ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής- Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Τρισδιάστατου Παιχνιδιού με Υπερήρωα.** **Development of a 3D video game with a Super Hero.**
Ονοματεπώνυμο Φοιτητή	**Χατζηευλογημένος Γεώργιος**
Πατρώνυμο	**Ευστράτιος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 21060**
Επιβλέπων	**Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Δεκεμβριος 2023**

Τριμελής Εξεταστική Επιτροπή

Παναγιωτόπουλος Θεμιστοκλής Καθηγητής	Πικράκης Άγγελος Επίκουρος Καθηγητής	Τασούλας Ιωάννης Επίκουρος Καθηγητής

Ανάλυση παιχνιδιού:

Όνομα: Miami City Crime

Είδος: Προσομοίωση Open World

Σκοπός: Παίξτε ως υπερήρωας και σώστε τους ανθρώπους στην πόλη από εγκληματίες.

Game Engine: Unity

Platform: PC, Android & IOS

Περιεχόμενα

ΣΥΝΟΨΗ

Στην παρούσα διπλωματική εργασία δημιουργήθηκε μια τρισδιάστατη προσομοίωση, που μου παρέχει την ευκαιρία να εξοικειωθώ με το περιβάλλον το οποίο θα συναντήσω καθ' όλη τη διάρκεια των ακαδημαϊκών μου σπουδών. Κατά την ανάπτυξή της, επιλέχθηκε η Unity 3D ως game engine για τις ευκολίες που προσφέρει στην εκμάθηση, το απλοποιημένο σύστημα εισαγωγής υλικού και πακέτων, τα εντυπωσιακά τεχνικά γραφικά, τον προηγμένο φωτισμό και σύστημα particles, καθώς και άλλες λειτουργίες που είναι προσβάσιμες ακόμη και με βασική εκμάθηση.

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση της παρούσας πτυχιακής εργασίας απαιτούσε αρκετό χρόνο, όχι λόγω δυσκολιών στην υλοποίηση, αλλά εξαιτίας των αναρίθμητων προσπαθειών για τη βελτίωση κάθε γραμμής του κώδικα, επιδιώκοντας την άριστη απόδοσή του. Επιπρόσθετα, οι πολλαπλές δοκιμές διαφόρων μεθόδων για την επίτευξη συγκεκριμένων λειτουργιών ενίσχυσαν την ευελιξία και εμπειρία μας στη χρήση του Unity και της γλώσσας προγραμματισμού C#.

Θέλω να εκφράσω την ευγνωμοσύνη μου προς εκείνους που διαθέτουν σχετικά άρθρα και βίντεο στο διαδίκτυο, παρέχοντας σαφήνεια και καθοδήγηση, οι οποίες συνέβαλαν στην επίτευξη του στόχου μου. Ειδική ευχαριστία αξίζει και στον καθηγητή μου Θεμιστοκλής Παναγιωτόπουλο για τις σαφής οδηγίες του.

1.ΕΙΣΑΓΩΓΗ

Η ανάπτυξη βιντεοπαιχνιδιών αναφέρεται στη διαδικασία δημιουργίας ενός ηλεκτρονικού παιχνιδιού, είτε για υπολογιστές, κονσόλες, ή φορητές συσκευές. Αυτή η διαδικασία μπορεί να διεξάγεται από ατομικούς προγραμματιστές, μικρές ομάδες ή μεγάλες εταιρείες. Συνήθως, τα βιντεοπαιχνίδια χρηματοδοτούνται από εκδότες και απαιτούν αρκετά χρόνια για την ολοκλήρωσή τους. Ωστόσο, υπάρχουν και τα λεγόμενα Indie Games, δηλαδή παιχνίδια που αναπτύσσονται από ανεξάρτητες ομάδες ή μεμονωμένα άτομα, τα οποία συχνά απαιτούν λιγότερο χρόνο και χρηματοδότηση.

Οι πρώτες μηχανές για βιντεοπαιχνίδια εμφανίστηκαν στις αρχές της δεκαετίας του '90, όταν ξεκίνησαν να αναπτύσσονται τα 3D γραφικά, δίνοντας ώθηση στη βιομηχανία. Τον μέσο της δεκαετίας, ο όρος "game engine" εδραιώθηκε, με τα παιχνίδια "Doom" και "Quake" να αποτελούν πρωτοπόρα στον τομέα της ανάπτυξης παιχνιδιών. Ακολούθησαν πολλοί developers που άρχισαν να αγοράζουν βασικά στοιχεία των παιχνιδιών και να προσθέτουν δικά τους στοιχεία, όπως όπλα και γραφικά, αναπτύσσοντας και διαθέτοντας τα ως ξεχωριστά βιντεοπαιχνίδια.

Ο σχεδιασμός παιχνιδιών αποτελεί μια δημιουργική διαδικασία που συνδυάζει σχεδιασμό και αισθητική, προκειμένου να δημιουργηθεί ένα παιχνίδι για ψυχαγωγικούς, εκπαιδευτικούς, γυμναστικούς ή πειραματικούς σκοπούς. Ολοένα και περισσότερο, οι αρχές σχεδιασμού παιχνιδιών ενσωματώνονται σε άλλες μορφές αλληλεπίδρασης μέσω της gamification. Αν και η Unity 3D είναι μια μηχανή ανάπτυξης παιχνιδιών, η εν λόγω διπλωματική εστιάζει στον σχεδιασμό επιπέδων, χρησιμοποιώντας την προγραμματιστική γλώσσα C#.

Με την αυξημένη επεξεργαστική και γραφική ικανότητα των σύγχρονων κονσολών και υπολογιστικών συστημάτων, σε συνδυασμό με τις αυξημένες προσδοκίες των χρηστών, ο σχεδιασμός επιπέδων στα βιντεοπαιχνίδια έχει εξελιχθεί πέρα από την απλή δημιουργία ενός προϊόντος από έναν μεμονωμένο προγραμματιστή, οδηγώντας στην ανάγκη για ανάπτυξη από ομάδες.

Η παρακάτω διπλωματική είναι ένα Παιχνίδι προσομοίωσης υπερήρωα σε ανοιχτό κόσμο, όπου είσαι ένας υπερήρωας και πρέπει να παρακολουθείς την πόλη σου για να συλλάβεις εγκληματίες και να σώσεις αθώους ανθρώπους. Αρχίζεις εντοπίζοντας το σημείο όπου διαπράττεται ή έχει διαπραχθεί ένα έγκλημα και πετάς εκεί για να συλλάβεις τον εγκληματία, να πολεμήσεις μαζί του και να τον αντιμετωπίσεις. Έτσι, μπορείς να σώσεις τους αθώους ανθρώπους.

2.ΠΛΑΤΦΟΡΜΕΣ ΑΝΑΠΤΥΞΗΣ ΒΙΝΤΕΟΠΑΙΧΝΙΔΙΩΝ

Οι πλατφόρμες για την ανάπτυξη βιντεοπαιχνιδιών, γνωστές επίσης ως εργαλεία αρχιτεκτονικής παιχνιδιού ή frameworks παιχνιδιών, αποτελούν συστήματα ανάπτυξης λογισμικού που διευκολύνουν τους δημιουργούς στην παραγωγή βιντεοπαιχνιδιών. Αυτές οι πλατφόρμες είναι χρήσιμες για τη δημιουργία παιχνιδιών που προορίζονται για κονσόλες, φορητές συσκευές και υπολογιστές.

Τα βασικά στοιχεία που παρέχονται συνήθως από μια τέτοια πλατφόρμα περιλαμβάνουν μηχανές γραφικών για 2D ή 3D παρουσίαση, συστήματα φυσικής ή ανίχνευσης και απόκρισης σε συγκρούσεις, ηχητικά εφέ, ανίμεση, τεχνητή νοημοσύνη, δικτυακή λειτουργικότητα, ροή εργασιών, διαχείριση μνήμης, διάλογος, εντοπισμός θέσης, σκηνοθεσία και μπορεί να προσφέρουν και υποστήριξη για κινηματογραφικά βίντεο.

Οι δημιουργοί συχνά εξοικονομούν χρόνο κατά τη διαδικασία ανάπτυξης επαναχρησιμοποιώντας ή προσαρμόζοντας τις ίδιες πλατφόρμες για τη δημιουργία διάφορων παιχνιδιών ή για να διευκολύνουν τη μεταφορά παιχνιδιών σε διάφορες πλατφόρμες. Η τεχνολογία στον τομέα των βιντεοπαιχνιδιών έχει εξελιχθεί σημαντικά, με αποτέλεσμα να υπάρχουν σήμερα περισσότερες από 400 τέτοιες πλατφόρμες.

Ωστόσο, όχι όλες προσφέρουν τις ίδιες δυνατότητες ή δικαιώματα. Κάθε πλατφόρμα έχει τα δικά της πλεονεκτήματα και μειονεκτήματα.

Οι πιο δημοφιλείς πλατφόρμες στις ημέρες μας περιλαμβάνουν:
• Unity3D
• UDK - Unreal Engine
• Godot
• Gamemaker

3.ΤΑ ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΗΣ UNITY;

Κάθε πλατφόρμα ανάπτυξης παιχνιδιών προσφέρει τα δικά της προτερήματα και προκλήσεις, με ειδική προσοχή στην κοινότητα και την υποστήριξη προς τους προγραμματιστές. Επιλέχθηκε η Unity 3D, λόγω της διαθεσιμότητας μιας δωρεάν εκδόσης, της ύπαρξης ενός ενεργού φόρουμ υποστήριξης στο forum.unity3d.com, όπου είναι δυνατή η εύρεση λύσεων σε πιθανές δυσκολίες. Προστίθεται επίσης η πληθώρα οδηγών στο YouTube και η πρόσβαση σε ένα asset store που προσφέρει δωρεάν γραφικά, μοντέλα και κώδικα προγραμματισμού.

4.ΕΞΕΛΙΞΗ ΤΗΣ UNITY GAME ENGINE

Η ανάπτυξη της Unity Game Engine ξεκίνησε από τρία φιλόδοξα μυαλά, τους David Helgason, Joachim Ante και Nicholas Francis, οι οποίοι, με περιορισμένα οικονομικά μέσα, συνεργάστηκαν σε ένα υπόγειο γραφείο για να δημιουργήσουν ένα λογισμικό που θα είχε τεράστια επίδραση στον χώρο των βιντεοπαιχνιδιών. Αυτή η τριάδα συνέθεσε την Over the Edge Entertainment (OTEE), που αργότερα μετονομάστηκε σε Unity Technologies, και άρχισε να προσελκύει ταλαντούχους μηχανικούς λογισμικού. Κατά την περίοδο των πρώιμων δοκιμών της μηχανής, η OTEE αξιοποίησε την Unity για την ανάπτυξη του βιντεοπαιχνιδιού GooBall το 2005, αποδεικνύοντας τις ικανότητές της (Nicoll, B., Keogh, B. & Springerlink, 2019).

Η παρουσίαση της Unity στο κοινό έγινε επίσημα κατά τη διάρκεια της Παγκόσμιας Συνδιάσκεψης Προγραμματιστών της Apple. Από τότε, η Unity έχει προκαλέσει δραστικές μεταβολές στον κλάδο των βιντεοπαιχνιδιών και έχει γίνει διαθέσιμη σε πληθώρα πλατφορμών όπως Windows (PC), Mac, Universal Windows Platform (UWP), Linux Standalone, iOS, Android, ARKit, ARCore, Microsoft HoloLens, Windows Mixed Reality, Magic Leap (Lumin), Oculus, PlayStation VR, PS5, PS4, Xbox One, Xbox X|S, Nintendo Switch, Google Stadia, και Linux QNX (Unity, 2022).

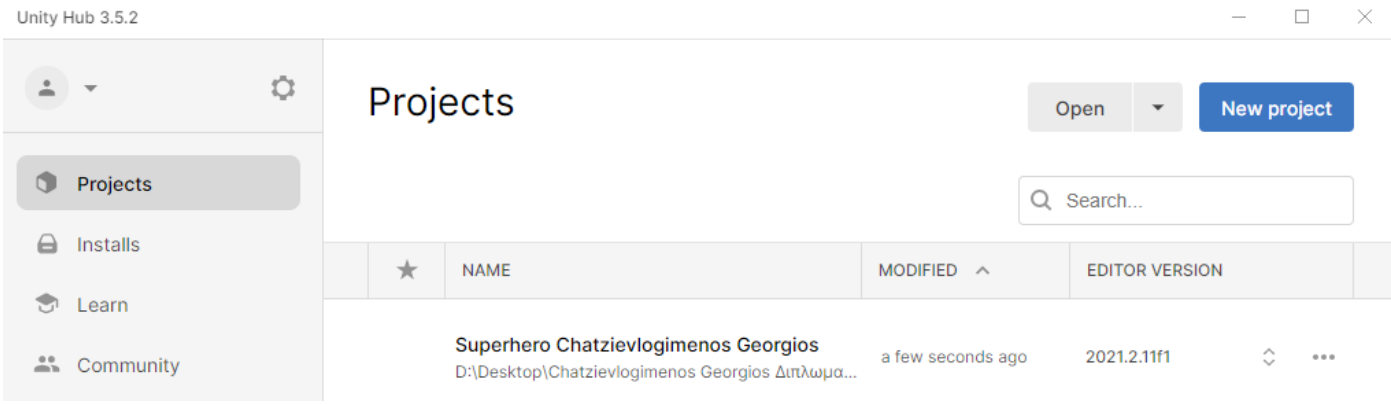
https://unity.com/solutions/multiplatform

4.1 UNITY HUB

Το Unity Hub είναι μια ανεξάρτητη εφαρμογή που ευχερεί τη διαδικασία εύρεσης, λήψης και διαχείρισης των έργων Unity και των σχετικών εγκαταστάσεων. Μέσω του Hub, μπορείτε επίσης να συμπεριλάβετε μη αυτοματοποιημένες εκδόσεις του Unity Editor που έχετε ήδη εγκαταστήσει στον υπολογιστή σας.

Χρησιμοποιήστε το Unity Hub για να:
• Διαχειριστείτε τις άδειες χρήσης του λογαριασμού Unity και του Unity Editor.
• Αναπτύξετε το έργο σας, συνδέοντας μια προεπιλεγμένη έκδοση του Unity Editor με το έργο σας, και διαχειριστείτε πολλαπλές εγκαταστάσεις του Editor.
• Εκκινήσετε διάφορες εκδόσεις του Unity απευθείας από την επισκόπηση του έργου (Project View).
• Διαχειριστείτε και επιλέξτε στόχους κατασκευής για τα έργα σας χωρίς να εκκινήσετε τον Editor.
• Τρέξτε παράλληλα δύο διαφορετικές εκδόσεις του Unity, προσέχοντας να μην ανοίγετε ένα έργο

παράλληλα σε δύο διαφορετικές παρουσίες του Editor για να αποφύγετε τυχόν συγκρούσεις.
• Εγκαταστήστε επιπρόσθετα στοιχεία σε υφιστάμενες εγκαταστάσεις του Editor. Όταν κατεβάζετε μια έκδοση του Editor μέσω του Unity Hub, μπορείτε να βρείτε και να προσθέσετε επιπλέον στοιχεία (όπως εξειδικευμένη υποστήριξη πλατφόρμας, Visual Studio, offline έγγραφα και βασικά συστατικά) είτε κατά τη διάρκεια της αρχικής εγκατάστασης είτε αργότερα.
• Χρησιμοποιήστε προκαθορισμένα πρότυπα έργων για να ξεκινήσετε τη διαδικασία ανάπτυξης κοινών τύπων έργων. Τα πρότυπα έργων της Unity προσφέρουν προεπιλεγμένες ρυθμίσεις για κοινά σενάρια ανάπτυξης όταν δημιουργείτε ένα νέο έργο.



4.2 UNITY EDITOR

Ο Unity Editor διαθέτει πολλαπλά πάνελ, τα οποία διευκολύνουν την διαχείριση του έργου. Ο χρήστης μπορεί να προσαρμόσει το περιβάλλον εργασίας του, καθώς επιτρέπεται η μετακίνηση και επιλογή των πάνελ που επιθυμεί να εμφανίζονται. Ανάμεσα στα πιο σημαντικά πάνελ, τα οποία θα εξετάσουμε περαιτέρω, περιλαμβάνονται:
• Προβολή Σκηνής (Scene View)
• Προβολή Παιχνιδιού (Game View)
• Ιεραρχία (Hierarchy)
• Έργο (Project)
• Επιθεωρητής (Inspector)

4.2.1 SCENE PANEL

Η "σκηνή" στο Unity αποτελεί τον κύριο χώρο ανάπτυξης για το παιχνίδι, όπου ο δημιουργός μπορεί να περιηγηθεί και να επεξεργαστεί τον χώρο με τη χρήση των πλήκτρων Q, W, E, R. Αυτό το παράθυρο είναι κρίσιμο στην Unity, καθώς αποτελεί το βασικό παράθυρο σχεδίασης.

Πλήκτρο Q (Εργαλείο Πλοήγησης): 

Με αυτό το πλήκτρο, που εικονίζεται ως χέρι, μετακινούμε την κάμερα στον χώρο. Κρατώντας πατημένο το αριστερό κλικ του ποντικιού, μετακινούμε την κάμερα σε διάφορες κατευθύνσεις, ενώ συνδυάζοντας με το πλήκτρο alt επιτυγχάνουμε περιστροφή γύρω από το επικεντρωμένο αντικείμενο. Με το δεξί κλικ του ποντικιού, περιστρέφουμε την κάμερα στην επιθυμητή κατεύθυνση και πατώντας παράλληλα το alt, επιτυγχάνουμε ζουμ.

Πλήκτρο W (Εργαλείο Μετακίνησης): 

Χρησιμοποιείται για τη μετακίνηση ενός επιλεγμένου αντικειμένου προς την κατεύθυνση που υποδεικνύουν τα βελάκια στο αντικείμενο, σύμφωνα με τους άξονες x, y, z.

Πλήκτρο E (Εργαλείο Περιστροφής): 

Επιτρέπει την περιστροφή ενός αντικειμένου γύρω από τους διάφορους άξονες.

Πλήκτρο R (Εργαλείο Κλίμακας): 

Με τη χρήση του πλήκτρου R, ρυθμίζουμε τις διαστάσεις του αντικειμένου σε κάθε άξονα ξεχωριστά, καθώς και την συνολική κλίμακα μεγεθύνοντας ή σμικρύνοντας το αντικείμενο.

4.2.2 GAME PANEL

Το πάνελ "Game" στο Unity είναι η ενότητα που εμφανίζει το παιχνίδι στην τελική του εκδοχή, κάτι που επιτυγχάνεται μέσω της επιλογής "Play" που βρίσκεται στην μπάρα εργαλείων, δίπλα στις επιλογές "Pause" και "Next Frame" του Unity. Πιο αναλυτικά:

Play: 

Με αυτή την επιλογή, ο χρήστης μπορεί να εκτελέσει το παιχνίδι και να πάρει μια εικόνα της τελικής του μορφής.

Pause: 

Πατώντας αυτό το κουμπί, το παιχνίδι παύει σε ένα συγκεκριμένο σημείο, δίνοντας την ευκαιρία στον χρήστη να παρατηρήσει λεπτομέρειες και να τροποποιήσει μεταβλητές που επηρεάζουν τη λειτουργία του παιχνιδιού, όπως την ταχύτητα του χαρακτήρα, τη βαρύτητα, τον φωτισμό κ.λπ.

Next Frame: 

Χρησιμοποιώντας αυτή την επιλογή, είναι δυνατή η λεπτομερής προβολή και επεξεργασία κάθε επιμέρους frame του παιχνιδιού.

Στην μπάρα εργαλείων του πάνελ "Game", στην αρχή βρίσκεται ένα αναδυόμενο μενού που επιτρέπει την προεπισκόπηση του παιχνιδιού σε επιλεγμένη ανάλυση. Δεξιότερα, υπάρχει η επιλογή "maximize on play", η οποία επιτρέπει την πλήρη οθόνη κατά την εκτέλεση του παιχνιδιού, καθώς και το "Stats", που προσφέρει στατιστικά στοιχεία σχετικά με τα γραφικά.

4.2.3 HIERARCHY PANEL

Το πάνελ "Hierarchy View" στο Unity είναι ένα εργαλείο για την οργάνωση και αποθήκευση διαφόρων στοιχείων μέσα στη σκηνή ενός παιχνιδιού, όπως κάμερες, 2D και 3D αντικείμενα, πηγές φωτός, κουτιά,

σφαίρες, κύβους, μοντέλα, δομές, επιφάνειες κ.ά. Κατά τη δημιουργία ενός νέου παιχνιδιού, συνήθως δημιουργείται αυτόματα μια νέα σκηνή με την προσθήκη της κύριας κάμερας στο Hierarchy View της σκηνής. Το Hierarchy View απεικονίζει όλα τα στοιχεία του παιχνιδιού που βρίσκονται στην ενεργή σκηνή. Οποιαδήποτε αντικείμενα δημιουργούνται δυναμικά και προστίθενται ή αφαιρούνται από την ενεργή σκηνή, εμφανίζονται εδώ ανάλογα με την κατάσταση ενεργοποίησής τους στη σκηνή.

4.2.4 PROJECT PANEL

Αυτή η ενότητα αποτελείται από όλα τα στοιχεία που έχουν αποθηκευτεί στον κύριο φάκελο ("root") του παιχνιδιού. Στο πάνελ "Project" μπορούμε να βρούμε ποικιλία αρχείων, όπως assets, σκηνές, γραμματοσειρές, ήχους, scripts, prefabs, άδειες χρήσης και ακόμα κείμενα σε απλή μορφή.

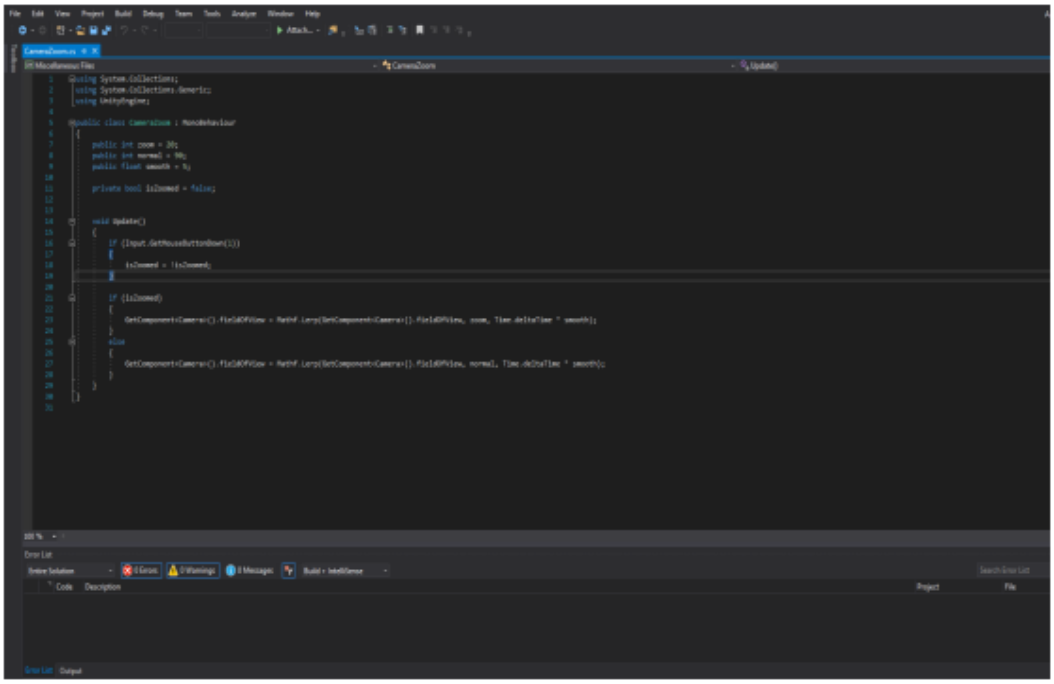
4.2.5 INSPECTOR PANEL

Στον Inspector εμφανίζονται τα διάφορα χαρακτηριστικά των αντικειμένων που βρίσκονται στο πάνελ Hierarchy. Ο χρήστης έχει τη δυνατότητα να τροποποιήσει τις τιμές αυτών των χαρακτηριστικών, να αλλάξει τη θέση τους, καθώς και να προσθέσει ή να αφαιρέσει λειτουργίες από τα επιλεγμένα αντικείμενα. Μια πολύτιμη λειτουργία του Inspector είναι το κουμπί “Lock”, το οποίο επιτρέπει το κλείδωμα της εστίασης σε ένα συγκεκριμένο αντικείμενο, ενώ ταυτόχρονα γίνεται διαχείριση άλλων αντικειμένων στο Hierarchy. Για παράδειγμα, μπορεί να γίνει drag and drop ενός αντικειμένου σε ένα στοιχείο του Inspector που είναι εστιασμένο σε ένα διαφορετικό αντικείμενο.

5. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Η Unity προσφέρει μια πληθώρα επιλογών επεξεργασίας και προσαρμογής, επιτρέποντας τη δημιουργία προηγμένων σχεδίων χωρίς την ανάγκη για εκτεταμένο προγραμματισμό. Ωστόσο, για την πλήρη εκμετάλλευση των δυνατοτήτων της και την επίτευξη εξειδικευμένων αποτελεσμάτων, ο προγραμματισμός είναι συχνά απαραίτητος. Κατά την εγκατάσταση της Unity, προτείνεται η επιλογή εγκατάστασης του Microsoft Visual Studio ως compiler, αν και οι χρήστες έχουν την ελευθερία να επιλέξουν άλλους compilers για την επεξεργασία των scripts στην Unity.

Η Unity κυρίως υποστηρίζει τη γλώσσα προγραμματισμού C#, αλλά είναι δυνατή και η χρήση άλλων γλωσσών προγραμματισμού όπως το BOO και το JavaScript, τα οποία βασίζονται στο .NET framework. Ωστόσο, για την ομαλή λειτουργία τους στο περιβάλλον της Unity μπορεί να απαιτούνται ειδικά plug-ins, γεγονός που καθιστά τη C# ως την προτιμώμενη επιλογή. Για αυτούς τους λόγους, σε αυτό το έργο, χρησιμοποίησα την C# και το Microsoft Visual Studio.



Περιβάλλον Microsoft Visual Studio

6. SCRIPTING ΣΤΗΝ UNITY

Η Unity, αξιοποιώντας το Mono, μια ανοικτού κώδικα έκδοση του .NET framework, παρέχει σημαντικές δυνατότητες στους προγραμματιστές. Αν και υποστηρίζει πολλαπλές γλώσσες προγραμματισμού, στην Unity κυρίως χρησιμοποιούνται C#, Boo και JavaScript. Η C# αποτελεί μια δημοφιλή επιλογή λόγω της εκτεταμένης διαθεσιμότητας εκπαιδευτικού υλικού και της συμβατότητάς της με το .NET και Mono, καθιστώντας την ιδανική για εμπορικές και ανοικτού κώδικα εφαρμογές. Γι' αυτούς τους λόγους, επέλεξα την C# και το Microsoft Visual Studio για το συγκεκριμένο έργο.

Στην Unity, τα scripts καθοδηγούν τη συμπεριφορά των GameObjects και τον τρόπο που αλληλεπιδρούν μεταξύ τους, δημιουργώντας τη λογική του παιχνιδιού. Το scripting στη Unity διαφέρει από τον τυπικό προγραμματισμό, καθώς δεν απαιτείται η δημιουργία κώδικα για την εκτέλεση της εφαρμογής - η Unity αναλαμβάνει αυτή τη λειτουργία. Έτσι, τα scripts εστιάζουν στους μηχανισμούς της εφαρμογής, όπως την κίνηση και τις αλληλεπιδράσεις.

Η Unity λειτουργεί μέσω ενός εκτενούς βρόχου, αναλύοντας όλα τα δεδομένα μιας σκηνής παιχνιδιού, όπως φωτισμό, σκιές, αντικείμενα και τις συμπεριφορές τους. Όταν δημιουργείται ένα νέο script C# στην Unity, αυτό επιφέρει τη δημιουργία ενός νέου αρχείου script στο φάκελο Assets, με τη δυνατότητα τοποθέτησής του σε οποιοδήποτε φάκελο επιλέγει ο χρήστης. Ένα διπλό κλικ στο αρχείο ανοίγει το νέο script στο MonoDevelop ή άλλο περιβάλλον προγραμματισμού C#, με το MonoDevelop να προσθέτει αυτόματα τις απαραίτητες κλάσεις και δομές.

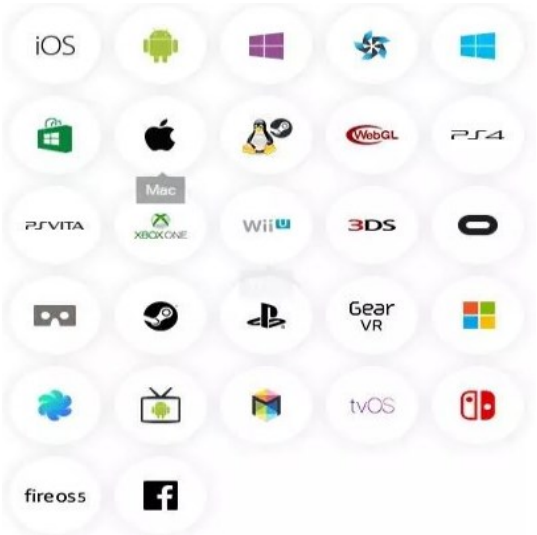
```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

Αρχικό περιεχόμενο ενός C# script στη Unity

Η μέθοδος Start() ενεργοποιείται την πρώτη φορά που το συγκεκριμένο script εκκινεί. Από την άλλη πλευρά, η μέθοδος Update είναι ένα από τα πιο κρίσιμα στοιχεία σε κάθε πλατφόρμα ανάπτυξης παιχνιδιών. Αυτή η μέθοδος, η οποία αποτελεί μέρος του συστήματος του παιχνιδιού, επανεξετάζεται σε κάθε καρέ του παιχνιδιού κατά τη διάρκεια της εκτέλεσής του, για να προσδιορίσει αν πρέπει να γίνουν ενέργειες ή αλλαγές. Για παράδειγμα, η Update μπορεί να ενεργοποιήσει αντίστοιχα κινούμενα σχέδια ή δράσεις βάσει της εκπλήρωσης ορισμένων συνθηκών ή ελέγχων.

7. PUBLISHING (ΔΗΜΟΣΙΕΥΣΗ)

Ένα σημαντικό πλεονέκτημα της Unity είναι η ευελιξία που προσφέρει στους προγραμματιστές να παράγουν το παιχνίδι τους για μια ποικιλία πλατφορμών. Ειδικά με την έκδοση Unity 5, η υποστήριξη επεκτάθηκε σε πάνω από 21 διαφορετικές πλατφόρμες.



Μέσα από το μενού Project Settings, υπάρχει η δυνατότητα προσαρμογής των γραφικών και των διαστάσεων του παιχνιδιού, ιδιαίτερα όταν αναφερόμαστε σε web player. Ένα άλλο σημαντικό πλεονέκτημα για τις εφαρμογές ιστού είναι ότι η Unity διαθέτει τον δικό της Web Player, ο οποίος έχει εγκατασταθεί από πάνω από 60 εκατομμύρια χρήστες. Τέλος, αξίζει να αναφερθεί ότι η εφαρμογή μας θα αναπτυχθεί τόσο για PC Standalone - Windows όσο και για Web player.

8. ΤΟ ΠΑΙΧΝΙΔΙ

Το "Miami City Crime Simulator: City Mafia War" είναι ένας δυναμικός τίτλος στο είδος των παιχνιδιών μάχης και στρατηγικής, όπου ο παίκτης αναλαμβάνει τον έλεγχο της μοίρας μιας πόλης που βρίσκεται στο επίκεντρο των συγκρούσεων της μαφίας.

Το παιχνίδι, που ανήκει στο είδος των gangster games, σας προκαλεί να κυριαρχήσετε στο Μαϊάμι, μια πόλη γνωστή για τις έντονες μάχες της, τις σκοτεινές συμφωνίες και τους αδίστακτους μαφιόζους. Διαθέτετε όμως δυναμικές δυνατότητες που ανταποκρίνονται σε ενός "super hero", δηλαδή κάποιον που μπορεί να πηδάει και να τρέχει δεξιοτεχνικά μέσα από λαβύρινθους δρόμων γεμάτων εγκληματικότητα.

Για τους λάτρεις των παιχνιδιών πτήσης, το παιχνίδι προσφέρει επίσης μια μοναδική εμπειρία, καθώς με το πάτημα του κουμπιού 'F', ο ήρωας απελευθερώνεται από τους εδάφινους δεσμούς, πετώντας ψηλά πάνω από τα ουρανοξύστη του Μαϊάμι. Με αυτή τη μοναδική οπτική γωνία, μπορείτε να σχεδιάσετε πολυπλόκαμες τακτικές ενάντια στις εχθρικές συμμορίες για να ανακτήσετε την πόλη σας στο χρονικό διάστημα τις κάθε πίστας που πρέπει να φτάσετε μέχρι το τελικό προορισμό.

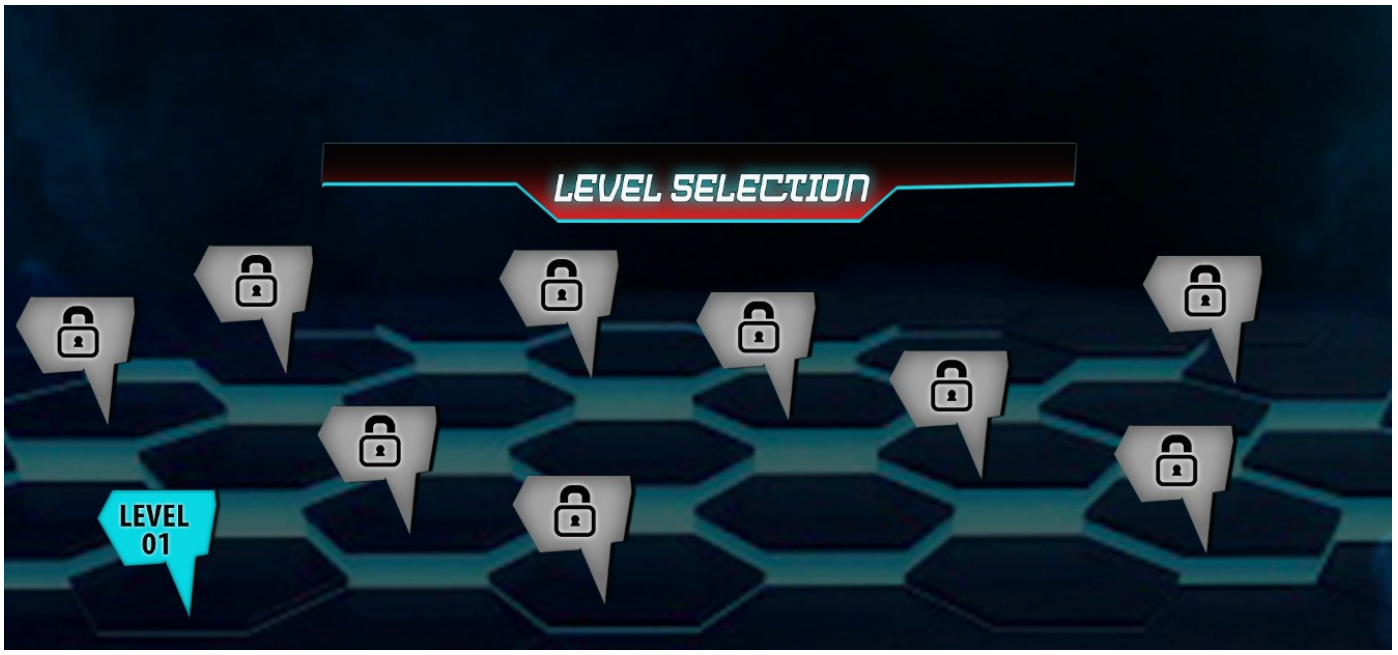
Επιπλέον, το παιχνίδι απαιτεί στρατηγική σκέψη και επιδεξιότητα, στοιχεία που προέρχονται απευθείας από τα skill games. Ο ήρωας δεν βασίζεται μόνο στη σωματική δύναμη· κάθε χτύπημα και κάθε άλμα πρέπει να είναι προσεκτικά σχεδιασμένο.

Σύμφωνα με τα πρότυπα των κορυφαίων παιχνιδιών επιβίωσης, το παιχνίδι απαιτεί επιμονή αντιμετωπίζοντας τις προκλήσεις. Η γεύση της νίκης έρχεται μόνο όταν αντέχετε ακλόνητοι ανάμεσα στους γίγαντες ανταγωνιστές του Βέγκας και τους χαοτικούς πολέμους της μαφίας.

Για τους φαν των παιχνιδιών προσομοίωσης, το παιχνίδι προσφέρει μια ρεαλιστική απεικόνιση του αστικού τοπίου και της εγκληματικής οργάνωσης του Μαϊάμι, καταδεικνύοντας τις προηγμένες μηχανικές προσομοίωσής του.

Το "Miami City Crime Simulator: City Mafia War" είναι μια περιπετειώδης και στρατηγική εμπειρία δράσης μέσα στον εγκληματικό κόσμο του Μαϊάμι, απαιτώντας από τους παίκτες να αντιμετωπίσουν με στρατηγική και τολμηρή δράση τις αδίστακτες μαφίες, προσφέροντας ταυτόχρονα ελευθερία και περιπέτεια με τη δυνατότητα πτήσης. Πρόκειται για μια μοναδική συνδυασμό που ικανοποιεί τους λάτρεις των παιχνιδιών μάχης, gangster, πτήσης, επιβίωσης και προσομοίωσης.

8.1 ΣΤΑΔΙΑ ΠΙΣΤΩΝ



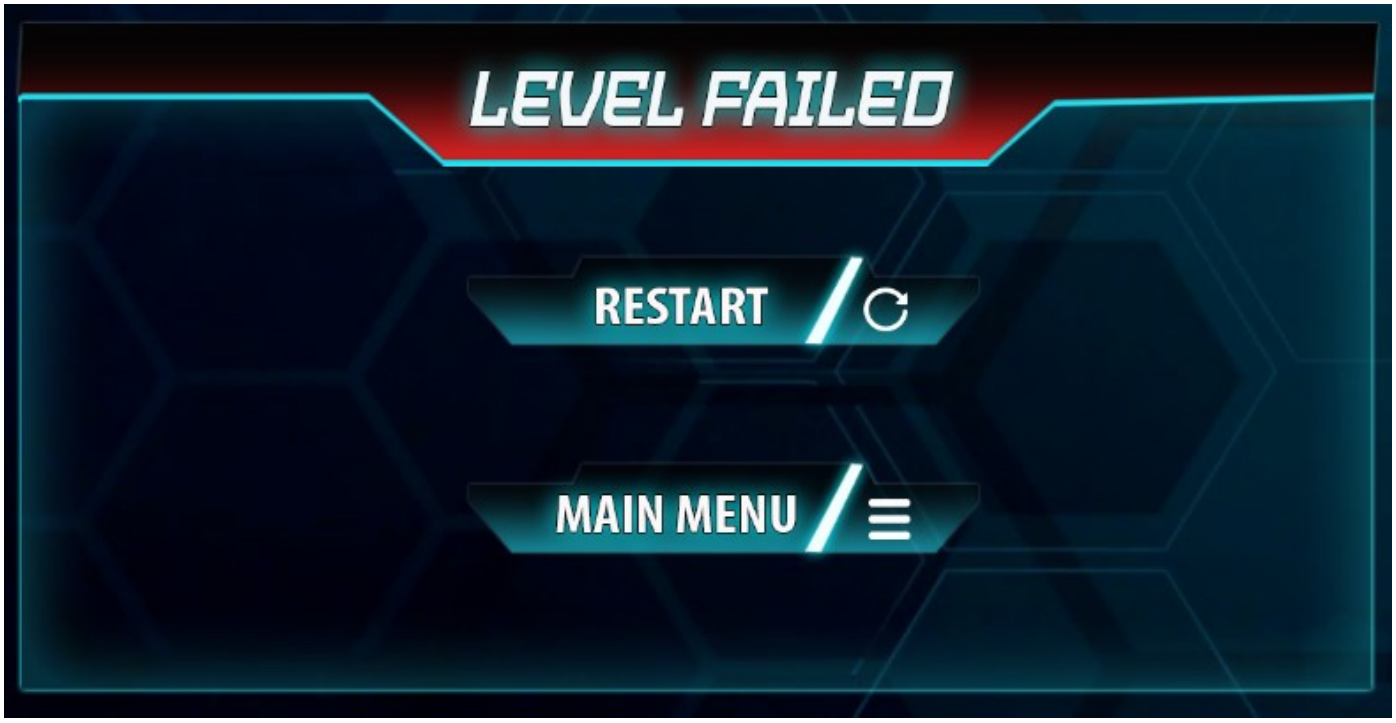
Ξεκινάει ο super ήρωας από το level 1 και όταν γράσει εγκυρα στον προορισμό του και μέσα στα χρονικά πλαίσια του χρόνου ξεκλειδώνει το επόμενο level για να ξεκλειδώσει την επόμενη αποστολή του.

Στόχος είναι να φτάσει στο κόκκινο στίγμα του χάρτη πριν τελειώσει ο χρόνος για να είναι επιτυχής η αποστολή.

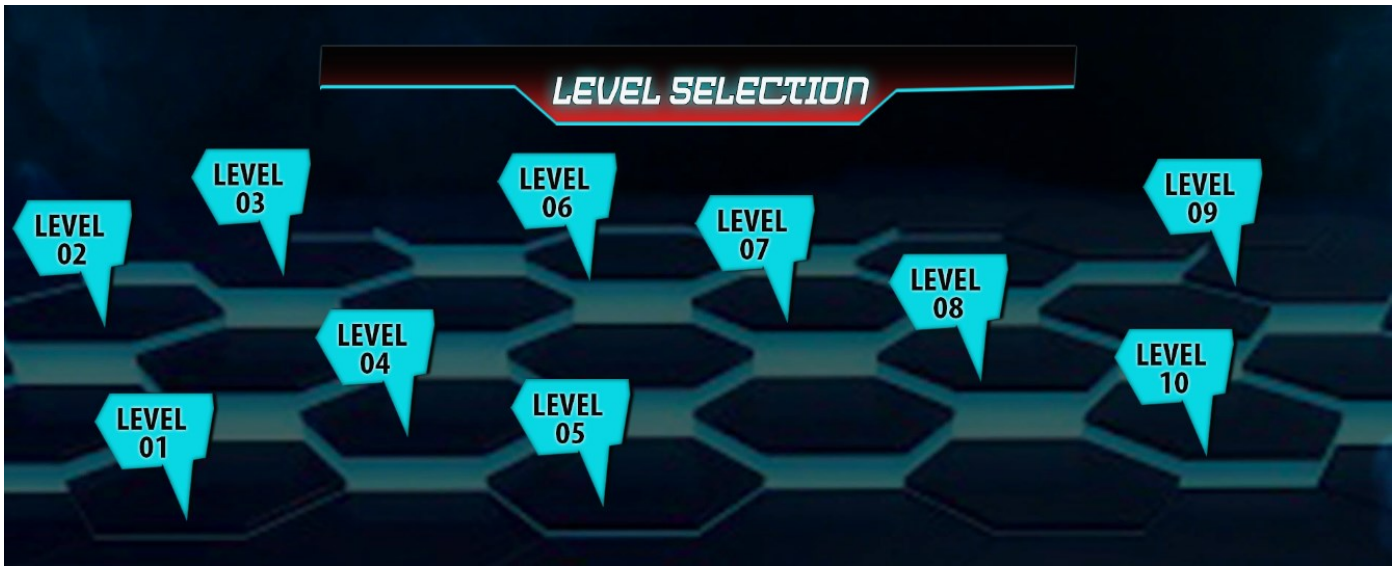
Χάρτης Πίστας

Χρόνος εκτέλεσης αποστολής

Εάν δεν φτάσει έγκυρα αποτυγχάνει και πρέπει να προσπαθήσει ξανά.



Το παιχνίδι ολοκληρώνετε όταν εκτελέσει 10 επιτυχείς αποστολές.



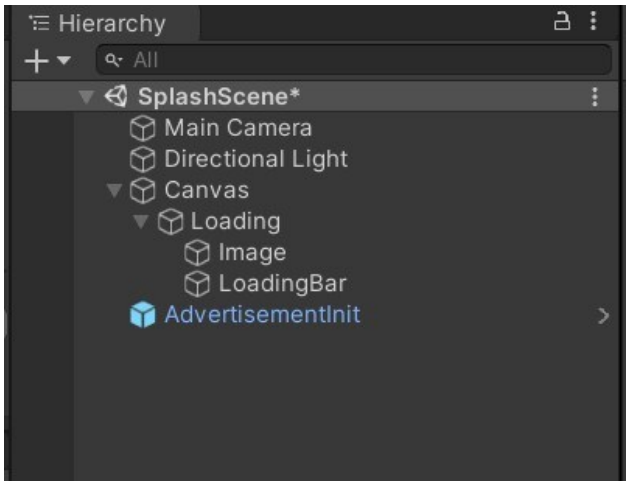
9.CONTROLS

Μπορείτε να πλοηγηθείτε σε τέσσερις κατευθύνσεις (αριστερά, δεξιά, εμπρός και πίσω) (W,A,S,D) χρησιμοποιώντας το joystick στην αριστερή πλευρά της οθόνης, ενώ μπορείτε να πηδήσετε (SPACE), να πετάξετε (F) και να τρέξετε γρήγορα (Shift) χρησιμοποιώντας τα αντίστοιχα κουμπιά.

10.SCENES

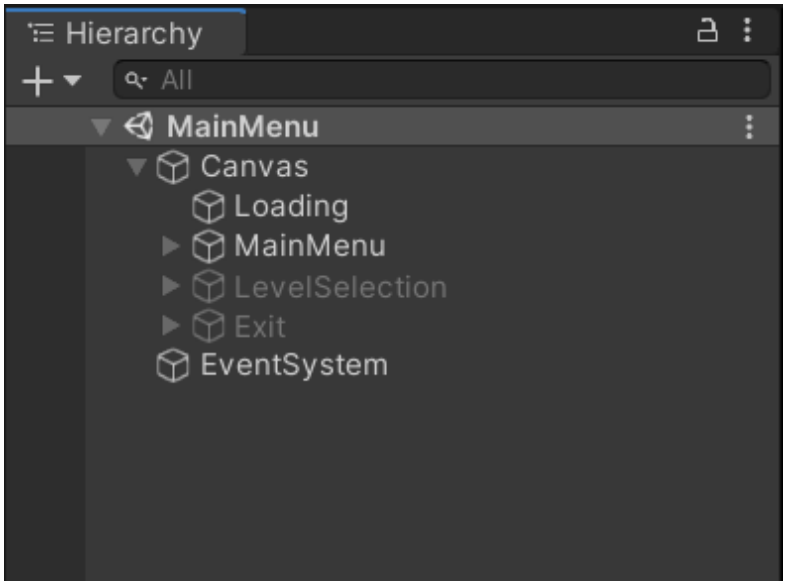
10.1 SPLASHSCENE

Γνωστή και ως σκηνή των Plugins, όπου ενεργοποιούνται όλα τα πρόσθετα τρίτων μερών ή βασικά στοιχεία του παιχνιδιού όπως διαφημίσεις, αναλυτικά στοιχεία και άλλα παρόμοια στοιχεία. Όλες οι κλάσεις περιπτώσεων του παιχνιδιού και τα singleton αρχικοποιούνται σε αυτή τη σκηνή.



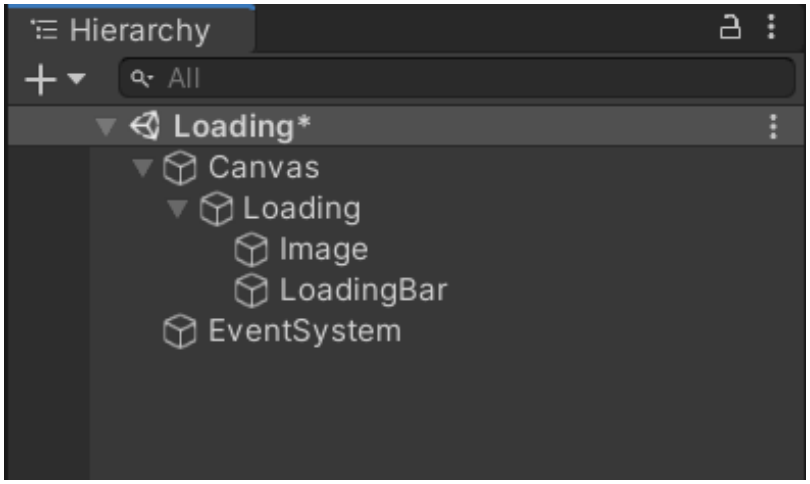
10.2 MAIN MENU

Η σκηνή που είναι υπεύθυνη για την διαχείριση των μενού για την έναρξη του παιχνιδιού. Μενού όπως επιλογή παίκτη, επιλογή επιπέδου, ρυθμίσεις, ημερήσιες ανταμοιβές βρίσκονται συχνά σε αυτή τη σκηνή.



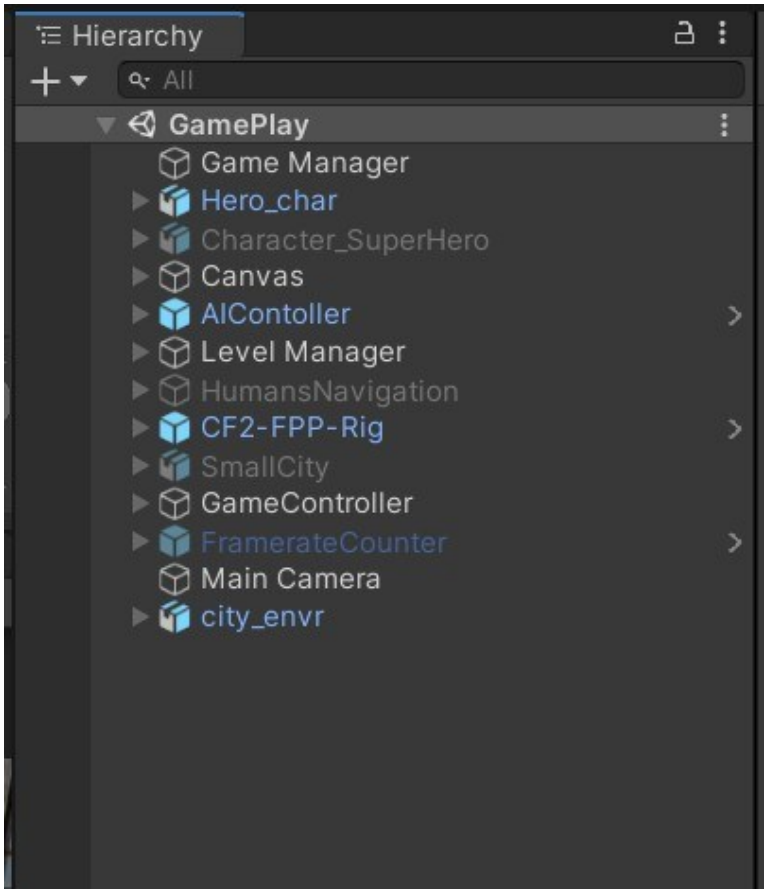
10.3 LOADING

Η σκηνή που είναι υπεύθυνη για τη διαχείριση της ασύγχρονης φόρτωσης άλλων σκηνών στο παιχνίδι.



10.4 GAMEPLAY

Η κύρια σκηνή που περιέχει τον παίκτη, το περιβάλλον και όλα τα στοιχεία του gameplay όπως εχθροί, εμπόδια και αποστολές που πρέπει να ολοκληρωθούν.



11.ΒΑΣΙΚΗ ΔΟΜΗ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ – SCRIPTS

11.1 PREFERENCES.cs

Αυτό το σενάριο έχει σχεδιαστεί για να διαχειρίζεται και να αποθηκεύει τις προτιμήσεις του παίκτη και την πρόοδο του παιχνιδιού. Με πιο απλά λόγια, το σενάριο προσφέρει μια μέθοδο για την αποθήκευση και διαχείριση των προτιμήσεων του παίκτη και της προόδου του στο παιχνίδι, όπως οι ρυθμίσεις ήχου, το τρέχον επίπεδο του παιχνιδιού, τα ξεκλειδωμένα επίπεδα και το σκορ του παίκτη. Χρησιμοποιεί το σύστημα PlayerPrefs της Unity για την αποθήκευση και φόρτωση αυτών των τιμών. Το σενάριο σχεδιάστηκε ως singleton, εξασφαλίζοντας ότι υπάρχει μόνο μία περίπτωση για τη διαχείριση αυτών των προτιμήσεων σε όλο το παιχνίδι. Επιτρέπει επίσης την επαναφορά όλων των προτιμήσεων στις προεπιλεγμένες τους τιμές.

```
2 references
void Load()
{
    _sound = PlayerPrefs.GetInt ("_sound");
    _Level = PlayerPrefs.GetInt ("_Level");
    _LevelUnlock = PlayerPrefs.GetInt ("_LevelUnlock");
    _score = PlayerPrefs.GetInt ("_score");
}

4 references
void Save()
{
    PlayerPrefs.SetInt ("_sound", _sound);
    PlayerPrefs.SetInt ("_Level", _Level);
    PlayerPrefs.SetInt ("_LevelUnlock", _LevelUnlock);
    PlayerPrefs.SetInt ("_score", _score);
}

Unity Message | 0 references
public void Reset()
{
    PlayerPrefs.DeleteAll ();
    PlayerPrefs.Save ();
    Load ();
}
```

11.2 LOADING.cs

Αυτό το σενάριο χρησιμοποιείται για τη διαχείριση της φόρτωσης μιας νέας σκηνής. Ενσωματώνει τη χρήση της δημοφιλούς βιβλιοθήκης απεικόνισης κινήσεων DOTween για ομαλές κινήσεις κατά τη διάρκεια της διαδικασίας φόρτωσης. Σε πιο απλοποιημένους όρους, αυτό το σενάριο είναι υπεύθυνο για την ομαλή μετάβαση από την τρέχουσα σκηνή στην καθορισμένη 'sceneToLoad' ενώ εμφανίζει μια γραμμή φόρτωσης που γεμίζει καθώς φορτώνεται η νέα σκηνή. Χρησιμοποιεί κινήσεις DOTween για να δημιουργήσει μια

οπτικά ευχάριστη εμπειρία φόρτωσης. Η γραμμή φόρτωσης γεμίζει γρήγορα όταν η πρόοδος φόρτωσης φτάσει το 90% ή περισσότερο, και εξασφαλίζει ότι η νέα σκηνή ενεργοποιείται μόνο όταν η φόρτωση είναι σχεδόν ολοκληρωμένη. Αν η πρόοδος φόρτωσης είναι κάτω από το 90%, η φόρτωση συνεχίζεται στο επόμενο πλαίσιο για να αποφευχθεί το πάγωμα του παιχνιδιού κατά τη διαδικασία.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using DG.Tweening;

Unity Script (2 asset references) | 0 references
public class Loading : MonoBehaviour
{
    //  public GameObject pnLoading;
    public string sceneToLoad = "GamePlay";
    public Image fillImag;
    //  public Image fill;

    AsyncOperation async;

    Unity Message | 0 references
    void Start()
    {

        Time.timeScale = 1;
        async = SceneManager.LoadSceneAsync(sceneToLoad);
        async.allowSceneActivation = false;

        _Loading();
    }

    1 reference
    public void _Loading()
    {
        if (async.progress >= 0.9f)
        {
            fillImag.DOFillAmount(1, 1.0f).OnComplete(() =>
                {
                    //                  pnLoading.SetActive(false);
                    async.allowSceneActivation = true;

                }).WaitForCompletion();
        }
        else
        {
            fillImag.DOFillAmount(async.progress, 1.0f).OnComplete(() =>
                {

                    Invoke("_Loading", 0.5f);

                }).WaitForCompletion();
        }
    }

}
```

11.3 MISSIONSAI.cs

Αυτό το σενάριο λειτουργεί σαν διαχειριστής αποστολών στο παιχνίδι. Ασχολείται με διάφορες αποστολές και τους κανόνες τους. Κάθε αποστολή έχει τις δικές της ενέργειες και κανόνες, και όταν ολοκληρωθεί μια αποστολή, εμφανίζεται ένα μήνυμα. Το σενάριο χρησιμοποιεί αντικείμενα παιχνιδιού για τις εργασίες της αποστολής και διασφαλίζει ότι ο AIController του παιχνιδιού συμπεριφέρεται σωστά για κάθε αποστολή. Επίσης, εμφανίζει το τρέχον επίπεδο του παιχνιδιού χρησιμοποιώντας το Preferences.Instance.Level.

```
public Mission mission;


Unity Message | 0 references
void Start () {
    switch (mission) {
    case Mission.Mission3:
        MissionObject.SetActive (true);
        break;
    case Mission.Mission7:
        MissionObject.GetComponent<AIContoller> ().maxVehicles = 1;
        System.Array.Resize (ref MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs, 1);
        MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs.SetValue (MissionOtherObject, 0);
        break;
    case Mission.Mission8:
        MissionObject.GetComponent<AIContoller> ().maxVehicles = 1;
        System.Array.Resize (ref MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs, 1);
        MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs.SetValue (MissionOtherObject, 0);
        break;
    case Mission.Mission9:
        MissionObject.GetComponent<AIContoller> ().maxVehicles = 1;
        System.Array.Resize (ref MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs, 1);
        MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs.SetValue (MissionOtherObject, 0);
        break;
    case Mission.Mission10:
        MissionObject.GetComponent<AIContoller> ().maxVehicles = 1;
        System.Array.Resize (ref MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs, 1);
        MissionObject.GetComponent<AIContoller> ().vehiclesPrefabs.SetValue (MissionOtherObject, 0);
        break;
    }
    print ("******* Level NO : " + Preferences.Instance.Level);
}

0 references
public void MissionDone()
{
    switch (mission) {

    case Mission.Mission1:
        print ("Mission 1 Done");
            break;

        case Mission.Mission2:
            print("Mission 2 Done");
            break;
    }
}
```

11.4 LEVELMANAGER.cs

Αυτό το σενάριο αναλαμβάνει τη διαχείριση και παρουσίαση διαφόρων επιπέδων του παιχνιδιού. Λειτουργεί σε δύο καταστάσεις:

• Σε κατάσταση δοκιμής (όταν η μεταβλητή "isTesting" έχει την τιμή true), ορίζει το τρέχον επίπεδο του παιχνιδιού σε ένα συγκεκριμένο επίπεδο δοκιμής και εμφανίζει τα σχετικά στοιχεία του παιχνιδιού.
• Σε κανονική κατάσταση, περνά από μια λίστα με επίπεδα του παιχνιδιού και εμφανίζει τα στοιχεία του παιχνιδιού για το επίπεδο στο οποίο βρίσκεται ο παίκτης, βάσει της προόδου του.

Τα στοιχεία του παιχνιδιού για τα άλλα επίπεδα παραμένουν κρυμμένα. Αυτό εξασφαλίζει ότι κατά τη

διάρκεια της κανονικής παιχνιδιού, μόνο το σωστό επίπεδο φαίνεται στην οθόνη.

```
public class LevelManager : MonoBehaviour {

    public GameObject[] Levels;
//  public GameObject sirenBtn;
    public bool isTesting = false;
    public int TestLevel;

    Unity Message | 0 references
    void Start ()
    {
        print (".........Levels LEngth "+Levels.Length);
        if (isTesting)
        {
            Preferences.Instance.Level = TestLevel;
            Levels[TestLevel - 1].SetActive (true);

        }
        else
        {
            for (int i = 0; i < Levels.Length; i++)
            {
                if (i == Preferences.Instance.Level - 1) {
                    Levels [i].SetActive (true);
                }
                else
                    Levels [i].SetActive (false);
            }
        }
    }

}
```

11.5 LEVELCOMPLETE.cs

Αυτό το σενάριο χρησιμεύει για να διαπιστώσει πότε ο χαρακτήρας του παίκτη μπαίνει σε μια συγκεκριμένη ζώνη ενεργοποίησης. Όταν συμβαίνει αυτό, ακολουθούνται τα εξής βήματα:
• Ελέγχεται αν το αντικείμενο που εισέρχεται έχει ετικέτα "Player."

Εάν το αντικείμενο είναι όντως ο χαρακτήρας του παίκτη, εκτελούνται οι εξής ενέργειες:

• Απενεργοποιείται το στοιχείο Animator του γονικού αντικειμένου, το οποίο μπορεί να ελέγχει τις κινήσεις.
• Κρύβεται το γονικό αντικείμενο απενεργοποιώντας το στοιχείο MeshRenderer του, κάνοντάς το αόρατο.

• Καλείται η μέθοδος LevelComplete στον GameManager για να δηλώσει ότι το επίπεδο έχει ολοκληρωθεί με επιτυχία.

```
using UnityEngine;
using System.Collections;

Unity Script | 0 references
public class LevelComplete : MonoBehaviour {

    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player") {
            transform.GetComponentInParent<Animator>().enabled = false;
            transform.GetComponentInParent<MeshRenderer>().enabled = false;
            GameManager.instance.LevelComplete ();
        }
    }
}
```

11.6 GAMEOVER.cs

Η λειτουργία αυτού του σεναρίου είναι να ανιχνεύει πότε αντικείμενα με συγκεκριμένες ετικέτες εισέρχονται σε μια ζώνη ενεργοποίησης. Όταν ένα αντικείμενο με την κατάλληλη ετικέτα μπει στη ζώνη, το σενάριο εκτελεί τις ακόλουθες ενέργειες:
• Ελέγχει την ετικέτα του εισερχόμενου αντικειμένου με μια λίστα ετικετών.
• Ψάχνει για αντιστοιχία συγκρίνοντας την ετικέτα του εισερχόμενου αντικειμένου με κάθε ετικέτα στη λίστα.
Εάν βρεθεί αντιστοιχία, το σενάριο εντοπίζει το αντικείμενο GameManager στη σκηνή και ενεργοποιεί τη μέθοδο GameOver του. Αυτό συνήθως χρησιμοποιείται για να δηλώσει το τέλος του παιχνιδιού όταν συγκεκριμένα αντικείμενα με ειδικές ετικέτες εισέρχονται στη ζώνη ενεργοποίησης.

```
using UnityEngine;
using System.Collections;

Unity Script | 0 references
public class GameOver : MonoBehaviour {

    public string[] tags;

    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        foreach (string st in tags) {
            if (other.tag == st)
                {
                    GameObject.FindObjectOfType<GameManager> ().GameOver ();
                }
        }
    }
}
```

11.7 GAEMANAGER.cs

Αυτό το σενάριο διαδραματίζει έναν κρίσιμο ρόλο σε ένα παιχνίδι Unity, καθώς επιβλέπει μια ευρεία γκάμα από εργασίες διαχείρισης παιχνιδιού, όπως ολοκλήρωση επιπέδων, καταστάσεις τέλους παιχνιδιού, χρονομέτρηση και αναπαραγωγή ήχου. Λειτουργεί ως κεντρικός κόμβος για τον συντονισμό πολλών γεγονότων του παιχνιδιού, όπως το πάγωμα και η επανέναρξη του παιχνιδιού, η διαχείριση των καταστάσεων τέλους παιχνιδιού και ολοκλήρωσης επιπέδου, η παρακολούθηση χρονομέτρων για κάθε επίπεδο και πολλά άλλα. Επιπλέον, ενσωματώνει δυνατότητες αναπαραγωγής ήχου και στοιχεία διεπαφής χρήστη για ρυθμίσεις και οδηγίες. Για τη δημιουργία ομαλών κινήσεων, το σενάριο βασίζεται στη βιβλιοθήκη DOTween και είναι επίσης άμεσα συνδεδεμένο με στοιχεία διαφήμισης όπως banner και διαφημίσεις interstitial από το σενάριο Advertisement.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.Events;
using DG.Tweening;
Unity Script (2 asset references) | 12 references
public class GameManager : MonoBehaviour
{

    //Details
    //text used: Copperplate Gothic ...
    public GameObject dailPause, dailGamePlay, dailGameOver, dialLevelComplete, pnWait, pnSetting, nextBtn, hrnBtn, sirenBtn;
    public RectTransform msnPanel;
    public Text gameMsgs;
//  public AudioSource audioSource;
//  public AudioClip buttonSound;
    Vector2 newPos= new Vector2(0f, 1500f), orgPos;
    Image playerHealth;
    float currentHealth, totalHealth;
    public string[] MSg;
    public AudioClip horn, siren, btnSound;
    AudioSource _audiosrc;

    public static GameManager instance;

    Unity Message | 0 references
    void Awake(){
        instance = this;
        AnaabiTechAds.Instance.HideBanner();
        AnaabiTechAds.Instance.ShowInterstitial();

    }
```

11.8 PLAYERCOLLISION.cs

Στο παιχνίδι μου στην Unity, αυτό το σενάριο είναι υπεύθυνο για τη διαχείριση των συγκρούσεων που συμβαίνουν με τον χαρακτήρα του παίκτη. Καθορίζει τον τρόπο αλληλεπίδρασης του παίκτη με αντικείμενα και συμπαίκτες καθώς προχωράει σε διάφορα επίπεδα. Βεβαιώνεται ότι ο παίκτης πληροί συγκεκριμένα κριτήρια πριν προχωρήσει στο επόμενο επίπεδο, όπως την παρουσία κοντά σε συμπαίκτες ή την ολοκλήρωση στόχων επιπέδου. Επιπλέον, διαχειρίζεται συγκρούσεις κατά τη διάρκεια της πτήσης και παρέχει καθοδηγητικά μηνύματα βασισμένα στην πρόοδο του παιχνιδιού.

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

Unity Script (3 asset references) | 0 references
public class PlayerCollision : MonoBehaviour {

    public GameObject level2CompPoint,level3CompPoint,level4CompPoint,level5CompPoint,level6CompPoint,level7CompPoint;
    public GameObject level2Medic,level3Cat,level4Girl,level5Boy,level6Doctor,level7BoyGril;

    public float checkCompainonDistance = 10;
    public string MsgForCompanionDistance = "Your Companion(s) is Far From you, Come with your Companion(s)";
    GameObject Temp;

    Unity Message | 0 references
    void OnTriggerEnter(Collider other){
        if (!GetComponent<FlyBehaviour> ().fly) {
            if (other.tag == "Point") {
                switch (Preferences.Instance.Level - 1) {
                case 0:
                    GameManager.instance.LevelComplete ();
                    break;
                case 1:
                    GameManager.instance.LevelComplete ();
                    break;
                case 2:
                    GameManager.instance.LevelComplete ();
                    break;
                default:
                    if (Vector3.Distance (transform.position, Temp.transform.position) > checkCompainonDistance) {
                        GameManager.instance.MSg [Preferences.Instance.Level - 1] = MsgForCompanionDistance;
                        GameManager.instance.ShowMsnBriefing ();
                    } else
                        GameManager.instance.LevelComplete ();
                    break;

                }
            }
```

12. FOR PLAYERCONTROLS

12.1 AIMBEHAVIOURBASIC.cs

Σε αυτό το παιχνίδι, το συγκεκριμένο script χρησιμοποιείται για τον έλεγχο του τρόπου στόχευσης του χαρακτήρα του παίκτη. Δίνει τη δυνατότητα στον παίκτη να στοχεύσει με το όπλο του ή να προσαρμόσει την οπτική του γωνία μέσω ενός σταυρού στόχευσης. Επιπλέον, προσφέρει την επιλογή εναλλαγής των ώμων και στόχευσης με διαφορετικά πλάνα κάμερας. Το script διαχειρίζεται διάφορες πτυχές της στόχευσης για τον χαρακτήρα του παίκτη, συμπεριλαμβανομένων της ενεργοποίησης και απενεργοποίησης της στόχευσης, της εναλλαγής ώμων, της εμφάνισης του σταυρού στόχευσης και της προσαρμογής της περιστροφής του χαρακτήρα βάσει της προσανατολισμού της κάμερας. Χρησιμοποιεί coroutines για την επίτευξη ομαλών μεταβάσεων μεταξύ των διαφορετικών καταστάσεων στόχευσης.

```
// Update is used to set features regardless the active behaviour.
Unity Message | 0 references
void Update ()
{
    // Activate/deactivate aim by input.
    if (ControlFreak2.CF2Input.GetAxisRaw(aimButton) != 0 && !aim)
    {
        StartCoroutine(ToggleAimOn());
    }
    else if (aim && ControlFreak2.CF2Input.GetAxisRaw(aimButton) == 0)
    {
        StartCoroutine(ToggleAimOff());
    }

    // No sprinting while aiming.
    canSprint = !aim;

    // Toggle camera aim position left or right, switching shoulders.
    if (aim && ControlFreak2.CF2Input.GetButtonDown (shoulderButton))
    {
        aimCamOffset.x = aimCamOffset.x * (-1);
        aimPivotOffset.x = aimPivotOffset.x * (-1);
    }

    // Set aim boolean on the Animator Controller.
    behaviourManager.GetAnim.SetBool (aimBool, aim);
}
```

12.2 BASICBEHAVIOUR.cs

Αυτό το σενάριο επιτρέπει στον παίκτη να εναλλάσσεται μεταξύ διαφόρων δράσεων, όπως το περπάτημα, το τρέξιμο, τη στόχευση και άλλες. Ακολουθούν τα κύρια στοιχεία και λειτουργίες αυτού του σεναρίου:

12.2.1 BASICBEHAVIOUR class:

- Διαχειρίζεται την τρέχουσα ενεργή ή υπερκείμενη συμπεριφορά του παίκτη.
- Περιλαμβάνει ρυθμίσεις και κοινές λειτουργίες που χρησιμοποιούνται από όλες τις συμπεριφορές του παίκτη.
- Παρέχει αναφορές σε συστατικά του παίκτη όπως η κάμερα, η κίνηση, το Rigidbody και άλλα.
- Χειρίζεται την είσοδο για οριζόντια και κάθετη κίνηση και τρέξιμο.
- Ελέγχει το πεδίο οπτικής γωνίας (FOV) της κάμερας του παίκτη.
- Εντοπίζει αν ο παίκτης βρίσκεται στο έδαφος.
- Καθορίζει λειτουργίες για την εγγραφή, καταγραφή, απεγγραφή, υπερκάλυψη και ανάκληση συμπεριφορών.
- Διαχειρίζεται προσωρινούς κλειδώματα συμπεριφοράς για μεταβάσεις όπως το άλμα ή η στόχευση.
- Περιλαμβάνει βοηθητικές λειτουργίες για τον έλεγχο αν ο παίκτης τρέχει, κινείται και άλλα.

12.2.2 GENERICBEHAVIOUR CLASS (ABSTRACT):

• Αποτελεί τη βάση για όλες τις συμπεριφορές του παίκτη.
• Παρέχει αναφορές σε βασικά συστατικά, όπως το Animator και τον διαχειριστή BasicBehaviour.
• Αναθέτει έναν μοναδικό κωδικό συμπεριφοράς για κάθε ειδική κλάση συμπεριφοράς που προέρχεται από αυτό.
• Επιτρέπει στις παράγωγες συμπεριφορές να καθορίσουν αν επιτρέπεται το τρέξιμο.

Βασικά, το BasicBehaviour συντονίζει τις συμπεριφορές του παίκτη, τις ενεργοποιεί/απενεργοποιεί και διαχειρίζεται τις μεταβάσεις ανάμεσά τους. Το GenericBehaviour λειτουργεί ως πρότυπο για όλες τις συμπεριφορές του παίκτη, προσφέροντας κοινές λειτουργικότητες και υπηρετώντας ως αφετηρία για προσαρμοσμένες συμπεριφορές όπως το περπάτημα, το τρέξιμο και η στόχευση.

Τα σενάρια προσαρμοσμένης συμπεριφοράς κληρονομούν από το GenericBehaviour και ορίζουν την μοναδική τους λειτουργικότητα για διάφορες καταστάσεις στο παιχνίδι, όπως το περπάτημα, το τρέξιμο και η στόχευση. Το σύστημα υποστηρίζει επίσης την προσωρινή υπερκάλυψη συμπεριφορών για δράσεις όπως η στόχευση ή το άλμα.

```
// This class manages which player behaviour is active or overriding, and call its local functions.
// Contains basic setup and common functions used by all the player behaviours.
Unity Script (10 asset references) | 2 references
public class BasicBehaviour : MonoBehaviour
{
    public Transform playerCamera;                      // Reference to the camera that focus the player.
    public float turnSmoothing = 0.06f;                 // Speed of turn when moving to match camera facing.
    public float sprintFOV = 100f;                      // the FOV to use on the camera when player is sprinting.
    public string sprintButton = "Sprint";              // Default sprint button input name.

    private float h;                                    // Horizontal Axis.
    private float v;                                    // Vertical Axis.
    private int currentBehaviour;                       // Reference to the current player behaviour.
    private int defaultBehaviour;                       // The default behaviour of the player when any other is not active.
    private int behaviourLocked;                        // Reference to temporary locked behaviour that forbids override.
    private Vector3 lastDirection;                      // Last direction the player was moving.
    private Animator anim;                              // Reference to the Animator component.
    private ThirdPersonOrbitCamBasic camScript;             // Reference to the third person camera script.
    private bool sprint;                                // Boolean to determine whether or not the player activated the sprint mode.
    private bool changedFOV;                            // Boolean to store when the sprint action has changed de camera FOV.
    private int hFloat;                                 // Animator variable related to Horizontal Axis.
    private int vFloat;                                 // Animator variable related to Vertical Axis.
    private List<GenericBehaviour> behaviours;          // The list containing all the enabled player behaviours.
    private List<GenericBehaviour> overridingBehaviours; // List of current overriding behaviours.
    private Rigidbody rBody;                            // Reference to the player's rigidbody.
    private int groundedBool;                           // Animator variable related to whether or not the player is on the ground.
    private Vector3 colExtents;                         // Collider extents for ground test.

    // Get current horizontal and vertical axes.
    2 references
    public float GetH { get { return h; } }
    2 references
    public float GetV { get { return v; } }
```

12.3 FLYBEHAVIOUR.cs

Το εν λόγω σενάριο ελέγχει τη συμπεριφορά πτήσης σε ένα παιχνίδι Unity. Ακολουθούν τα κύρια συστατικά του και ο τρόπος λειτουργίας του:

12.3.1 FLYBEHAVIOUR CLASS:

- Κληρονομεί από το GenericBehaviour, δηλώνοντας ότι αποτελεί μια συγκεκριμένη συμπεριφορά του παίκτη.
- Διαχειρίζεται τη συμπεριφορά πτήσης του παίκτη, επιτρέποντας την εναλλαγή μεταξύ των καταστάσεων πτήσης και μη πτήσης.
- Προσφέρει παραμέτρους όπως το κουμπί πτήσης, την ταχύτητα πτήσης και τις περιορισμένες κάθετες γωνίες.
- Διαχειρίζεται τις εισόδους του παίκτη για την ενεργοποίηση της κατάστασης πτήσης.
- Ρυθμίζει τη βαρύτητα όταν ο παίκτης πετάει για να επιτρέψει την ελεύθερη κίνηση στον αέρα.
- Διαχειρίζεται τη μετατόπιση της κάμερας και την κατεύθυνση του collider ανάλογα με την κατάσταση πτήσης.
- Τροποποιεί το στερεό σώμα (rigid body) του παίκτη με δυνάμεις για τον έλεγχο της κίνησής του στον αέρα.
- Προσαρμόζει την περιστροφή του παίκτη για να ταιριάζει με την κατεύθυνση της κάμερας κατά την πτήση.
- Αντικαθιστά την βασική μέθοδο OnOverride της βασικής κλάσης για να εξασφαλίσει ότι η κάθετη κατεύθυνση του collider είναι σωστή.
- Υλοποιεί την συνάρτηση LocalFixedUpdate, η οποία διαχειρίζεται τους μηχανισμούς πτήσης και την κίνηση του παίκτη κατά τη διάρκεια της πτήσης.
- Ορίζει τη συνάρτηση Rotating, η οποία περιστρέφει τον παίκτη για να ταιριάζει με τη σωστή προσανατολισμό βάσει της κατεύθυνσης της κάμερας και της εισόδου.

12.3.2 GENERICBEHAVIOUR CLASS (ABSTRACT):

- Λειτουργεί ως η βασική κλάση για όλες τις συμπεριφορές του παίκτη.
- Προσφέρει αναφορές σε κρίσιμα συστατικά, όπως το Animator και τον διαχειριστή BasicBehaviour.
- Αναθέτει έναν μοναδικό κωδικό συμπεριφοράς σε κάθε κλάση συμπεριφοράς που προέρχεται από αυτή.
- Δίνει τη δυνατότητα στις παράγωγες συμπεριφορές να καθορίσουν εάν επιτρέπεται το τρέξιμο.

Στην πράξη, αυτό το σενάριο επιτρέπει στον παίκτη να ενεργοποιεί την κατάσταση πτήσης πατώντας ένα ορισμένο κουμπί. Στην κατάσταση πτήσης, ο παίκτης μπορεί να κινηθεί ελεύθερα στον αέρα, και η κάμερα καθώς και ο προσανατολισμός του παίκτη προσαρμόζονται ανάλογα με την κατεύθυνση της κίνησης. Το σενάριο διαχειρίζεται τους μηχανισμούς πτήσης, συμπεριλαμβανομένης της ταχύτητας κίνησης και του προσανατολισμού του παίκτη, και εξασφαλίζει ομαλές μεταβάσεις μεταξύ των καταστάσεων πτήσης και μη πτήσης.

```
// Update is used to set features regardless the active behaviour.
Unity Message | 0 references
void Update()
{
    // Toggle fly by input, only if there is no overriding state or temporary transitions.
    if (ControlFreak2.CF2Input.GetButtonDown(flyButton) && !behaviourManager.IsOverriding()
        && !behaviourManager.GetTempLockStatus(behaviourManager.GetDefaultBehaviour))
    {
        fly = !fly;

        // Force end jump transition.
        behaviourManager.UnlockTempBehaviour(behaviourManager.GetDefaultBehaviour);

        // Obey gravity. It's the law!
        behaviourManager.GetRigidBody.useGravity = !fly;

        // Player is flying.
        if (fly)
        {
            // Register this behaviour.
            behaviourManager.RegisterBehaviour(this.behaviourCode);
        }
        else
        {
            // Set collider direction to vertical.
            col.direction = 1;
            // Set camera default offset.
            behaviourManager.GetCamScript.ResetTargetOffsets();

            // Unregister this behaviour and set current behaviour to the default one.
            behaviourManager.UnregisterBehaviour(this.behaviourCode);
        }
    }

    // Assert this is the active behaviour
    fly = fly && behaviourManager.IsCurrentBehaviour(this.behaviourCode);

    // Set fly related variables on the Animator Controller.
    behaviourManager.GetAnim.SetBool(flyBool, fly);
}
```

12.4 MOVEBEHAVIOUR.cs

Αυτό το σενάριο ελέγχει την κίνηση ενός χαρακτήρα σε ένα παιχνίδι Unity. Ακολουθεί περιγραφή της λειτουργίας του:

12.4.1 MOVEBEHAVIOUR CLASS:

- Κληρονομεί από το GenericBehaviour, προσδιορίζοντας ότι αποτελεί μια συμπεριφορά του παίκτη για το περπάτημα και το τρέξιμο.
- Διαχειρίζεται την κίνηση του χαρακτήρα, συμπεριλαμβανομένων του περπατήματος, του τρεξίματος και του πηδήματος.
- Προσφέρει παραμέτρους όπως ταχύτητα περπατήματος, ταχύτητα τρεξίματος, ταχύτητα σπριντ, ύψος πηδήματος και ρυθμίσεις σχετικές με το άλμα.

- Διαχειρίζεται τις εισόδους του παίκτη, συμπεριλαμβανομένων του πηδήματος και των αλλαγών ταχύτητας λόγω εισόδων ή άλλων παραγόντων.
- Προσαρμόζει τις παραμέτρους κίνησης της ανίματης εικόνας του παίκτη, όπως την ταχύτητα κίνησης.
- Ελέγχει την περιστροφή του χαρακτήρα για να ταιριάζει με την κατεύθυνση της κάμερας και την είσοδο του παίκτη.
- Ανιχνεύει συγκρούσεις για να τοποθετήσει τον χαρακτήρα στο έδαφος και να προσαρμόσει τη συμπεριφορά του παίκτη ανάλογα με την κατάσταση επί του εδάφους.

12.4.2 GENERICBEHAVIOUR CLASS (ABSTRACT):

- Αποτελεί τη βάση για όλες τις συμπεριφορές του παίκτη.
- Δίνει πρόσβαση σε θεμελιώδη στοιχεία όπως το Animator και τον διαχειριστή BasicBehaviour.
- Καθορίζει έναν μοναδικό κωδικό συμπεριφοράς για κάθε συγκεκριμένη κλάση συμπεριφοράς που προέρχεται από αυτή.
- Επιτρέπει στις παράγωγες συμπεριφορές να καθορίζουν αν επιτρέπεται το σπριντ.

Στην πρακτική εφαρμογή, αυτό το σενάριο επιτρέπει στον χαρακτήρα του παίκτη να κινείται, να πηδάει και να ελέγχει την ταχύτητα κίνησης με βάση τις εντολές του χειριστή. Διαχειρίζεται διάφορες καταστάσεις κίνησης, όπως το περπάτημα και το τρέξιμο, διαχειρίζεται τον προσανατολισμό του παίκτη, τις παραμέτρους της κίνησης και την ανίχνευση συγκρούσεων για να εξασφαλίσει ότι ο χαρακτήρας είναι σταθερά στο έδαφος και συμπεριφέρεται σωστά. Το σενάριο συνεργάζεται με τις κλάσεις BasicBehaviour και FlyBehaviour για να δημιουργήσει ένα πλήρες σύστημα κίνησης του χαρακτήρα στο παιχνίδι Unity.

```
// Start is always called after any Awake functions.
Unity Message | 0 references
void Start()
{
    // Set up the references.
    jumpBool = Animator.StringToHash("Jump");
    groundedBool = Animator.StringToHash("Grounded");
    behaviourManager.GetAnim.SetBool (groundedBool, true);

    // Subscribe and register this behaviour as the default behaviour.
    behaviourManager.SubscribeBehaviour (this);
    behaviourManager.RegisterDefaultBehaviour (this.behaviourCode);
    speedSeeker = runSpeed;
}

// Update is used to set features regardless the active behaviour.
Unity Message | 0 references
void Update ()
{
    // Get jump input.
    if (!jump && ControlFreak2.CF2Input.GetButtonDown(jumpButton) && behaviourManager.IsCurrentBehaviour(this.behaviourCode) && !behaviourManager.IsOverriding())
    {
        jump = true;
    }
}
```

13. ΣΤΙΓΜΙΟΤΥΠΑ ΠΑΙΧΝΙΔΙΟΥ (SCREENSHOTS)

Figure 1 Black Hero Flying

Figure 2 Black Hero Flying 2

Figure 3 Black Hero Standing 1

Figure 4 Black Hero Standing 2

Figure 5 Black Hero Flying towards Mission point

Figure 6 Looking for enemies

Figure 7 Red Hero

Figure 8 Trying to catch criminal

Figure 9 Talking with NPC

Figure 10 Red Hero jumping

Figure 11 Starting to fly

Figure 12 Checking the situation

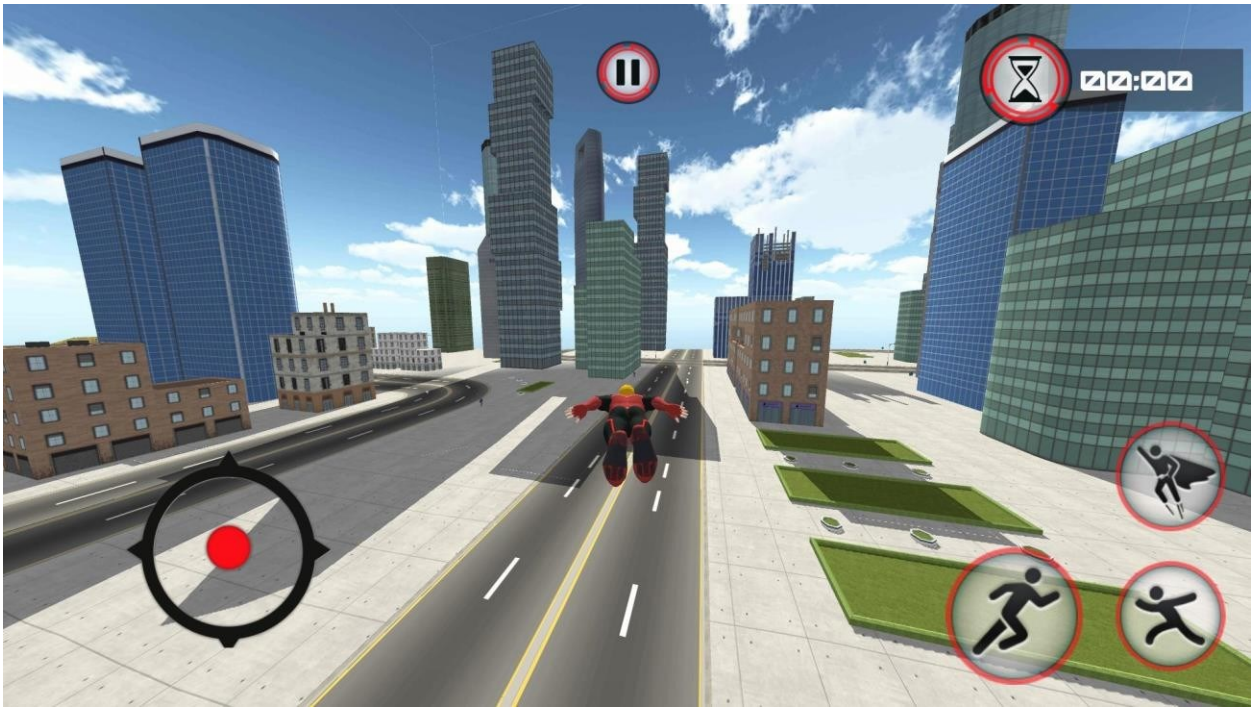


Figure 13 Going to target

Figure 14 Red hero standing

Figure 15 Red hero running

Figure 16 Running

Figure 17 Watching van for criminals

Figure 18 Landing

Figure 19 Following biker

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η Unity αποτελεί ένα εκτεταμένο πακέτο που προσφέρει στους προγραμματιστές την ευκαιρία να υλοποιήσουν το δικό τους όραμα στον χώρο των 2D ή 3D παιχνιδιών με σχετική ευκολία. Η Unity ξεχωρίζει ως μία από τις κορυφαίες πλατφόρμες ανάπτυξης παιχνιδιών, ιδιαίτερα για Android και iOS. Με την πιο πρόσφατη έκδοσή της και την έντονη έμφαση στο οπτικό τμήμα, αναμένεται να αποκτήσει σημαντική παρουσία τόσο στους υπολογιστές όσο και στις κονσόλες παιχνιδιών, καθώς απευθύνεται τόσο σε αρχάριους όσο και σε πιο επιδέξιους προγραμματιστές.

Ένα σημαντικό στοιχείο που καθιστά τη Unity πιο προσβάσιμη είναι ο πλούτος των διαθέσιμων οδηγιών και οδηγιών που μπορούν να βρεθούν online, καθώς και η δυναμική κοινότητα του Community Forum που παρέχει άμεση και αποτελεσματική βοήθεια. Η δωρεάν έκδοση της πλατφόρμας, καθώς και η ευρεία υποστήριξη πολλαπλών πλατφορμών, την καθιστά εξαιρετικά δημοφιλή μεταξύ των μικρότερων εταιριών ανάπτυξης σε μια βιομηχανία παιχνιδιών που βρίσκεται σε συνεχή ανάπτυξη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://unity.com/
2. https://forum.unity.com/
3. https://learn.unity.com/tutorials
4. https://en.wikipedia.org/wiki/Unity_Technologies
5. https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf
6. https://whatis.techtarget.com/definition/scrip
7. Unity - Scripting API: MonoBehaviour (unity3d.com)
8. Unity: Analysing the First Game Engine IPO (naavik.co)