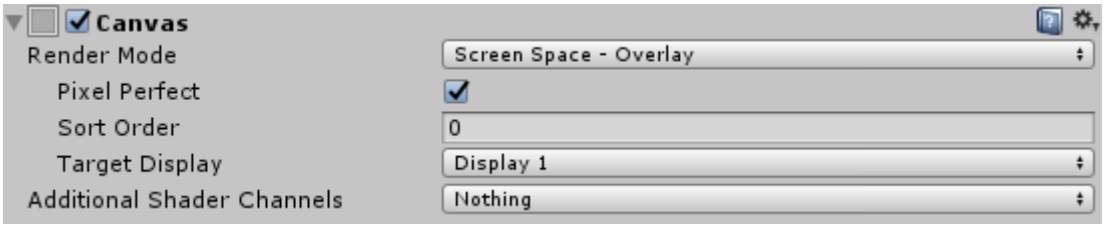
Όταν το Rigidbody προστεθεί σε ένα Game Object, του δίνει την δυνατότητα να επηρεάζεται από την φυσική, πιο συγκεκριμένα από ροπές και δυνάμεις, δίνοντάς του έτσι έναν ρεαλιστικό τρόπο κίνησης.



Σχήμα 13: Το Rigidbody Component

Canvas

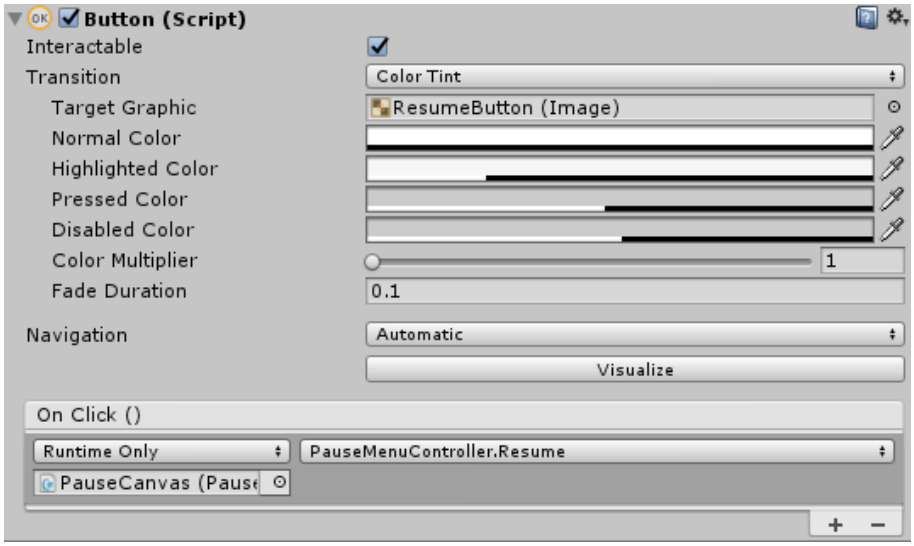
Το Canvas Component δημιουργεί έναν 2D καμβά στο παιχνίδι, πάνω στον οποίον μπορούν να τοποθετηθούν διάφορα UI Elements όπως κουμπιά, sliders, και εικόνες.



Σχήμα 14: Το Canvas Component

Button

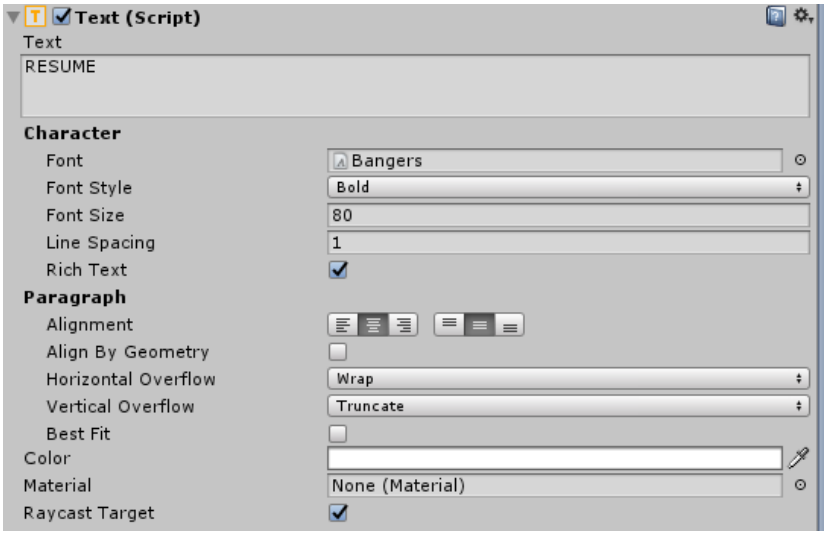
Το Button Component αποτελεί το στοιχείο ενός κουμπιού στο οποίο ο χρήστης έχει την δυνατότητα να ορίσει μια ενέργεια που θα εκτελείται όταν πατηθεί. Τα κουμπιά συνήθως περιέχουν μια εικόνα για φόντο, καθώς και ένα πολύ σύντομο κομμάτι κειμένου (συνήθως μια ή δυο λέξεις) που περιγράφει την λειτουργικότητά τους.



Σχήμα 15 Το Button Component

Text

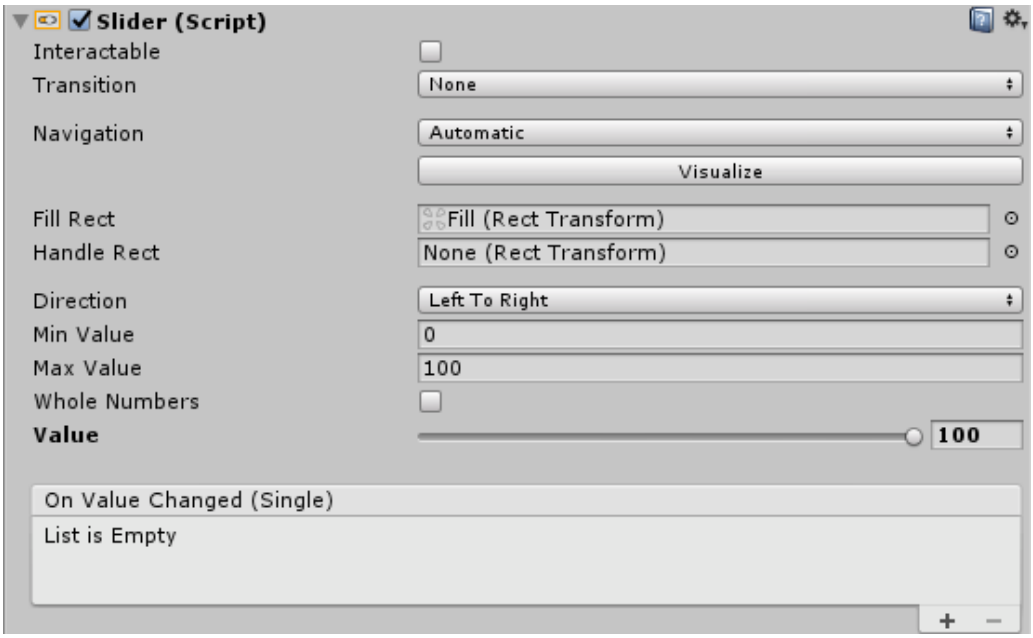
Το Text Component απεικονίζει απλά ένα κομμάτι κειμένου μέσα στο παιχνίδι. Μία από τις δυνατότητες που έχει ο χρήστης είναι να αλλάξει τύπο γραμματοσειράς, μέγεθος και χρώμα στο κείμενο που έχει ορίσει.



Σχήμα 16: Το Text Component

Slider

Slider ονομάζεται το αντικείμενο μιας μπάρας που υπάρχει σε έναν καμβά. Η μπάρα αυτή γεμίζει και αδειάζει αναπαριστώντας την αλλαγή στο περιεχόμενο μιας τιμής. Συνήθως χρησιμοποιείται για την απεικόνιση της ζωής του παίκτη, ή για την πρόοδο φόρτωσης μιας πίστας. Τα sliders συνοδεύονται και από έναν πήχη ο οποίος δίνει την δυνατότητα στον παίκτη να αλλάζει την τιμή της μπάρας χειροκίνητα. Μπορεί όμως να αφαιρεθεί.



Σχήμα 17: Το Slider Component

3. Εισαγωγή στο παιχνίδι

Στο παρόν κεφάλαιο παρουσιάζονται και περιγράφονται τα στοιχεία και οι μηχανισμοί που απαρτίζουν το κεντρικό παιχνίδι. Δίνεται βάση στα δύο οχήματα που ελέγχει ο παίκτης, στις ιδιότητές τους, στους γειωμένους και ιπτάμενους εχθρούς που πρέπει να αντιμετωπιστούν, καθώς επίσης και στον "αρχηγό". Στην συνέχεια αναφέρονται τα αντικείμενα που συλλέγει ο παίκτης, ο τρόπος με τον οποίο επηρεάζονται τα στατιστικά του, στις παγίδες που υπάρχουν μέσα στις πίστες και τέλος το περιβάλλον των ίδιων των πιστών.

3.1 Τα οχήματα του παίκτη

Τα οχήματα που ελέγχει ο παίκτης σε αυτό το παιχνίδι είναι δύο. Ένα γρήγορο αυτοκίνητο και ένα πολεμικό ελικόπτερο εξοπλισμένα με οπλικά συστήματα.

Και οι δύο πίστες του παιχνιδιού ξεκινούν δίνοντας στον παίκτη τον έλεγχο του αυτοκινήτου, και σε ορισμένα σημεία μέσα στις πίστες, το αυτοκίνητο μεταμορφώνεται σε ελικόπτερο, για να περάσει από δυσπρόσιτα σημεία όπου δεν υπάρχει άσφαλτος. Μόλις ξεπεραστεί ο κίνδυνος και ο παίκτης πλησιάζει πάλι σε τμήματα δρόμου, το ελικόπτερο θα μεταμορφωθεί σε αυτοκίνητο και θα συνεχίσει πάνω στην άσφαλτο. Οι διαφορές τους αφορούν αφενός στην εξωτερική τους εμφάνιση, και αφετέρου στον τρόπο με τον οποίο ελέγχονται και δρουν.

Η βασική τους ομοιότητα εντοπίζεται στα όπλα που φέρουν, καθώς και στο αυτόματο γκάζι που διαθέτουν, το οποίο κάνει και τα δύο οχήματα να προχωρούν συνεχώς προς την κατεύθυνση που είναι στραμμένα. Ένα κανόνι που εκτοξεύει βλήματα για να βλάψουν μερικώς τους εχθρούς τους, συνοδευόμενο από ένα σύστημα εκτόξευσης πυραύλων που τους καταστρέφουν με ένα χτύπημα είναι τα όπλα τα οποία παρέχονται στο ελικόπτερο και στο αυτοκίνητο. Το κανόνι αποτελεί το βασικό όπλο και πυροβολεί χωρίς περιορισμούς, ενώ το σύστημα πυραύλων δίνει περιορισμένο αριθμό βολών, ο οποίος ανανεώνεται από την συλλογή ενός συγκεκριμένου αντικειμένου μέσα στο παιχνίδι.

Οι διαφορές που παρουσιάζονται μεταξύ τους συναντώνται, αρχικά, στους άξονες κίνησης. Το αυτοκίνητο μπορεί να κινηθεί προς τα εμπρός και να αλλάξει κατεύθυνση στρίβοντας, ενώ το ελικόπτερο διαθέτει επίσης αυτές τις δυνατότητες, αλλά ταυτόχρονα μπορεί να κινηθεί και κατακόρυφα. Επίσης, το αυτοκίνητο μπορεί να φρενάρει, μειώνοντας έτσι την ταχύτητά του σχεδόν στο μισό, ενώ το ελικόπτερο δεν έχει αυτήν την δυνατότητα.

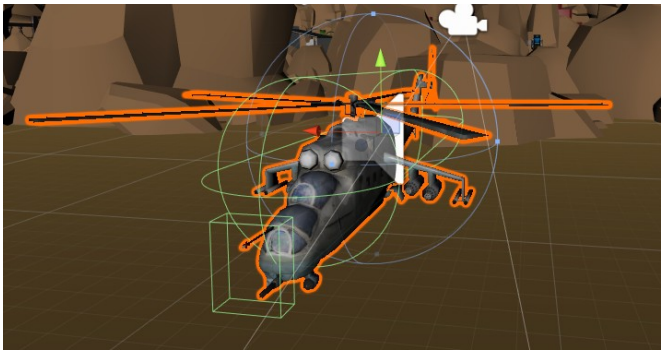
Παράλληλα, το αυτοκίνητο κινείται, προφανώς, στο έδαφος, συγκεκριμένα σε τμήματα ασφάλτινου δρόμου, ενώ το ελικόπτερο κινείται στον αέρα, σε μέρη όπου δεν υπάρχουν δρόμοι και το αυτοκίνητο αδυνατεί να φτάσει.

Ταυτόχρονα, και τα δύο οχήματα διαθέτουν κινούμενα μέρη και εκπέμπουν ήχους ενώ είναι σε λειτουργία. Από το αυτοκίνητο μπορούν να ακουστούν οι ήχοι της μηχανής και των λάστιχων και να παρατηρηθεί η κίνηση και κύλιση των τροχών, ενώ από το ελικόπτερο ξεχωρίζουν ο ήχος και η περιστροφή του έλικα.

Ακόμα, σε αντίθεση με το ελικόπτερο, το αυτοκίνητο διαθέτει μπροστινούς και πίσω προβολείς, οι οποίοι είναι διαρκώς ενεργοποιημένοι, ενώ οι πίσω λάμπουν περισσότερο όταν ο παίκτης φρενάρει.

Τέλος, το κάθε όχημα διαθέτει μια ξεχωριστή ιδιότητα. Όσο ο παίκτης ελέγχει το αυτοκίνητο, με το πάτημα του κατάλληλου πλήκτρου το θέτει σε λειτουργία drift, κατά την οποία το αυτοκίνητο θα στρίβει λίγο παραπάνω από το συνηθισμένο όριο, με αποτέλεσμα να εισέρχεται στις στροφές "με το πλάι", διατηρώντας ταυτόχρονα την ταχύτητα και την ισορροπία του. Η τακτική αυτή αποδεικνύεται χρήσιμη στις περιπτώσεις που ο παίκτης καταδιώκεται από εχθρούς και επιθυμεί να γυρίσει να τους αντιμετωπίσει, ενώ ταυτόχρονα του δίνει την δυνατότητα να εκτελέσει μια κανονική στροφή με πιο εντυπωσιακό τρόπο. Η ξεχωριστή ιδιότητα του ελικοπτέρου παρουσιάζεται σαν σύστημα αποφυγής. Πάλι με το πάτημα του κατάλληλου πλήκτρου, το ελικόπτερο μπορεί να κινείται στο πλάι για να αποφεύγει εμπόδια και εχθρικά πυρά.

Σχήμα 18: Το αυτοκίνητο του παίκτη



Σχήμα 19: Το ελικόπτερο του παίκτη

3.2 Εχθροί

Οι εχθροί του παιχνιδιού χωρίζονται σε τρεις κατηγορίες. Τους ιπτάμενους εχθρούς και τους εχθρούς του εδάφους, οι οποίοι αντιμετωπίζονται από το ελικόπτερο και από το αυτοκίνητο αντίστοιχα, ενώ υπάρχει και ο "αρχηγός" που αντιμετωπίζεται και από τα δύο οχήματα. Καθένας από αυτούς διαθέτει τα δικά του όπλα, τα οποία χρησιμοποιεί για να εξουδετερώσει τον παίκτη. Σε κάθε περίπτωση, όλοι οι εχθροί εμφανίζονται μόλις ο παίκτης περάσει από ένα Trigger Collider. Υπάρχουν έξι εχθρικά οχήματα σε αυτό το παιχνίδι, με τα τρία από αυτά να είναι του εδάφους, και τα άλλα τρία (συμπεριλαμβανομένου και του "αρχηγού") να είναι ιπτάμενα.

Οι ιπτάμενοι εχθροί παρουσιάζουν την ίδια συμπεριφορά καθ' όλη την διάρκεια του παιχνιδιού, η οποία τους υπαγορεύει να κινούνται συνεχώς προς μια κατεύθυνση εκτοξεύοντας πυρά κατά του παίκτη. Οι εν λόγω εχθροί κινούνται πάντα κατά πάνω στον παίκτη, ο οποίος έχει την επιλογή είτε να τους εξοντώσει, είτε να τους αποφύγει. Μπορεί, ακόμα, να συγκρουστεί μαζί τους, έτσι θα προκληθεί ένα μικρό ποσοστό ζημιάς στον παίκτη, αλλά ο εχθρός που θα εμπλακεί στην σύγκρουση θα καταστραφεί. Όλα τα εχθρικά οχήματα αυτής της κατηγορίας ελέγχονται από το ίδιο script, και έχουν περιορισμένο χρόνο ζωής, έτσι ώστε αν ο παίκτης επιλέξει να τους αποφύγει, θα εξαφανιστούν από την πίστα μετά από λίγα δευτερόλεπτα, αφού δεν παρουσιάζουν κάποια άλλη χρησιμότητα.

Αντίθετα, η συμπεριφορά των εχθρών του εδάφους διαφέρει αναλόγως την πίστα στην οποία βρίσκεται ο παίκτης. Στην πρώτη πίστα λαμβάνουν χώρα συνηθισμένες μάχες οχημάτων μέσα σε αρένες, με τους εχθρούς να ακολουθούν τον παίκτη όπου πηγαίνει, πυροβολώντας τον ταυτόχρονα, ενώ ο παίκτης δεν μπορεί να φύγει από την αρένα αν δεν τους καταστρέψει, έτσι δεν έχει άλλη επιλογή από το να τους πολεμήσει. Σε αυτές τις μάχες οι εχθροί είναι προγραμματισμένοι να πυροβολούν μόνο αν είναι στραμμένοι προς τον παίκτη. Τα δύο από τα τρία οχήματα ελέγχονται από το ίδιο script, ενώ το τρίτο ελέγχεται από διαφορετικό. Όμως η μοναδική διαφορά που εντοπίζεται σε αυτά τα δύο scripts, είναι ότι τα πυρά των δύο πρώτων εχθρών πηγάζουν από ένα κανόνι, ενώ του τρίτου εχθρού από δύο.

Έπειτα, στην δεύτερη πίστα του παιχνιδιού, οι εχθροί του εδάφους παρουσιάζουν την ίδια συμπεριφορά με τους ιπτάμενους εχθρούς. Πιο συγκεκριμένα, κινούνται συνεχώς προς μια κατεύθυνση, κατά πάνω στον παίκτη, εκτοξεύοντας ασταμάτητα σφαίρες, ενώ έχουν και εκείνοι περιορισμένο χρόνο ζωής. Όπως συμβαίνει με τους ιπτάμενους εχθρούς, ο παίκτης έχει την δυνατότητα να συγκρουστεί μαζί τους στην δεύτερη πίστα, φέροντας τα ίδια αποτελέσματα. Ελέγχονται από το ίδιο script των ιπτάμενων εχθρών.

Παράλληλα, ο "αρχηγός" εμφανίζεται μόνο στην δεύτερη πίστα και διαθέτει δύο ρόλους. Ο πρώτος από αυτούς εξυπηρετεί καθαρά σκοπούς εμφάνισης. Ο "αρχηγός" φαίνεται μονάχα να περνά από την πίστα, μόνο και μόνο για να κάνει αισθητή την παρουσία του, χωρίς να αποτελεί κανέναν κίνδυνο. Ο δεύτερος ρόλος του χαρακτηρίζεται από μια διαφορετική συμπεριφορά. Ο

“αρχηγός” θα εμφανιστεί να πετάει σε πολύ χαμηλό υψόμετρο ακριβώς πίσω από τον παίκτη, κυνηγώντας τον και πυροβολώντας από δύο κανόνια ταυτόχρονα. Ο παίκτης δεν μπορεί να τον αντιμετωπίσει άμεσα, παρά μόνο να τον αποφύγει. Ο “αρχηγός” κινείται ελαφρώς πιο γρήγορα από τον παίκτη, έτσι ώστε να του δημιουργείται η αίσθηση ότι τον πλησιάζει. Αν ο παίκτης είναι απρόσεκτος και ο “αρχηγός” τον φτάσει, τότε το όχημα του παίκτη καταστρέφεται, με αποτέλεσμα να χάσει σε αυτήν την πίστα.

Τέλος, όλοι οι εχθροί, εκτός του “αρχηγού”, διαθέτουν δικά τους έτοιμα animations, όπως και ένα φωτεινό κόκκινο βελάκι που αιωρείται και αναπηδά από πάνω τους για να είναι πιο εύκολα ορατοί στον παίκτη.

Παρακάτω ακολουθεί μια λίστα με όλους τους εχθρούς που υπάρχουν στο παιχνίδι, μαζί με γενικές πληροφορίες και τις εικόνες τους.

1. **Εχθρικό Τανκ**
   Κατηγορία: Έδαφος
   Ζωή: 15
   Ζημιά πυρών: 10
   Ζημιά από σύγκρουση: 15 (μόνο στην δεύτερη πίστα)
   Ταχύτητα: 6
   Αριθμός κανονιών: 1
   Συχνότητα Επίθεσης:: 1 βολή / 1.1 δευτερόλεπτα
   Χρόνος Ζωής: 6” στην δεύτερη πίστα, απεριόριστος χρόνος στην πρώτη πίστα

2. **Εχθρικός Τροχός**
   Κατηγορία: Έδαφος
   Ζωή: 30
   Ζημιά πυρών: 10
   Ζημιά από σύγκρουση: 15 (μόνο στην δεύτερη πίστα)
   Ταχύτητα: 6
   Αριθμός κανονιών: 1
   Συχνότητα Επίθεσης:: 1 βολή / 1.1 δευτερόλεπτα
   Χρόνος Ζωής: 6” στην δεύτερη πίστα, απεριόριστος χρόνος στην πρώτη πίστα

3. **Εχθρικό Βαρύ Τανκ**
   Κατηγορία: Έδαφος
   Ζωή: 30
   Ζημιά πυρών: 10
   Ταχύτητα: 6
   Αριθμός κανονιών: 2
   Συχνότητα Επίθεσης:: 2 βολές / 1.1 δευτερόλεπτα

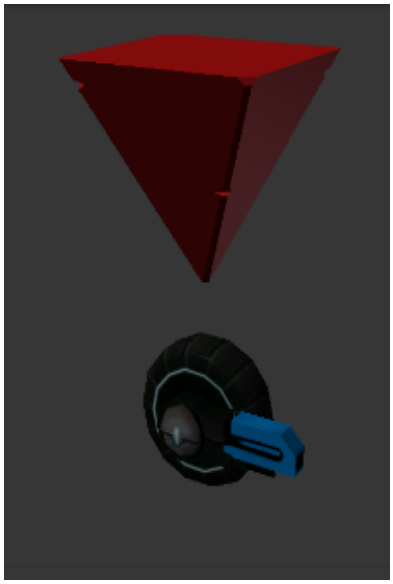
4. **Εχθρικό Τζετ**
   Κατηγορία: Ιπτάμενο
   Ζωή: 15
   Ζημιά πυρών: 10
   Ζημιά από σύγκρουση: 15
   Ταχύτητα: 6
   Αριθμός κανονιών: 1
   Συχνότητα Επίθεσης:: 1 βολή / 1.1 δευτερόλεπτα
   Χρόνος Ζωής: 10”

5. **Εχθρικό Ελικόπτερο**

Κατηγορία: Ιπτάμενο
Ζωή: 15
Ζημιά πυρών: 10
Ζημιά από σύγκρουση: 15
Ταχύτητα: 6
Αριθμός κανονιών: 1
Συχνότητα Επίθεσης:: 1 βολή / 1.1 δευτερόλεπτα
Χρόνος Ζωής: 10”

6. **Αρχηγός**
Κατηγορία: Αρχηγός
Ζωή: 100
Ζημιά πυρών: 10
Ζημιά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
Ταχύτητα: 12.8
Αριθμός κανονιών: 2
Συχνότητα Επίθεσης:: 2 βολές / 1.1 δευτερόλεπτα
Χρόνος ζωής: 12” έως 17”

Σχήμα 20: Εχθρικό Τανκ (έδαφος)

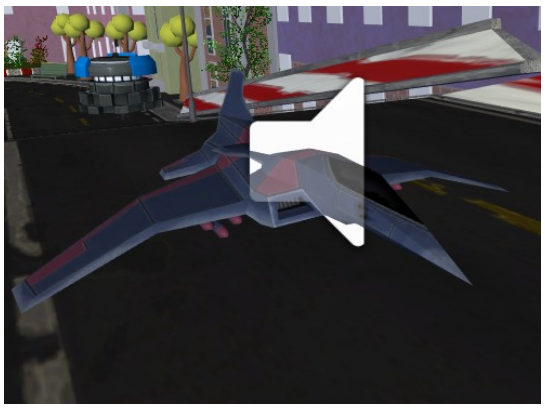


Σχήμα 21: Εχθρικός Τροχός (έδαφος)

Σχήμα 22: Εχθρικό Βαρύ Τανκ (έδαφος)

Σχήμα 23: Εχθρικό Τζετ (ιπτάμενο)

Σχήμα 24: Εχθρικό Ελικόπτερο (ιπτάμενο)



Σχήμα 25: Αρχηγός

3.3 Εμπόδια και παγίδες

Τα εχθρικά οχήματα δεν είναι η μοναδική απειλή που έχει να αντιμετωπίσει ο παίκτης, καθώς μέσα στις πίστες υπάρχουν και αρκετά εμπόδια και παγίδες. Τα εμπόδια χωρίζονται σε δύο κατηγορίες: εδάφους, και ιπτάμενα, ενώ οι παγίδες σε ορατές και αόρατες.

Όλα τα εμπόδια συνοδεύονται από το δικό τους έτοιμο animation, καθώς και από μια πηγή φωτός για να ξεχωρίζουν και να δίνουν μεγαλύτερη βαρύτητα στην παρουσία τους. Ο παίκτης δεν μπορεί να τα καταστρέψει, παρά μόνο να τα αποφύγει. Μια σύγκρουση μαζί τους οδηγεί σε μείωση ζωής του παίκτη.

Τα εμπόδια εδάφους είναι διασκορπισμένα στον ασφάλτινο δρόμο, ενώ τα ιπτάμενα βρίσκονται στον αέρα. Στην πρώτη πίστα όλα τα εμπόδια εδάφους είναι ακίνητα, με εξαίρεση το animation τους, το οποίο δεν αλλάζει την θέση τους, ενώ τα ιπτάμενα εμπόδια έχουν μια ελαφριά κατακόρυφη κίνηση που δίνει στον παίκτη την εντύπωση πως αιωρούνται.

Αντίθετα, στην δεύτερη πίστα υπάρχουν ακίνητα και κινούμενα εμπόδια εδάφους, ενώ υπάρχουν και ιπτάμενα εμπόδια τα οποία εκτελούν μεγαλύτερη και γρηγορότερη κίνηση από τα υπόλοιπα. Οι προσθήκες αυτές στην δεύτερη πίστα συμβάλλουν στην αύξηση του βαθμού δυσκολίας της.

Παράλληλα με τις παραπάνω δύο κατηγορίες, θανάσιμες παγίδες αποτελούν το έδαφος των πιστών (με εξαίρεση τους δρόμους), ένας αόρατος τοίχος αρκετά ψηλά, για να αποτρέπει τον παίκτη από το να συνεχίσει μια μεγάλη ανοδική πορεία που θα τον βγάλει εκτός πίστας, βράχοι και φαράγγια στην πρώτη πίστα, και κτήρια καθώς και κατακόρυφοι αόρατοι τοίχοι στην δεύτερη πίστα. Μια σύγκρουση με κάποια από αυτές τις παγίδες θα καταστρέψει πλήρως τον παίκτη, με αποτέλεσμα να χάσει. Για αυτόν τον λόγο, εμφανίζεται μια προειδοποίηση στην οθόνη όταν ο παίκτης ανεβαίνει πολύ ψηλά και υπάρχει κίνδυνος να χτυπήσει σε κάποιον οριζόντιο αόρατο τοίχο.

Τέλος, αν ο παίκτης συγκρουστεί με εμπόδιο εδάφους, τότε αυτό θα παραμείνει ανέπαφο, ενώ ένα ιπτάμενο εμπόδιο καταστρέφεται μετά την σύγκρουση.

Παρακάτω ακολουθεί μία λίστα με τις παγίδες και τα εμπόδια που βρίσκονται στις δύο πίστες του παιχνιδιού:

1. **Εμπόδιο Εδάφους**
   Κατηγορία: Εμπόδιο

Τύπος: Εδάφους
Ζημιά μετά από σύγκρουση: 15
Καταστροφή μετά από σύγκρουση: Όχι

2. **Ιπτάμενο Εμπόδιο**
   Κατηγορία: Εμπόδιο
   Τύπος: Ιπτάμενο
   Ζημιά μετά από σύγκρουση: 20
   Καταστροφή μετά από σύγκρουση: Ναι

3. **Έδαφος**
   Κατηγορία: Παγίδα
   Τύπος: Ορατή
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

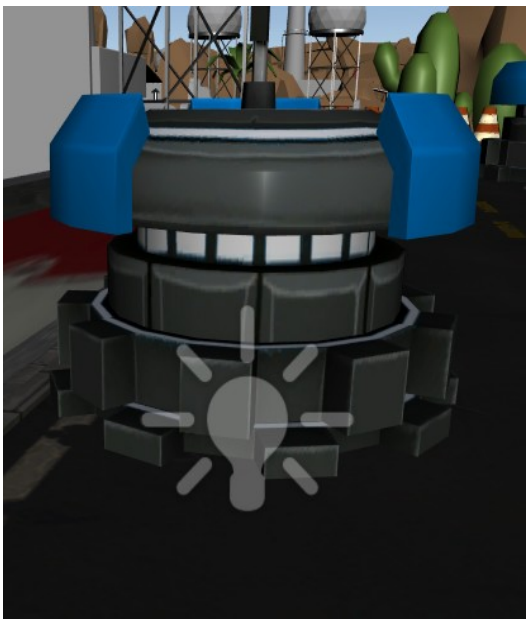
4. **Οριζόντιος αόρατος τοίχος**
   Κατηγορία: Παγίδα
   Τύπος: Αόρατη
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

5. **Κατακόρυφος αόρατος τοίχος**
   Κατηγορία: Παγίδα
   Τύπος: Αόρατη
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

6. **Βράχος**
   Κατηγορία: Παγίδα
   Τύπος: Ορατή
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

7. **Φαράγγι**
   Κατηγορία: Παγίδα
   Τύπος: Ορατή
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

8. **Κτήριο**
   Κατηγορία: Παγίδα
   Τύπος: Ορατή
   Ζημιά μετά από σύγκρουση: Καταστρέφει πλήρως τον παίκτη
   Καταστροφή μετά από σύγκρουση: Όχι

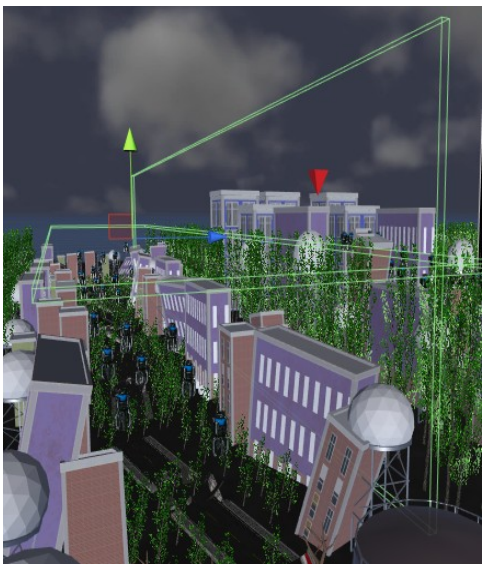


Σχήμα 26: Εμπόδιο εδάφους

Σχήμα 27: Ιπτάμενο Εμπόδιο

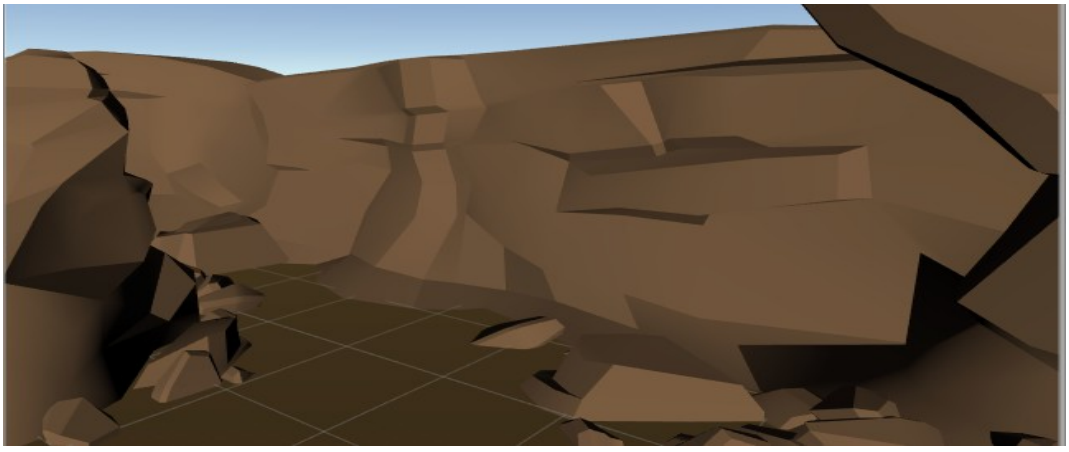
Σχήμα 28: Βράχος

Σχήμα 29: Κτήρια



Σχήμα 30: Οριζόντιος και κατακόρυφος αόρατος τοίχος

Σχήμα 31: Έδαφος



Σχήμα 32: Φαράγγι

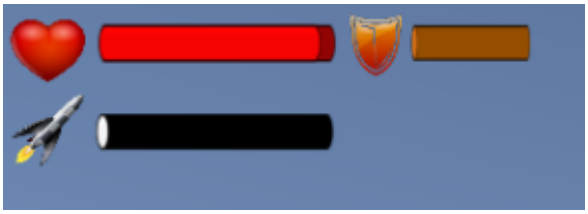
3.4 Αντικείμενα και στατιστικά παίκτη

Στον παίκτη παρέχονται τα στατιστικά των οχημάτων του με την μορφή τριών μπαρών - sliders. Πιο συγκεκριμένα, διαθέτει μια κόκκινη μπάρα ζωής, μια πορτοκαλί μπάρα ασπίδας και μια λευκή μπάρα για τους πυραύλους οι οποίες έχουν τοποθετηθεί πάνω αριστερά στην οθόνη.

Το παιχνίδι ξεκινά πάντα με την μπάρα ζωής γεμάτη και τις άλλες δύο άδειες. Η μπάρα ασπίδας λειτουργεί σαν προστατευτικό μέσο για την μπάρα ζωής. Όταν ο παίκτης δεχτεί χτύπημα από ένα εμπόδιο ή από εχθρικά πυρά η τιμή της μπάρας ζωής του θα μειωθεί, αν όμως η μπάρα ασπίδας διαθέτει μια τιμή μεγαλύτερη του 0, τότε θα μειωθεί εκείνη. Η μπάρα ζωής θα μένει εκτός κινδύνου εφόσον η μπάρα ασπίδας δεν είναι άδεια. Όταν μπάρα ζωής φτάσει στο 0, ο παίκτης εξουδετερώνεται, με αποτέλεσμα να χάσει.

Η μπάρα των πυραύλων είναι απαραίτητο να έχει τιμή μεγαλύτερη ή ίση του 10 για να μπορέσει ο παίκτης να εκτοξεύσει τους πυραύλους του. Αυτό συμβαίνει επειδή οι πύραυλοι "καταναλώνουν" την μπάρα αυτήν. Κάθε φορά που χρησιμοποιούνται μειώνουν την τιμή της κατά 10.

Όλες οι μπάρες έχουν ως ελάχιστη τιμή το 0 και είναι αδύνατον να περιέχουν μια αρνητική τιμή. Η μπάρα ζωής και η μπάρα πυραύλων έχουν ως μέγιστη τιμή το 100, ενώ η μπάρα ασπίδας έχει ως μέγιστη τιμή το 50 και είναι, επίσης, αδύνατον να έχει μια τιμή μεγαλύτερη της μέγιστης.



Σχήμα 33: Οι μπάρες με τα στατιστικά του παίκτη

Ο παίκτης έχει την επιλογή να αναπληρώσει τις μπάρες των στατιστικών του συλλέγοντας συγκεκριμένα αντικείμενα τα οποία έχουν τοποθετηθεί σε στρατηγικά σημεία μέσα στις πίστες για να διευκολύνουν το παιχνίδι. Τα διαθέσιμα αντικείμενα είναι μια πορτοκαλί ασπίδα που αναπληρώνει την μπάρα της ασπίδας, ένα ζευγάρι με σφυριά που αναπληρώνει την μπάρα της ζωής, καθώς και ένας μαύρος πύραυλος που αναπληρώνει την μπάρα των πυραύλων.

Τα αντικείμενα αυτά διαθέτουν ένα μικρό ειδικό εφέ που αναπαριστά φυσαλίδες, φωτισμό, καθώς και ένα script που τους δίνει την δυνατότητα να αναπηδούν και να περιστρέφονται καθώς αιωρούνται. Μόλις ο παίκτης συλλέξει κάποιο από αυτά τα αντικείμενα θα ακουστεί ένα σύντομο ηχητικό εφέ και το αντικείμενο θα εξαφανιστεί.

Σχήμα 34: Το αντικείμενο ασπίδας

Σχήμα 35: Το αντικείμενο ζωής

Σχήμα 36: Το αντικείμενο πυραύλων

3.5 Οι πίστες και ο σκοπός του παιχνιδιού

Όπως αναφέρθηκε προηγουμένως, υπάρχουν δύο πίστες μέσα στο παιχνίδι, με την πρώτη να απαρτίζεται από μάχες οχημάτων ανάμεσα στον παίκτη και στους εχθρούς, ενώ η δεύτερη αποτελεί μια αναμέτρηση ανάμεσα στον παίκτη και στον "αρχηγό". Κατά την διάρκεια των πιστών το όχημα του παίκτη υποβάλλεται σε αρκετές μεταμορφώσεις μέχρι να φτάσει στο τέλος τους.

Η πρώτη πίστα έχει δημιουργηθεί για να θυμίζει περιβάλλον ερήμου. Το βασικό έδαφος αποτελείται από άμμο, ενώ ο χώρος περιτριγυρίζεται από φαράγγια. Σαν διακόσμηση έχουν τοποθετηθεί διάφορα κτίσματα τα οποία μπορεί κανείς να συναντήσει σε εργοστασιακές και στρατιωτικές εγκαταστάσεις που βρίσκονται σε ερήμους, κατεστραμμένα κτήρια, διαλυμένα τανκς, κάκτοι, φοίνικες, όπως και τρεις πυραμίδες που συνοδεύονται από οβελίσκους και ερείπια.

Το αυτοκίνητο του παίκτη κινείται πάνω σε ασφάλτινο δρόμο, ο ποιος περιβάλλεται από κατακόρυφους αόρατους τοίχους για να μην βγει εκτός πίστας, καθώς και σε γέφυρες που περιβάλλονται από μπάρες και κώνους δρόμων για τον ίδιο λόγο. Στους δρόμους και στις γέφυρες έχουν τοποθετηθεί εμπόδια τα οποία πρέπει να αποφευχθούν.

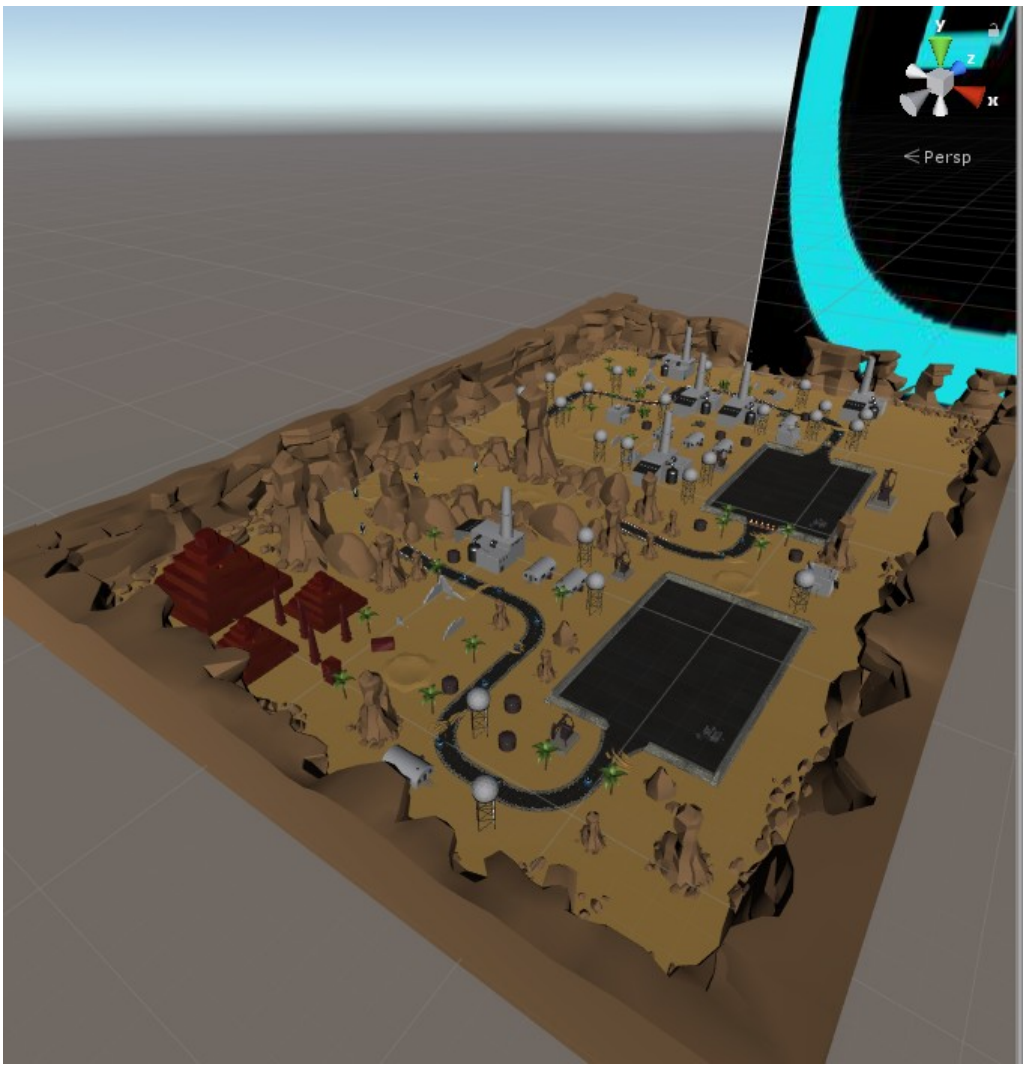
Ταυτόχρονα, υπάρχουν δύο αρένες στις οποίες λαμβάνουν χώρα οι μάχες εδάφους σε αυτήν την πίστα. Οι αρένες αποτελούνται από ένα μεγάλο ορθογώνιο κομμάτι ασφάλτου και περιτριγυρίζονται από πέτρινους τοίχους, ενώ διαθέτουν και ένα διαλυμένο τανκ σαν διακόσμηση. Μόλις ο παίκτης μπει μέσα σε κάποια από αυτές τότε θα κλείσει ένας τοίχος πίσω του, αποκόπτοντάς τον από τον δρόμο επιστροφής, ενώ την ίδια στιγμή τον θέτει αντιμέτωπο με εχθρικά οχήματα εδάφους που επιχειρούν να τον εξοντώσουν. Αφού καταστραφούν όλοι οι εχθροί της πρώτης αρένας, ένας φράχτης από κώνους δρόμων θα καταστραφεί, ανοίγοντας στον παίκτη τον δρόμο για να προχωρήσει στην πίστα.

Ακολουθεί η πρώτη μεταμόρφωση αυτής της πίστας, με το αυτοκίνητο να μεταμορφώνεται σε ελικόπτερο. Ο παίκτης πρέπει να καθοδηγήσει το ελικόπτερό του μέσα από επικίνδυνα φαράγγια και βράχους αντιμετωπίζοντας ιπτάμενους εχθρούς και αποφεύγοντας ιπτάμενα εμπόδια. Στη συνέχεια το ελικόπτερο μεταμορφώνεται σε αυτοκίνητο και ο παίκτης συνεχίζει πάνω στον δρόμο.

Τέλος, ο παίκτης φτάνει στην δεύτερη αρένα της πίστας, στην οποία παγιδεύεται καθώς πρέπει να πολεμήσει ενάντια στα τελευταία εχθρικά οχήματα της πίστας. Όταν επιτύχει, αντί να ανοίξει κάποιος δρόμος για να συνεχιστεί η πίστα, το αυτοκίνητο σταματά να κινείται και στην οθόνη εμφανίζεται το μήνυμα επιτυχίας συνοδευόμενο από ένα σύντομο μουσικό κομμάτι και ένα κουμπί που μεταφέρει τον παίκτη πίσω στο αρχικό μενού.

Σχήμα 37: Η πρώτη αρένα της πρώτης πίστας

Σχήμα 38: Το σημείο οδήγησης ελικοπτέρου



Σχήμα 39: Η πρώτη πίστα

Η δεύτερη πίστα έχει δημιουργηθεί για να θυμίζει περιβάλλον κατεστραμμένης πόλης, και σε αντίθεση με την πρώτη, δεν διαθέτει καθόλου αρένες για μάχη. Σε αυτήν την πίστα ο παίκτης έρχεται αντιμέτωπος με τον "αρχηγό", ο οποίος διαθέτει και αυτός την δική του μπάρα ζωής.

Η πίστα ολοκληρώνεται όταν η μπάρα ζωής του "αρχηγού" πέσει στο 0. Όμως ο παίκτης δεν μπορεί να τον πολεμήσει άμεσα. Για αυτό πρέπει να χρησιμοποιήσει προς όφελός του τέσσερα "φιλικά" κανόνια που έχουν τοποθετηθεί σε στρατηγικά σημεία μέσα στην πίστα. Αυτά αποτελούν το μοναδικό μέσο για να μειώσει ο παίκτης την μπάρα ζωής του "αρχηγού".

Ο "αρχηγός" κάνει την εμφνάνισή του πρώτη φορά στην αρχή της πίστας, καθώς φαίνεται να πετά πάνω από τα κτήρια της πόλης. Μόλις ο παίκτης φτάσει σε μία μεγάλη ευθεία, η κάμερα του παιχνιδιού αλλάζει θέση, με αποτέλεσμα να φαίνεται το αυτοκίνητο του παίκτη από την μπροστινή μεριά. Αυτό συμβαίνει γιατί μετά από δύο δευτερόλεπτα, ο "αρχηγός" εμφανίζεται πίσω από τον παίκτη και τον κυνηγά εκτοξεύοντας σφαίρες ταυτόχρονα. Ο παίκτης πρέπει να αποφύγει τις σφαίρες του, καθώς και εμπόδια εδάφους που υπάρχουν εκεί.

Μόλις ο κίνδυνος περάσει, η κάμερα επιστρέφει στην προκαθορισμένη της θέση, και ο παίκτης θα περάσει από το πρώτο κανόνι το οποίο θα εκτοξεύσει τρεις πυραύλους που θα μειώσουν την ζωή του "αρχηγού" κατά 1/4. Ο παίκτης συνεχίζει αντιμετωπίζοντας ένα κύμα εχθρών εδάφους που έρχονται κατά πάνω του. Στην συνέχεια, το αυτοκίνητο θα μεταμορφωθεί σε ελικόπτερο. Τότε θα ξαναφανεί ο αρχηγός που περνά πάνω από τα κτήρια.

Μόλις ο παίκτης ξεφύγει από τα ιπτάμενα εμπόδια και τους ιπτάμενους εχθρούς, το ελικόπτερο θα μεταμορφωθεί σε αυτοκίνητο και θα αρχίσει ακόμα ένα κυνηγητό με τον "αρχηγό" σε μια

μεγάλη ευθεία. Στο τέλος της ευθείας ο παίκτης περνάει από ένα ακόμα κανόνι, το οποίο θα μειώσει και αυτό την ζωή του “αρχηγού” κατά 1/4.

Ακολουθούν ακόμα δύο μεταμορφώσεις από αυτοκίνητο σε ελικόπτερο και το αντίστροφο, δύο απλές εμφανίσεις του “αρχηγού”, και το τρίτο κανόνι, καθώς και κάποια κινούμενα εμπόδια εδάφους. Στο τέλος της πίστας ο παίκτης οδηγεί το ελικόπτερο προς έναν ουρανοξύστη, ο οποίος διαθέτει ένα φωτεινό κόκκινο βελάκι που αναπηδά από πάνω του. Στην κορυφή του ουρανοξύστη υπάρχει ένας χώρος προσγείωσης ελικόπτερου, καθώς και το τελευταίο κανόνι. Μόλις ο παίκτης φτάσει πάνω από τον ουρανοξύστη, η ταχύτητά του σταματά, και το ελικόπτερο πρέπει να προσγειωθεί. Όταν συμβεί αυτό, η κάμερα του παιχνιδιού δείχνει το ελικόπτερο από μπροστά, στο βάθος φαίνεται ο αρχηγός που πλησιάζει τον παίκτη απειλητικά, και λίγο πριν τον φτάσει το τέταρτο κανόνι τον πυροβολεί, μειώνοντας την ζωή του στο 0 και καταστρέφοντάς τον ταυτόχρονα. Η πίστα τελειώνει με το ίδιο μήνυμα επιτυχίας.

Σχήμα 40: Ένα "φιλικό" κανόνι

Σχήμα 41: Η μπάρα ζωής του "αρχηγού"

Σχήμα 42: Ο ουρανοξύστης με τον χώρο προσγείωσης και το τελευταίο κανόνι

Σχήμα 43: Η δεύτερη πίστα

4. Ανάπτυξη Παιχνιδιού

Σε αυτό το κεφάλαιο παρουσιάζεται η διαδικασία κατασκευής και ανάπτυξης του παιχνιδιού. Περιγράφεται η βασική ιδέα πίσω από αυτό, η δημιουργία των πιστών, η διαχείριση των στοιχείων του παιχνιδιού όπως των μοντέλων και των ηχητικών κομματιών, η κατασκευή της παύσης, των οθονών φόρτωσης, των διάφορων μενού, καθώς και των ρυθμίσεων που μπορεί να αλλάξει ο παίκτης.

4.1 Η βασική ιδέα

Βασική πηγή έμπνευσης για την δημιουργία του παιχνιδιού αποτέλεσαν συγκεκριμένοι ήρωες των παιδικών μου χρόνων, όπως ο Action Man[8], ο Max Steel[9] και οι G.I. JOE[10], οι οποίοι διέθεταν πολλά οχήματα εξοπλισμένα με διάφορα οπλικά συστήματα, πολλά εκ των οποίων μεταμορφώνονταν σε άλλα οχήματα.

Εξίσου σημαντικό ρόλο έπαιξε και η επιρροή που μου άσκησε σειρά βιντεοπαιχνιδιών δράσης SpyHunter, πιο συγκεκριμένα, τα παιχνίδια με τίτλο “SpyHunter”[11], “SpyHunter 2”[12] και “SpyHunter: Nowhere to Run”[13] που αναπτύχθηκαν και κυκλοφόρησαν από την Midway Games για πολλαπλές πλατφόρμες τις χρονιές 2001-2003, 2003-2004 και 2006 αντίστοιχα. Στα παιχνίδια αυτά οι παίκτες μπορούσαν να πάρουν τον έλεγχο ενός οχήματος που φέρει διάφορους τύπους όπλων, και έχει την δυνατότητα να μεταμορφωθεί σε πολλά άλλα οχήματα. Οι παίκτες χρησιμοποιούσαν το όχημα αυτό ως μέσο για να ολοκληρώσουν τις διάφορες πίστες των παιχνιδιών και να κατατροπώσουν τους εχθρούς με τους οποίους ερχόντουσαν αντιμέτωποι.

Σχήμα 44: Στιγμιότυπο από gameplay του SpyHunter 2

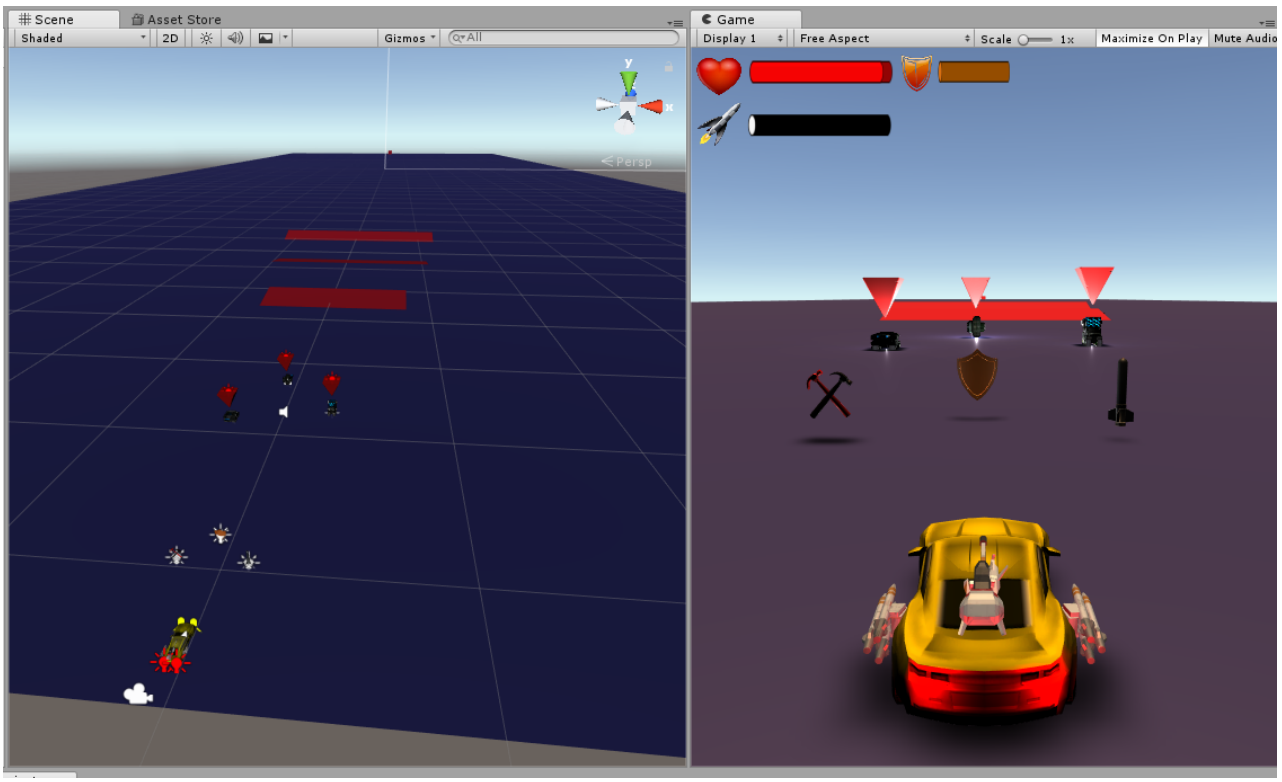
Σχήμα 45: Στιγμιότυπο από gameplay του SpyHunter: Nowhere to Run

4.2 Εισαγωγή και επεξεργασία μοντέλων

Όπως αναφέρθηκε προηγουμένως, τα περισσότερα μοντέλα λήφθηκαν έτοιμα από το Asset Store της Unity. Όμως σχεδόν όλα υπέστησαν ένα μικρό ποσοστό επεξεργασίας μέσα από την εφαρμογή της Unity πριν χρησιμοποιηθούν στο τελικό παιχνίδι. Σε αυτήν την ενότητα θα γίνει μια σύντομη αναφορά στην επεξεργασία που έγινε στα οχήματα που ελέγχει ο παίκτης, στους εχθρούς, καθώς και σε διάφορα άλλα μοντέλα που υπάρχουν στο παιχνίδι..

Αρχικά, για να πραγματοποιηθεί η επεξεργασία αυτή, έπρεπε να δημιουργηθεί μια μικρή πίστα η οποία εκτελούσε μονάχα σκοπούς δοκιμών και δεν ξαναχρησιμοποιήθηκε μετά την δημιουργία των

κανονικών πιστών. Στην πίστα αυτή επεξεργάστηκαν και δοκιμάστηκαν τα οχήματα του παίκτη, οι ικανότητές τους, οι μεταμορφώσεις τους, τα όπλα τους, οι εχθροί εδάφους, τα αντικείμενα που συλλέγει ο παίκτης, τα στατιστικά του παίκτη, καθώς και τα Scripts που συνδέονται με όλα τα παραπάνω.



Σχήμα 46: Πίστα δοκιμών

Στην συνέχεια ξεκίνησε η κατασκευή του αυτοκινήτου. Ένα έτοιμο μοντέλο φορτώθηκε στην Unity, και του προστέθηκαν τα όπλα που φαίνονται και παραπάνω. Έπειτα γράφτηκε και δοκιμάστηκε το script που συνδέεται με τον έλεγχο του, του προστέθηκαν οι προβολείς, οι ήχοι της μηχανής και των λάστιχων, καθώς και σαν επιπλέον λειτουργία οι τροχοί του αυτοκινήτου κυλούσαν ταυτόχρονα με την κίνηση του, ενώ οι μπροστινοί τροχοί έστριβαν σύμφωνα με τις στροφές του αυτοκινήτου.

Ακολούθησε η δημιουργία του τζετ του παίκτη, διότι αρχικός σκοπός του παιχνιδιού ήταν το αυτοκίνητο να μεταμορφώνεται σε τζετ, όμως αυτό άλλαξε λόγω προσωπικής προτίμησης, και το τζετ αντικαταστήθηκε από το πολεμικό ελικόπτερο. Το τζετ χρησιμοποιήθηκε αργότερα για την κατασκευή του "αρχηγού" στην δεύτερη πίστα. Παρ' όλα αυτά όμως, ο κώδικας που γράφτηκε για το τζετ ήταν εφικτό, με ελάχιστες αλλαγές, να χρησιμοποιηθεί για το ελικόπτερο. Από το ελικόπτερο έπρεπε να αφαιρεθούν οι ρόδες που διέθετε στην κάτω μεριά, να προστεθεί ένα ηχητικό κομμάτι που αναπαριστούσε ήχο από έλικα ελικοπτέρου και να δοθεί στον έλικα η ικανότητα να περιστρέφεται. Αφού το ελικόπτερο ήταν έτοιμο, δημιουργήθηκε η διαδικασία μεταμόρφωσης των οχημάτων. Πιο συγκεκριμένα, τα οχήματα περνούν μέσα από ένα αόρατο Trigger Collider, το οποίο πυροδοτεί την μεταμόρφωση.

Σειρά είχε η δημιουργία των στατιστικών του παίκτη. Οι μπάρες αντιπροσωπεύονται από το 2D αντικείμενο της Unity, Slider. Τα Sliders τοποθετήθηκαν πάνω σε έναν 2D αντικείμενο Canvas, άλλαξε το χρώμα τους και τους αφαιρέθηκε ο πήχης, ενώ οι εικόνες που τις συνοδεύουν βρέθηκαν στο διαδίκτυο, μετατράπηκαν στην κατηγορία εικόνων "sprite" μέσω της εφαρμογής της Unity, και τοποθετήθηκαν πάνω στον καμβά μέσω χρήσης του 2D αντικειμένου της Unity, Image.

Έπειτα εισάχθηκε στην πίστα δοκιμών το αντικείμενο που αργότερα θα αντιπροσώπευε το εχθρικό τανκ. Στο τανκ αυτό προστέθηκε το navigator tool της Unity, το οποίο του επιτρέπει να

ακολουθεί έναν στόχο (στην συγκεκριμένη περίπτωση, τον παίκτη), δίνοντάς του, ταυτόχρονα, την δυνατότητα να αποφεύγει εμπόδια που του ορίζει ο χρήστης. Αργότερα του δόθηκε η ικανότητα να πυροβολεί τον παίκτη μόνο εφόσον είναι στραμμένο προς αυτόν, μειώνοντας την μπάρα ζωής και ασπίδας του.

Τα αντικείμενα των σφαιρών του παίκτη και των εχθρών χρησιμοποιήθηκαν από το δωρεάν εκπαιδευτικό project της Unity με τίτλο "Tanks tutorial", ενώ για τους σκοπούς του παιχνιδιού υπέστησαν μια μικρή αλλαγή στο σχήμα τους, και τους προστέθηκε κόκκινος και μπλε φωτισμός. Ο κόκκινος φωτισμός χαρακτηρίζει τις βολές των εχθρών, ενώ ο μπλε τις βολές του παίκτη.

Το αντικείμενο του εχθρικού τανκ εμπλουτίστηκε με ένα στοιχείο φωτισμού, ήχο μηχανής, καθώς και από ένα κόκκινο βελάκι που υποδείκνυε στον παίκτη την θέση του. Αφού το εχθρικό τανκ ήταν, πλέον, έτοιμο, στην πίστα δοκιμών μπήκαν και ο εχθρικός τροχός μαζί με το εχθρικό βαρύ τανκ, τα οποία υπέστησαν ακριβώς την ίδια επεξεργασία και ίδιες προσθήκες με το εχθρικό τανκ.

Στην συνέχεια, τα αντικείμενα των εχθρών έπρεπε να απενεργοποιηθούν για να ξεκινήσει η δημιουργία των αντικειμένων που θα σύλλεγε ο παίκτης. Αρχικά, τα αντικείμενα αυτά αντιπροσωπεύονταν από κάποια έτοιμα 3D αντικείμενα της Unity. Πιο συγκεκριμένα, ένας κόκκινος κύβος αντιπροσώπευε το αντικείμενο που αναπλήρωνε την ζωή του παίκτη, μια πορτοκαλί σφαίρα το αντικείμενο της ασπίδας, και μια μαύρη κάψουλα το αντικείμενο των πυραύλων. Πάνω στα προαναφερθέντα αντικείμενα γράφτηκε και δοκιμάστηκε το script των τελικών εκδοχών τους. Δεν άργησαν να αντικατασταθούν με 3D μοντέλα που βρέθηκαν από το Asset Store της Unity και από την ιστοσελίδα turbosquid.com. Μετά την αντικατάστασή αυτή, τους προστέθηκε ο φωτισμός, το ειδικό εφέ μέσω του Particle System και το ηχητικό εφέ που παίζει όταν ο παίκτης τα συλλέξει.

Τέλος, ως επιπλέον λεπτομέρειες, προστέθηκαν ειδικά και ηχητικά εφέ από Particles Systems και ηχητικά αντικείμενα τα οποία συνοδεύουν τα χτυπήματα από τις σφαίρες στους εχθρούς και στα οχήματα του παίκτη καθώς και τις εκρήξεις από τις καταστροφές των οχημάτων αυτών.

Η δημιουργία όλων των κατηγοριών εμποδίων και παγίδων, του αρχηγού, των ιπτάμενων εχθρών και τον δρόμων έλαβε χώρα κατά την δημιουργία των πιστών.

4.3 Δημιουργία πιστών

4.3.1 Πρώτη πίστα

Ακολούθησε η κατασκευή των πιστών. Και στις δύο πίστες πρωταγωνιστούν 3D μοντέλα που απεικονίζουν κομμάτια δρόμων και βρέθηκαν δωρεάν από το Asset Store της Unity. Αρχικά τα μοντέλα δοκιμάστηκαν στην πίστα δοκιμών, με την κατασκευή ενός πολύ πρόχειρου δρόμου, και έπειτα δημιουργήθηκε μια νέα σκηνή για να στεγάσει την πρώτη πίστα.

Εξίσου σημαντικό ρόλο στην δημιουργία της πίστας αυτής, έπαιξαν και τα assets που διαθέτονταν με το δωρεάν εκπαιδευτικό πακέτο Tanks tutorial, καθώς το έδαφος, τα φαράγγια, οι βράχοι και όλα τα στοιχεία διακόσμησης (εκτός των πυραμίδων) προέρχονται από εκεί.

Πριν τοποθετηθούν οι δρόμοι στην κατάλληλη διάταξη, έπρεπε πρώτα να υποστούν και αυτοί ένα μικρό ποσοστό επεξεργασίας. Τα 3D μοντέλα της ασφάλτου που θα χρησιμοποιούνταν ήταν συνηθισμένοι δρόμοι και γέφυρες, στα προστέθηκαν κατακόρυφοι αόρατοι τοίχοι και φράχτες δρόμων αντίστοιχα (δεν αποτελούν παγίδες). Τα μοντέλα που προέκυψαν αποθηκεύτηκαν στους φακέλους του παιχνιδιού για μελλοντική χρήση.

Έπειτα προστέθηκε η πρώτη αρένα με χρήση ενός μεγάλου ορθογωνίου μοντέλου ασφάλτου και ο δρόμος που οδηγούσε προς την πρώτη μεταμόρφωση. Η αρένα περικυκλώθηκε από πέτρινους φράχτες, και τοποθετήθηκαν οι πρώτοι εχθροί. Έγινε η απαραίτητη τροποποίηση στο navigator tool της Unity, έτσι ώστε οι εχθροί να ακολουθούν τον παίκτη μόνο μέσα στην αρένα. Σαν διακοσμητικό στοιχείο προστέθηκε ένα διαλυμένο τανκ.

Στη συνέχεια το έδαφος της πίστας περιφράχτηκε από φαράγγια, ενώ αργότερα δημιουργήθηκε το κομμάτι της πίστας όπου ο παίκτης θα έπαιρνε τον έλεγχο του ελικοπτέρου. Ο χώρος αυτός

περιφράχτηκε από διάφορα μοντέλα βράχων, ενώ προστέθηκαν και οι αόρατοι τοίχοι μαζί με ένα Trigger Collider που προειδοποιούσε τον παίκτη ότι πλησίαζε επικίνδυνα κοντά τους.

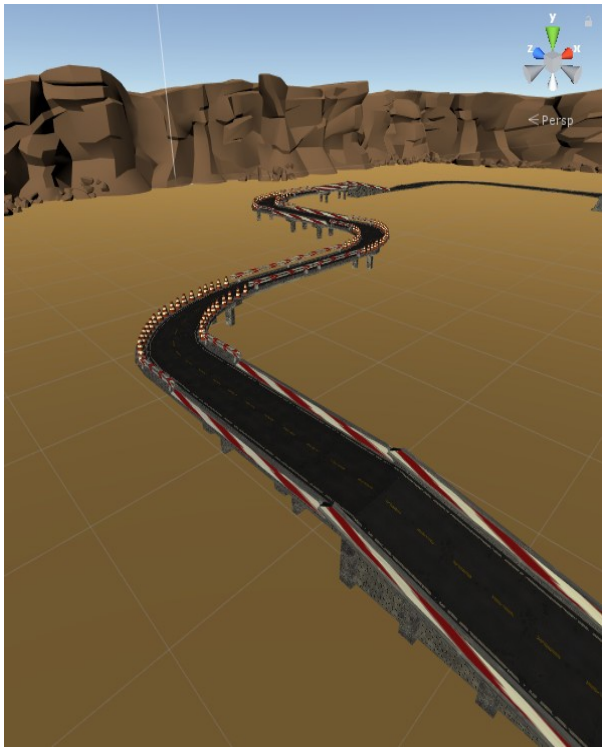
Σειρά είχε η συνέχεια του δρόμου μέχρι την τελική αρένα, η οποία αντιγράφτηκε αυτούσια από την πρώτη και χρησιμοποιήθηκε μονάχα με δύο αλλαγές. Ενώ στην πρώτη, ανάμεσα σε δύο πέτρινους τοίχους υπήρχε ένας τοίχος από κώνους δρόμων που καταστρεφόταν όταν ο παίκτης εξουδετέρωνε όλους τους εχθρούς, στην δεύτερη τοποθετήθηκε απλά ένας μεγάλος τοίχος που θα κάλυπτε εκείνη την μεριά της αρένας. Η δεύτερη διαφορά είναι ο αριθμός των εχθρών, με την δεύτερη αρένα να έχει περισσότερους εχθρούς από την πρώτη.

Ακολούθησε η τοποθέτηση των αντικειμένων που συλλέγει ο παίκτης, η επεξεργασία και η διάταξη των εμποδίων εδάφους, των ιπτάμενων εμποδίων και των ιπτάμενων εχθρών. Και στις δύο κατηγορίες εμποδίων προστέθηκε φωτισμός, ενώ στα ιπτάμενα δόθηκε η δυνατότητα να αναπηδούν ελαφρώς στην θέση τους. Τα μοντέλα εμποδίων και ιπτάμενων εχθρών που προέκυψαν αποθηκεύτηκαν στους φακέλους του παιχνιδιού για μελλοντική χρήση.

Αργότερα προστέθηκαν τα στοιχεία διακόσμησης τα οποία κάλυψαν όλη την υπόλοιπη πίστα δίνοντας έτσι περισσότερο βάρος στην εμφάνισή της. Επίσης προστέθηκαν τα Trigger Colliders που ευθύνονται για την εμφάνιση των εχθρών και για τις μεταμορφώσεις των οχημάτων.

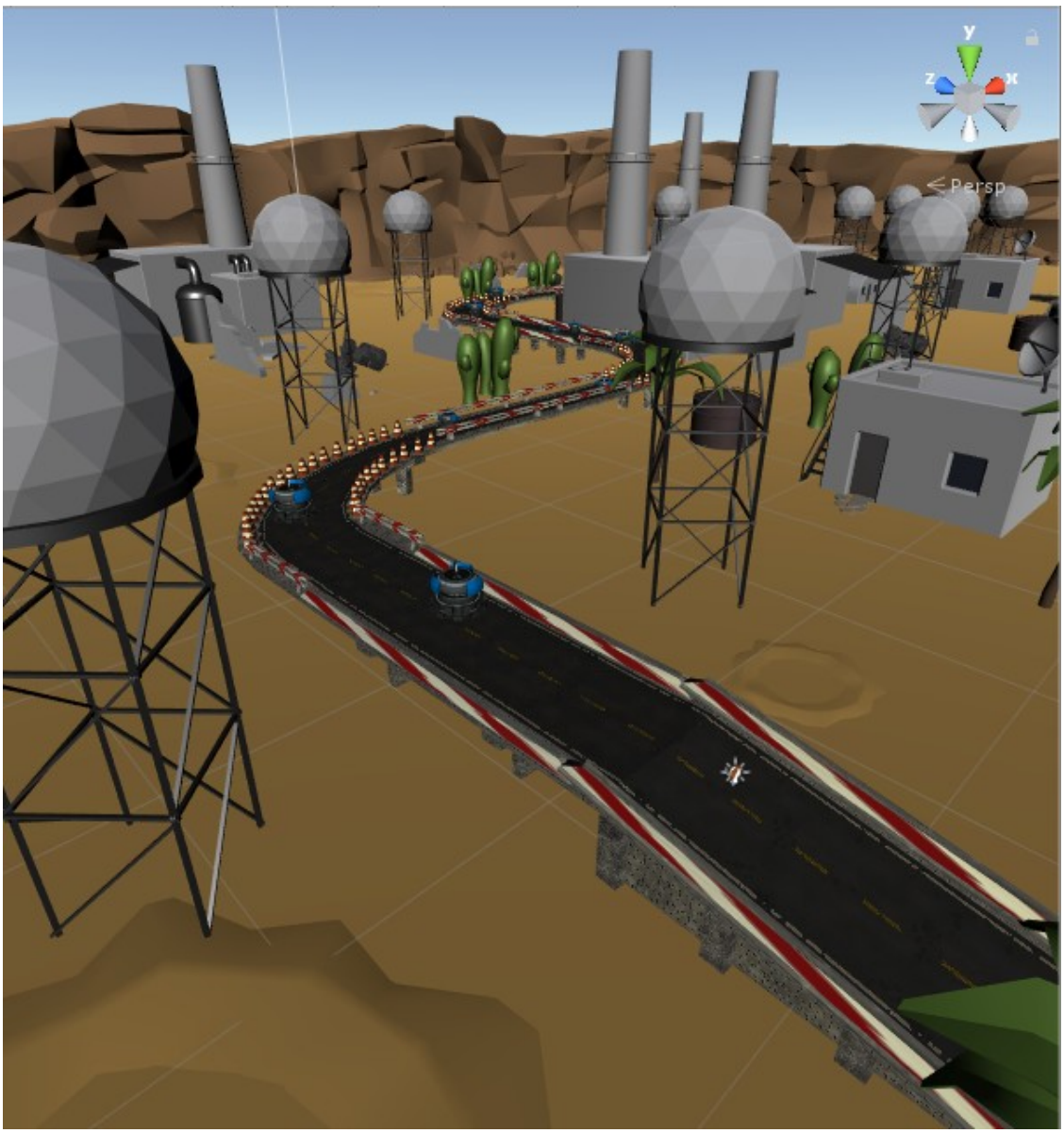
Ακόμα, προστέθηκε ένα Trigger Collider στο αυτοκίνητο και στο ελικόπτερο, το οποίο συνέβαλε στην πυροδότηση διάφορων γεγονότων όταν ερχόταν σε επαφή με άλλα Trigger Colliders, όπως την εμφάνιση εχθρών.

Τέλος, προστέθηκε η μουσική επένδυση της πίστας με χρήση ενός μουσικού κομματιού που λήφθηκε από την ιστοσελίδα youtube.com (βλέπε Disclaimer στην σύνοψη), και τα αντικείμενα Canvas με τις εικόνες αρχής, νίκης και ήττας, που το καθένα από τα δύο τελευταία συνοδευόταν από το αντίστοιχο μουσικό κομμάτι, τα οποία αποθηκεύτηκαν και αυτά στους φακέλους του παιχνιδιού για μελλοντική χρήση.



Σχήμα 47: Πρώτο στάδιο κατασκευής πρώτης πίστας

Σχήμα 48: Δεύτερο στάδιο κατασκευής πρώτης πίστας



Σχήμα 49: Τελικό στάδιο κατασκευής πρώτης πίστας

4.3.2 Δεύτερη πίστα

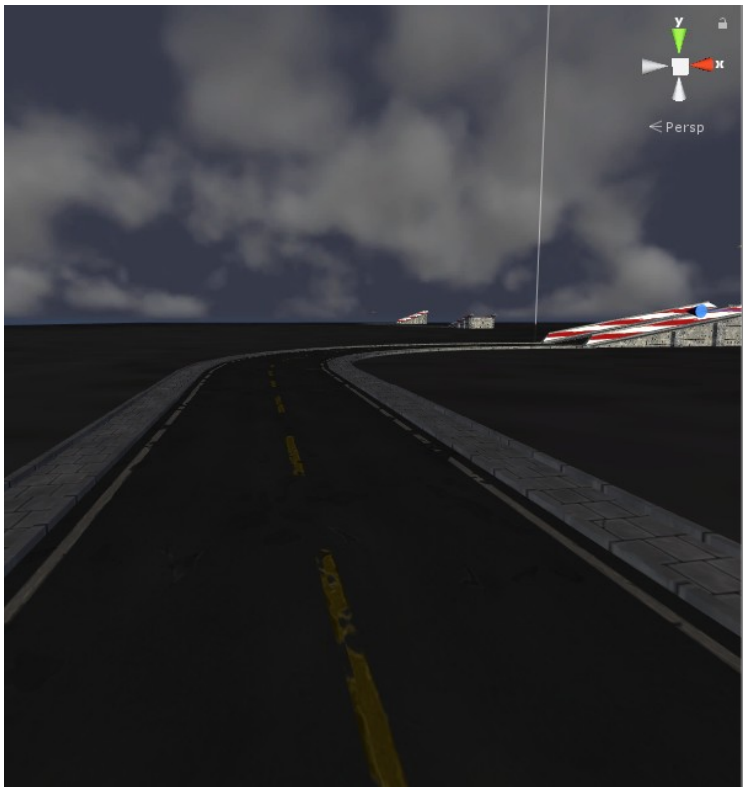
Αρχικά, εκτιμήθηκε πως η κατασκευή της δεύτερης πίστας θα ήταν πολύ πιο εύκολη από την κατασκευή της πρώτης, δεδομένου ότι τα πιο πολλά αντικείμενα που θα χρησιμοποιούνταν είχαν ήδη υποστεί την απαραίτητη επεξεργασία και είχαν αποθηκευτεί στους φακέλους του παιχνιδιού. Όμως η εκτίμηση αυτή αποδείχθηκε εσφαλμένη, αφού η δημιουργία της δεύτερης πίστας αποτέλεσε ένα αρκετά πιο δύσκολο εγχείρημα λόγω του μεγάλου μεγέθους της και της μεγάλης έκτασης που έπρεπε να καλυφθεί με στοιχεία διακόσμησης.

Η διαδικασία που ακολουθήθηκε για την δημιουργία της δεύτερης πίστας είναι παρόμοια με την διαδικασία της πρώτης. Πρώτα προστέθηκαν το έδαφος και οι δρόμοι όπως είχαν δημιουργηθεί στην πρώτη πίστα, με τους αόρατους τοίχους και τους φράχτες να τους συνοδεύουν. Έπειτα ακολούθησαν τα μοντέλα της πόλης και ενός νυχτερινού Sky Box για να δημιουργηθεί η αίσθηση της νύχτας, που λήφθηκαν δωρεάν από το Asset Store της Unity. Ακόμα, προστέθηκαν τα εμπόδια εδάφους, καθώς και τα ιπτάμενα εμπόδια, μαζί με τα αντικείμενα που συλλέγει ο παίκτης. Σειρά είχε η τοποθέτηση των εχθρών. Κατά την δημιουργία αυτής της πίστας, δημιουργήθηκαν και οι εχθροί εδάφους που συμπεριφέρονται όπως οι ιπτάμενοι εχθροί. Στην συνέχεια ακολούθησε το πιο

δύσκολο κομμάτι της πίστας αυτής, που ήταν η διάταξη των στοιχείων διακόσμησης. Αυτά αποτελούσαν αντικείμενα που θα έβρισκε κανείς στους δρόμους μιας πόλης, καθώς και πυκνή βλάστηση η οποία είχε πάρει υπό τον έλεγχο της τα καταστραμμένα τμήματα πόλης στην πίστα. Προστέθηκαν, επίσης, μερικά μοντέλα από το Tanks tutorial.

Στη συνέχεια προστέθηκαν και επεξεργάστηκαν το μοντέλο του αρχηγού, η μπάρα ζωής τους με χρήση ενός 2D αντικειμένου Slider της Unity και η εικόνα του τα οποία τοποθετήθηκαν πάνω στον καμβά που περιείχε τις μπάρες του παίκτη, οι κάμερες που δείχνουν τα οχήματα του παίκτη από την μπροστινή τους μεριά, τα "φιλικά κανόνια, ο ουρανοξύστης στον τερματισμό, οι αόρατοι τοίχοι με τις προειδοποιήσεις τους, καθώς και όλα τα Trigger Colliders που ενεργοποιούν τις δυνατότητές τους όταν έρθουν σε επαφή με τα Trigger Colliders που υπάρχουν στα οχήματα του παίκτη.

Τέλος, έγινε μια μετατροπή στην κάμερα του αυτοκινήτου, προστέθηκε άλλο ένα μουσικό κομμάτι που λήφθηκε από την ιστοσελίδα youtube.com (βλέπε Disclaimer στην σύνοψη) για να καλύψει την μουσική επένδυση της πίστας, καθώς και οι καμβάδες με τις εικόνες αρχής, νίκης και ήττας, όπως και τα μουσικά κομμάτια των δύο τελευταίων.



Σχήμα 50: Πρώτο σταδιο κατασκευής δεύτερης πίστας

Σχήμα 51: Δεύτερο σταδιο κατασκευής δεύτερης πίστας

Σχήμα 52: Τρίτο σταδιο κατασκευής δεύτερης πίστας

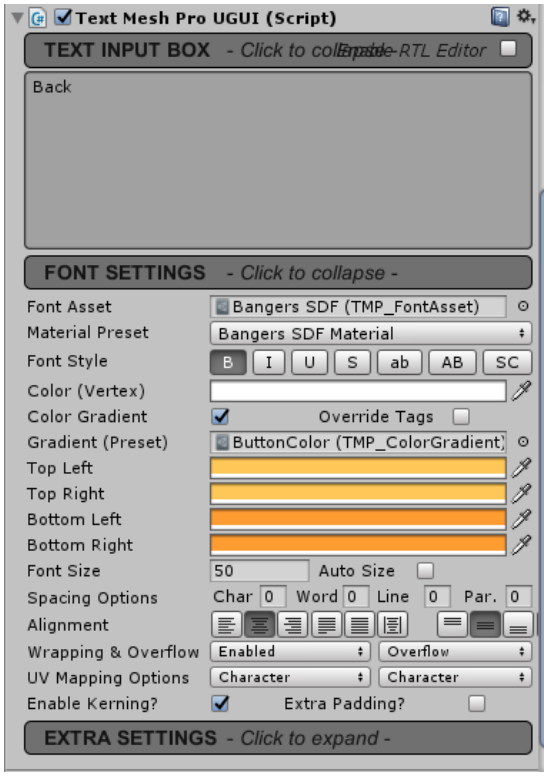
4.4 Δημιουργία των μενού, οθόνης φόρτωσης και βίντεο εισαγωγής

Απαραίτητη κρίθηκε η ύπαρξη ενός αρχικού μενού, του μενού παύσης και της οθόνης φόρτωσης για την σωστή λειτουργία του παιχνιδιού καθώς και για την ομαλότερη διεπαφή του παιχνιδιού με τον παίκτη. Το βίντεο εισαγωγής λήφθηκε από το youtube.com (βλέπε Disclaimer στην σύνοψη) και προστέθηκε αργότερα ως μέσω ολοκλήρωσης του παιχνιδιού.

4.4.1 Αρχικό μενού

Για την κατασκευή του αρχικού μενού κατασκευάστηκε μια νέα σκηνή και χρησιμοποιήθηκε ένα 2D στοιχείο Canvas της Unity, το οποίο θα περιείχε τα αντίστοιχα κουμπιά που πρέπει να περιέχει το μενού ενός παιχνιδιού. Αρχικά προστέθηκε μια μπλε εικόνα για φόντο, ένα μουσικό κομμάτι (βλέπε Disclaimer στην σύνοψη) καθώς και μια δεύτερη εικόνα με το λογότυπο του Action Man (βλέπε Disclaimer στην σύνοψη) που τοποθετήθηκε στην κορυφή. Έπειτα κατασκευάστηκαν και διατάχτηκαν τα κουμπιά, το κείμενο των οποίων γράφτηκε με χρήση του Text Mesh Pro, ενός επιπρόσθετου στοιχείου από το Asset Store, και όχι του προκαθορισμένου στοιχείου text της Unity.

Το στοιχείο αυτό χρησιμοποιήθηκε για όλα τα κουμπιά και όλα τα κομμάτια κειμένου στο αρχικό μενού.



Σχήμα 53: Το στοιχείο Text Mesh Pro που χρησιμοποιήθηκε για το αρχικό μενού

Το αρχικό μενού διαθέτει τέσσερα κουμπιά, τρία από τα οποία οδηγούν σε άλλα τρία υπομενού, ενώ το τέταρτο κουμπί κλείνει την εφαρμογή. Τα υπομενού που υπάρχουν είναι το μενού ρυθμίσεων, το μενού επιλογής πίστας, καθώς και το μενού με οδηγίες για το πως παίζεται το παιχνίδι. Όλα τα υπομενού διαθέτουν ένα κουμπί που μεταφέρει τον παίκτη πίσω στο αρχικό μενού.

Παράλληλα, το μενού επιλογής πίστας διαθέτει δύο κουμπιά, ένα για κάθε πίστα, με τον τίτλο της κάθε πίστας να αναγράφεται πάνω στο κουμπί. Το μενού με οδηγίες περιέχει μονάχα στοιχεία κειμένου και δεν διαθέτει κάποιο επιπλέον κουμπί, πέρα από εκείνο που μεταφέρει τον παίκτη στο αρχικό μενού.

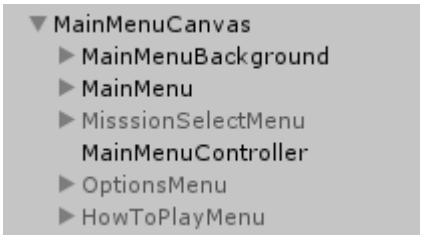
Ακόμα, το μενού ρυθμίσεων παρέχει στον χρήστη την δυνατότητα να αλλάξει την ποιότητα των γραφικών, την ανάλυση της οθόνης, την ένταση των ήχων του παιχνιδιού, καθώς και να θέσει το παιχνίδι σε λειτουργία πλήρης οθόνης ή παραθύρου. Η αλλαγή ποιότητας των γραφικών και ανάλυσης οθόνης έγινε με το 2D στοιχείο Dropdown της Unity, η αλλαγή έντασης του ήχου υλοποιήθηκε με ένα Slider που συνδεόταν με ένα στοιχείο Audio Mixer, το οποίο συνδεόταν με όλα τα στοιχεία Audio Source που χρησιμοποιήθηκαν στο παιχνίδι, ενώ η εναλλαγή από λειτουργία πλήρης οθόνης σε λειτουργία παραθύρου πραγματοποιήθηκε με χρήση του 2D στοιχείου Toggle της Unity.

Αφού ολοκληρώθηκε η διαδικασία δημιουργίας και διάταξης των στοιχείων των μενού, γράφτηκαν δύο scripts για την λειτουργία τους. Να σημειωθεί πως η μετάβαση από το ένα υπομενού στο άλλο υλοποιήθηκε απευθείας με λειτουργίες των κουμπιών και δεν χρειάστηκε η δημιουργία κάποιου script για αυτό.

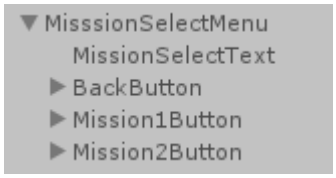
Για την λειτουργία μετάβασης από το αρχικό μενού στις πίστες ήταν αναγκαίο να τοποθετηθούν πρώτα τα scenes του μενού και των πιστών στα Build Settings του παιχνιδιού και έπειτα η κατασκευή της οθόνης φόρτωσης μαζί με το script της.

Μετά την δημιουργία των μενού,προστέθηκαν στις εικόνες ήττας και νίκης τα κουμπιά μετάβασης στο αρχικού μενού και φόρτωσης της τρέχουσας της πίστας από την αρχή.

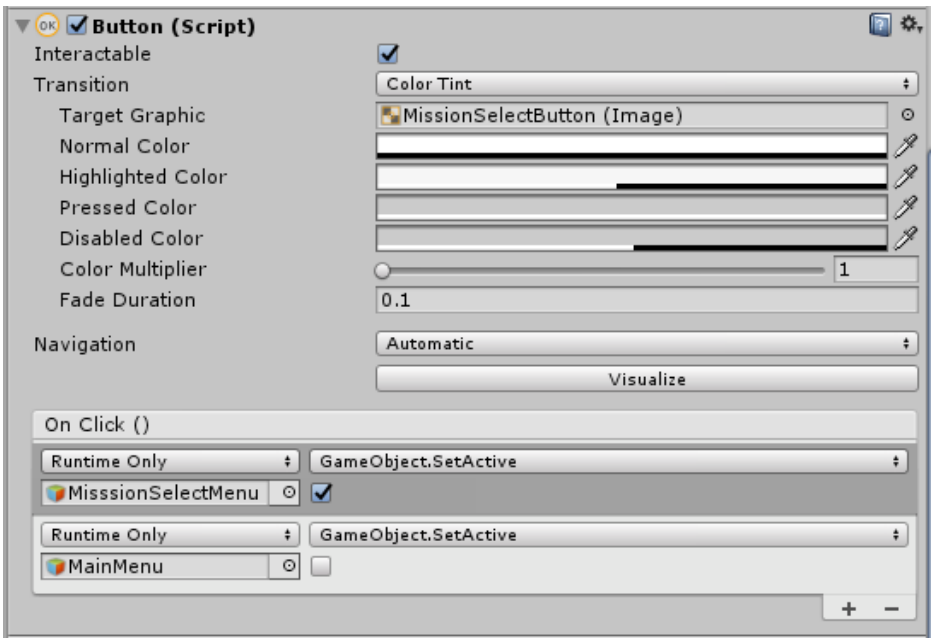
Σχήμα 54: Το αρχικό μενού του παιχνιδιού



Σχήμα 55: Τα κυριότερα στοιχεία του κεντρικού μενού



Σχήμα 56: Τα στοιχεία του μενού επιλογής πίστας



Σχήμα 57: Το κουμπί που μεταφέρει τον παίκτη στο μενού επιλογής πίστας

4.4.2 Οθόνη φόρτωσης

Η οθόνη φόρτωσης χρησιμοποιείται για την ομαλή μετάβαση από την μια σκηνή στην άλλη, πιο συγκεκριμένα για την μετάβαση από την σκηνή του αρχικού μενού σε κάποια πίστα και το αντίστροφο, όπως για την εναλλαγή από την σκηνή που περιέχει το βίντεο εισαγωγής στην σκηνή του αρχικού μενού.

Αποτελείται από ένα αντικείμενο Canvas το οποίο περιέχει μια εικόνα για φόντο, ένα Slider που γεμίζει σταδιακά, καθώς και ένα στοιχείο κειμένου που παρουσιάζει με χρήση αριθμών την πρόοδο φόρτωσης της σκηνής.

Σχήμα 58: Η οθόνη φόρτωσης

4.4.3 Μενού παύσης

Όσο ο παίκτης παίζει κάποια από τις δύο πίστες του παιχνιδιού, έχει την δυνατότητα να πατήσει το κατάλληλο πλήκτρο για να θέσει το παιχνίδι σε παύση. Όταν γίνεται αυτό, η οθόνη σκοτεινιάζει ελαφρώς, το παιχνίδι σταματά πλήρως, η ένταση της μουσικής χαμηλώνει, και εμφανίζεται στην οθόνη το μενού της παύσης.

Το μενού αυτό περιέχει τρία κουμπιά, ένα που αν πατηθεί εξαφανίζει το μενού και το παιχνίδι συνεχίζει όπως πριν, ένα που κάνει την τρέχουσα πίστα να ξαναφορτώσει από την αρχή και το τελευταίο που μεταφέρει τον παίκτη πίσω στο αρχικό μενού.

Το μενού παύσης δημιουργήθηκε με χρήση του αντικειμένου Canvas που περιέχει ένα ημιδιάφανο 2D αντικείμενο Panel, πάνω στο οποίο έχουν τοποθετηθεί τα αντικείμενα των κουμπιών.

4.4.4 Βίντεο εισαγωγής

Το πρώτο που αντικρίζει ο παίκτης ανοίγοντας το παιχνίδι είναι ένα βίντεο εισαγωγής, το οποίο μπορεί να παραληφθεί με το πάτημα οποιουδήποτε πλήκτρου. Το βίντεο αυτό λήφθηκε από το youtube.com και αποτελεί το βίντεο εισαγωγής της παιδικής σειράς κινουμένων σχεδίων Action Man (βλέπε Disclaimer στην σύνοψη).

Το βίντεο χρησιμοποιήθηκε με χρήση του αντικειμένου Video Player της Unity, και αφού τελειώσει ή παραληφθεί, τότε ο παίκτης μεταφέρεται στο μενού μέσω μιας σύντομης οθόνης φόρτωσης.

Σχήμα 59: Στιγμιότυπο από το βίντεο εισαγωγής

4.5 Scripts

Σε αυτήν την ενότητα παρουσιάζονται και επεξηγούνται σύντομα τα σημαντικότερα scripts του παιχνιδιού, τα οποία είναι αναγκαία για την ορθή λειτουργία του. Τα scripts παρουσιάζονται αναλυτικότερα στο Παράρτημα Α, στην σελίδα 50.

PlayerCarController

Αυτό το script αναλαμβάνει να υλοποιήσει τις λειτουργίες ελέγχου του αυτοκινήτου του παίκτη. Περιέχει τις απαραίτητες εντολές για την επιτάχυνσή, το φρενάρισμα, τις στροφές, την λειτουργία drift, τα όπλα του καθώς και την κύλιση και στροφή των τροχών. Η κίνηση και οι στροφές του αυτοκινήτου εκτελούνται με την χρήση συναρτήσεων. Ταυτόχρονα περιέχει τις εντολές για διάφορα γεγονότα που ενεργοποιούνται όταν το αυτοκίνητο έρχεται σε επαφή με συγκεκριμένα Trigger Colliders, όπως την μεταμόρφωσή του σε ελικόπτερο, την ενεργοποίηση του πίσω τοίχου στις αρένες, και την εναλλαγή της κάμερας όταν καταδιώκεται από τον "αρχηγό" της δεύτερης πίστας. Κατά την διάρκεια μεταμόρφωσης, τα στατιστικά του αυτοκινήτου μεταφέρονται στο ελικόπτερο. Ακόμα, σε αυτό το script δημιουργείται ένα στιγμιότυπο των script StatsAndPickUps και JetStatsAndPickUps, τα οποία περιέχουν τα στατιστικά τόσο του αυτοκινήτου,όσο και του ελικοπτέρου. Το πρώτο χρησιμοποιείται για τον έλεγχο της μπάρας των πυραύλων όταν ο χρήστης θέλει να εκτοξεύσει τους πυραύλους του, ενώ το δεύτερο χρησιμοποιείται μαζί με το πρώτο για την μετάβαση των στατιστικών κατά την μεταμόρφωση.

PlayerChopperController

Σε αυτό το script εμπεριέχονται οι εντολές που υλοποιούν τις λειτουργίες ελέγχου του ελικοπτέρου του παίκτη. Παρόμοια με το script του αυτοκινήτου, σε αυτό το script υπάρχουν οι εντολές για την επιτάχυνση, την κατακόρυφη κίνηση, την στροφή, την λειτουργία αποφυγής τα όπλα, όπως και για την περιστροφή του έλικα. Επίσης, εμπεριέχονται και οι εντολές δύο λειτουργιών που πυροδοτούνται με την επαφή του ελικοπτέρου με τα αντίστοιχα Trigger Colliders. Η μια από αυτές είναι η μεταμόρφωση σε αυτοκίνητο, ενώ η δεύτερη αφορά στην εμφάνιση της προειδοποίησης όσο ο παίκτης πετά κοντά σε κάποιον οριζόντιο αόρατο τοίχο. Όπως και στο script του αυτοκινήτου, όταν εκτελείται η μεταμόρφωση, τα στατιστικά του ελικοπτέρου μεταφέρονται στα στατιστικά του αυτοκινήτου. Παράλληλα, και σε αυτό το script δημιουργούνται τα στιγμιότυπα

των script JetStatsAndPickUps και StatsAndPickUps, τα οποία έχουν ακριβώς την ίδια χρήση με αυτήν που είχαν στο script του αυτοκίνητου.

StatsAndPickUps και JetStatsAndPickUps

Τα δύο αυτά scripts περιέχουν τον κώδικα για τα στατιστικά του αυτοκινήτου και του ελικοπτέρου του παίκτη, αντίστοιχα, καθώς και την αύξηση των τιμών τους από τα αντικείμενα που συλλέγει ο παίκτης από τις πίστες. Υπάρχουν περιορισμοί οι ανάλογοι περιορισμοί για την μη υπέρβαση των μέγιστων τιμών όπως και για την μη μείωση των τιμών τους κάτω από τις ελάχιστες τιμές. Η επίδραση των αντικειμένων ενεργοποιείται μέσω της επαφής με τα Trigger Colliders που διαθέτουν. Παράλληλα, τόσο το αυτοκίνητο όσο και το ελικόπτερο καταστρέφονται αν αγγίξουν το έδαφος. Επιπλέον, το ελικόπτερο καταστρέφεται αν έρθει σε επαφή και με τους αόρατους τοίχους.

DamageController και JetDamageController

Αυτά τα δύο scripts αντιπροσωπεύουν την ζημία που θα υποστεί το αυτοκίνητο και το ελικόπτερο του παίκτη, αντίστοιχα, από τις διάφορες απειλές που βρίσκονται στις πίστες του παιχνιδιού. Οι λειτουργίες αυτές υλοποιούνται μέσω των Trigger Colliders που υπάρχουν στις σφαίρες των εχθρών και στα εμπόδια. Σε αυτά τα scripts δημιουργούνται στιγμιότυπα των προαναφερθέντων scripts StatsAndPickUps και JetStatsAndPickUps για το αυτοκίνητο και το ελικόπτερο, αντίστοιχα. Αυτό συμβαίνει επειδή η ζημιά που θα δεχτεί ο παίκτης θα επηρεάσει τις τιμές των στατιστικών του μέσω αυτών των στιγμιότυπων. Πιο συγκεκριμένα, τόσο και το αυτοκίνητο δέχονται ζημιά από την επαφή με τα πυρά των εχθρών και τα εμπόδια.

Enemy και EnemyHeavyTank

Αυτά τα δύο scripts είναι πανομοιότυπα, και διαχειρίζονται την λειτουργία των εχθρών εδάφους που ο παίκτης αντιμετωπίζει στις αρένες. Διαχειρίζονται την κίνηση των εχθρών, την λειτουργία του navigator tool της Unity, την ικανότητα τους να πυροβολούν τον παίκτη μόνο εφόσον είναι στραμμένοι προς αυτόν, όπως και την ζημιά που δέχονται από τις σφαίρες και τους πυραύλους του παίκτη. Η μοναδική διαφορά τους εντοπίζεται στις βολές που εκτοξεύουν. Το εχθρικό τανκ και ο τροχός εκτοξεύουν σφαίρες από ένα κανόνι, ενώ το εχθρικό βαρύ τανκ από δύο.

StraightMovingEnemy

Σε αυτό το script εμπεριέχονται οι εντολές λειτουργίας όλων των εχθρών που κινούνται σε μια ευθεία, δηλαδή όλων των ιπτάμενων εχθρών και των εχθρών εδάφους της δεύτερης πίστας. Όπως τα δύο προηγούμενα scripts, διαχειρίζεται την κίνησή αυτών των εχθρών καθώς και την ζημιά που δέχονται από τα όπλα του παίκτη. Επιπλέον, τους δίνουν έναν προκαθορισμένο χρόνο ζωής, δεν διαθέτουν εντολές για το navigator tool και πυροβολούν συνεχώς, δίχως να ελέγχουν αν είναι στραμμένοι προς τον παίκτη ή όχι.

GroundEnemyActivator και FinalBattleActivator

Αυτά τα δύο scripts αναλαμβάνουν να υλοποιήσουν τις εντολές για τις μάχες στις αρένες που λαμβάνουν χώρα στην πρώτη πίστα του παιχνιδιού. Εφαρμόζονται στο αόρατο Trigger Collider που υπάρχει στην είσοδο κάθε αρένας, και μόλις έρθει σε επαφή με τον παίκτη, τα scripts αυτά κλείνουν τους τοίχους πίσω του και ενεργοποιούν τους εχθρούς.

Η διαφορά τους εντοπίζεται στο γεγονός που πυροδοτείται μετά την εξόντωση των εχθρών. Πιο συγκεκριμένα, εφόσον όλοι οι εχθροί μιας αρένας είναι “παιδιά” ενός άλλου αντικειμένου, τα scripts ελέγχουν συνεχώς για το αν το αντικείμενο αυτό έχει “παιδιά”. Όταν ανακαλύψει πως δεν έχουν, δηλαδή οι εχθροί έχουν εξουδετερωθεί, το GroundEnemyActivator απενεργοποιεί το αντικείμενο των κώνων δρόμου που φράζουν τον δρόμου του παίκτη, ενώ το FinalBattleActivator εκκινεί την διαδικασία τερματισμού της πίστας. Σε αυτό το script δημιουργείται ένα στιγμιότυπο του PlayerCarController, το οποίο χρησιμοποιείται για να μηδενιστεί η ταχύτητα του αυτοκινήτου,

έπειτα μειώνεται η ένταση της μουσικής της πίστας, μέχρι να σταματήσει τελείως και τέλος,εμφανίζει τον καμβά με το μήνυμα νίκης.

Boss, BossChase και BossAppearance

Αυτά τα scripts αντιπροσωπεύουν όλες της λειτουργίες του "αρχηγού" της δεύτερης πίστας. Το Boss Appearance εφαρμόζεται σε αόρατα Trigger Colliders και οι εντολές του ενεργοποιούν το μοντέλο του αρχηγού όταν ο παίκτης έρθει σε επαφή μαζί τους.

Το BossChase καλύπτει τις περιπτώσεις καταδίωξης. Είναι πανομοιότυπο με το script StraightMovingEnemy, με τις κατάλληλες τροποποιήσεις για να καλύψει μια διαφορά. Ο "αρχηγός" πυροβολεί από δύο κανόνια, ενώ οι εχθροί που χρησιμοποιούν το προαναφερθέν script πυροβολούν από ένα.

Παράλληλα, το Boss αναλαμβάνει μονάχα την κίνηση για τις απλές εμφανίσεις του "αρχηγού" μέσα στην πίστα, όπως και την καταστροφή του από τους πυραύλους των "φιλικών" κανονιών, όταν έρθει σε επαφή με τα Trigger Colliders που διαθέτουν.

Επίσης, τα δύο τελευταία scripts δίνουν στο μοντέλο του "αρχηγού" περιορισμένο χρόνο ζωής.

GameController

Το GaneController διαχειρίζεται δύο βασικές καταστάσεις: την αρχή της κάθε πίστας και την καταστροφή των οχημάτων του παίκτη όταν η ζωή τους πέσει στο 0. Πιο συγκεκριμένα, όταν ξεκινήσει μια πίστα το GameController δείχνει για δύο δευτερόλεπτα την εικόνα αρχής, και αμέσως μετά ενεργοποιεί το αυτοκίνητο του παίκτη.

Παράλληλα, ελέγχει συνεχώς αν η ζωή του αυτοκινήτου ή του ελικοπτέρου είναι ίση με το 0. Αν συμβεί αυτό, τότε χρησιμοποιεί μια εφεδρική κάμερα για να την κάμερα του αντίστοιχου οχήματος, επειδή εκείνη θα καταστραφεί μαζί με αυτό μέσα σε μια έκρηξη και μετά από δύο δευτερόλεπτα σταματά την μουσική της πίστας και εμφανίζεται στην οθόνη η εικόνα ήττας.

OptionsMenuController

Το τελευταίο script που παρουσιάζεται σε αυτό το κεφάλαιο, είναι αυτό των ρυθμίσεων του παιχνιδιού. Το script αυτό αναλαμβάνει να υλοποιήσει τις εντολές που χρειάζονται για να μπορέσει ο παίκτης να πραγματοποιήσει τις αλλαγές που επιθυμεί. Οι λειτουργίες εκτελούνται με χρήση συναρτήσεων.

Για την αλλαγή της ανάλυσης οθόνης, το script διαβάζει τις αναλύσεις του υπολογιστή, τις αποθηκεύει σε μια λίστα και τις παρουσιάζει στο Dropdown του μενού των ρυθμίσεων. Έπειτα εφαρμόζει την επιλογή του παίκτη στο παιχνίδι.

Ταυτόχρονα, για την ένταση των ήχου, το script συνδέεται με ένα στοιχείο Audio Mixer, στο οποίο είναι συνδεδεμένα όλα τα Audio Sources του παιχνιδιού. Επίσης, οι επιλογές των γραφικών και της πλήρης οθόνης, συνδέονται άμεσα με τις διαθέσιμες επιλογές που παρέχει η Unity.

5 Τρέξιμο παιχνιδιού

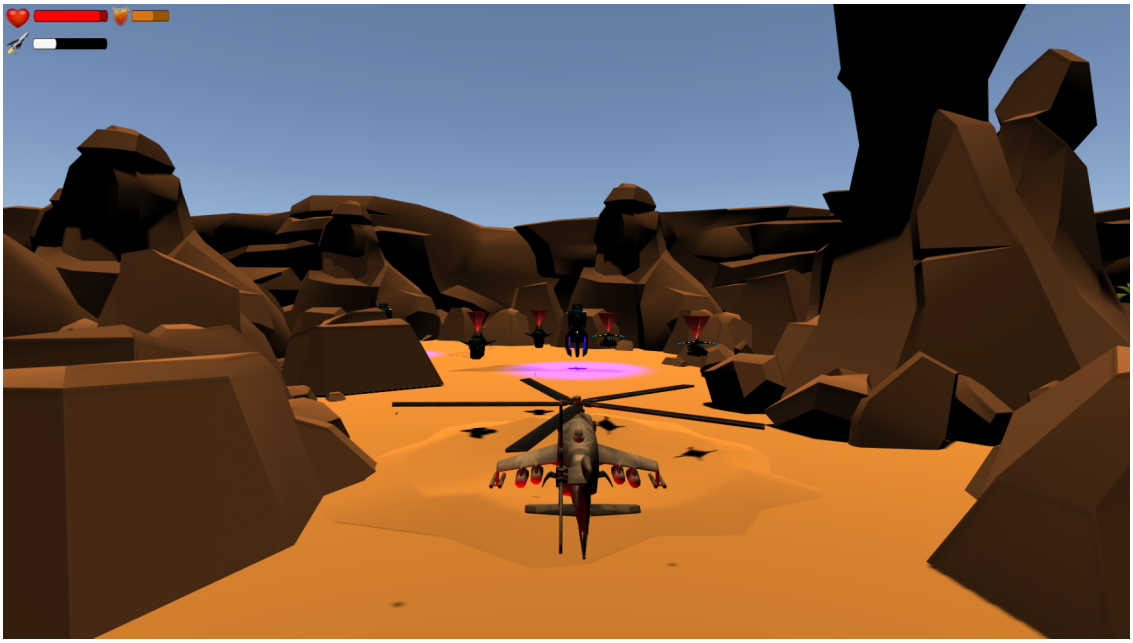
Αυτό το κεφάλαιο αποτελεί το τελευταίο της παρούσας πτυχιακής εργασίας. Παρουσιάζεται το τελικό αποτέλεσμα του παιχνιδιού με χρήση εικόνων που απεικονίζουν διάφορα στιγμιότυπα από το gameplay, τα μενού, καθώς και τις εικόνες αρχής, ήττας και νίκης. Στην συνέχεια περιγράφονται οι δυσκολίες που εμφανίστηκαν κατά την υλοποίησή του, και τέλος, αναφέρονται κάποιες πιθανές μελλοντικές τροποποιήσεις που θα μπορούσαν να επεκτείνουν και να βελτιώσουν το παιχνίδι.

5.1 Gameplay

Παρακάτω παρουσιάζονται μερικές σκηνές από το gameplay του παιχνιδιού, συνοδευόμενες από σύντομες περιγραφές.

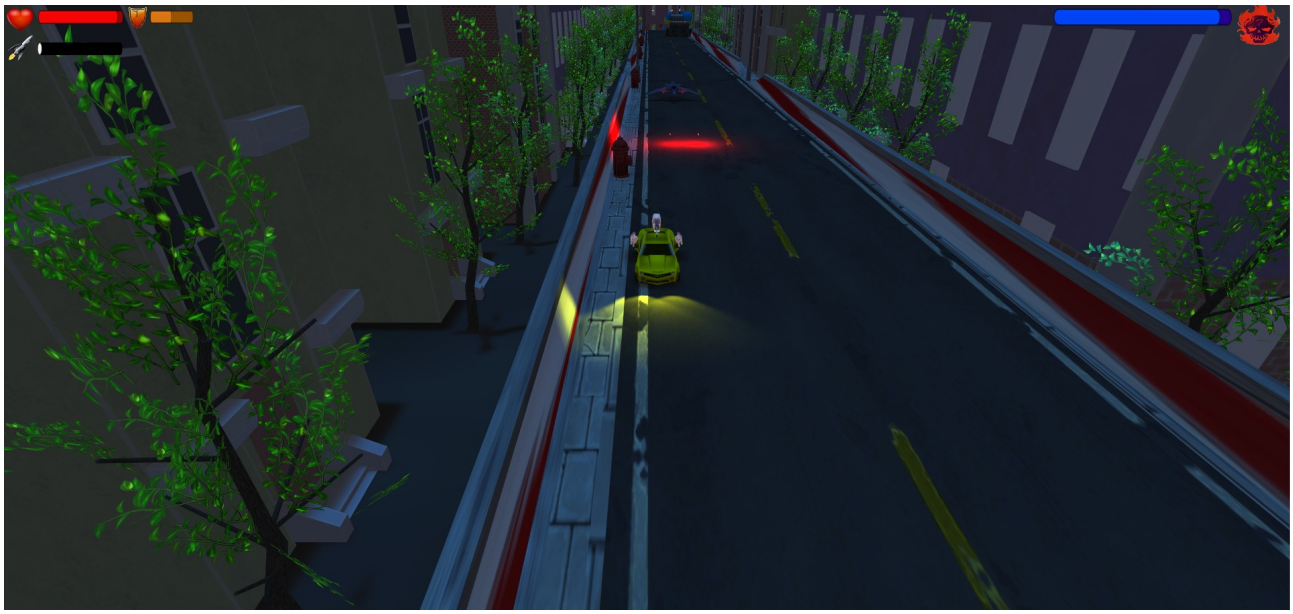
Σχήμα 60: Στιγμιότυπο από gameplay της πρώτης πίστας με το αυτοκίνητο

Στην παραπάνω εικόνα παρουσιάζεται ένα δείγμα της πρώτης πίστας όσο ο παίκτης ελέγχει το αυτοκίνητο. Η εικόνα τραβήχτηκε λίγα δευτερόλεπτα μετά την αρχή της πρώτης πίστας όπου το αυτοκίνητο είναι πάνω σε μια γέφυρα και προσπαθεί να αποφύγει ένα εμπόδιο εδάφους που βρίσκεται μπροστά του.



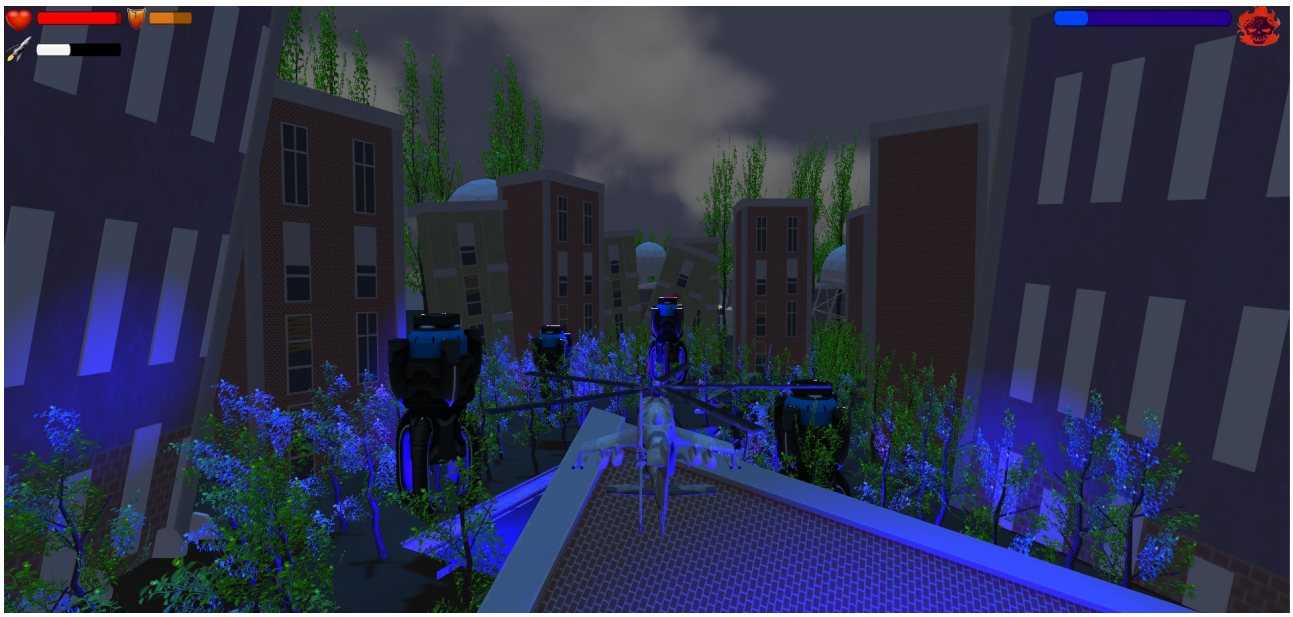
Σχήμα 61: Στιγμιότυπο από gameplay της πρώτης πίστας με το ελικόπτερο

Σε αυτήν την εικόνα φαίνεται το ελικόπτερο του παίκτη ενώ τίθεται αντιμέτωπο με κάποιους ιπτάμενους εχθρούς στην πρώτη πίστα, ενώ ακριβώς πίσω τους βρίσκεται ένα ιπτάμενο εμπόδιο.



Σχήμα 62: Στιγμιότυπο από καταδίωξη "αρχηγού" στην δεύτερη πίστα

Το σχήμα 62 αποθανατίζει την πρώτη σκηνή καταδίωξης στην δεύτερη πίστα. Η κάμερα δείχνει το αυτοκίνητο από μπροστά, και ακριβώς πίσω από το αυτοκίνητο βρίσκεται ο “αρχηγός” που το κυνηγά και εκτοξεύει δύο σφαίρες ταυτόχρονα.



Σχήμα 63: Στιγμιότυπο από gameplay της δεύτερης πίστας με το ελικόπτερο

Στην παραπάνω εικόνα φαίνεται το τελευταίο κομμάτι οδήγησης του ελικοπτέρου της δεύτερης πίστας. Μπροστά από τον παίκτη βρίσκονται μερικά ιπτάμενα εμπόδια.

Σχήμα 64: Εικόνα αρχής

Σχήμα 65: Εικόνα νίκης

Σχήμα 66: Εικόνα ήττας

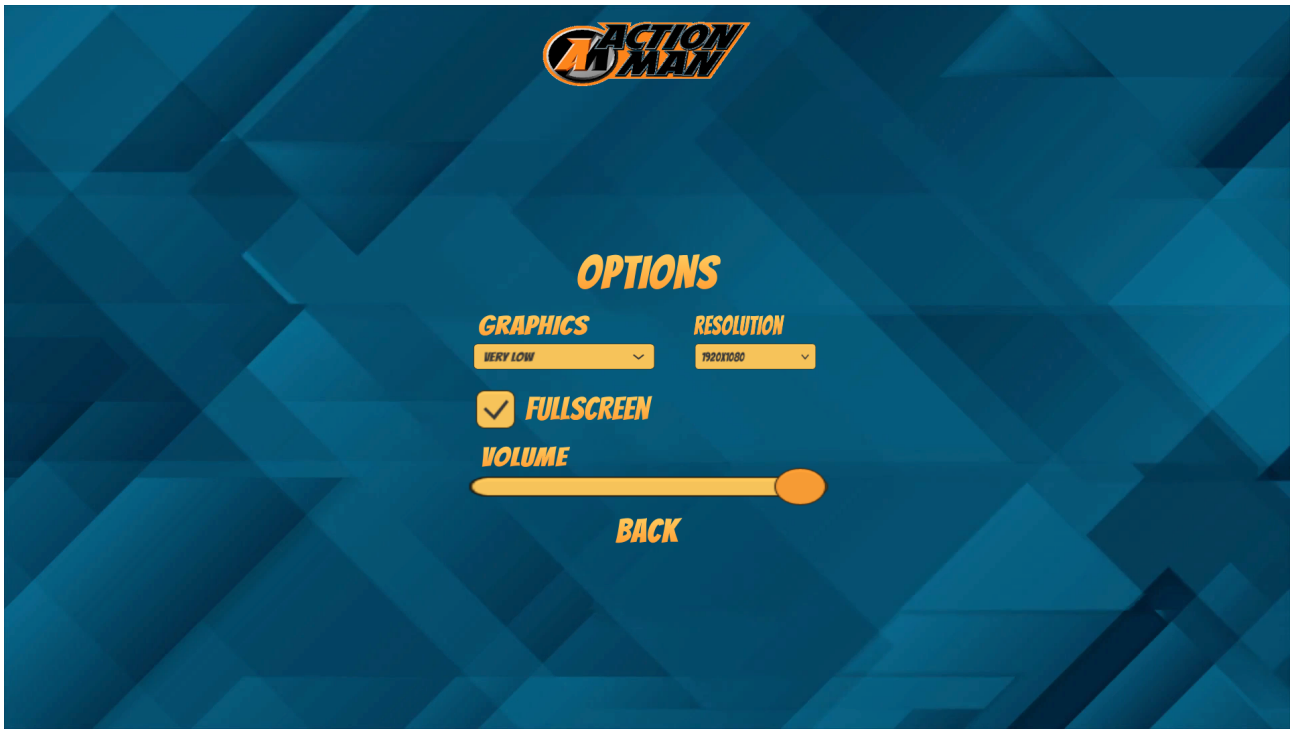
Στα παραπάνω σχήματα απεικονίζονται οι εικόνες αρχής, νίκης και ήττας, με τις δύο τελευταίες να διαθέτουν τα ανάλογα κουμπιά για την επιστροφή στο κεντρικό μενού και φόρτωση της τρέχουσας πίστας από την αρχή.

5.2 Μενού

Σε αυτήν την υποενότητα παρουσιάζονται εικόνες που απεικονίζουν τα διάφορα μενού του παιχνιδιού, εκτός του αρχικού, που έχει παρουσιαστεί σε προηγούμενο κεφάλαιο.

Σχήμα 67: Το μενού οδηγιών

Σχήμα 68: Το μενού επιλογής πίστας



Σχήμα 69: Το μενού ρυθμίσεων

Σχήμα 70: Το μενού παύσης

5.3 Δυσκολίες που παρουσιάστηκαν

Είναι λογικό και αναμενόμενο να υπάρχουν διάφορες δυσκολίες κατά την ανάπτυξη και υλοποίηση του παιχνιδιού. Αρχικά ήταν η εξοικείωση και η απόκτηση των απαραίτητων γνώσεων σχετικά με την Unity, το περιβάλλον της, τις δυνατότητες που προσφέρει στον χρήστη καθώς και η εξοικείωση με την γλώσσα προγραμματισμού C#. Αυτό το εγχείρημα ήταν ακόμα πιο δύσκολο, διότι δεν ήταν δυνατή η δήλωση και η παρακολούθηση του αντίστοιχου μαθήματος που διαθέτει η σχολή λόγω εξωτερικών παραγόντων. Ήταν απαραίτητη η μελέτη της γλώσσας, του περιβάλλοντος της Unity, καθώς και η εξάσκηση με διάφορα tutorials που υπάρχουν στην ιστοσελίδα της εφαρμογής.

Πολύτιμη αποδείχτηκε η βοήθεια που προσέφεραν διάφορες ιστοσελίδες του διαδικτύου σχετικά με τις αμέτρητες απορίες που προέκυψαν κατά την υλοποίηση του project. Σε αυτές τις ιστοσελίδες κατατάσσονται το documentation[14] της Unity, όπως και το Unity Answers[15].

Ακόμα, απαραίτητη κρίθηκε η χρήση τριών ακόμα προγραμμάτων για την επεξεργασία μερικών εικόνων, και ηχητικών αρχείων. Για τις εικόνες χρησιμοποιήθηκε το πρόγραμμα Gimp[16], όπως και η εφαρμογή της ζωγραφικής των Windows 10, ενώ για την επεξεργασία ηχητικών αρχείων χρησιμοποιήθηκε το πρόγραμμα iTunes[17] της Apple.

Τέλος, η διαδικασία εύρεσης ορισμένων 3D μοντέλων και του ήχου μηχανής του αυτοκινήτου που χρησιμοποιήθηκαν στο παιχνίδι ήταν αρκετά χρονοβόρα αφού απαιτούσε δοκιμή τόσο των μοντέλων όσο και του ήχου μέσα στην εφαρμογή της Unity.

5.4 Πιθανές τροποποιήσεις

Το παιχνίδι μπορεί να δεχτεί αρκετές τροποποιήσεις που μπορούν τόσο να το βελτιώσουν σχετικά με την τρέχουσα μορφή του, όσο να το επεκτείνουν προσθέτοντάς του διάφορα επιπλέον στοιχεία.

Μια πιθανή βελτίωση είναι η διόρθωση ενός προβλήματος που υπάρχει σχετικά με τις συγκρούσεις του αυτοκινήτου. Όταν το αυτοκίνητο κάνει drift για πολλά δευτερόλεπτα

ακουμπώντας έναν τοίχο, τότε το αυτοκίνητο μπορεί να περάσει μέσα από τον τοίχο και να βρεθεί εκτός πίστας. Μια άλλη βελτίωση είναι η καλύτερη διαχείριση του ήχου της μηχανής του αυτοκινήτου ώστε να ακούγεται πιο ρεαλιστικό, αφού σε αυτό το στάδιο του παιχνιδιού, το αυτοκίνητο δίνει την εντύπωση πως αλλάζει για πάντα ταχύτητες με τον ίδιο ρυθμό.

Σαν επέκταση μπορεί να θεωρηθεί η προσθήκη παραπάνω πιστών στο παιχνίδι, οι οποίες θα αναπαριστούν διαφορετικά περιβάλλοντα όπως ζούγκλες και αρκτικά τοπία. Ακόμα καθίσταται δυνατή η προσθήκη παραπάνω οχημάτων από τα οποία θα μπορεί να διαλέγει ο παίκτης ποιο θα χρησιμοποιήσει, όπως και η δυνατότητα αναβάθμισή τους.

5.5 Στοιχεία που χρησιμοποιήθηκαν

5.5.1 Μοντέλα και αντικείμενα

Μοντέλο αυτοκινήτου:
https://www.assetstore.unity3d.com/en/#!/content/12133
Μοντέλο ελικοπτέρου:
https://www.assetstore.unity3d.com/en/?stay#!/content/8405
Μοντέλα εχθρών και εμποδίων:
https://www.assetstore.unity3d.com/en/?stay#!/content/2358
Μοντέλα πόλης:
https://www.assetstore.unity3d.com/en/?stay#!/content/67029
Μοντέλα ερήμου και σφαιρών:
https://www.assetstore.unity3d.com/en/?stay#!/content/46209
Αντικείμενα εκρήξεων και ηχητικών τους εφέ:
https://www.assetstore.unity3d.com/en/?stay#!/content/13866
Μοντέλο πυραύλων:
https://www.assetstore.unity3d.com/en/?stay#!/content/71890
Μοντέλα δρόμων:
https://www.assetstore.unity3d.com/en/?stay#!/content/67475
Μοντέλα όπλων αυτοκινήτου:
https://www.assetstore.unity3d.com/en/?stay#!/content/25525
Μοντέλο “αρχηγού”:
https://www.assetstore.unity3d.com/en/?stay#!/content/23174
Αντικείμενο ασπίδας:
https://www.assetstore.unity3d.com/en/?stay#!/content/61351
Αντικείμενο πυραύλου:
https://www.assetstore.unity3d.com/en/?stay#!/content/72692
Αντικείμενο ζωής:
https://www.turbosquid.com/AssetManager/Index.cfm?
stgAction=getFiles&subAction=Download&intID=935557&intType=3
SkyBox νυχτερινού ουρανού:
https://www.assetstore.unity3d.com/en/?stay#!/content/79066
Πυραμίδες, οβελίσκοι και βελάκια εχθρών:
https://www.assetstore.unity3d.com/en/?stay#!/content/86482
“Φιλικά” κανόνια:
https://www.assetstore.unity3d.com/en/?stay#!/content/62750
Text Mesh Pro:
https://www.assetstore.unity3d.com/en/?stay#!/content/84126

5.5.2 Βίντεο και ηχητικά κλιπ

(Βλέπε Disclaimer στην Σύνοψη)
Μουσική πρώτης πίστας:

https://www.youtube.com/watch?v=ZTidn2dBYbY
Μουσική δεύτερης πίστας:
https://www.youtube.com/watch?v=UpSHC1dqX1o
Μουσική μενού:
https://www.youtube.com/watch?v=vX0D36bWKFE
Ήχος μεταμόρφωσης από αυτοκίνητο σε ελικόπτερο:
https://www.youtube.com/watch?v=pkdry54C0oU
Ήχος μεταμόρφωσης από ελικόπτερο σε αυτοκίνητο:
https://www.youtube.com/watch?v=MHHlRkZvOPg
Ήχοι πυραύλων:
https://www.youtube.com/watch?v=1qUN59Oebe0
Έκρηξη παίκτη:
https://www.youtube.com/watch?v=XjOtGgxaX5g
Έκρηξη "αρχηγού":
https://www.youtube.com/watch?v=Gqoz5G-ryFc
Ήχοι σφαιρών και λοιπών εκρήξεων:
https://www.assetstore.unity3d.com/en/?stay#!/content/13866
Βίντεο εισαγωγής:
https://www.youtube.com/watch?v=6C_0xLGmkEo
Ήχος αντικειμένων:
https://www.youtube.com/watch?v=URaCbswg7Ks
Ήχος νίκης:
https://www.youtube.com/watch?v=ohhxE0762Ko
Ήχος ήττας:
https://www.youtube.com/watch?v=MHr3ag7B410
Ήχος μηχανής αυτοκινήτου:
https://www.youtube.com/watch?v=N_5MMzx5rOs
Ήχοι λάστιχων:
Πακέτο Standard Assets (διατίθεται μαζί με την εφαρμογή της Unity)

5.5.3 Εικόνες που χρησιμοποιήθηκαν στο παιχνίδι
(Βλέπε Disclaimer στην Σύνοψη)
Εικόνα αρχής:
https://www.ready.gov/sites/default/files/file_attach/AF_BeReadyLogo.jpg
Εικόνα νίκης:
https://vignette.wikia.nocookie.net/halo/images/a/a8/Halo_spartan_Assault_mission_complete.png/revision/latest?cb=20130604164611
Εικόνα ήττας:
https://i.ytimg.com/vi/aqXRGQJLsCk/maxresdefault.jpg
Εικόνα φόρτωσης:
http://1.bp.blogspot.com/-fAtepVmMou8/Up0oemAUybI/AAAAAAAAANY/JqpxjCNyoaw/s1600/ActionMan1.jpg
Λογότυπο Action Man:
http://pluspng.com/img-png/logo-action-man-png-action-man-logo-615.png
Λογότυπο Action Man για το μενού:
https://vector.me/files/images/1/3/13485/action_man.png
Φόντο μενού:
https://d2v9y0dukr6mq2.cloudfront.net/video/thumbnail/itCjTBE/videoblocks-abstract-dark-blue-corporate-geometric-motion-graphic-background-video-animation-ultra-hd-4k-3840x2160_bhio7vxwz_thumbnail-full01.png
Εικόνα ζωής του παίκτη:

http://download.seaicons.com/download/i6331/custom-icon-design/valentine/custom-icon-design-valentine-heart.ico
Εικόνα πυραύλων του παίκτη:
http://icons.iconarchive.com/icons/oxygen-icons.org/oxygen/128/Actions-fork-icon.png
Εικόνα ασπίδας του παίκτη:
https://lh3.googleusercontent.com/soZneqTc4l8NyffD2emP9-YS4jA4vRtG9yCfSt52sM7BxI7MEgUA1AdAMQPKBIqs3QwF=s85
Εικόνα κινδύνου:
http://4.bp.blogspot.com/-eicX6TMTW5w/UWT5NwW-ImI/AAAAAAAACsk/uzf3CCRfPyI/s640/7505_danger.jpg
Εικόνα ζωής "αρχηγού"
https://orig00.deviantart.net/923f/f/2016/214/d/1/el_diablo_skull_suicide_squad_transparent_cropped_by_classicgameguys-dacaxhb.png

5.5.4 Εξωτερικές εικόνες που χρησιμοποιήθηκαν στην παρούσα γραπτή αναφορά
(Βλέπε Disclaimer στην Σύνοψη)
https://d2q63o9r0h0ohi.cloudfront.net/images/media/screenshots/ss1-med-975cbaf0ad0d112e7665728d494a62a8764679fa76546c390afb8970d7ce261269fa6d31dc46b0f3c3942024ed8897f4cdcdce7efdf4f463247a342228b15319.jpg

https://i.kinja-img.com/gawker-media/image/upload/s--2bK09ynW--/c_scale,fl_progressive,q_80,w_800/bckkhydhrqoqxlmeymjf.jpg

https://connectingtohost.co.uk/assets/images/2017/cuphead.jpg

http://is1.mzstatic.com/image/thumb/Purple7/v4/61/1e/67/611e6765-8877-c937-dadd-5ea55f2a75f3/source/800x500bb.jpg

https://i.ytimg.com/vi/JzO6-1zdXoE/hqdefault.jpg

https://humblebundle.imgix.net/misc/files/hashed/70908e3f7482ea2334b10fd047d882be23bf430d.jpg?auto=compress,format&fit=crop&h=353&w=616&s=64eb23534c2852ecb6de828efb3318d8

http://www.mobygames.com/images/shots/l/623945-spy-hunter-nowhere-to-run-windows-screenshot-game-starts.jpg

Αναφορές

[1] Unity, ιστοσελίδα Unity3D: https://unity3d.com/
[2] Asset Store, ιστοσελίδα AssetStore.Unity3D: https://www.assetstore.unity3d.com/en/
[3] Hearthstone, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Hearthstone
[4] Pillars of Eternity, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Pillars_of_Eternity
[5] Furi, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Furi
[6] Hand of Fate, ιστοσελίδα Wikipedia:
https://en.wikipedia.org/wiki/Hand_of_Fate_(video_game)#Reception
[7] Cuphead, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Cuphead
[8] Action Man, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Action_Man
[9] Max Steel, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/Max_Steel
[10] G.I. JOE, ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/G.I._Joe
[11] SpyHunter (2001), ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/SpyHunter
[12] SpyHunter 2 (2003), ιστοσελίδα Wikipedia: https://en.wikipedia.org/wiki/SpyHunter_2
[13] SpyHunter Nowhere to Run (2006), ιστοσελίδα Wikipedia:
https://en.wikipedia.org/wiki/SpyHunter:_Nowhere_to_Run
[14] Documentation της Unity, ιστοσελίδα Docs.Unity3D:
https://docs.unity3d.com/Manual/index.html
[15] Unity Answers, ιστοσελίδα Answers.Unity: https://answers.unity.com/index.html
[16] Gimp, ιστοσελίδα Gimp: https://www.gimp.org/
[17] iTunes, ιστοσελίδα: iTunes: https://www.apple.com/gr/itunes/

Παράρτημα Α: Επεξήγηση κώδικα

Σε αυτό το κεφάλαιο θα παρουσιαστεί ο κώδικας των scripts που περιγράφηκαν στο κεφάλαιο 4, η επεξήγηση του οποίου πραγματοποιείται μέσα από σχόλια που διαθέτει το κάθε script. Η παρουσίαση του κώδικα γίνεται με χρήση εικόνων.

PlayerCarController

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

public class PlayerCarController : MonoBehaviour
{
    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΗΝ ΚΙΝΗΣΗ
    public float speed; //ταχύτητα αυτκινήτου
    public float turnSpeed; //ταχύτητα στροφής του αυτοκινήτου

    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ
    public GameObject shot; //εκεί καταχωρείται το αντικείμενο της σφαίρας
    public GameObject frontLeftWheel; //εκεί καταχωρείται το αντικείμενο της μπροστινής αριστερής ρόδας
    public GameObject frontRightWheel; //εκεί καταχωρείται το αντικείμενο της μπροστινής δεξιάς ρόδας
    public GameObject backLeftWheel; //εκεί καταχωρείται το αντικείμενο της πίσω δεξιάς ρόδας
    public GameObject backRightWheel; //εκεί καταχωρείται το αντικείμενο της πίσω δεξιάς ρόδας
    public Transform primaryShotSpawn; //από εκεί εκτοξεύεται η σφαίρα
    public Transform rightSuperShotSpawn; //από εκεί εκτοξεύεται ο δεξιός πύραυλος
    public Transform leftSuperShotSpawn; //από εκεί εκτοξεύεται ο αριστερός πύραυλος
    public float primaryFireRate; //ρυθμός βολών
    public float superFireRate; //ρυθμός εκτόξευσης πυραύλων
    public GameObject missile; //εκεί καταχωρείται το στιγμιότυπο του πυραύλου
    public GameObject chopper; //εκεί καταχωρείται το στιιγμιότυπο του ελικόπτερου για την μεταμόρφωση
    public ParticleSystem effect; //εκεί καταχωρείται το εφέ της μεταμόφωσης
    public AudioSource skid; //εκεί αταχωρείται ο ήχος που κάνουν τα λάστιχα κατά τη διάρκεια του drift
    public GameObject terrain; //εκεί καταχωρείται το μοντέλο του εδάφους
    public GameObject arena; //εκεί καταχωρείται το μοντέλο της αρένας
    public float yMinimumMovement; //εκεί καταχωρείται η ελάχιστη τμή του άξονα των ψ
    public Light leftBackLight; //εκεί καταχωρείται η τιμή του πίσω αριστερά φαναριού
    public Light rightBackLight; //εκεί καταχωρείται η τιμή του πίσω δεξιά φαναριού
    public GameObject reverseCamera; //εκεί καταχωρείται το μοντέλο της αντίστροφης κάμερας που χρησιμοποιείται στις καταδιώξεις
    public GameObject camera; //εκεί καταχωρείται το μοντέλο της κανονικής κάμερας που απενεργοποιείται στις καταδιώξεις


    public StatsAndPickUps statsAndPickUpsScript; //εδώ δημιουργείται ένα στιγμιότυπο του script που περιέχει τα στατιστικά του αυτοκινήτου
                                                  //γιατί όταν οπαίκτης εκτοξεύει τους πυραύλους, μειώνει την τιμή της μπάρας των πυραύλων

    public JetStatsAndPickUps jetStatsAndPickUpsScript; //στιγμιότυπο του script με τα στατιστικά του ελικοπτέρου για την μεταμόρφωση
```

Δείγμα Κώδικα 1: PlayerCarController script

```
private float primaryNextFire; //σε πόσο διάστημα θα εκτοξευτεί η μια βολή μετά την άλλη
private float superNextFire; //σε πόσο διάστημα θα εκτοξευτεί ο ένας πύραυλος μετά τον άλλο
private Rigidbody rb; //εκεί καταχωρείται το rigidbody component του αυτοκινήτου
private float moveHorizontal; //σε αυτήν την τιμή καταχωρείται οάξονας των χ για την οριζόντια κίνηση
private Vector3 eulerAngles; //για την καταχώρηση των αξόνων του αυτοκινήτου

private void Awake ()
{
    rb = GetComponent<Rigidbody>(); //καταχώρηση του rigidbody
    yMinimumMovement = terrain.transform.position.y;
}
```

Δείγμα Κώδικα 2: PlayerCarController script

```
void Update ()
{
        if(transform.rotation.x < -16 || transform.rotation.x > 16)
        {
            transform.Rotate (new Vector3 (0.0f, transform.rotation.y, transform.rotation.z));
        }

        //πρωτεύον όπλο - πυρ
        //αν πατηθεί το αριστερό κλικ, και ο χρόνος που πέρασε από την ποηγούμενη φορά που πατήθηκε είναι μεγαλύτερος από τον χόνο που έχει τεθεί,
        //τότε το όπλο πυροβολεί
        if (Input.GetButton("Fire1") && Time.time > primaryNextFire && PauseMenuController.gameIsPaused == false)
        {
            //θέτω τον ελάχιστο χρόνο που θα πρέπει να μεσολαβεί ανάμεσα στις βολές
            primaryNextFire = Time.time + primaryFireRate;
            //δημιουργείται ένα στιγμιότυπο της βολής
            Instantiate(shot, primaryShotSpawn.position, primaryShotSpawn.rotation);
        }

        //πύραυλοι - πυρ
        //αν πατηθεί το δεξί κλικ, και ο χρόνος που πέρασε από την ποηγούμενη φορά που πατήθηκε είναι μεγαλύτερος από τον χόνο που έχει τεθεί,
        //και υπάρχει το απαραίτητο απόθεμα ενέργειας πυραύλων,τότε θα εκτοξευτούν δύο πύραυλοι
        if (Input.GetButton("Fire2") && Time.time > superNextFire && statsAndPickUpsScript.currentSuper >10 && PauseMenuController.gameIsPaused == false)
        {
            //θέτω τον ελάχιστο χόνο που θα πρέπει να μεσολαβεί ανάμεσα στις βολές
            superNextFire = Time.time + superFireRate;
            //δημιουργούνται δύο στιγμιότυπα των πυραύλων
            Instantiate(missile, rightSuperShotSpawn.position, rightSuperShotSpawn.rotation);
            Instantiate(missile, leftSuperShotSpawn.position, leftSuperShotSpawn.rotation);
            //και μειώνεται η τιμή μπάρας των πυραύλων
            statsAndPickUpsScript.currentSuper -= 10;
        }
```

Δείγμα Κώδικα 3: PlayerCarController script

```
//περιστροφή μπροστινών τροχών όσο στίβει αριστερά
if((Input.GetKey(KeyCode.A)) || (Input.GetKey(KeyCode.LeftShift) && Input.GetKey(KeyCode.D)) )
{
    //έλεγχος για το αν οι μπορστινοι τροχοί ξεπερνούν το όριο περιστροφης κατά την αριστερή στροφή του αυτοκινήτου
    if(frontLeftWheel.transform.eulerAngles.y <= transform.eulerAngles.y + 75.0f
    && frontRightWheel.transform.eulerAngles.y <= transform.eulerAngles.y + 75.0f
    && frontLeftWheel.transform.eulerAngles.y >= transform.eulerAngles.y - 75.0f
    && frontRightWheel.transform.eulerAngles.y >= transform.eulerAngles.y - 75.0f)
    {
        //περιστροφή μπροστινών τροχών σύμφωνα με τον άξονα ψ του αυτοκινήτου ενώ κυλάνε ταυτόχρονα (όσο περνάει ο χρόνος)
        frontLeftWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, frontLeftWheel.transform.eulerAngles.y -15.0f, 0.0f);
        frontRightWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, frontRightWheel.transform.eulerAngles.y -15.0f, 0.0f);
    }
}
//περιστροφή μπροστινών τροχών όσο στίβει δεξιά
else if((Input.GetKey(KeyCode.D)) || (Input.GetKey(KeyCode.LeftShift) && Input.GetKey(KeyCode.A)) )
{
    //έλεγχος για το αν οι μπορστινοι τροχοί ξεπερνούν το όριο περιστροφης κατά την δεξιά στροφή του αυτοκινήτου
    if(frontLeftWheel.transform.eulerAngles.y <= transform.eulerAngles.y + 75.0f
    && frontRightWheel.transform.eulerAngles.y <= transform.eulerAngles.y + 75.0f
    && frontLeftWheel.transform.eulerAngles.y >= transform.eulerAngles.y - 75.0f
    && frontRightWheel.transform.eulerAngles.y >= transform.eulerAngles.y - 75.0f)
    {
        //περιστροφή μπροστινών τροχών σύμφωνα με τον άξονα ψ του αυτοκινήτου ενώ κυλάνε ταυτόχρονα (όσο περνάει ο χρόνος)
        frontLeftWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, frontLeftWheel.transform.eulerAngles.y + 15.0f, 0.0f);
        frontRightWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, frontRightWheel.transform.eulerAngles.y + 15.0f, 0.0f);
    }
}
//επαναφορά μπροστινών τροχών στην αρχική τους θέση (όταν σταματήσει το στρίψιμο)
else
{
    frontLeftWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, transform.eulerAngles.y, 0.0f);
    frontRightWheel.transform.rotation = Quaternion.Euler(1000000.0f * Time.deltaTime, transform.eulerAngles.y, 0.0f);
}
```

Δείγμα Κώδικα 4: PlayerCarController script

```
    //κύλιση τροχών (περιστρέφονται στον άξονα των χ) όσο περνάει ο χρόνος
    frontLeftWheel.transform.Rotate (new Vector3 (1000000.0f, 0.0f, 0.0f) * Time.deltaTime);
    frontRightWheel.transform.Rotate (new Vector3 (1000000.0f, 0.0f, 0.0f) * Time.deltaTime);
    backLeftWheel.transform.Rotate (new Vector3 (1000000.0f, 0.0f, 0.0f) * Time.deltaTime);
    backRightWheel.transform.Rotate (new Vector3 (1000000.0f, 0.0f, 0.0f) * Time.deltaTime);
}

void FixedUpdate()
{
    //καταχώρηση εντολών για την ελάχιστη τιμή κίνησης του αυτοκινήτου στον άξονα των ψ
    rb.position = new Vector3
    (
        rb.position.x,
        Mathf.Clamp (rb.position.y, yMinimumMovement, rb.position.y),
        rb.position.z
    );

    //καταχώρηση αξόνων για οριζόντια κίνηση
    moveHorizontal = Input.GetAxis("Horizontal");

    //δημιουργία άξονα για διατήρηση μηδενικής περιστρφής στον άξονα των Z
    eulerAngles = transform.eulerAngles;

    //κλήση συναρτήσεων που εκτελούν στοφή και κίνηση του αυτοκινήτου
    move();
    turn();
}
```

Δείγμα Κώδικα 5: PlayerCarController script

```
//εκτέλεση συνάρτησης κίνησης του ατοκινήτου
private void move()
{
    //διατήρηση μηδενικής περιστροφής στον άξονα των Z
    eulerAngles.z = eulerAngles.z - eulerAngles.z;
    transform.eulerAngles = eulerAngles;
    //κίνηση του αυτοκινήτου σύμφωνα και με την ταχύτητα που έχει οριστεί
    Vector3 movement = transform.forward * speed * Time.deltaTime;
    //εφαρμογή rigidbody
    rb.MovePosition(rb.position + movement);
    //φεένο. όταν πατιέται το φρένο, η ταχύτητα του αυτοκινήτου μειώνεται και τα πίσω φανάρια ανάβουν πιο έντονα
    if(Input.GetKey(KeyCode.S))
    {
        speed = 7;
        rightBackLight.intensity += 0.5f;
        leftBackLight.intensity += 0.5f;
    }
    //και όταν δεν πατιέται το φρένο, η ταχύτητα και η φωτεινότητα των πίσω φαναριών γυρνάνε στις κανονικές τους τιμές
    else
    {
        speed = 12;
        rightBackLight.intensity = 4;
        leftBackLight.intensity = 4;
    }
}
```

Δείγμα Κώδικα 6: PlayerCarController script

```
    //εκτέλεση συνάρτησης στροφής του αυτοκινήτου
    private void turn()
    {
        //διατήρηση μηδενικής περιστροφής στον άξονα των Z
        eulerAngles.z = eulerAngles.z - eulerAngles.z;
        transform.eulerAngles = eulerAngles;
        //περιστροφή του αυτοκινήτου
        float turn = moveHorizontal * turnSpeed * Time.deltaTime;
```

Δείγμα Κώδικα 7: PlayerCarController script

```
    //μόνο στον άξονα των ψ
    //μηχανισμός για drift
    //(όσο είναι πατημένο το αριστερό shift, το αυτοκίνητο θα γυρνάει περισσότερο, διατηρώντας ισορροπία και ταχύτητα)
    if(Input.GetKey(KeyCode.LeftShift))
    {
        Quaternion turnRotation = Quaternion.Euler(0.0f, turn * 1.3f, 0.0f);
        //και το αυτοκίνητο θα πηγαίνει "με το πλάι" αναλόγως την κατεύθυνση που στρίβει
        if(Input.GetKey(KeyCode.A))
        {
            //παίζει το ηχητικό εφέ με τον ήχο των λάστιχων
            skid.Play();
            Vector3 movement = transform.right * speed * 0.4f * Time.deltaTime;
            //εφαρμογή στο rigidbody
            rb.MovePosition(rb.position + movement);
        }
        else if(Input.GetKey(KeyCode.D))
        {
            //αίζει το ηχητικό εφέ με τον ήχο των λάστιχων
            skid.Play();
            Vector3 movement = transform.right * speed * -0.4f * Time.deltaTime;
            //εφαρμογή στο rigidbody
            rb.MovePosition(rb.position + movement);
        }
        //εφαρμογή στο rigidbody
        rb.MoveRotation(rb.rotation * turnRotation);
    }
    //αλλιώς θα γυρνάει κανονικά
    else
    {
        //σταματάει να παίζει το ηχητικό εφέ με τον ήχο των λάστιχων
        skid.Stop();
        //περιστροφή
        Quaternion turnRotation = Quaternion.Euler(0.0f, turn, 0.0f);
        //εφαρμογή rigidbody
        rb.MoveRotation(rb.rotation * turnRotation);
    }
}
```

Δείγμα Κώδικα 8: PlayerCarController script

```
void OnTriggerEnter(Collider other)
{
    //όταν το αυτοκίνητο μεταμορφώνεται σε ελικόπτερο,
    //το μοντέλο του ελικόπτερου παίρνει την ίδια θέση με το αυτοκίνητο,
    //το αυτοκίνητο απενεργοποιείται και ενεργοποιείται το ελικόπτερο,
    //τα στατιστικά του αυτοκίνητου μεταφέρονται στο ελικόπτερο
    //και παίζει ένα μικρό ειδικό εφέ μαζί με ένα ηχητικό εφέ για την μεταμόρφωση
    if (other.tag == "Transformation")
    {
        jetStatsAndPickUpsScript.currentHealth = statsAndPickUpsScript.currentHealth;
        jetStatsAndPickUpsScript.currentSuper = statsAndPickUpsScript.currentSuper;
        jetStatsAndPickUpsScript.currentShield = statsAndPickUpsScript.currentShield;
        Instantiate(effect, transform.position, transform.rotation);
        chopper.transform.position = transform.position;
        chopper.transform.rotation = other.transform.rotation;
        gameObject.SetActive(false);
        chopper.SetActive(true);
    }
```

Δείγμα Κώδικα 9: PlayerCarController script

```
        //λόγω περίεργων συγκρούσεων,καμιά φορά ο παίκτης πετάγεται έξω από την πίστα
        //όταν ο παίκτης μπει σε μια αρένα δεν μπορεί να πάει κάτω από αυτήν
        if(other.tag == "ArenaEnter")
        {
            yMinimumMovement = arena.transform.position.y;
        }
        //και όταν βγει δεν μπορεί να πάει κάτω από το έδαφος
        if(other.tag == "ArenaExit")
        {
            yMinimumMovement = terrain.transform.position.y;
        }
        //όταν οεράσει μέσα από το Trigger Collider ChaseEnter θα ενεργοποιηθεί η αντίστροφη κάμερα και θα απενεργοποιηθεί η κανονική
        if(other.tag == "ChaseEnter")
        {
            camera.SetActive(false);
            reverseCamera.SetActive(true);
        }
        //και όταν περάσει από το Trigger Collider ChaseExit θα ενεργοποιηθεί η κανονική κάμερα και θα απενεργοποιηθεί η αντίστροφη
        if(other.tag == "ChaseExit")
        {
            camera.SetActive(true);
            reverseCamera.SetActive(false);
        }
    }

}
```

Δείγμα Κώδικα 10: PlayerCarController script

PlayerChopperController

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

public class PlayerChopperController : MonoBehaviour
{
    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΗΝ ΚΙΝΗΣΗ
    public float speed; //ταχύτητα του ελικόπτερου
    public float turnSpeed; //ταχύτητα στροφής του ελικόπτερου

    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ
    public GameObject car; //εκεί καταχωρείται το μοντέλο αυτοκινήτου του παίκτη
    public GameObject shot; //εκεί καταχωρείται το αντικείμενο της σφαίρας
    public Transform shotSpawn; //από εκεί εκτοξεύεται η σφαίρα
    public Transform superShotSpawn1; //από εκεί εκτοξεύεται ενας πύραυλος
    public Transform superShotSpawn2; //από εκεί εκτοξεύεται ο άλλος πύραυλος
    public float fireRate; //ρυθμός βολών
    public GameObject missile; //εκεί καταχωρείται το μοντέλο του πυραύλου
    public float superFireRate; //ρυθμός βολών των πυραύλων
    public GameObject topRotor; //εκεί καταχωρείται το μοντέλο του έλικα
    public ParticleSystem effect; //εκεί καταχωρείται το εφέ για την μεταμόρφωση
    public GameObject warningImage;

    public JetStatsAndPickUps jetStatsAndPickUpsScript; //εδώ δημιουργείται ένα στιγμιότυπο του script που περιέχει τα στατιστικά του ελικόπτερου
                                                        //γιατί όταν οπαίκτης εκτοξεύει τους πυραύλους, μειώνει την τιμή της μπάρας των πυραύλων

    public StatsAndPickUps carStatsAndPickUpsScript; //εδώ δημιουργείται ένα στιγμιότυπο του script που περιέχει τα στατιστικά του αυτοκινήτου

    private Rigidbody rb;
    private float nextFire; //για την επόμενη βολή
    private float moveHorizontal; //κίνηση οριζόντια
    private float moveVertical; //κίνηση κάθετα
    private float superNextFire; //για την επόμενη βολή των πυραύλων
    private Vector3 eulerAngles; //για την καταχώρηση των αξόνων του ελικόπτερου

    private void Awake ()
    {
        rb = GetComponent<Rigidbody>(); //καταχώρηση του rigidbody
    }
```

Δείγμα Κώδικα 11: PlayerChopperController script

```
void Update()
{
    //πρωτεύον όπλο - πυρ
    //αν πατηθεί το αριστερό κλικ, και ο χρόνος που πέρασε από την ποηγούμενη φορά που πατήθηκε είναι μεγαλύτερος από τον χόνο που έχει τεθεί,
    //τότε το όπλο πυροβολεί
    if (Input.GetButton("Fire1") && Time.time > nextFire && PauseMenuController.gameIsPaused == false)
    {
        //θέτω τον ελάχιστο χρόνο που θα πρέπει να μεσολαβεί ανάμεσα στις βολές
        nextFire = Time.time + fireRate;
        //δημιουργείται ένα στιγμιότυπο της βολής
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
    }

    //πύραυλοι - πυρ
    //αν πατηθεί το δεξί κλικ, και ο χρόνος που πέρασε από την ποηγούμενη φορά που πατήθηκε είναι μεγαλύτερος από τον χόνο που έχει τεθεί,
    //και υπάρχει το απαραίτητο απόθεμα ενέργειας πυραύλων,τότε θα εκτοξευτούν δύο πύραυλοι
    if (Input.GetButton("Fire2") && Time.time > superNextFire && jetStatsAndPickUpsScript.currentSuper >10 && PauseMenuController.gameIsPaused == false)
    {
        //θέτω τον ελάχιστο χόνο που θα πρέπει να μεσολαβεί ανάμεσα στις βολές
        superNextFire = Time.time + superFireRate;
        //δημιουργούνται δύο στιγμιότυπα των πυραύλων
        Instantiate(missile, superShotSpawn1.position, superShotSpawn1.rotation);
        Instantiate(missile, superShotSpawn2.position, superShotSpawn2.rotation);
        //και μειώνεται η τιμή μπάρας των πυραύλων
        jetStatsAndPickUpsScript.currentSuper -= 10;
    }
    //περιστροφή του έλικα (όσο πρνάει ο χρόνος)
    topRotor.transform.Rotate (new Vector3 (0.0f, 500.0f, 0.0f) * Time.deltaTime);
}
```

Δείγμα Κώδικα 12: PlayerChopperController script

```
void FixedUpdate()
{
    //καταχώρηση αξόνων για οριζόντια κίνηση
    moveHorizontal = Input.GetAxis("Horizontal");
    //καταχώρηση αξόνων για κάθετη κίνηση
    moveVertical = Input.GetAxis("Vertical");
    //δημιουργία άξονα για διατήρηση μηδενικής περιστρφής στον άξονα των Z
    eulerAngles = transform.eulerAngles;
    //κλήση συναρτήσεων που εκτελούν στοφή και κίνηση του ελικόπτερου
    move();
    turn();
}

private void move()
{
    //διατήρηση μηδενικής περιστροφής στον άξονα των Z
    eulerAngles.z = eulerAngles.z - eulerAngles.z;
    transform.eulerAngles = eulerAngles;
    //κίνηση του ελικόπτερου σύμφωνα και με την ταχύτητα που έχει οριστεί
    Vector3 movement = transform.forward * speed * Time.deltaTime;
    //εφαρμογή sto rigidbody
    rb.MovePosition(rb.position + movement);
}
```

Δείγμα Κώδικα 13: PlayerChopperController script

```
private void turn()
{
    //διατήρηση μηδενικής περιστροφής στον άξονα των Z
    eulerAngles.z = eulerAngles.z - eulerAngles.z;
    transform.eulerAngles = eulerAngles;
    //περιστροφή και κατακόρυφη κίνηση του ελικόπτερου (αν δεν είναι πατημένο το αριστερό shift)
    if(!Input.GetKey(KeyCode.LeftShift))
    {
        float turn = moveHorizontal * turnSpeed * Time.deltaTime;
        //μόνο στον άξονα των Ψ
        //περιστροφή
        Quaternion turnRotation = Quaternion.Euler(0.0f, turn, 0.0f);
        //κατακόρυφη κίνηση του ελικόπτερου
        Vector3 verticalMovement = new Vector3 (0.0f, -moveVertical/3, 0.0f);
        //εφαρμογή rigidbody
        rb.MoveRotation(rb.rotation * turnRotation);
        rb.MovePosition(rb.position + verticalMovement);
    }
    //αλλιώς, αν είναι πατημένο το αριστερό shift, το ελικόπτερο θα μπει σε λειτουργία αποφυγής και θα κινείται οριζόντια
    else if(Input.GetKey(KeyCode.LeftShift) && Input.GetKey(KeyCode.D))
    {
        rb.MovePosition(rb.position + transform.right/2);
    }
    else if(Input.GetKey(KeyCode.LeftShift) && Input.GetKey(KeyCode.A))
    {
        rb.MovePosition(rb.position - transform.right/2);
    }
}
```

Δείγμα Κώδικα 14: PlayerChopperController script

```
    void OnTriggerEnter(Collider other)
    {
        //όταν το ελικόπτερο μεταμορφώνεται σε αυτοκίνητο,
        //το μοντέλο του ελικόπτερου παίρνει την ίδια θέση με το αυτοκίνητο,
        //το ελκόπτερο απενεργοποιείται και ενεργοποιείται το αυτοκίνητο,
        //τα στατιστικά του ελικόπτερου μεταφέρονται στο αυτοκίνητο
        //και παίζει ένα μικρό ειδικό εφέ μαζί με ένα ηχητικό εφέ για την μεταμόρφωση
        if (other.tag == "Transformation2")
        {
            Instantiate(effect, transform.position, transform.rotation);
            carStatsAndPickUpsScript.currentHealth = jetStatsAndPickUpsScript.currentHealth;
            carStatsAndPickUpsScript.currentSuper = jetStatsAndPickUpsScript.currentSuper;
            carStatsAndPickUpsScript.currentShield = jetStatsAndPickUpsScript.currentShield;
            car.transform.position = transform.position;
            car.transform.rotation = transform.rotation;
            gameObject.SetActive(false);
            car.SetActive(true);
        }

        if(other.tag == "WarningArea")
        {
            warningImage.SetActive(true);
        }
    }
    //όταν το ελικόπτερο πλησιάζει πολύ σε οριζόντιο αόρατο τοίχο μεφανίζεται προειδοποίηση στην οθόνη
    void OnTriggerExit(Collider other)
    {
        if(other.tag == "WarningArea")
        {
            warningImage.SetActive(false);
        }
    }
}
```

Δείγμα Κώδικα 15: PlayerChopperController script

StatsAndPickUps και JetStatsAndPickUps

Επειδή τα δύο scripts είναι πανομοιότυπα θα παρουσιαστεί μονάχα ο κώδικας του script StatsAndPickUps με την μοναδική προσθήκη που υπάρχει στο JetStatsAndPickUps να ακολουθεί.

StatsAndPickUps

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

public class StatsAndPickUps : MonoBehaviour
{
    public AudioSource soundEffect; //ηχητικό εφέ που παίζει όταν ο παίκτης συλλέγει ένα αντικείμενο

    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΗΝ ΖΩΗ ΤΟΥ ΠΑΙΚΤΗ
    public Slider healthSlider; //το slider που απεικονίζει την ζωή του παίκτη
    public int currentHealth; //η τιμή ζωής του παίκτη

    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΗΝ ΑΣΠΙΔΑ ΤΟΥ ΠΑΙΚΤΗ
    public Slider shieldSlider; //το slider που απεικονίζει την ασπίδα του παίκτη
    public int currentShield; //η τιμή ασπίδας του παίκτη

    //ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΟΥΣ ΠΥΡΑΥΛΟΥΣ ΤΟΥ ΠΑΙΚΤΗ
    public Slider superSlider; //το slider που απεικονίζει τους πυραύλους του παίκτη
    public int currentSuper; //η τιμή πυραύλων του παίκτη
```

Δείγμα Κώδικα 16: StatsAndPickUps script

```
    void Update()
    {
        //τα sliders παίρνουν τις τιμές των στατιστικών σε κάθε frame
        superSlider.value = currentSuper;
        shieldSlider.value = currentShield;
        healthSlider.value = currentHealth;

        //οι τιμές των στατιστικών δεν μπορούν να πέσουν κάτω από το 0
        if( currentHealth < 0 )
        {
            currentHealth = 0;
        }
        if( currentShield < 0 )
        {
            currentShield = 0;
        }
        if( currentSuper < 0 )
        {
            currentSuper = 0;
        }
    }
```

Δείγμα Κώδικα 17: StatsAndPickUps script

```
    void OnTriggerEnter(Collider other)
    {
        //όταν ο παίκτης συλλέγει το αντικείμενο που του αναπληρώνει την ζωή
        if (other.gameObject.CompareTag("HealthPickUp"))
        {
            //παίζει το ηχητικό εφέ
            soundEffect.Play();
            other.gameObject.SetActive(false); //τότε αυτό το αντικείμενο θα εξαφανίζεται
            currentHealth += 25; //και η ζωή του παίκτη θα ανέβει κατά 25
            //η ζωή του παίκτη δεν μπορεί να είναι πάνω από 100
            if( currentHealth > 100 )
            {
                currentHealth = 100;
            }
            //και αυτό θα φαίνεται στο slider της ζωής
        }
```

Δείγμα Κώδικα 18: StatsAndPickUps script

```
        //όταν ο παίκτης συλλέγει το αντικείμενο που του αναπληρώνει την ασπίδα
        if (other.gameObject.CompareTag("ShieldPickUp"))
        {
            //παίζει το ηχητικό εφέ
            soundEffect.Play();
            other.gameObject.SetActive(false); //τότε αυτό το αντικείμενο θα εξαφανίζεται
            currentShield += 25; //και η ασπίδα του παίκτη θα ανέβει κατά 25
            //η ασπίδα του παίκτη δεν μπορεί να είναι πάνω από 50
            if( currentShield > 50 )
            {
                currentShield = 50;
            }
            //και αυτό θα φαίνεται στο slider της ασπίδας
        }

        //όταν ο παίκτης συλλέγει το αντικείμενο που του αναπληρώνει τους πυραύλους
        if (other.gameObject.CompareTag("SuperPickUp"))
        {
            //παίζει το ηχητικό εφέ
            soundEffect.Play();
            other.gameObject.SetActive(false); //τότε αυτό το αντικείμενο θα εξαφανίζεται
            currentSuper += 25; //και η τιμή των πυραύλων του παίκτη θα ανέβει κατά 25
            //η τιμή των πυραύλων του παίκτη δεν μπορεί να είναι πάνω από 100
            if( currentSuper > 100 )
            {
                currentSuper = 100;
            }
            //και αυτό θα φαίνεται στο slider των πυραύλων
        }

        //αν ο παίκτης πέσει έξω από την πίστα λόγω σφάλματος στις συγκρούσεις, τότε η ζωή του θα μηδενιστεί
        if(other.gameObject.CompareTag("DeathTerrain"))
        {
            currentHealth = 0;
        }
    }
}
```

Δείγμα Κώδικα 19: StatsAndPickUps script

JetStatsAndPickUps

Η επιπλέον προσθήκη αυτού του script εντοπίζεται στο τέλος της συνάρτησης void OnTriggerEnter (Collider other) και περιλαμβάνει το αποτέλεσμα της σύγκρουσης του ελικοπτέρου με κάποιον αόρατο τοίχο

```
        //αν ο παίκτης αγγιξει το έδαφος, τότε η ζωή του θα μηδενιστεί
        if(other.gameObject.CompareTag("DeathTerrain"))
        {
            currentHealth = 0;
        }
        //αν ο παίκτης αγγιξει το κάποιον αόρατο τοίχο, τότε η ζωή του θα μηδενιστεί
        if(other.gameObject.CompareTag("DeathWall"))
        {
            currentHealth = 0;
        }
    }
}
```

Δείγμα Κώδικα 20: StatsAndPickUps script

DamageController και JetDamageController

Επειδή υπάρχει μεγάλη ομοιότητα ανάμεσα και σε αυτά τα δύο scripts, θα παρουσιαστεί ο κώδικας του DamageController και στην συνέχεια θα ακολουθήσουν τα κομμάτια κώδικα του JetDamageController που διαφέρουν από το πρώτο script.

DamageController

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

public class DamageController : MonoBehaviour
{
    public StatsAndPickUps statsAndPickUpsScript; //στιγμιότυπο του script με τα στατιστικά του αυτοκινήτο
    public ParticleSystem hitEffect; //εφέ για τα χτυπήματα που δέχεται ο παίκτης

    void OnTriggerEnter(Collider other)
    {
        //αν ο παίκτης δεχτεί χτύπημα από πυρά εχθρού, θα πέσει πρώτα η τιμή της ασπίδας (αν δεν είναι 0)
        //και μετά η κανονική ζωή του. Επίσης, θα καταστραφεί η βολή και θα παίξει το αντίστοιχο εφέ
        if ( other.tag == "EnemyBolt" )
        {
            if( statsAndPickUpsScript.currentShield != 0 )
            {
                statsAndPickUpsScript.currentShield -= 10;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            else
            {
                statsAndPickUpsScript.currentHealth -= 10;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            Destroy(other.gameObject);
        }
```

Δείγμα Κώδικα 21: DamageController script

```
        //αν ο παίκτης δεχτεί χτύπημα από εμπόδιο εδάφους, θα πέσει πρώτα η τιμή της ασπίδας (αν δεν είναι 0)
        //και μετά η κανονική ζωή του. Επίσης, θα παίξει το αντίστοιχο εφέ
        if ( other.tag == "Obstacle" )
        {
            if( statsAndPickUpsScript.currentShield != 0 )
            {
                statsAndPickUpsScript.currentShield -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            else
            {
                statsAndPickUpsScript.currentHealth -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
        }
        //αν ο παίκτης δεχτεί χτύπημα σύγρουσης από εχθρό εδάφους που κινείται στην ίδια ευθεία συνέχεια, θα πέσει πρώτα η τιμή της ασπίδας (αν δεν είναι 0)
        //και μετά η κανονική ζωή του. Επίσης, θα καταστραφεί ο εχθρός και θα παίξει το αντίστοιχο εφέ
        if ( other.tag == "StraightMovingGroundEnemy" )
        {
            if( statsAndPickUpsScript.currentShield != 0 )
            {
                statsAndPickUpsScript.currentShield -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            else
            {
                statsAndPickUpsScript.currentHealth -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            Destroy(other.gameObject);
        }
        //αν ο παίκτης συγρουστεί με τον "αρχηγό" τότε η ζωή του θα μηδενιστεί
        if( other.tag == "Boss" )
        {
            statsAndPickUpsScript.currentHealth = 0;
        }

    }
}
```

Δείγμα Κώδικα 22: DamageController script

JetDamageController

Οι διαφορές ανάμεσα στα δύο scripts εντοπίζονται τόσο στην αρχή του κώδικα,όπου γίνεται η δήλωση μιας ακόμα μεταβλητής, όσο και στο τέλος, όπου αντί για την συνθήκη σύγκρουσης με εμπόδια και εχθρούς εδάφους, εξετάζεται η σύγκρουση με ιπτάμενα εμπόδια και εχθρούς.

```
public GameObject explosionParticles; //εφέ εκρηξης για την καταστροφή των ιπτάμενων εμποδίων
```

Δείγμα Κώδικα 23: JetDamageController script

```
        //αν ο παίκτης δεχτεί χτύπημα από ιπτάμενο εμπόδιο, θα πέσει πρώτα η τιμή της ασπίδας (αν δεν είναι 0)
        //και μετά η κανονική ζωή του. Επίσης, το εμπόδιο θα καταστραφεί θα παίξει το αντίστοιχο εφέ
        if ( other.tag == "FlyingObstacle" )
        {
            if( jetStatsAndPickUpsScript.currentShield != 0 )
            {
                jetStatsAndPickUpsScript.currentShield -= 20;
            }
            else
            {
                jetStatsAndPickUpsScript.currentHealth -= 20;
            }
            Instantiate(explosionParticles, other.transform.position, other.transform.rotation);
            Destroy(other.gameObject);
        }
        //αν ο παίκτης δεχτεί χτύπημα σύγρουσης από ιπτάμενο εχθρό, θα πέσει πρώτα η τιμή της ασπίδας (αν δεν είναι 0)
        //και μετά η κανονική ζωή του. Επίσης, θα καταστραφεί ο εχθρός και θα παίξει το αντίστοιχο εφέ
        if( other.tag == "FlyingEnemy" )
        {
            if( jetStatsAndPickUpsScript.currentShield != 0 )
            {
                jetStatsAndPickUpsScript.currentShield -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            else
            {
                jetStatsAndPickUpsScript.currentHealth -= 15;
                Instantiate(hitEffect, transform.position, transform.rotation);
            }
            Destroy(other.gameObject);
        }
    }
}
```

Δείγμα Κώδικα 24: JetDamageController script

Enemy και EnemyHeavyTank

Παρουσιάζεται ο κώδικας του script Enemy και έπειτα ένα απόσπασμα από το EnemyHeavyTank με το κομμάτι κώδικα που δεν υπάρχει στο Enemy.

Enemy

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

public class Enemy : MonoBehaviour {

    public Transform player; //εκεί καταχωρείται το transform του παίκτη
    public Transform enemyShotSpawn; //από εκεί θα βγει η βολή
    public GameObject enemyShot; //εκεί καταχωρείται το μοντέλο της βολής
    public float fireRate; //ρυθμός βολών
    public float startingHealth = 15.0f; //εκεί καταχωρείται η αρχικη ζωή του εχθρού
    public float currentHealth; //εκεί καταχωρείται η τρέχουσα ζωή του εχθού
    public ParticleSystem explosionParticles; //εκεί καταχωρείται το εφέ έκρηξης

    private UnityEngine.AI.NavMeshAgent enemyNavigator; //για το navigation
    private float nextFire; //σε ππόσο διάστημαθα εκτοξευτεί η μια βολή μετά την άλλη


    //για να ελέγχω αν ο εχθρός κοιτάζει τον παίκτη
    private Vector3 dirFromEnemyToPlayer;
    private float dotProd;

    void Awake()
    {
        //καταχώρηση του navigator για να ακολουθεί τον παίκτη
        enemyNavigator = GetComponent<UnityEngine.AI.NavMeshAgent>(); //navigation
        //καταχώρηση ζωής του εχθρού
        currentHealth = startingHealth;
    }
```

Δείγμα Κώδικα 25: Enemy script

```
void Update ()
{
    enemyNavigator.SetDestination(player.position); //ο εχθρός ακολουθεί τον παίκτη
    //οι τιμές που δείχνουν αν ο εχθρός κοιτάζει ον παίκτη
    dirFromEnemyToPlayer = (player.position - transform.position).normalized;
    dotProd = Vector3.Dot(dirFromEnemyToPlayer, transform.forward);

    //αν ο εχθρός κοιτάζει τον παίκτη, πυροβολεί
    if(dotProd > 0.99f && Time.time > nextFire)
    {
        //ο ελάχιστος χρόνος που πρέπει να μεσολαβεί ανάμεσα στις βολές
        nextFire = Time.time + fireRate;
        //δημιουργείται ένα στιγμιότυπο της βολής
        Instantiate(enemyShot, enemyShotSpawn.position, enemyShotSpawn.rotation);
    }

    //αν η ζωή του εχθρού έχει μηδενιστεί
    if( currentHealth <= 0.0f )
    {
        //παίζει το εφέ έκρηξης
        Instantiate(explosionParticles, transform.position, transform.rotation);
        //και ο εχθρός καταστρέφεται
        Destroy(gameObject);
    }
    //για να μην πηγαίνει ο εχθρός πιο πάνω ή πιο κάτω από το όχημα του παίκτη
    Vector3 tempPosition = new Vector3();
    tempPosition = transform.position;
    tempPosition.y = player.position.y;

}
```

Δείγμα Κώδικα 26: Enemy script

```
void OnTriggerEnter(Collider other)
{
    //αν δεκτεί χτύπημα από την βολή του παίκτη, θα χάσει 5 ζωή και η σφαίρα θα καταστραφεί
    if ( other.tag == "PrimaryBolt" )
    {
        currentHealth -= 5;
        Instantiate(explosionParticles, transform.position, transform.rotation);
        Destroy(other.gameObject);
    }
    //ενώ αν δεχτεί χτύπημα από πύραυλο η ζωή του θα μηδενιστεί και θα καταστραφεί ο πύραυλος
    if (other.tag == "Missile")
    {
        currentHealth = 0.0f;
        Destroy(other.gameObject);
    }
}

}
```

Δείγμα Κώδικα 27: Enemy script

EnemyHeavyTank

Η διαφορά στα δύο scripts είναι πως το βαρύ εχθρικό τανκ πυροβολεί από δύο κανόνια, σε αντίθεση με το εχθρικό τανκ και τον εχθρικό τροχό που πυροβολούν από ένα. Αυτή η διαφορά φαίνεται στην δήλωση της μεταβλητής για το δεύτερο κανόνι και στην εντολή που πυροδοτεί την δεύτερη βολή.

```
public Transform leftEnemyShotSpawn; //από εκεί θα βγει η αριστερή βολή
public Transform rightEnemyShotSpawn; //από εκεί θα βγει η δεξιά βολή
```

Δείγμα Κώδικα 28: EnemyHeavyTank script

```
    //αν ο εχθρός κοιτάζει τον παίκτη, πυροβολεί
    if(dotProd > 0.99f && Time.time > nextFire)
    {
        //ο ελάχιστος χρόνος που πρέπει να μεσολαβεί ανάμεσα στις βολές
        nextFire = Time.time + fireRate;
        //δημιουργούνται δύο στιγμιότυπα της βολής
        Instantiate(enemyShot, leftEnemyShotSpawn.position, leftEnemyShotSpawn.rotation);
        Instantiate(enemyShot, rightEnemyShotSpawn.position, rightEnemyShotSpawn.rotation);
    }
```

Δείγμα Κώδικα 29: EnemyHeavyTank script

StraightMovingEnemy

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

public class StraightMovingEnemy : MonoBehaviour
{
    public float speed; //ταχύτητα του εχθρού
    public float fireRate; //ρυθμός βολών
    public float startingHealth = 15.0f; //εκεί καταχωρείται η αρχικη ζωή του εχθρού
    public float currentHealth; //εκεί καταχωρείται η τρέχουσα ζωή του εχθού
    public ParticleSystem explosionParticles; //εκεί καταχωρείται το εφέ έκρηξης
    public Transform enemyShotSpawn; //από εκεί θα βγει η βολή
    public GameObject enemyShot; //εκεί καταχωρείται το μοντέλο της βολής
    public float lifetime = 10.0f; //εκεί καταχωρείται ο χρόνος ζωής του εχθρού (τα 10" είναι ενδεικτικά)

    private float nextFire; //σε πόσο διάστημα θα εκτοξευτεί η μια βολή μετά την άλλη
```

Δείγμα Κώδικα 30: StraightMovingEnemy script

```
void Awake()
{
    currentHealth = startingHealth; //καταχώρηση ζωής του εχθρού
}

void Update ()
{

    GetComponent<Rigidbody>().velocity = transform.forward * -speed; //κίνηση του εχθρού
    Destroy(gameObject, lifetime); //ο εχθρός καταστρέφεται όταν περάσει ο χρόνος ζωής του

    if(Time.time > nextFire)
    {
        //ο ελάχιστος χρόνος που πρέπει να μεσολαβεί ανάμεσα στις βολές
        nextFire = Time.time + fireRate;
        //δημιουργείται ένα στιγμιότυπο της βολής
        Instantiate(enemyShot, enemyShotSpawn.position, enemyShotSpawn.rotation);
    }
    //αν η ζωή του εχθρού έχει μηδενιστεί
    if( currentHealth <= 0.0f )
    {
        //παίζει το εφέ έκρηξης
        Instantiate(explosionParticles, transform.position, transform.rotation);
        //και ο εχθρός καταστρέφεται
        Destroy(gameObject);
    }
}
```

Δείγμα Κώδικα 31: StraightMovingEnemy script

```
    void OnTriggerEnter(Collider other)
    {
        //αν δεκτεί χτύπημα από την βολή του παίκτη, θα χάσει 5 ζωή και η σφαίρα θα καταστραφεί
        if ( other.tag == "PrimaryBolt" )
        {
            currentHealth -= 5;
            Instantiate(explosionParticles, transform.position, transform.rotation);
            Destroy(other.gameObject);
        }
        //ενώ αν δεχτεί χτύπημα από πύραυλο η ζωή του θα μηδενιστεί και θα καταστραφεί ο πύραυλος
        if (other.tag == "Missile")
        {
            currentHealth = 0.0f;
            Destroy(other.gameObject);
        }
    }
}
```

Δείγμα Κώδικα 32: StraightMovingEnemy script

GroundBattleActivator

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

public class GroundEnemyActivator : MonoBehaviour
{
    public GameObject enemies; //εκεί καταχωρείται το αντικείμενο που περιέχει τους εχθρούς
    public GameObject startWall; //εκεί καταχχωρείται ο τοίχος στην αρχή της αρένας
    public GameObject endWall; //εκεί καταχχωρείται ο τοίχος κώνων στο τέλος της αρένας

    void Update()
    {
        //αν έχουν καταστραφεί οι εχθροί
        //(επειδή όλοι είναι καταχωρημένοι σαν "παιδιά" ενός αντικειμένου, ελέγχεται αν το αντικείμενο αυτό έχει "παιδιά")
        if(enemies.transform.childCount == 0)
        {
            //ο τοίχος κώνων απενεργοποιείται και ο παίκτης μπορεί να συνεχίσει στην πίστα
            endWall.SetActive(false);
        }
    }

    void OnTriggerEnter( Collider other )
    {
        //όταν ο παίκτης περάσει από αυτό το Trigger Collider θα ενεργοποιηθούν οι εχθροί και ο τοίχος στην αρχή της αρένας
        if (other.tag == "CarActivator")
        {
            enemies.SetActive(true);
            startWall.SetActive(true);
        }
    }


}
```

Δείγμα Κώδικα 33: GroundBattleActivator script

FinalBattleActivator

Αυτό το script παρουσιάζει κάποιες ομοιότητες με το προηγούμενο. Πιο συγκεκριμένα, στις εντολές που αναλαμβάνουν την ενεργοποίηση των εχθρών, του τοίχου στην αρχή της αρένας και στον έλεγχο για την εξουδετέρωση των εχθρών. Όμως, ταυτόχρονα, διαχειρίζεται τον επιτυχή τερματισμό της πρώτης πίστας.

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

public class FinalBattleActivator : MonoBehaviour
{
    public GameObject enemies; //εκεί καταχωρείται το αντικείμενο που περιέχει τους εχθρούς
    public GameObject startWall; //εκεί καταχχωρείται ο τοίχος στην αρχή της αρένας
    public PlayerCarController playerCarControllerScript; //ένα στιγμιότυπο του script με τα στατιστικά του αυτοκινήτου
    public AudioSource carSound; //ο ήχος του αυτοινήτου
    public AudioSource backgroundMusic; //η μουσική της πίστας
    public GameObject winImage; //η εικόνα και η μουσική για τη νίκη
```

Δείγμα Κώδικα 34: FinalBattleActivator script

```
    void Update ()
    {
        //αν έχουν καταστραφεί οι εχθροί
        //(επειδή όλοι είναι καταχωρημένοι σαν "παιδιά" ενός αντικειμένου, ελέγχεται αν το αντικείμενο αυτό έχει "παιδιά")
        if(enemies.transform.childCount == 0)
        {
            //το αυτοκίνητο θα σταματήσει να κινείται
            playerCarControllerScript.speed = 0;
            //όπως και ο ήχος του θα σταματήσει
            carSound.Stop();
            //η μουσική της πίστας θα χαμηλώνει όσο περνάει ο χρόνος
            backgroundMusic.volume -= backgroundMusic.volume * Time.deltaTime;
            //και ξεκινάει ένα coroutine για την εμφάνιση της εικόνας και την μουσική της νίκης
            StartCoroutine(Waiting());
        }
    }

    IEnumerator Waiting()
    {
        //μόλις περάσουν 2 δευτερόλεπτα
        yield return new WaitForSecondsRealtime(2);
        //θα ενεργοποιηθεί η εικόνα και η μουσική της νίκης
        winImage.SetActive(true);
        //και θα σταματήσει τελείως η μουσική της πίστας
        backgroundMusic.Stop();
    }

    void OnTriggerEnter( Collider other )
    {
        //όταν ο παίκτης περάσει από αυτό το Trigger Collider θα ενεργοποιηθούν οι εχθροί και ο τοίχος στην αρχή της αρένας
        if (other.tag == "CarActivator")
        {
            enemies.SetActive(true);
            startWall.SetActive(true);
        }
    }
}
```

Δείγμα Κώδικα 35: FinalBattleActivator script

BossAppearance

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

public class BossAppearance : MonoBehaviour
{
    public GameObject boss; //εκεί καταχωρείται το μοντέλο του ¨"αρχηγού"

    void OnTriggerEnter( Collider other )
    {
        //αν το Trigger Collider του αυτοινήτου ή του ελικοπτέρου
        //περάσει από το αόρατο Trigger Collide στο οποίο έχει ανατεθεί αυτό το script
        if(other.tag == "CarActivator" || other.tag == "ChopperActivator" )
        {
            //τότε ενεργοποιείται το μοντέλο του "αρχηγού"
            boss.SetActive(true);
        }
    }
}
```

Δείγμα Κώδικα 36: BossAppearance script

BossChase

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

public class BossChase : MonoBehaviour
{
    public GameObject enemyShot; //εκεί καταχωρείται το μοντέλο της βολής
    public Transform shotSpawn1; //από εκεί θα βγει η μία βολή
    public Transform shotSpawn2; //από εκεί θα βγει η άλλη βολή
    public float speed; //ταχύτητα του "αρχηγού"
    public float fireRate; //ρυθμός βοώλν
    public float lifetime; //εκεί καταχωρείται ο χρόνος ζωής του μοντέλου του "αρχηγού"

    private float nextFire; //σε ππόσο διάστημα θα εκτοξευτεί η μια βολή μετά την άλλη

    void Update ()
    {
        GetComponent<Rigidbody>().velocity = transform.forward * speed; //κίνηση του "αρχηγού"
        //το μοντέλο του "αρχηγού" καταστρέφεται όταν περάσει ο χρόνος ζωής του
        Destroy(gameObject, lifetime);

        if(Time.time > nextFire)
        {
            //ο ελάχιστος χρόνος που πρέπει να μεσολαβεί ανάμεσα στις βολές
            nextFire = Time.time + fireRate;
            //δημιουργούνται δύο στιγμιότυπα της βολής
            Instantiate(enemyShot, shotSpawn1.position, shotSpawn1.rotation);
            Instantiate(enemyShot, shotSpawn2.position, shotSpawn2.rotation);
        }
    }
}
```

Δείγμα Κώδικα 37: BossChase script

Boss

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

public class Boss : MonoBehaviour
{
    public float speed; //ταχύτητα του "αρχηγού"
    public ParticleSystem explosion; //εκεί καταχωρείται το εφέ έκρηξης
    public Slider bossSlider; //εκεί καταχωρείται το αντικείμενο slider που αντιπροσωπεύει την ζωή του "αρχηγού"

    void Update ()
    {
        GetComponent<Rigidbody>().velocity = transform.forward * speed; //κίνηση του "αρχηγού"
    }
```

Δείγμα Κώδικα 38: Boss script

```
     void OnTriggerEnter( Collider other )
     {
         //όταν ο "αρχηγός έρθει σε επαφή με τα Trigger Colliders που διαθέτουν οι πύραυλοι των "φιλικών" κανονιών
         //θα καταστραφούν τόσο οι πύραλοι, όσο και ο "αρχηγός"
         //θα παίξει το εφέ της έκρηξης
         //και η ζωή τουα ρχηγού θα μηδενιστεί
         if(other.tag == "Missile" )
         {
             Destroy(other.gameObject);
             Instantiate(explosion, transform.position, transform.rotation);
             Destroy(gameObject);
             bossSlider.value = 0;
         }
     }
 }
```

Δείγμα Κώδικα 39: Boss script

GameController

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

public class GameController : MonoBehaviour
{
    public StatsAndPickUps statsAndPickUpsScript; //στιγμιότυπο του script με τα στατιστιά του αυτοινήτου
    public JetStatsAndPickUps jetStatsAndPickUpsScript; //στιγμιότυπο του script με τα στατιστικά του ελικοπτέρου
    public GameObject carCamera; //εκεί καταχωρείται η κάμερα του αυτοινήτου
    public GameObject jetCamera; //εκεί καταχωρείται η κάμερα του ελικοπτέρου
    public GameObject deathCamera; //εκεί καταχωρείται η κάμερα για τον "θάνατο" του παίκτη
    public GameObject playerCar; //εκεί καταχωρείται το μοντέλο του αυτοκινήτου
    public GameObject playerJet; //εκεί καταχωρείται το μοντέλο του ελικοπτέρου
    public GameObject loseImage; //εκεί καταχωρείται η εικόνα και  μουσική της ήττας
    public GameObject startImage; //εκεί καταχωρείται η εικόνα για την αρχή της πίστας
    public GameObject gameHUD; //εκεί καταχωρείται το HUD του παιχνιδιού με την μπάρα ζωής, ασπίδας και πυραύλων
    public AudioSource backgroundMusic; //εκεί καταχωρείται η μουσική της πίστας
    public ParticleSystem explosion; //εκεί καταχωρείται το εφέ της έκρηξης

    void Start()
    {
        //ξεκινάει ένα coroutine για να περάσουν 2 δευτερόλεπτα,
        //να απενεργοποιηθεί η εικόνα αρχής και να ξεκινήσει η πίστα με την ενεργοποίηση του αυτοκινήτου
        StartCoroutine(StartWaiting());
    }
```

Δείγμα Κώδικα 40: GameController script

```
    void Update ()
    {
        //αν μηδενιστεί η ζωή του αυτοκινήτου
        if( playerCar.active && statsAndPickUpsScript.currentHealth == 0 )
        {
            //η μουσική της πίστας χαμηλώνει όσο περνάει ο χρόνος
            backgroundMusic.volume -= backgroundMusic.volume * Time.deltaTime;
            //η κάμερα θανάτου παίρνει την θέση της κάμερας του αυτοκινήτου
            deathCamera.transform.position = carCamera.transform.position;
            deathCamera.transform.rotation = carCamera.transform.rotation;
            //η κάμερα θανάτου ενεργοποιείται
            deathCamera.SetActive(true);
            //παίζει το εφέ έκρηξης
            Instantiate(explosion, playerCar.transform.position, playerCar.transform.rotation);
            //το αυτοκίνητο καταστρέφεται
            Destroy(playerCar);
            //και ξεινάει το coroutine για την ήττα
            StartCoroutine(Waiting());
        }
```

Δείγμα Κώδικα 41: GameController script

```
        //αν μηδενιστεί η ζωή του ελικοπτέρου
        if( playerJet.active && jetStatsAndPickUpsScript.currentHealth == 0 )
        {
            //η μουσική της πίστας χαμηλώνει όσο περνάει ο χρόνος
            backgroundMusic.volume -= backgroundMusic.volume * Time.deltaTime;
            //η κάμερα θανάτου παίρνει την θέση της κάμερας του ελικοπτέρου
            deathCamera.transform.position = jetCamera.transform.position;
            deathCamera.transform.rotation = jetCamera.transform.rotation;
            //η κάμερα θανάτου ενεργοποιείται
            deathCamera.SetActive(true);
            //παίζει το εφέ έκρηξης
            Instantiate(explosion, playerJet.transform.position, playerJet.transform.rotation);
            //το ελικόπτερο καταστρέφεται
            Destroy(playerJet);
            //και ξεινάει το coroutine για την ήττα
            StartCoroutine(Waiting());
        }
    }
```

Δείγμα Κώδικα 42: GameController script

```
    IEnumerator Waiting()
    {
        //αφού παίκτης "πεθάνει", θα περάσουν 2 δευτερόλεπτο
        yield return new WaitForSecondsRealtime(2);
        //η μουσική της πίστας θα σταματήσει
        backgroundMusic.Stop();
        //και θα εμφανιστεί η εικόνα και η μουσική της ήττας
        loseImage.SetActive(true);
    }

    IEnumerator StartWaiting()
    {
        //η αρχική εικόνα είναι ενεργή για 2 δευτερόλεπτα, και μετα η πίστα ξεκινά κανονικά,
        //αυτό βοηθά στο να μην ξεκινήσει η πίστα αμέσως και μπερδευτεί ο παίκτης
        yield return new WaitForSecondsRealtime(2);
        //το HUD της πίστας ενεργοποείται
        gameHUD.SetActive(true);
        //η κάμερα θανάτου απενεργοποιείται
        //(ήταν ενεργοποιημένη στην αρχή για να δειχνει την εικόνα αρχής)
        deathCamera.SetActive(false);
        //η εικόνα αρχής απενεργοποιείται
        startImage.SetActive(false);
        //και ενεργοποιείται το αυτοκίνητο του παίκτη
        playerCar.SetActive(true);
    }
}
```

Δείγμα Κώδικα 43: GameController script

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

public class OptionsMenuController : MonoBehaviour
{
    public AudioMixer audioMixer; //εκεί καταχωρείται το audiomixer για την ρύθμιση της έντασης του ήχου
    Resolution[] resolutions; //ο πίνακας που κρατά τις αναλύσεις του υπολογιστή
    public Dropdown resolutionDropdown; //η dropdown λίστα στην οποία θα παρουσιάζονται οι αναλύσεις
```

Δείγμα Κώδικα 44: OptionsMenuController script

```
//το script διαβάζει τις διαθέσιμες αναλύσεις του υπολογιστή και τις αποθηκεύει στον πίνακα
//τις μετατρέπει σε string (συμβλοσειρές) για να παρουσιαστούν στο παιχνίδι
//και τις ενσωματώνει στην dropdown λίστα του μενού
void Start()
{
    resolutions = Screen.resolutions;
    resolutionDropdown.ClearOptions();
    List<string> options = new List<string>();

    int currentResolutionIndex = 0;

    for(int i = 0; i < resolutions.Length; i++)
    {
        string option = resolutions[i].width + "x" + resolutions[i].height;
        options.Add(option);

        if(resolutions[i].width == Screen.currentResolution.width && resolutions[i].height == Screen.currentResolution.height)
        {
            currentResolutionIndex = i;
        }
    }

    resolutionDropdown.AddOptions(options);
    resolutionDropdown.value = currentResolutionIndex;
    resolutionDropdown.RefreshShownValue();
}
```

Δείγμα Κώδικα 45: OptionsMenuController script

```
    //ρύθμιση για την ανάλυση οθόνης
    public void SetResolution(int resolutionIndex)
    {
        Resolution resolution = resolutions[resolutionIndex];
        Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);
    }
    //ρύθμιση έντασης ήχου
    public void SetVolume(float volume)
    {
        audioMixer.SetFloat("volume", volume);
    }
    //ρύθμιση ποιότητας γραφικών
    public void SetQuality(int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }
    //ρύθμιση για τις λειτουργίες πλήρης οθόνης και παραθύρου
    public void SetFullScreen(bool isFullScreen)
    {
        Screen.fullScreen = isFullScreen;
    }
}
```

Δείγμα Κώδικα 46: OptionsMenuController script