ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Ανάπτυξη παιγνίου τριών διαστάσεων σε περιβάλλον Unity για λειτουργικό windows

Πτυχιακή εργασία

Παπούλια Σοφία

Αριθμός Μητρώου: 21672

Αθήνα, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

3D Game development in Unity for Windows operating system

Bachelor thesis

Papoulia Sophia, 21672

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Επιβλέπων Καθηγητής : Βασίλειος Δαλάκας

Ε.ΔΙ.Π, Τμήμα Πληροφορικής & Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Τσερπές Κωνσταντίνος

Αναπληρωτής Καθηγητής, Τμήμα πληροφορικής και τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Τσαδήμας Ανάργυρος

Ε.Δ.Ι.Π., Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Η Παπούλια Σοφία, δηλώνω υπεύθυνα ότι:

1)Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2)Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Η παρούσα εργασία είναι αφιερωμένη στην οικογένειά μου, για την στήριξή τους και την απερίγραπτη υπομονή που έχουν δείξει κατά τη διάρκεια της συγγραφής της.

ΠΕΡΙΛΗΨΗ

Διανύουμε μία εποχή υψηλών ταχυτήτων και μεγάλων αλλαγών. Ιδιαίτερα τις τελευταίες δεκαετίες έχουν σημειωθεί εκπληκτικοί πρόοδοι στον τεχνολογικό τομέα. Πληθώρα από τις αλλαγές αυτές, είχαν άμεσο αντίκτυπο στον τρόπο με τον οποίο ζουν, εργάζονται και ψυχαγωγούνται. Εργαλεία επικοινωνίας όπως είναι τα κινητά τηλέφωνα, οι δορυφόροι και το διαδίκτυο, επιτρέπουν στον άνθρωπο να παραμένει σε σταθερή επαφή με άλλους σε μεγαλύτερες αποστάσεις, ακόμα και σε πιο απομακρυσμένα μέρη. Οι πρόσφατες τεχνολογικές εξελίξεις έχουν σε πολλές περιπτώσεις, αλλάξει δραστικά τον τρόπο με τον οποίο οι άνθρωποι βιώνουν τον κόσμο γύρω τους. Αυτό διότι με το πάτημα ενός κουμπιού μπορούν να επικοινωνήσουν σε κάποιον που μπορεί να βρίσκεται στην άλλη μεριά της υδρογειου με αστραπιαία ταχύτητα, να αποκτήσει πρόσβαση σε μία πληρώθα πληροφοριών ή και να γίνει ακόμα άβαταρ σε έναν ψηφιακό κόσμο δικής του δημιουργίας.

Σε έναν κόσμο που αποτελείται από 0 και 1, τα βιντεοπαιχνίδια προσφέρουν μοναδικές ευκαιρίες μέσω της διαδραστικοτητας τους με τον κόσμο τους, κάνοντάς τα ένα μοναδικό μέσο μετάδοσης. Η παρούσα πτυχιακή εργασία μελετά και ασχολείται με την ανάλυση και την σχεδίαση παιχνιδιών τρόμου που ήταν μία ιδιαίτερη δημοφιλή κατηγορία από την αρχές της δεκαετίας του 90 (όπου και πρωτοεμφανίστηκε), μέχρι και σήμερα. Πραγματοποιήθηκε αναδρομή τόσο στο υλικοτεχνικό κομμάτι των κονσολών, τόσο και στα γραφικά, ενώ στη συνέχεια έγινε αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν. Βασικός στόχος της αποτελεί η αξιοποίηση της έρευνας που πραγματοποιήθηκε για τη δημιουργία ενός ρεαλιστικού περιβάλλοντος, τόσο στο οπτικό κομμάτι των γραφικών, όσο και στην ατμόσφαιρα. Στα πλαίσια υλοποίησής της, έγινε η χρήση του προγράμματος Unity και του Vegas Pro, για τη συντέλεση των ήχων.

ABSTRACT

We live in an era of high speeds and enormous alterations. Especially the past few decades have seen some amazing advances in technology. Many of these changes have had a direct impact on the way people live,work, and get entertained. Communication tools, such as cell phones, satellites, and the Internet, allow people to keep in constant contact across longer distances and from the most remote places. Recent technological developments have, in many cases, drastically changed the way people experience the world around them. This is because at the push of a button he can communicate with someone who may be on the other side of the globe at lightning speed, gain access to a wealth of information or even become an avatar in a digital world of his own creation.

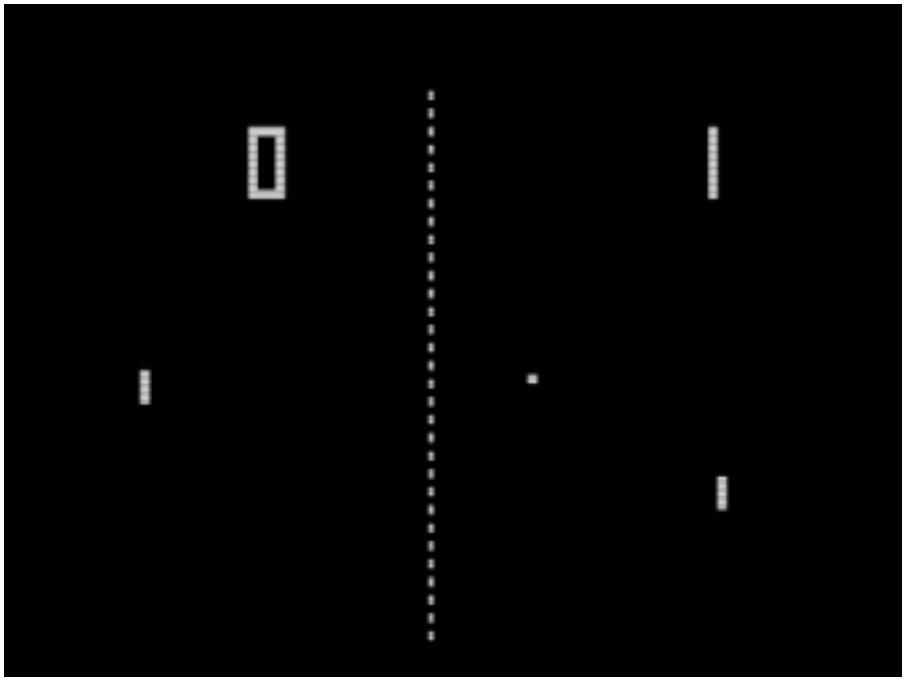
In a world of 0s and 1s, video games offer unique opportunities through their interaction with their world, making them a unique medium of transmission. The present thesis deals with the analysis and design of horror games that have been a particularly popular category since the early 90's (where it first appeared), until today. It was done retrospectively both in the hardware part of the consoles and in the graphics, while at the same time the technologies that were used were referenced. Its main goal is to utilize the research carried out to create a realistic environment, both in the visual part of the graphics and in the atmosphere. As part of its implementation, the Unity program was used to create the game and Vegas Pro was used to compose the sounds.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΜΕΡΟΣ Α’

1. ΜΙΑ ΣΥΝΤΟΜΗ ΑΝΑΔΡΟΜΗ ΣΤΗΝ ΕΞΕΛΙΞΗ ΤΩΝ ΓΡΑΦΙΚΩΝ ΣΤΑ ΠΑΙΧΝΙΔΙΑ

Κάποιος θα μπορούσε να πει ότι τα γραφικά ενός παιχνιδιού προδίδουν την τεχνολογία τους, καθώς επίσης και την εποχή από την οποία προήλθαν. Μέσα στα τελευταία 50 χρόνια της ύπαρξής τους, ξεκινήσαμε από δύο γραμμές που μπορούσαν μόνο να μετακινηθούν σε κάθετο άξονα και τώρα έχουμε φτάσει στο σημείο να υφίστανται ψηφιακοί κόσμοι με μεγάλη λεπτομέρεια, που μπορεί κάποιος να εξερευνήσει. Ποιες ήταν οι θεμελιώδεις γραφικές υπάρξεις και πως η τεχνολογία της εκάστοτε εποχής διαμόρφωσε τον τύπο των παιχνιδιών που παίζουμε;



[Εικόνα 1]: Στιγμιότυπο από το pong που κυκλοφόρησε το 1972

Η ιστορία των βιντεοπαιχνιδιών ξεκινάει από πιο παλιά. Συγκεκριμένα βρισκόμαστε στην Ιαπωνία το 1889, όπου το κράτος έχει θέσει απαγόρευση κατά των χαρτοπαιγνίων, καθώς τότε ήταν άμεσα συνδεδεμένα με τον τζόγο. Στις 23 Σεπτεμβρίου, λοιπόν, ο Fusajiro Yamauchi ιδρύει την εταιρία Marufuku, της οποίας το κύριο εμπόρευμα ήταν το Hanafuda, ένα Ιαπωνικό παιχνίδι

καρτών. Αρκετά χρόνια αργότερα, και συγκεκριμένα το 1929, ο Yamauchi, όντας μεγάλης ηλικίας, μεταβιβάζει την εταιρία του στον γαμπρό του, Sekiryo Kaneda.

Φθάνοντας στις αρχές της δεκαετίας του 50, ο Kaneda, μετονόμασε την εταιρεία από Marufuku, στην γνωστή σε όλους μας, Nintendo (που σημαίνει: "Άσε την τύχη στον παράδεισο"). Μερικά χρόνια αργότερα, και συγκεκριμένα το 1959, έχουμε την παραγωγή του πρώτου υπολογιστή από την DEC (Digital Equipment Corporation), τον PDP-1 (Programmed Data Processor-1) με 9 kilobytes υπολογιστική ισχύς. Η ανάγκη για τη δημιουργία εφαρμογών ήταν εμφανής και για αυτό το λόγο η DEC στράφηκε προς τους φοιτητές του MIT (Massachusetts Institute of Technology).

1960 - 1970

Στις αρχές της νέας δεκαετίας, το 1961, ένας φοιτητής από τους προαναφερόμενους, ο Steve Russell, δημιουργεί το πρώτο διαδραστικό παιχνίδι για υπολογιστή το 'Spacewar!'. Σε αυτό, τα δύο διαστημόπλοια η 'σφήνα' και η 'βελόνα' κινούνταν ελεύθερα στην οθόνη και έριχναν πυρομαχικά μεταξύ τους, στην προσπάθεια να καταρριφθεί το ένα από τα δύο. Έπειτα, μετά την κυκλοφορία του, το ένα άτομο, μετατράπηκε σε ομάδα, η οποία δούλευε στο να μπορεί να τρέξει το παιχνίδι και σε άλλες εκδόσεις του PDP-1, κάνοντάς το μάλιστα και το πρώτο παιχνίδι που μπορούσε να τρέξει σε πολλαπλές υπολογιστικές εγκαταστάσεις.

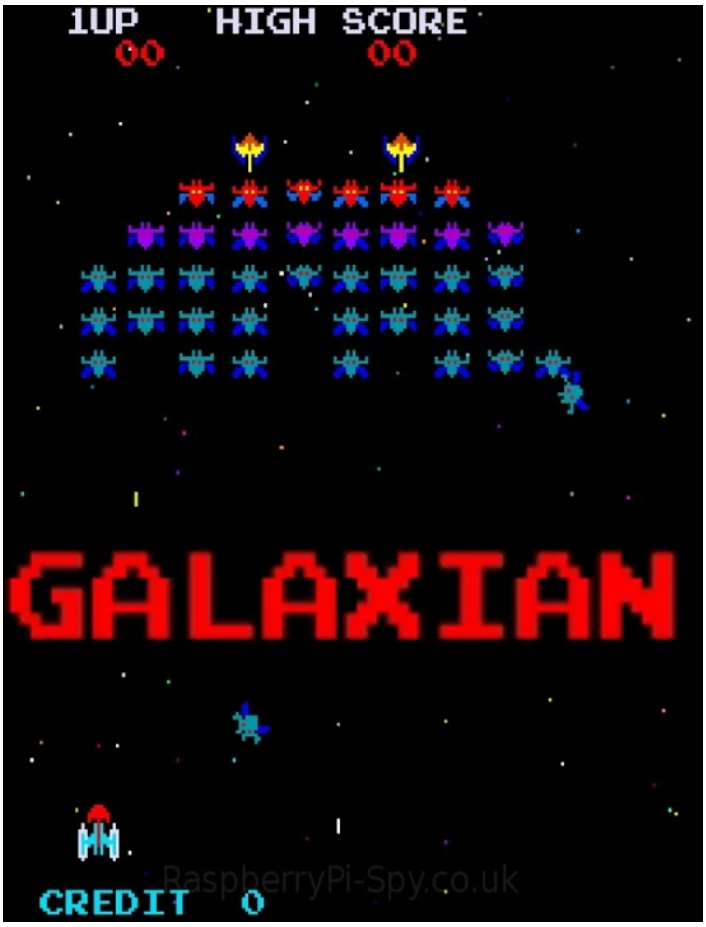
[Εικόνα 2]: Διαφήμιση του Computer space

1970-1980

Ξεκινώντας, μεταφερόμαστε στην εποχή των “70s”, όπου η Intel κυκλοφόρησε το πρώτο της προγραμματίσιμο μικροεπεξεργαστή στην αγορά (Intel 4004) και η γλώσσα προγραμματισμού C γεννήθηκε. Μία εποχή που παρείχε τα σωστά θεμέλια, ώστε να ευδοκιμήσει η βιομηχανία των βιντεοπαιχνιδιών. Τότε βέβαια, τα πράγματα ήταν πολύ απλά σε αυτόν τον τομέα και, γι αυτό τον λόγο, ακόμα και το να μετακινείται το φως στην οθόνη, θεωρούνταν ως κάτι απίστευτο.

Το παιχνίδι pong, που βρήκε το 1972, ήταν η πρώτη μεγάλη επιτυχία στη βιομηχανία των βιντεοπαιχνιδιών. Παρά την επιτυχία του, η καινοτομία αυτή δεν κράτησε πολύ και οι ανάγκες για νέες τεχνολογίες και πιο ρεαλιστικά γραφικά άρχισαν να γίνονται όλο και πιο έντονες. Ενώ οι έγχρωμες τηλεοράσεις υπήρχαν και πριν από το δεύτερο παγκόσμιο πόλεμο, τα πρώτα παιχνίδια ήταν ασπρόμαυρα. Το μόνο που μπορούσε να γίνει, ήταν η τοποθέτηση έγχρωμων ημιδιαφανών

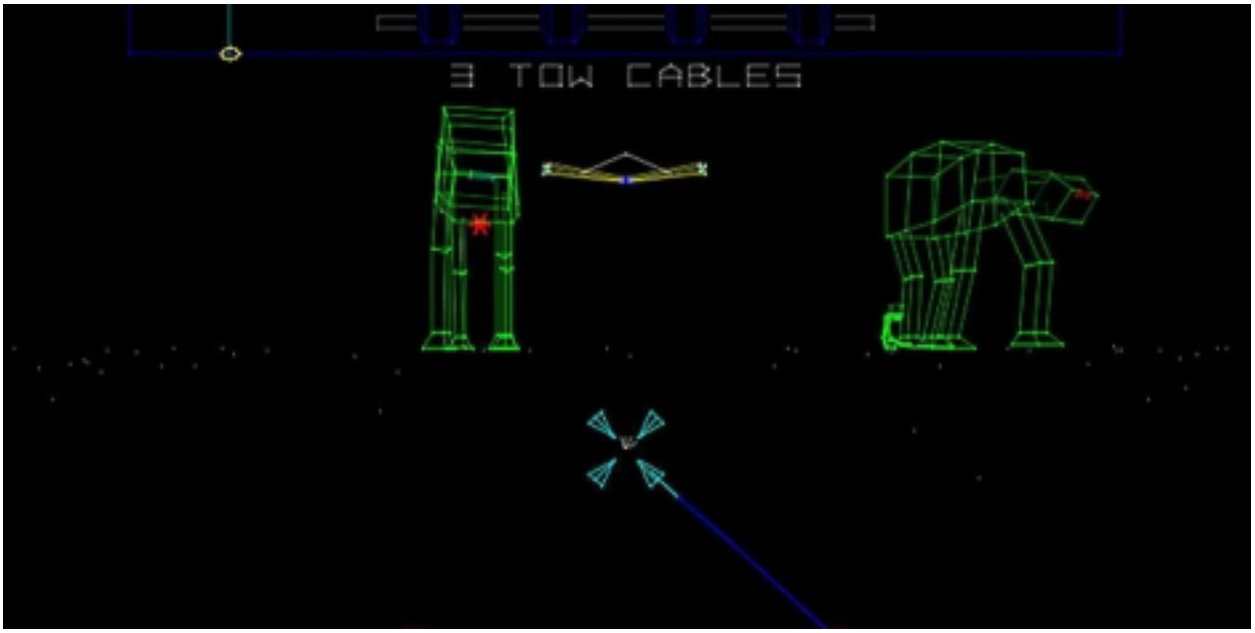
πλαστικών σελίδων πάνω από την ασπρόμαυρη οθόνη. Μία περιορισμένη λύση, αλλά και μια οικονομική.



[Εικόνα 3]: Στιγμιότυπο από το παιχνίδι Galaxian

Δεν είναι εύκολο να προσδιορίσουμε ποιο ήταν το πρώτο έγχρωμο arcade παιχνίδι. Υπήρχε μία έγχρωμη παραδοχή του Gotcha (1972), σε κάποια multiplayers υπήρχαν χρώματα - ώστε να διαφοροποιείται ο ένας παίκτης από τους άλλους (Indy 4, 1976) και το Car Polo (1977) ήταν το πρώτο που χρησιμοποιούσε μικροεπεξεργαστή. Το πρώτο επιτυχημένο RGB, όμως, ήταν το Galaxian (1979). Ουσιαστικά αποτελούσε μία πιο όμορφη έκδοση του space invaders, όπου κάθε διαστημόπλοιο μπορούσε να μετακινηθεί ελεύθερα μέσα στην οθόνη. Το πιο εντυπωσιακό ήταν η χρήση πολλαπλών χρωμάτων που υπήρχε σε κάθε στοιχείο.

1980 - 1990

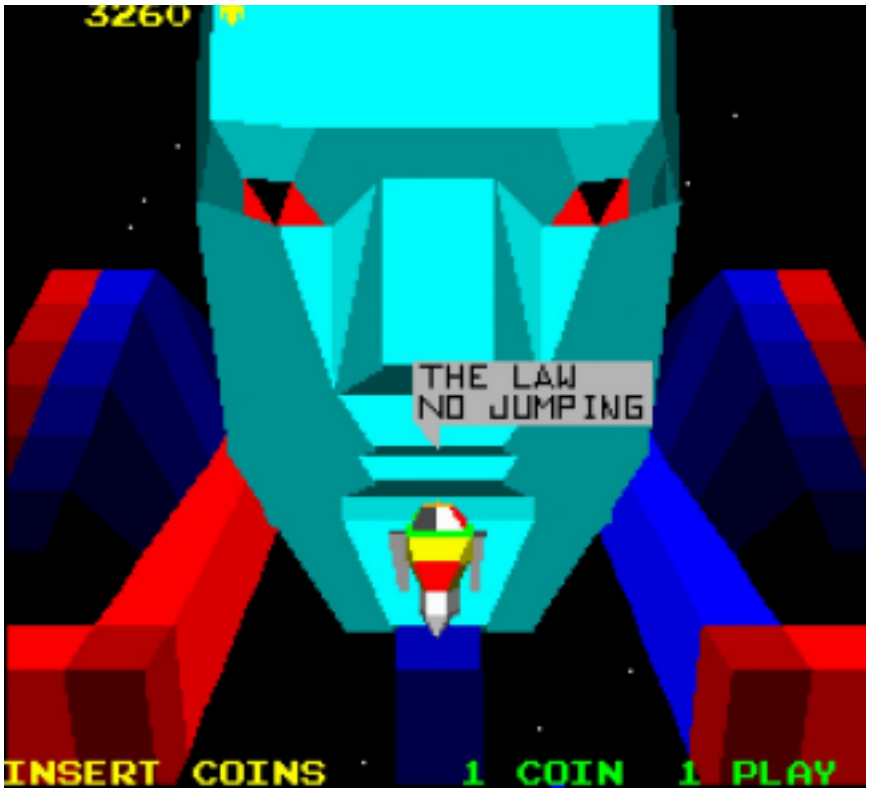


[Εικόνα 4]: Στιγμιότυπο από το παιχνίδι Star Wars για arcade (1983)

Το 1980 κατά κανόνα πλέον, κυριαρχούν τα πολύχρωμα γραφικά. Χαρακτηριστικό παράδειγμα το pacman, το οποίο, άλλωστε δεν θα ήταν το ίδιο χωρίς τα πολύχρωμα φαντάσματα ή τον κίτρινο πρωταγωνιστή του. Την ίδια περίοδο οι οι developers αρχίζουν να κλείνουν προς τον κόσμο των τριών διαστάσεων και έτσι έχουμε την εμφάνιση των πολυγωνων ως μέσο για την απεικόνιση χώρων και αντικειμένων. Η επεξεργαστική ισχύς των τότε μηχανημάτων, αποτελούσε ακόμα τροχοπέδη για τη ανάπτυξη τους και για να ‘τρέξουν’ έπρεπε να γίνουν και υποχωρήσεις. Τα πρώτα, ήταν περιορισμένα σε γραμμική απεικόνιση και, αν και απλά, μπορούσαν να δημιουργήσουν ολόκληρες σκηνές, απλά και μόνο με την χρήση μερικών διανυσματικων γραμμών. Παρεμφερής τεχνολογία χρησιμοποιήθηκε και στο παιχνίδι Star Wars για Arcade του 1983, όπου έβαζε τον παίκτη στην θέση του πιλότου. Ακόμα και οικιακοί υπολογιστές των 8-bit διαχειρίζονταν την τεχνική αυτή και αν και δεν οι απεικονίσεις ήταν πιο λιτές, προσδίδοντας

βάθος, παρέχοντας στον παίκτη την δυνατότητα της εξερεύνησης.

Λογικό επακόλουθο μετά την γραμμές αυτές, ήταν το γέμισμά τους με επίπεδη σκίαση (flat shading), απλό σαν λογική, αλλά ζόρικο για τα τότε μηχανήματα, μιας και μπορούσε εύκολα να πέσει ο ρυθμός των καρέ σε αρκετά χαμηλά επίπεδα. Το πρώτο παιχνίδι που παρουσίασε αυτό τον τρόπο σχεδίασης ήταν το I Robot (1983) καθιστώντας το μπροστά για την εποχή από την εποχή του, αν και δεν αποτέλεσε μεγάλη εμπορική επιτυχία. Όσο αυξάνονταν οι απαιτήσεις στο υλικοτεχνικό κομμάτι, μειωνόταν το ενδιαφερον για τα arcade παιχνίδια, έχοντας ως αποτέλεσμα στο τέλος της δεκαετίας να κυριαρχήσουν τα 3d γραφικά. Οι προσομοιωτές πτήσεων εκμεταλεύονταν την πρόοδο που είχε σημειωθεί στο υλικοτεχνικό κομμάτι των οικιακών υπολογιστών και του flat shading, έχοντας ελευθερία κινήσεων σε λεπτομερή arcade περιβάλλοντα. Μερικά παιχνίδια οδήγησης υιοθέτησαν την τεχνική αυτή αντίστοιχα. Στο stunt car racer (1989) του Geoff Crammond ο παίκτης τρέχει με ιλιγγιώδη ταχύτητα.



[Εικόνα 5]: Στιγμιότυπο από το παιχνίδι I Robot (1983)

Η έλλειψη στην ποικιλία των απεικονίσεων με την τεχνική της επίπεδη σκίασης, οδήγησε αρκετούς developers στον να χρησιμοποιήσουν υλικοτεχνικές πατέντες, με στόχο την προσομοίωση τρισδιάστατων κόσμων. Παράδειγμα αποτελεί το λεγόμενο Mode 7 που χρησιμοποιούσε το Super Entertaining System της Nintendo, που εγκαθίδρυσε μία στοιχειώδη, για την τότε εποχή, μορφή χαρτογράφησης των υφών (texture mapping). Αυτό μπορεί να ήταν ένα μισό μέτρο, αλλά αποτελεί έναν ιδανικό τρόπο στην εισαγωγή τρισδιάστατου πεδίου σε κλασική δράση δύο διαστάσεων. Παιχνίδια όπως το Super Mario Kart, διατήρησε έναν υγιή αριθμό καρέ, ενώ πρόδιδε την ψευδαίσθηση βάθους στην οθόνη.

1990-2000

Τη δεκαετία του 90 έχουμε την κυκλοφορία και εφαρμογή του SuperFX επεξεργαστή στα φυσίγγια του Super Nintendo Entertaining System όπου εξυπηρετούσε την ενεργοποίηση τρισδιάστατων πολυγώνων για την απεικόνιση γραφικών αναμειγμένο με εφέ κλιμάκωσης των στοιχείων άλλα και δύο διαστάσεων. Η εκφόρτωση γραφικών σε άλλον επεξεργαστή θα αποδειχθεί χρήσιμη τεχνική μελλοντικά. Οι υπολογιστές συμβατοί με την IBM είχαν το πλεονέκτημα ενός αρθρωτού σχεδιασμού, μαζί με μια τιμή πολύ μακριά από αυτή των κονσολών. Αυτό σήμαινε ότι στις αρχές της δεκαετίας του '90 μπορούσαν να αρχίσουν να ξεπερνούν τα τότε γραφικά όρια.

Τα πρώτα παιχνίδια σε υπολογιστές αρκετές φορές έκαναν χρήση pre-rendered φόντων. Παιχνίδια όπως το Alone In the Dark, δέσμευε μνήμη μόνο για τα πολύγωνα που σχετίζονταν με τον βασικό χαρακτήρα και τους εχθρούς, με τον υπόλοιπο κόσμο να είναι ζωγραφισμένος ως bitmap. Αυτή η τεχνική αποδείχθηκε χρήσιμη, στην διαχείριση περιορισμένης υπολογιστικής δύναμης. Οι δημιουργοί που χρησιμοποιούσαν αυτή την τεχνική, μπορούσαν αντί να αποδώσουν μια πλήρη τρισδιάστατη σκηνή, να στρέφουν την προσοχή του σε πιο λεπτομερή μοντέλα χαρακτήρων. Ορισμένα πρώιμα παιχνίδια ήταν πιο φιλόδοξα, λαμβάνοντας μια οπτική γωνία πρώτου προσώπου, αντί για μια σταθερή προβολή κάμερας.

Μία τεχνική όπου έκανε τα πρώτα παιχνίδια εξερεύνησης σε ένα χάρτη βιώσιμα, είναι η λεγόμενη τεχνική της ‘Ακτίνας’, ή αλλιώς Raycasting, οπως είναι διαδεδομένη με αυτή την

ονομασία μέχρι και σήμερα. Είναι μια αποτελεσματική προσέγγιση στην απόδοση της σκηνής, όπου εστιάζει αποκλειστικά σε αυτό που βλέπει ο παίκτης, τα όρια του οπτικού του πεδίου δηλαδή. Αυτό μαζί σε συνδυασμό με απλή γεωμετρία πίστας, μπορεί να γίνει αρκετά αποδοτικό. Παράδειγμα χρήσης αυτή της τεχνικής αποτελεί το Doom, της id Software, που εκδόθηκε το 1993. Αναπτύχθηκε χρησιμοποιώντας την μηχανή του Wolfenstein. Το Doom επέκτεινε το σύνολο χαρακτηριστικών του για να επιτρέπει έναν πιο οργανικό σχεδιασμών επιπέδων. Αυτό σήμαινε όχι το τέλος των σταθερών χαρτών και την προσθήκη φωτισμού στο περιβάλλον και στοιχείων σε διαφορετικά υψόμετρα. Αυτό είχε σαν αποτέλεσμα το Doom να είναι πιο ατμοσφαιρικό, οι τοποθεσίες του πιο απιστευτη και σε συνδυασμό με την υψηλή του δράση, αποδείχθηκε μεγάλη επιτυχία και αποτέλεσε πηγή έμπνευσης για έναν τεράστιο αριθμό κλώνων.

[Εικόνα 6]: Στιγμιότυπο από το παιχνίδι Doom (1993)

Πολλά από αυτά τα πρώιμα παιχνίδια στηρίζονταν σε πατέντες ώστε να προσομοιώσουν έναν τρισδιάστατο κόσμο. Βαδίζοντας στα ίχνη των arcade, η πέμπτη γενιά κονσολών είχε την δυνατότητα να αναπαραστήσει εξ ολοκλήρου γραφικά τριών διαστάσεων. Έτσι, πλατφόρμες όπως το Playstation και το Nintendo 64 παρατήρησαν άνοδο των πολυγώνων σε ένα οικιακό

περιβάλλον. Τα αληθινά 3d παιχνίδια δεν ήταν πλέον μια καινοτομία, και αντ' αυτού αποτέλεσαν αναπόσπαστο μέρος του mainstream gaming. Ενώ το υλικό της κονσόλας διαχωρίζεται σε διακριτές γενιές, ο ρυθμός ανάπτυξης των υπολογιστών είναι συνεχής και σε συνδυασμό με την επιτυχία του Doom, δεν υπήρχε έλλειψη σε 3D τίτλους.

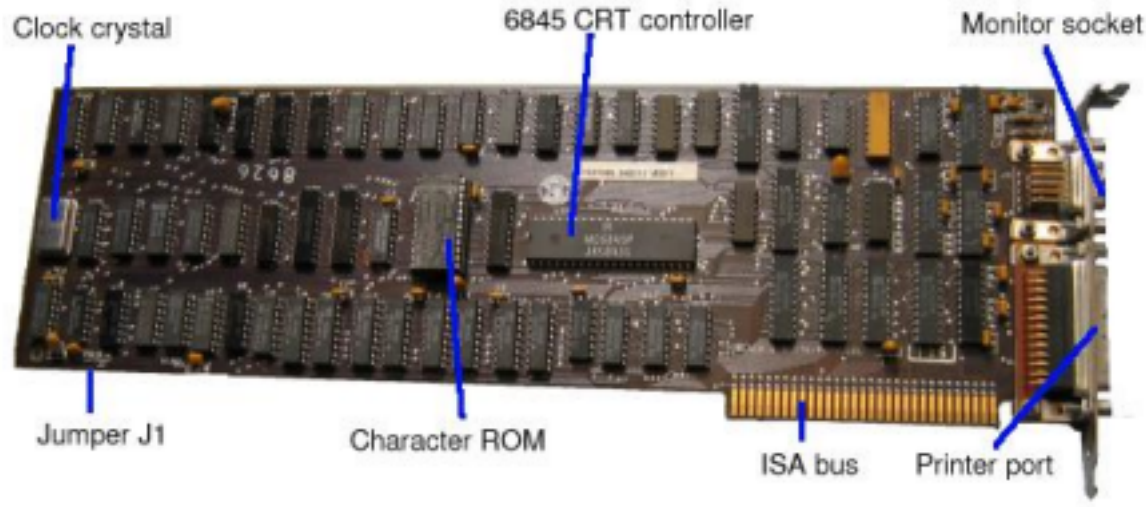
Μια τελευταία υποσημείωση που αξίζει να αναφερθεί, είναι τα voxels. Τα voxels είναι ογκομετρικά εικονοστοιχεία, τα οποία αποτέλεσαν μία εναλλακτική προσέγγιση στα πολύγωνα. Αντί για τριγωνικές όψεις, τα αντικείμενα δημιουργούνται από τρισδιάστατα εικονοστοιχεία, ουσιαστικά, δομικά στοιχεία. Αν και τα τρισδιάστατα γραφικά ήταν το προφανές μέλλον, ωστόσο η παραδοσιακή αρχιτεκτονική τον τότε υπολογιστικών συστημάτων δεν ήταν σχεδιασμένη ώστε να ταιριάξει.

Οι πρώτες κάρτες γραφικών επιτάχυνσης, όπως η Voodoo της 3dfx, ξεκλείδωσαν όρια απόδοσης που είχαν οι επεξεργαστες και παρατηρήθηκε μία άνθηση στις δυνατότητες των παιχνιδιών τριών διαστάσεων. Έτσι υπήρχε σαν αποτέλεσμα, γρηγορότερο ρυθμό ανανέωσης, καλύτερες αναλύσεις, όπως επίσης και γραφικά που δεν είχαν υπάρχει πάλι.

2. ΜΙΑ ΣΥΝΤΟΜΗ ΑΝΑΔΡΟΜΗ ΣΤΟ ΥΛΙΚΟΤΕΧΝΙΚΟ ΚΟΜΜΑΤΙ ΤΩΝ ΚΟΝΣΟΛΩΝ

Οι κάρτες γραφικών χωρίζονται σε κατηγορίες με κριτήρια όπως ο ρυθμός ανανέωσης, οι τύποι χρωμάτων κ.ο.κ. Η IBM είναι αυτή που όρισε τα πρότυπα των πρώτων καρτών για τους προσωπικούς υπολογιστές. Τα πρότυπα αυτά συμβάλλουν στο να βοηθήσουν τους κατασκευαστές οθονών να έχουν μία συμφωνία στις αναλύσεις, τα χρώματα, τους ρυθμούς ανανέωσης. Τα τελευταία χρόνια η Video Electronics Standards Association (VESA) είναι αυτή που ανέλαβε να ορίσει τα νέα πρότυπα μετά την αποχώρηση της IBM, σε συνδυασμό με την ανάγκη για δημιουργία νεότερων και ταχύτερων καρτών.

2.1 Monochrome Display Adapter (MDA)



[Εικόνα 7]: MDA κάρτα γραφικών

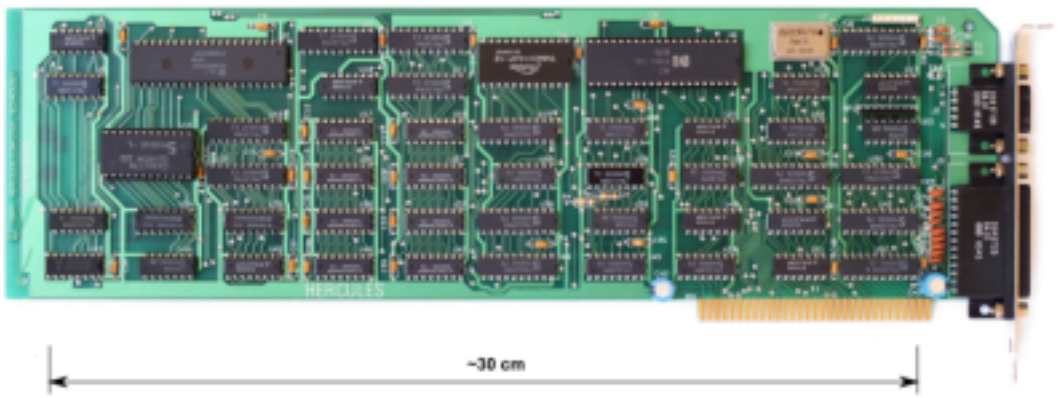
Με την κυκλοφορία του πρώτου υπολογιστή της IBM, έχουμε την εμφάνιση της πρώτης κάρτας γραφικών, τη Monochrome Display Adapter MDA). Η MDA αποτελεί ένα μονόχρωμο πρότυπο κειμένου (πράσινο), όπου επιτρέπει την απεικόνιση κειμένου με 80 (στήλες) x 25 (γραμμές) χαρακτήρες με μνήμη 4KB.

Κάθε χαρακτήρας αποτελείται από έναν πίνακα διαστάσεων: 9 κουκίδες εύρος και 14

κουκίδες ύψος, έχοντας σαν αποτέλεσμα μία ανάλυση που έφτανε τα 720x350 με ρυθμό ανανέωσης της εικόνας 50Hz. Απαρχαιωμένο πρότυπο, ωστόσο καλή λύση για τους πρώτους προσωπικούς υπολογιστές.

[Εικόνα 8]: Στιγμιότυπο από το παιχνίδι Starflight (1986) με την γραφικη απεικόνιση της MDA κάρτας γραφικών.

2.2 Hercules Graphics Card (HGC)

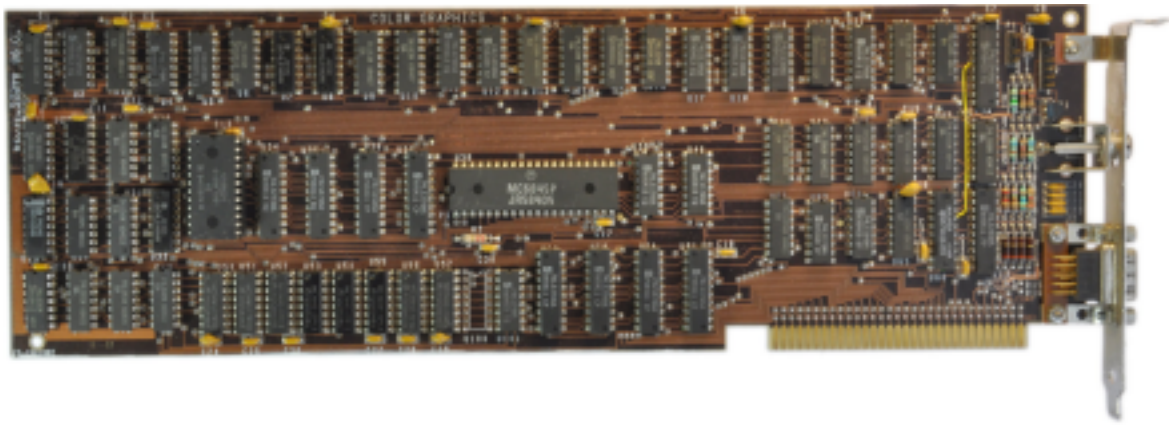


[Εικόνα 9]: Κάρτα γραφικών HCG

Η MDA δεν υποστηρίζει την ύπαρξη των γραφικών με την παραδοσιακή έννοια. Γι αυτό το λόγο, την δεκαετία του ’80, η εταιρία Hercules δημιούργησε μία κάρτα γραφικών συμβατή με την MDA, όπου υποστήριζε μονόχρωμα γραφικά πέρα από την τυπική μορφή κειμένου. Η κάρτα αυτή είχε μεγάλη υποστήριξη, ιδιαίτερα στα μέσα της δεκαετίας του 80, έχοντας σαν αποτέλεσμα, να εμφανιστούν και αρκετοί κλώνοι της.

[Εικόνα 10]: Στιγμιότυπο από το παιχνίδι Space Quest II με την γραφικη απεικόνιση της HCG κάρτας γραφικών

2.3 Color Graphics Adapter (CGA)



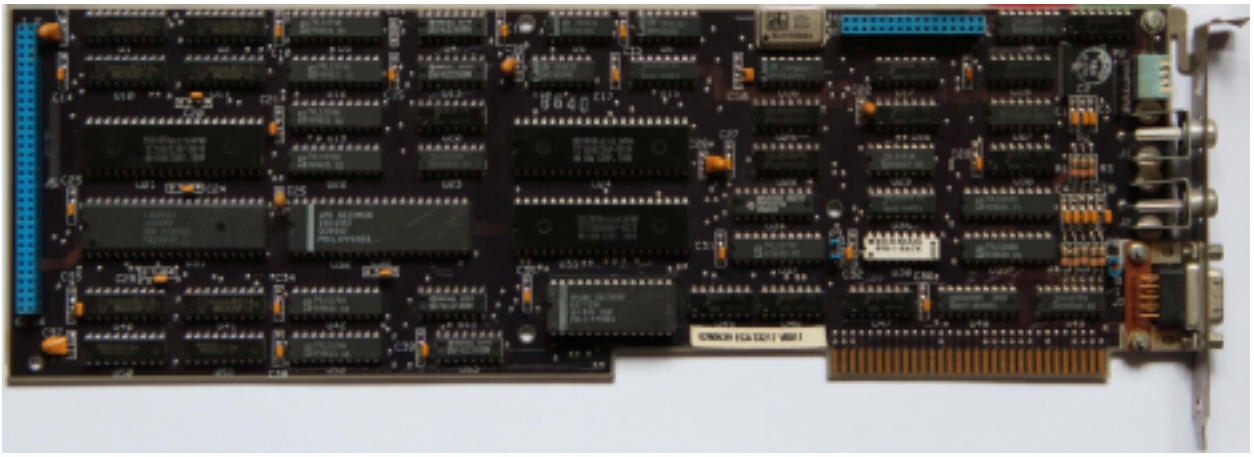
[Εικόνα 11]: Κάρτα γραφικών CGA

Η πρώτη και επικρατέστερη κάρτα γραφικών που υποστήριζε την ύπαρξη έγχρωμων γραφικών για

υπολογιστή ήταν η Color Graphics Adapter (ή αλλιώς CGA) της IBM. Η CGA υποστηρίζει αρκετές διαφορετικές μορφές και η υψηλότερη ανάλυση σε μορφή κειμένου είναι 80x25 χαρακτήρες σε 16 χρώματα. Οι γραφικές μορφές ποικίλουν από τη μονόχρωμη 640x200 (που είναι χειρότερη από αυτή της κάρτας Hercules) στην 160x200 σε 16 χρώματα. Η κάρτα έχει ρυθμό ανανέωσης 60Hz και μνήμη βίντεο 16KB.

[Εικόνα 12]: Στιγμιότυπο από το παιχνίδι Space Wars 2002 με την γραφικη απεικόνιση της CGA κάρτας γραφικών.

2.4. Enhanced Graphics Adapter (EGA)



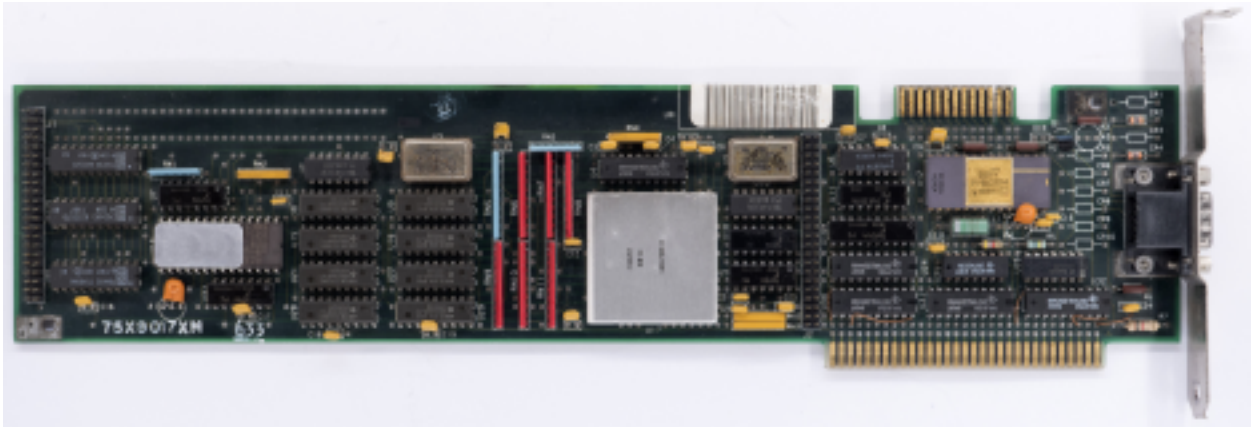
[Εικόνα 13]: Κάρτα γραφικών EGA

Η εξέλιξη της CGA έρχεται με την μορφή της Enhanced Graphics Adapter (EGA). Έχοντας καλύτερες αναλύσεις και μεγαλύτερο φάσμα χρωμάτων από τον προκάτοχό της, η EGA επέτρεπε την απεικόνιση περισσότερων από 16 χρωμάτων (μια παλέτα που φτάνει τα 64 χρώματα) και ανάλυση 640x350 με ρυθμό ανανέωσης τα 60Hz. Ωστόσο παρά τη πρόοδο στο οπτικό κομμάτι, δεν μπορούσε να ειπωθεί το ίδιο και για τις ικανότητές της, μιας και σε σύγκριση με τις σύγχρονες παρόμοιες συσκευές, ήταν ιδιαίτερα φτωχές.

Το επίπεδο γραφικών της EGA είναι το ελάχιστο για τις απαιτήσεις των Windows 3.x και έτσι ίσως κάποια πολύ παλιά συστήματα που χρησιμοποιούν Windows 3.0 ίσως έχουν EGA κάρτα. Δεν υπάρχει πάντως λόγος να διατηρούμε μια EGA κάρτα όταν είναι απαρχαιωμένη και οι VGA κάρτες αρκετά πιο οικονομικές και παρέχουν πολύ μεγαλύτερη απόδοση και συμβατότητα με λογισμικό.

[Εικόνα 14]: Οι πρωταγωνιστές του Alone in the dark με την γραφικη απεικόνιση της EGA κάρτας γραφικών.

2.5 Video Graphics Adapter (VGA)



[Εικόνα 15]: Κάρτα γραφικών VGA

Τέλος, ως διαδεδομένο πρότυπο της IBM, έχουμε τον αντικαταστάτη της EGA και το πιο ευρέως αποδεκτό πρότυπο της IBM, το Video Graphics Array (ή αλλιώς και VGA). Η VGA αποτελεί την βάση των σημερινών καρτών γραφικών που χρησιμοποιείται στους υπολογιστές. Παρουσιάστηκε το μοντέλο PS/2 της IBM και αποτέλεσε ‘πηγή έμπνευσης’ για πολλούς κατασκευαστές. Ακόμα και όταν η IBM έχασε την κυριαρχία της στην αγορά, το πρότυπο του

VGA διατηρήθηκε, εξελίχθηκε και προσαρμόστηκε με πολλούς διαφορετικούς τρόπους. Παρά το γεγονός ότι οι σημερινές τεχνολογίες στις κάρτες γραφικών παρέχουν φάσμα χρωμάτων και δυνατοτήτων πιο εξελιγμένων από αυτά που παρέχει το πρότυπο της VGA, υποστηρίζουν και την VGA, για λόγους συμβατότητας. Άλλωστε αποκαλούνται ‘συμβατές με VGA’ γι αυτό ακριβώς το λόγο. Υποστηρίζει 16 χρώματα σε ανάλυση 640x480 και 256 χρώματα σε ανάλυση 320x200, από μία παλέτα με 262144 χρώματα, μιας και χρησιμοποιούνται 6 bit ώστε να καθοριστεί το κάθε χρώμα, αντί για τα 8 που είναι το τυπικό σήμερα. Παράλληλα, χρησιμοποιεί σήματα που είναι τελείως διαφορετικά από αυτά που είχαν συνηθίσει τα παλαιότερα πρότυπα, δηλαδή αναλογικά αντί για ψηφιακά. Αυτό είχε σαν αποτέλεσμα, μεγαλύτερη ακρίβεια στην απεικόνιση του χρώματος. Παλαιότερες οθόνες που λειτουργούσαν με EGA ή παλαιότερες κάρτες, χρησιμοποιούσαν τη σήμανση που λεγόταν "λογική από transistor σε transistor" ("TTL" ή transistor-transistor logic) και δε λειτουργούν με VGA. Κάποιες οθόνες που δημιουργήθηκαν στα τέλη της δεκαετίας του '80, είχαν ένα διακόπτη επιλογής για να επιτρέπουν την επιλογή αναλογικής ή ψηφιακής εισόδου.

[Εικόνα 16]: Παράδειγμα γραφικής απεικόνισης προτύπου VGA, όπως φαίνεται στο παιχνίδι Mafia III

2.6 VESA Super VGA πρότυπα

[Εικόνα 17]: Κάρτα γραφικών VESA Super VGA

Με την αποχώρηση της IBM από την κυριαρχία στη βιομηχανία των καρτών γραφικών, δημιουργήθηκε ένα χάος ανάμεσα σε εταιρείες και κατασκευαστές, έχοντας σαν αποτέλεσμα την δημιουργία πληθώρας προτύπων που αρκετά δεν είναι και συμβατά μεταξύ τους. Σε αυτό το σημείο της αγοράς, εισήλθε η VESA, όπου συνέβαλε στην εύρεση νέων προτύπων για τις κάρτες γραφικών. Κύριος στόχος τους είναι η παροχή ενός τυποποιημένου προγράμματος εφαρμογών για την διασύνδεση ανάμεσα στο υλικοτεχνικό κομμάτι της κάρτας και το λογισμικό των εφαρμογών. Αυτό επιτρέπει στους κατασκευαστές λογισμικού να γράφουν τον κώδικά τους, έτσι ώστε να λειτουργεί με ένα απλό πρότυπο μοντέλο γραφικών, αντί να χρειάζεται να γράφουν ξεχωριστό κώδικα για κάθε κάρτα που κυκλοφορεί στην αγορά.

3. Η ΕΞΕΛΙΞΗ ΤΩΝ HORROR ΠΑΙΧΝΙΔΙΩΝ

3.1. ΑΝΑΦΟΡΑ ΣΕ ΤΙΤΛΟΥΣ ΠΟΥ ΠΡΟΣΔΙΟΡΙΣΑΝ ΤΟ ΕΙΔΟΣ

3.1.1 Alone in the Dark

[Εικόνα 18]: Το εξώφυλλο του Alone in the Dark για το 3DO (1992)

Ξεκινώντας, το ταξίδι μας στην αναδρομή των τίτλων που προσδιόρισαν το είδος, δεν έχουμε άλλο παρά το 'Alone in the dark', το οποίο κυκλοφόρησε το 1992 για την κονσόλα της Panasonic 3DO και αργότερα σε υπολογιστή. Η ιστορία, επηρεασμένη από το έργο του H.P. Lovecraft, εξελίσσεται σε μία στοιχειωμένη έπαυλη βικτωριανής εποχής την δεκαετία του 20'. Ο πρωταγωνιστής καλείται να επιλύσει γρίφους που θα συναντήσει στην προσπάθεια του να δραπετεύσει, προσπαθώντας παράλληλα, να αντιμετωπίσει τα φαντάσματα που στοιχειώνουν την έπαυλη.

Ο τρόπος με τον οποίο ήταν σχεδιασμένο το παιχνίδι, το έκανε να ξεχωρίζει σε σχέση με τα άλλα της εποχής του. Στα πρώιμα στάδια της ανάπτυξής του περιλαμβάνονταν 3d πλάσματα φτιαγμένα από πολύγωνα. Η ιδέα ήταν ότι ο παίκτης βρίσκεται στο 1920, όπου από την μία είχε μεν ανακαλυφθεί ο ηλεκτρισμός, αλλά όχι τα μέσα που εξασφαλίζουν την εύκολη επιβίωση του παίκτη (κινητό τηλέφωνο, διαδίκτυο). Την ίδια στιγμή, η απλότητα αυτή στο χώρο, βοηθούσε και από προγραμματιστικής άποψης. Αυτό διότι, η μηχανή ανάπτυξης του παιχνιδιού μπορούσε να διαχειριστεί έναν σχετικά μικρό αριθμό πολυγώνων εκείνη την εποχή.

"Έτσι σκέφτηκα ότι χρειαζόμουν φόντα τριών διαστάσεων, αλλά που δεν ήταν φτιαγμένα από πολύγωνα. Είχα την ιδέα να χρησιμοποιήσω φωτογραφίες από μία παλιά έπαυλη" είχε πει ο δημιουργός του παιχνιδιού Frederick Raynal.

Με αυτή την ιδέα, ο σχεδιαστής Didier Chanfray, σχεδίασε ένα ασπρόμαυρο κόνσεπτ, όπου στο κέντρο της βρισκόταν μία μοναχική φιγούρα να κρατά ένα φανάρι στο τέλος ενός τρομακτικού διαδρόμου. Ήταν η πρώτη φορά στην καριέρα του Raynal όπου το πεδίο του παιχνιδιού είχε σκοπό να φοβίσει και να χρειάζεται ιδέες όπου δεν υφίστανται εκείνη την εποχή. Ήξερε ότι χρειαζόταν ένα επαγγελματικό εργαλείο για σχεδιασμό τριών διαστάσεων, αλλά ό,τι υπήρχε, ήταν ακόμα σε πρώιμο στάδιο.

[Εικόνα 19]: Concept art για το Alone in the Dark από τον Dirier Chanfray

3.1.2 Resident Evil

[Εικόνα 20]: Το εξώφυλλο του Resident Evil 1 για το Playstation (1992)

Η σειρά Resident Evil είναι αυτή που εφηύρε τον τίτλο του 'Survival Horror' και αποτελεί το πρώτο παιχνίδι τρόμου με τεράστια εμπορική επιτυχία, όντας παράλληλα πηγή έμπνευσης για πολλά άλλα παιχνίδια που κυκλοφόρησαν στην συνέχεια. Συγκεκριμένα στους τίτλους αρχής του παιχνιδιού εμφανίζεται στο παίκτη το εξής μήνυμα:

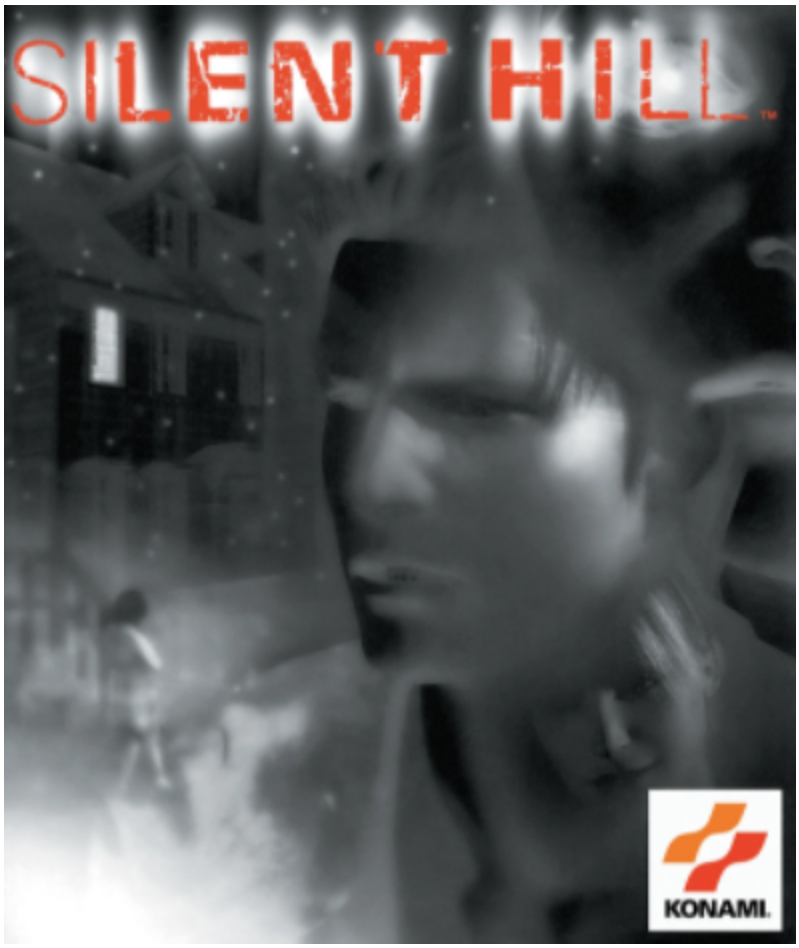
"You have once again entered...The World of Survival Horror...Good Luck".

Ο όρος αυτός, του survival horror, δεν είχε χρησιμοποιηθεί ποτέ ξανά πάλι για να περιγράψει κάποιο παιχνίδι και πολλοί αρθρογράφοι εκείνης της εποχής, για να περιγράψουν το είδος του παιχνιδιού, χρησιμοποιούσαν έννοιες όπως Δράσης/Περιπέτειας (Action/Adventure) και ακόμα Παιχνίδι Ρόλων (Role Playing Game - RPG), μόνο και μόνο επειδή υπάρχει η δυνατότητα

τροποποίησης του αποθηκευτικού χώρου της Jill Valentine. Η ανάπτυξη του παιχνιδιού ξεκίνησε το 1993 ως remake του Sweet Home της Capcom που είχε κυκλοφορήσει το 1989, μιας και τότε, εξαιτίας των τεχνολογικών περιορισμών, δεν είχε υλοποιηθεί με τον τρόπο που το οραματίζονταν οι δημιουργοί του. Το Resident Evil: Biohazard (ή αλλιώς Resident Evil 1, μιας και είναι το πρώτο παιχνίδι της σειράς) εν συντομία RE κυκλοφόρησε το 1996 από την CapCom για το Playstation 1, του Shinji Mikami. Η ιστορία διαδραματίζεται στην Raccoon City, όπου μία ομάδα ειδικών ερευνητών από την ομάδα S.T.A.R.S. (Special Tactics and Rescue Service), ερευνά μία σειρά δολοφονιών.

Χρησιμοποιώντας τις τεχνικές δυσκολίες εκείνης της εποχής (tank controls και ενοχλητικές γωνίες θέασης της κάμερας), το RE καταφέρνει να δημιουργήσει μία αίσθηση τρόμου και άγχους, μιας και κάθε φορά ο παίκτης βιώνει, αυτό που του παρέχεται από την κάμερα, δημιουργώντας μία άγνοια και μία αίσθηση μυστηρίου.

3.1.3 Silent Hill



[Εικόνα 21]: Το εξώφυλλο του παιχνιδιού Silent Hill (1999)

Το 1996 η ιαπωνική εταιρεία παιχνιδιών Konami (γνωστή για τις τότε κυκλοφορίες Metal Gear του Hideo Kojima και Castlevania του Hitoshi Akamatsu) επηρεασμένη από την επιτυχία του RE, επιθυμούσε την έκδοση ενός παιχνιδιού που θα γινόταν επιτυχία και στο Αμερικάνικο κοινό. Η ομάδα που δημιουργήθηκε αποτελούνταν από προγραμματιστές που είχαν αποτύχει στα προηγούμενα project τους και είχαν σκοπό να αποχωρήσουν από την εταιρεία, μιας και δεν τους επιτρεπόταν να υλοποιήσουν τα οράματά τους. Η ομάδα "Silent", όπως ονομάστηκε, εξαιτίας των περιορισμών από την Konami, δεν γνώριζε πως θα προχωρούσε με το εγχείρημα. Μετά από λίγο καιρό η εταιρεία άρχισε να χάνει το ενδιαφέρον της και έτσι τους παρείχε με αυτό τον τρόπο την ευκαιρία στην ομάδα να κινηθούν όπως επιθυμούσαν.

Αυτό που οραματίζονταν να δημιουργήσουν ήταν ένα παιχνίδι, όπου θα είχε αντίκτυπο στα συναισθήματα του παίκτη, μέσα από τον φόβο για το άγνωστο. Με αυτό το σκεπτικό, η πλοκή που είχε το παιχνίδι ήταν ασαφής και περιστασιακά είχε έναν αντιφατικό χαρακτήρα, αφήνοντας το πραγματικό του νόημα στο ‘σκοτάδι’ και επιτρέποντας στον παίκτη να αντικατοπτριστεί σε κομμάτια που δεν υπήρχε εξήγηση. Ο Keichiro Tomaya ήταν ο σκηνοθέτης και αυτός που δημιούργησε την βασική πλοκή του παιχνιδιού, ο Akira Yamaoka ήταν ο συνθέτης, ο Hiroyuki Owaku ήταν αυτός που έφτιαξε τους γρίφους, ενώ ο Takayoshi Sato σχεδίασε τους χαρακτήρες. Ο Tomaya δεν είχε εμπειρία με ταινίες τρόμου, αλλά ήταν λάτρης του David Lynch, ο οποίος αποτέλεσε πηγή έμπνευσης κατά τη διάρκεια της παραγωγής του παιχνιδιού.

Στο παιχνίδι, μέσα από τον ρόλο του Harry Mason, ο παίκτης καλείται να εξερευνήσει τους ομιχλώδεις δρόμους του Silent Hill. Εκεί, μετά από ένα αυτοκινητιστικό δυστύχημα, ο Harry στην προσπάθεια του να βρει την μικρή του κόρη Cheryl, έρχεται αντιμέτωπος με τα μυστηριώδη πλάσματα που καραδοκούν στους δρόμους της πόλης και με μία αίρεση.. Το παιχνίδι κατάφερε να πουλήσει πάνω από 2 εκατομμύρια παγκοσμίως και συμπεριλαμβάνεται στη λίστα με τα καλύτερα παιχνίδια όλων των εποχών.

3.1.4 Amnesia the Dark Descent

[Εικόνα 22]: Εξώφυλλο του Amnesia: The Dark Descent

Το εγχείρημα πέρασε από πολλά κύματα ώστε να είναι το τελικό αποτέλεσμα που έχουμε σήμερα και είχε κάνει μεγάλη επιτυχία το 2010. Άλλαξε δύο ονομασίες (Unknown) και το budget ήταν προβληματικά μικρό, σε σημείο που καθιστούσε την ολοκλήρωσή του σχεδόν αδύνατη και παραλίγο να μην κυκλοφορήσει ποτέ. Παρότι πέρασαν μία εφιαλτική κατάσταση και οι ίδιοι οι developers της frictional games πίστευε σε αυτό που έκανε και κατάφερε να δημιουργήσει ένα παιχνίδι που μιλάνε για αυτό ακόμα και σήμερα.

Η ανάπτυξη του παιγνίου ξεκίνησε στα τέλη του 2007, από μία τετραμελή ομάδα από την Ρωσία, όπου εκείνη την περίοδο ολοκλήρωσε τη δημιουργία του Plague: Panumbria. Επειδή τα προηγούμενα παιχνίδια τους δεν είχαν σημειώσει μεγάλη επιτυχία σε οικονομικό επίπεδο, η ομάδα, δεν είχε την πολυτέλεια να κάνει διαλείμματα μεταξύ των project της. Έπρεπε να σκεφτεί την επόμενη της δουλειά για να ξεκινήσει με το που θα έβγαινε το Plague στα ράφια.

Αν και στην αρχή δεν είχαν βρει κάτι συγκεκριμένο, οι τέσσερίς τους συμφώνησαν στο γεγονός ότι το επόμενο τους παιχνίδι έπρεπε να είναι κάτι απλό, μιας και οι προηγούμενες δουλειές τους αποδείχθηκαν αρκετά απαιτητικές για το περιορισμένο ανθρώπινο δυναμικό τους. Παράλληλα, είχαν βαρεθεί να φτιάχνουν παιχνίδια σαν το Panumbria όπου είχαν πολλούς γρίφους

και ήθελαν να δώσουν προσοχή σε ιδέες που περιλάμβαναν περισσότερους μηχανισμούς gameplay. Αυτό όχι μόνο θα έκανε την ανάπτυξή του ευκολότερη, αλλά ήταν και πιο προσιτό και σε εκδότες. Αυτό διότι είναι πιο εύκολο να εξηγήσεις τους μηχανισμούς του παιχνιδιού, παρά να διαπραγματευτείς με τον άλλον για το ενδιαφέρον της πλοκή της ιστορίας και την δυσκολία των γρίφων που μπορεί να έχει.

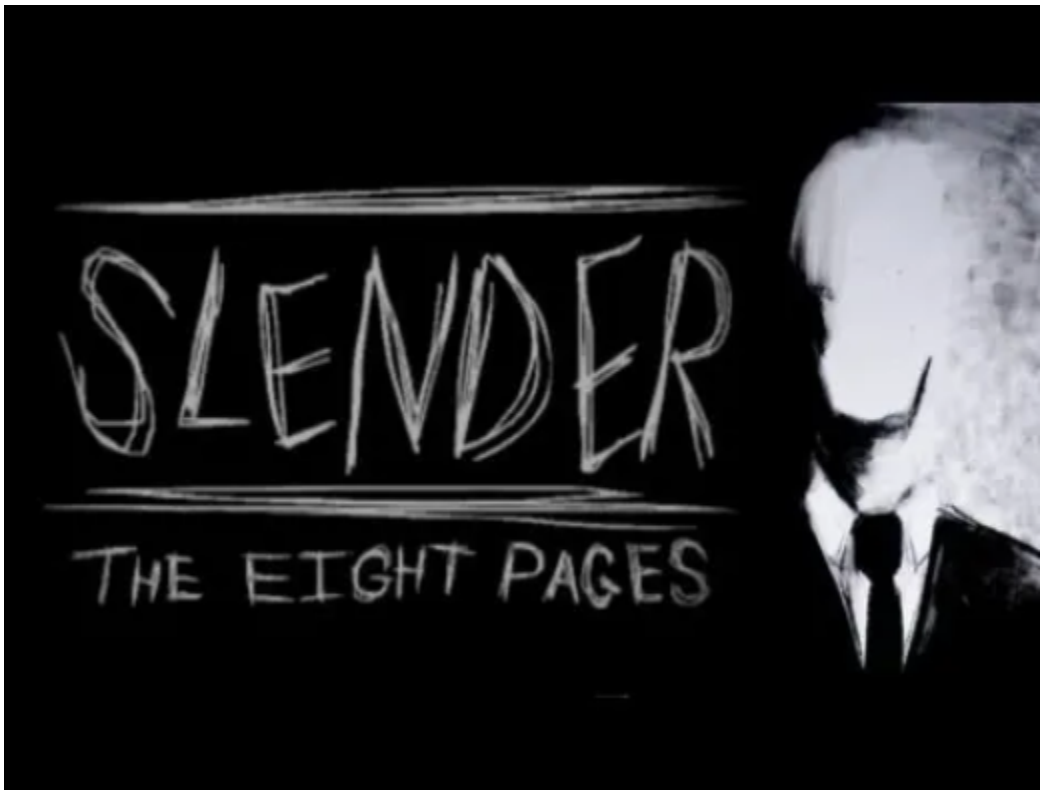
Τον Φεβρουάριο του 2008 όταν κυκλοφόρησε το Black Plague, δεν είχαν καταλήξει σε κάποια συγκεκριμένη ιδέα.Υπήρχε μία βασική τεχνολογία που μπορούσαν να τρέξουν, αλλά δεν υπήρχε κάποιο κόνσεπτ για τον αρτίστα και για τους δημιουργούς, ώστε να δουλέψουν πάνω σε αυτό. Ευτυχώς για αυτούς, το Black Plague πήγε αρκετά καλά και τους πλησίασε η εκδοτική εταιρία Paradox Interactive, ώστε να δουν αν η ομάδα της Frictional ενδιαφερόταν στο να φτιάξει ένα παιχνίδι που θα είχε σχέση με την σειρά παιχνιδιών Penumbria. Η Frictional άδραξε την ευκαιρία, μιας και ένα expansion θα μπορούσε να τους κρατήσει απασχολούμενους, αλλά και να τους δώσει χρόνο στο να βρουν ιδέες για το επόμενο μεγάλο τους project. Γρήγορα αποφάσισαν πως ήθελαν να ανήκει στην κατηγορία του horror, μιας και οι δημιουργοί ένιωθαν πιο άνετα σε αυτό το είδος και τους έδινε την δυνατότητα να χρησιμοποιήσουν όλη τη γνώση που είχαν συγκεντώσει από την σειρά Penumbria.

Αρκετά ενδιαφέρον είναι το γεγονός ότι η ομάδα δεν στόχευε στο να παράγει την επόμενη μεγάλη εμπορική επιτυχία, αλλά έκανε μερικά βήματα πίσω και συγκεντρώθηκε στο να κάνει τα πράγματα πιο απλά. Αυτό διότι οι προηγούμενοι τίτλοι δεν ήταν replayable, εξαιτίας της έλλειψης των gameplay μηχανισμών σε κάθε περιβάλλον. Στα προηγούμενα παιχνίδια η λογική ήταν ότι ο παίκτης θα ασχολείται με έναν γρίφο σε ένα καλά σχεδιασμένο χώρο, θα τον έλυνε, θα μάθαινε λίγο για την ιστορία (exposition) και θα προχωρούσε στον επόμενο χώρο. Εξαιτίας αυτού, οι developers δεν είχαν την δυνατότητα να εξερευνήσουν το χώρο που είχαν πασχίσει να σχεδιάσουν και να προγραμματίσουν. Γι αυτό το λόγο, εγκατέλειψαν τελείως την επιλογή των γρίφων και επικεντρώθηκαν στο να βρουν μηχανισμούς που θα μπορούσαν να χρησιμοποιηθούν με ποικίλους τρόπους.

Η έμπνευση για αυτή την νέα κατεύθυνση ήρθε από πολλές πηγές. Ο σκηνοθέτης του Amnesia , Thomas Grimp, θυμόταν την ανατριχιαστική ατμόσφαιρα των Silent Hill 1 και 2, κάτι που ήθελαν να αναδημιουργήσουν. Επιπλέον, παιχνίδια όπως το Zork έκανε την ομάδα να αναρωτιέται σχετικά με την δημιουργία γρίφων σε ένα 3d περιβάλλον όπου θα είχαν μία λογική

υπόστασή και θα φαίνονταν ότι ανήκουν στο χώρο, σε αντίθεση με τους γρίφους των παιχνιδιών του Resident Evil. Στα αρχικά στάδια της παραγωγής, κοιτούσαν ταινίες όπως το Saw και το Hostel, μιας και ήταν πολύ δημοφιλείς εκείνη την περίοδο. Η αργή και μινιμαλιστική πρόοδος που έχει το παιχνίδι στην τελική του μορφή, είναι επηρεασμένη από ταινίες όπως το The Haunting. Τρομαχτικά παιχνίδια όπως το Call of Cthulhu: Dark Corners of the Earth και γενικότερα ο H.P. Lovecraft επηρέασαν την γενικότερη ατμόσφαιρα, την αίσθηση και τον σχεδιασμό του παιχνιδιού.

3.1.5 Slender: The 8 pages



[Εικόνα 23]: Slender: the eight pages

Ο χαρακτήρας του Slenderman πλάστηκε για τις ανάγκες ενός διαγωνισμού photoshop το 2009 από τον Erik Nudsen (ή αλλιώς γνωστός από το όνομα που χρησιμοποιούσε Victor Surge). Μαζί με το εικονικό / παρουσιαστικό κομμάτι του χαρακτήρα (υπεράνθρωπα ψηλός, χλωμή-σχεδόν-γκρίζα επιδερμίδα, τα χαρακτηριστικά του προσώπου να απουσιάζουν και αφύσικα μακριά άκρα), του έδωσε επίσης ζωή δημιουργώντας του ένα παρελθόν. Αυτό σε συνδυασμό μαζί με την ανατριχιαστική ατμόσφαιρα, αλλά και τη χαμηλή ποιότητα των φωτογραφιών, τον έκαναν να μοιάζει σχεδόν αληθινός. Ο Slenderman βρίσκεται πάντα στο βάθος των φωτογραφιών , με την ποιότητα αρκετά χαμηλή, ώστε να μην μπορεί να καταλάβει ο θεατής, τι είναι αυτό που αντικρίζει. Πηγή έμπνευσης αποτέλεσαν οι ιστορίες του HP. Lovecraft και ο κοσμικός τρόμος, το οποίο μπορεί εύκολα να γίνει αντιληπτό από την συμπερίληψη πλοκαμιών στο παρουσιαστικό του

χαρακτήρα, αλλά και της pure - evil φύσης του.

Στις 26 Ιουνίου του 2012, ο Mark Hadley και η ομαδα της Parsec Productions κυκλοφορούν το περίφημο παιχνίδι ‘Slender: The eight pages’. Πολύ απλό και εμφανισιακά και χειριστικά, χωρίς καν κεντρικό μενού, μόνο το σκοτάδι και ένα δάσος. Οι ήσυχοι ατμοσφαιρικοί ήχοι να παίζουν στο σκοτάδι, σε συνδυασμό με το αδύναμο φως του φακού που κουβαλά ο παίκτης και έναν μόνο ένα μήνυμα “Σύλλεξε και τις 8 σελίδες ”, είναι αυτά που άλλαξαν μια και καλή τον κόσμο των ανεξάρτητων παιχνιδιών για πάντα. Ήταν εύκολα προσβάσιμο, μιας είχε γραφικά χαμηλής ποιότητας, πράγμα που σήμαινε ότι ένας υπολογιστής μέσων χαρακτηρικών μπορούσε να το σηκώσει. Ήταν εύκολο περιεχόμενο για content creators στο youtube, εξαιτίας αυτού κάνοντας τα Let’s play βιντεάκια δημοφιλή και εκτοξεύοντας την καριέρα γνωστών youtubers όπως είναι ο Markpliar και ο Pewdiepie. Το μόνο που χρειαζόταν ήταν μία κάμερα να σημαδεύει το πρόσωπό τους και κάθε φορά που πετάγονταν ο τρομακτικός άντρας στην οθόνη να ουρλιάζουν.

Αργότερα προστέθηκε και ένα κεντρικό μενού με συνδέσμους να παραπέμπουν σε περισσότερο περιεχόμενο και προστέθηκαν άλλες παραλλαγές του παιχνιδιού, όπως είναι το Daylight Mode, όπου παίζεις το πρωί, με τον πυρήνα του παιχνιδιού να μην αλλάζει. Ο παίκτης βρίσκεται σε ένα δάσος με οκτώ σημεία αναφοράς. Σε κάθε σημείο υπάρχει και από μία σελίδα που πρέπει να συλλέξει με σκοπό να κερδίσει το παιχνίδι. Με το που σηκώνει την πρώτη σελίδα ξεκινά να τον κυνηγάει ο Slenderman, που προσπαθεί να τον αποτρέψει από αυτό τον σκοπό και όσο προχωρά το παιχνίδι ο Slenderman γίνεται πιο επιθετικός, μπορεί να πεταχτεί αναπάντεχα, το φως του φακού γίνεται όλο και πιο αχνό, ενώ αν πλησιάσεις αρκετά κοντά ή τον κοιτάξεις για ένα χρονικό διάστημα, πεθαίνεις.

3.1.6 Outlast

Το 2008, οι game developers Philippe Morun και David Chateauneuf έχοντας δουλέψει σε μεγάλα εγχειρήματα, όπως το Uncharted: Drake's Fortune, Prince of Persia: Sands of Time, αλλά και στο πρώτο Assassin's Creed, δημιούργησαν την ομάδα Red Barrels. Επιθυμούσαν να μπουν δυναμικά στην βιομηχανία των παιχνιδιών, μιας και οι δυό τους είχαν αποκτήσει αρκετή εμπειρία από τις προηγούμενες δουλειές τους και είχαν αρκετές ιδέες, αλλά και όρεξη. Η πλειοψηφία αυτή των ιδεών δεν ήταν αρκετά καλές ή εύκολα υλοποιήσιμες, συνεπώς εγκαταλείφθηκαν αρκετά γρήγορα. Στο τέλος κατέληξαν να συμφωνήσουν στο να πειραματιστούν στα χωράφια παιγνίων τρόμου, όπου δεν υπήρχε και αρκετός ανταγωνισμός εκείνη την περίοδο στο εμπόριο. Οι προσδοκίες δεν ήταν υψηλές, μιας και το ανθρώπινο δυναμικό ήταν μικρό και το χρηματικό ποσό, που είχαν διαθέσιμο να δουλέψουν, χαμηλό.

[Εικόνα 24]: Το εξώφυλλο του παιχνιδιού outlast (2013)

Για την σύλληψη της βασικής ιδέας, οι developers κλήθηκαν να απαντήσουν στις εξής ερωτήσεις:

1) Ποιοι είναι οι κεντρικοί μηχανισμοί;

2) Ποιος ειναι ο πρωταγωνιστής;

3) Προς τα που κατευθύνεται το παιχνίδι και γιατί ο πρωταγωνιστής χρησιμοποιεί νυχτερινή όραση;

4) Ποιος ο στόχος;

Επειδή η ομάδα της Red Barrels αποτελούνταν από 10 άτομα, οι απαντήσεις σε αυτές τις ερωτήσεις έπρεπε να είναι ρεαλιστικές και υλοποιήσιμες για τις δυνατότητές τους. Αρχικά, κλήθηκαν να επιλέξουν ανάμεσα:

1) στους επιθετικούς μηχανισμούς του Resident Evil (με τη χρήση λιγότερων πυρομαχικών) και

2) στους αμυντικούς του Amnesia: The Dark Descent.

Νικητής αυτού του διλήμματος, βγήκε το Amnesia, πράγμα που σήμαινε ότι οι developers είχαν την ευκαιρία να αναπτύξουν μία πιο εστιασμένη εμπειρία.

Για την χρήση της νυχτερινής όρασης, έπρεπε να βρεθεί και ο κατάλληλος πρωταγωνιστής. Στα αρχικά στάδια σύλληψης της ιδέας, ο πρωταγωνιστής ήταν ένα μέλος ειδικής ομάδας αστυνομικών με εξοπλισμό νυχτερινής όρασης, αλλά γρήγορα απορρίφθηκε, μιας και δεν θα υπήρχε μάχη στο παιχνίδι (combat gameplay mechanics). Εκείνη την περίοδο, στον τομέα του κινηματογράφου, ήταν αρκετά δημοφιλείς οι ταινίες με υλικό που είναι γυρισμένο από βιντεοκάμερα και έτσι η ομάδα κινήθηκε προς αυτή την κατεύθυνση (μια βιντεοκάμερα, έχει λειτουργία νυχτερινής όρασης).

Επόμενο κρίσιμο κομμάτι που έπρεπε να επιλυθεί, ήταν η εύρεση της τοποθεσίας που θα λάμβαναν χώρα τα γεγονότα, από την οποία θα έπρεπε να δραπετεύσει ο παίκτης, αλλά να είναι υλοποιήσιμη για την ομάδα. ‘Η δημιουργία έρχεται πάντα με κάποιους περιορισμούς’ αναφέρει ο ίδιος ο Philippe Morun. Μετά από μία αντίστοιχη λίστα ιδεών με αυτή που είχε η ομαδα στην αρχή, σαν καλύτερη επιλογή κρίθηκε το ψυχιατρικό άσυλο. Όπως και με τις βιντεοκάμερες, έτσι και με το άσυλο, έχει χρησιμοποιηθεί αρκετά στον κινηματογράφο, αλλά όχι και στα παιχνίδια, σε ένα ρεαλιστικό περιβάλλον.

”We felt an asylum offered an opportunity to create really unique and compelling characters that the player would meet along his journey. “

Το τελευταίο πράγμα που ήθελαν να πετύχουν, ήταν οι ασθενείς που υπάρχουν στο άσυλο να είναι τρομακτικοί. Το τι κάνει κάποιον, όμως, τρομακτικό είναι καθαρά υποκειμενικό. Για κάποιους είναι το οπτικό κομμάτι (το πως μοιάζει εξωτερικά), για άλλους είναι η ψυχοσύνθεσή του, σε πολλούς οι πράξεις κ.ο.κ. Αποφάσισαν να χρησιμοποιήσουν το ψυχολογικό προφίλ ψυχασθενών ανθρώπων που είχαν διαπράξει εγκλήματα. Για να φανεί η προσωπικότητα κάθε χαρακτήρων, ήταν αναγκαία η ανάπτυξη πειστικών και όχι γελοίων διαλόγων. Σαν να βρίσκεται μαζί με τον Hannibal Lecter σε ένα κλειστό περιβάλλον.

Όμως, το φυσιολογικό παρουσιαστικό των ασθενών δεν θα τρόμαζε τον παίκτη. Γι αυτό το λόγο ξεκίνησαν να κάνουν έρευνα πάνω στο πρόγραμμα MKUltra που διεξάγεται μέχρι την δεκαετία του ’70 σε φυλακές και άσυλα και το άρθρο του Alan Turing πάνω στην Μορφογένεση. (Σύμφωνα με αυτό, με απλή εξήγηση, τα κύτταρα όταν διαχωρίζονται διαφοροποιούνται και πως από τα ίδια κύτταρα δημιουργούνται τα διαφορετικά όργανα, είδη κλπ)

Η ανάπτυξη πραγματοποιήθηκε στη μηχανή Unreal Engine. Πρώτα δημιουργήθηκε όλος ο χάρτης του ψυχιατρικού ασύλου με πολύγωνα και έπειτα σιγά σιγά, άρχισαν να προστίθενται τα assets, ο φωτισμός, οι ήχοι, τα εφέ, με στόχο τη δημιουργία μιας κινηματογραφικής ατμόσφαιρας.

3.1.7 Evil Within

[Εικόνα 25]: Φωτογραφία προώθησης του παιχνιδιού Evil Within 2, η συνέχεια του Evil Within.

Το 2010, ο Shinji Mikami αποχώρησε από την Capcom και μαζί με άλλους 12 developers δημιούργησαν την Tango Gameworks η οποία μετά από μερικούς μήνες αγοράστηκε από την Bethesda. Τότε ξεκίνησε και η παραγωγή του παιχνιδιού ‘The Evil Within’, έχοντας τότε τον τίτλο ‘Project Zwei’, με τον Mikami στη θέση του σκηνοθέτη. Εκείνη την περίοδο τα παιχνίδια επιβίωσης τρόμου, είχαν αρχίσει να γίνονται προβλεπόμενα και για αυτό τον λόγο, ο Mikami υποσχόταν ότι το The Evil Within, θα επαναπροσδιόριζε την τότε βιομηχανία των παιχνιδιών τρόμου. Αυτό μέσω των κλειστοφοβικών χώρων, της έλλειψης πυρομαχικών και παρουσιάζοντας αόρατους εχθρούς, θα ανάγκαζε τον παίκτη να υιοθετήσει μία πιο επιφυλακτική προσέγγιση όσο θα εξελίσσονταν τα γεγονότα του παιχνιδιού.

Ο Shinji Mikami από την αρχή ήθελε τα γεγονότα του παιχνιδιού να λαμβάνουν χώρα στο μυαλό κάποιου ή σε κάποιο περιβάλλον σαν το matrix, μιας και θα μπορούσε να δημιουργήσει πιο ευφάνταστες καταστάσεις, αλλά με τον ρεαλισμό να αποτελεί καταλυτικό παράγοντα.

3.1.8 Resident Evil 7: Biohazard

[Εικόνα 26]: Προωθητική εικόνα του παιχνιδιού Resident Evil 7: Biohazard (2017) στην πλατφόρμα Steam

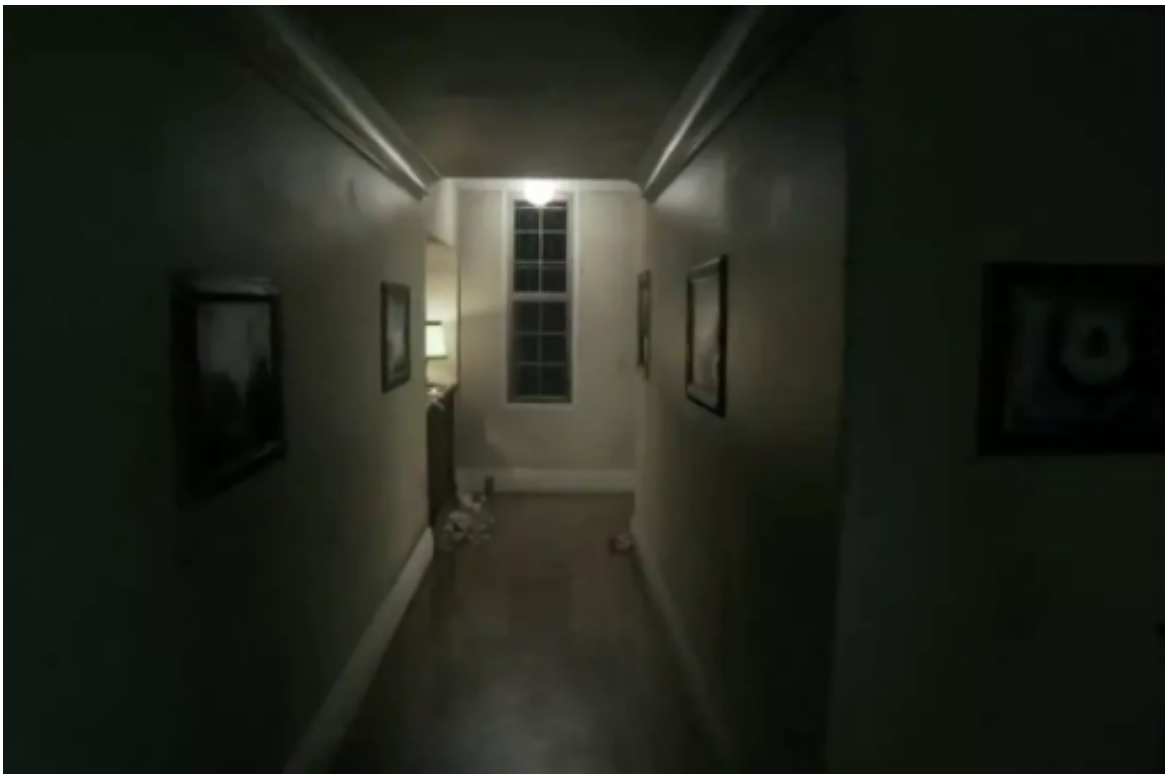
Με το resident evil να γίνεται τρομερή επιτυχία παγκοσμίως με την κυκλοφορία του, σύντομα, όπως ήταν λογικό, ακολούθησαν και άλλα παιχνίδια στην σειρά. Το ένα να διαδέχεται το άλλο συνεχίζουν να εξιστορούν τις περιπέτειες της Jill Valentine και του Chris Redfield από το πρώτο παιχνίδι, ενώ άλλα είχαν στο επίκεντρο, νέους χαρακτήρες, όπως είναι η Claire Redfield, αδερφή του Chris και τον αστυνομικό Leon Kennedy, προσδίδοντας στο franchise πλούσιο περιεχόμενο και βάθος. Τα Resident Evil 2, Resident Evil 3: Nemesis και Resident Evil 4 έμοιαζαν αρκετά στον προκάτοχό τους: ανατριχιαστική ατμόσφαιρα, τέρατα να καραδοκούν σε κάθε μεριά, περίπλοκοι γρίφοι και οικονομία στα πυρομαχικά, με στόχο ο παίκτης να νιώθει ευάλωτος και ότι απειλείται.

Τα επόμενα δύο απομακρύνθηκαν από αυτή την προσέγγιση, με την Capcom να επιλέγει να επενδύσει περισσότερο σε gameplay παρότι σε ατμόσφαιρα. Παρότι το Resident Evil 6, είναι το τέταρτο πιο κερδοφόρο παιχνίδι της εταιρείας (πουλώντας συνολικά 7,7 εκατομμύρια αντίτυπα μέχρι το Δεκέμβρη του 2020) πρόσθεσαν έναν πιο action χαρακτήρα στη σειρά, κάτι που αποξένωσε τους μακροχρόνιους φαν της σειράς. Παρακάτω φαίνονται οι βαθμολογίες του Metacritic και οι βαθμολογίες του κοινού από την μέρα που κυκλοφόρησαν οι τίτλοι μέχρι και το Δεκέμβρη του 2021, όπου συλλέχθηκαν και τα δεδομένα για την δημιουργία του.

Τίτλος	Metacritic	Βαθμολογία Κοινού
Resident Evil (PS1)	91/100	9.1/10
Resident Evil 2 (PS2)	89/100	9.2/10
Resident Evil 3: Nemesis (Dreamcast)	79/100	9/10
Resident Evil 4 (Game Cube/ PS2)	96/100	9.2/10
Resident Evil 5 (PS4)	69/100	7.3/10
Resident Evil 6 (PS4)	60/100	6.6
Resident Evil 7 (PS4)	86/100	8/10
Resident Evil 8 (PS5)	84/100	8.5/10

Οι μέτριες και διχασμένες κριτικές για τον τίτλο, είχαν σαν αποτέλεσμα η Capcom να ακυρώσει το sequel που είχε τότε στα σκάρια (action-based). Τον Ιανουάριο του 2014, ο CEO της εταιρίας προσέλαβε τον δημιουργό που είχε ασχοληθεί με το Resident Evil 5

3.2 Η ΙΣΤΟΡΙΑ ΤΩΝ LOOP HORROR ΠΑΙΧΝΙΔΙΩΝ



[Εικόνα 27]: Στιγμιότυπο από το παιχνίδι P.T. που απεικονίζει την αρχή του διαδρόμου

Το πρώτο παιχνίδι horror που όλο το gameplay λάμβανε μέρος σε έναν επαναλαμβανόμενο χώρο, είναι το P.T.. Τα αρχικά του P.T. σημαίνουν Playable Teaser και ήταν το demo για το Silent Hills της εταιρίας Konami, η τελευταία προσθήκη στη συλλογή παιχνιδιών Silent Hill. Αποτελούσε έναν πρωτότυπο τρόπο προώθησης του παιχνιδιού, καθώς ο παίκτης βίωνε την εμπειρία με διαδραστικό τρόπο, αντί να δει κάποιο trailer ή και στιγμιότυπα από το ίδιο το παιχνίδι. Κυκλοφόρησε στις 12 Αυγούστου του 2014 για την πλατφόρμα PS4 και ήταν διαθέσιμο για δωρεάν εγκατάσταση.

[Εικόνα 28]: Το φάντασμα που στοιχειώνει το σπίτι, η Lisa

Ήταν σκηνοθετημένο και σχεδιασμένο από τον Hideo Kojima με τη βοήθεια του μεξικανού σκηνοθέτη Guillermo del Toro, γνωστού για τις πρωτότυπες, ατμοσφαιρικές ταινίες τρόμου (ο Λαβύρινθος του Πάνα, Πορφυρός Λόφος, Η μορφή του νερού). Το εγχείρημα ακυρώθηκε, όμως, όταν ένα χρόνο αργότερα, ο Hideo Kojima ήρθε σε σύγκρουση με την Konami, λόγω δημιουργικών διαφορών πάνω στο project και το παιχνίδι απομακρύνθηκε άμεσα από το playstation store.

Σε αντίθεση με τα προηγούμενα παιχνίδια της σειράς, όπου ο παίκτης βίωνε την ιστορία σε τρίτο πρόσωπο, το P.T. χρησιμοποιεί μία πρωτοπρόσωπη οπτική. Είναι ένα διαδραστικό teaser, το οποίο επικεντρώνεται σε έναν άγνωστο χαρακτήρα, που τον ελέγχει ο παίκτης, ο οποίος ξυπνά σε ένα ένα στοιχειωμένο σπίτι και καλείται να το εξερευνήσει.

Τα γεγονότα του παιχνιδιού λαμβάνουν χώρο, ως επί το πλείστον, σε ένα διάδρομο σε σχήμα L, το μπάνιο και μία σκάλα που οδηγεί στο δωμάτιο που ξεκινά ο φαύλος κύκλος. Ο διάδρομος μπορεί να είναι είτε ο ίδιος ή κάτι να έχει αλλάξει σε αυτόν (λ.χ. διαφορετικός φωτισμός) ή να έχει πρόσβαση σε διαφορετικά σημεία του σπιτιού. Οι μόνες ενέργειες του παίκτη είναι να επικεντρώνεται σε ένα αντικείμενα και να περπατά. Με κάθε βήμα που κάνει ο παίκτης, με κάθε λούπα που περνά, απομακρύνεται όλο και πιο πολύ από την πραγματικότητα και

εισέρχεται στο κόσμο του σουρεαλισμού. Ο διάδρομος που διανύει, έχει τη μορφή των σκαλοπατιών Penrose, δεν ξέρεις που ξεκινούν και που τελειώνουν, οι νευρικοί ήχοι τον έχουν πάντα σε εγρήγορση, η εικόνα του φαναριού που κρέμεται από το ταβάνι και κουνιέται τον υπνωτίζει. Ένα καταληκτικός παράγοντας που το p.t. είναι τόσο ανατριχιαστικό, είναι στο γεγονός ότι δεν βασίζεται στην έκπληξη για να τρομάξει τον παίκτη.

Για να γίνει πρόοδος στο παιχνίδι, ο παίκτης πρέπει να λύσει γρίφους και να αποφύγει να τον ‘‘πιάσει’’ το φάντασμα που υπάρχει στο σπίτι, η Λίσα. Μόλις η λούπα ολοκληρωθεί με επιτυχία, ο διάδρομος αλλάζει εμφανισιακά, ενώ στην αντίθετη περίπτωση που τον πιάσει το φάντασμα, μεταφέρεται από εκεί που ξεκίνησε. Μία άλλη διαφορά του συγκεκριμένου παιχνιδιού σε σχέση με τα προηγούμενα Silent Hill, ο παίκτης δεν έχει κάποιον αμυντικό μηχανισμό ενάντια στην Λίσα. Αυτό συμβαίνει επειδή, όπως ανέφερε και ο Hideo Kojima: “Η κατοχή ενός όπλου μειώνει την αίσθηση του φόβου και ότι δεν μπορείς να κάνεις τίποτα”.

Παρά τη σύντομη διάρκεια ζωής του, το P.T. τάραξε τα νερά της τότε βιομηχανίας παιχνιδιού τρόμου και αποτελεί πηγή έμπνευσης για πολλά περισσότερα στην ίδια λογική με αυτό. Παράλληλα, έλαβε πολύ καλές κριτικές για τα φωτορεαλιστικά γραφικά του και την τρομακτική του ατμόσφαιρα.

3.2.1 LAYERS OF FEAR

[Εικόνα 29]: Το εξώφυλλο του παιχνιδιού Layers of fear (2016)

Το Layers of Fear είναι ένα ψυχολογικό παιχνίδι που κυκλοφόρησε από την Bloober Team στις 16 Φεβρουαρίου το 2016, εμφανώς επηρεασμένο από το βιβλίο του Όσκαρ Ουάιλντ 'Το πορτραίτο του Ντόριαν Γκρέι'. Εδώ, ο παίκτης παίρνει το ρόλο ενός ψυχολογικά διαταραγμένου ζωγράφου ο οποίος προσπαθεί να ολοκληρώσει το μεγαλύτερο αριστούργημα της καριέρας του (**magnum opus**). Καλείται να εξερευνήσει μία έπαυλη βικτωριανής εποχής, ενώ ταυτόχρονα ανακαλύπτει μυστικά και την ιστορία του ζωγράφου (μέσα από γράμματα και διαλόγους που λαμβάνουν χώρα στις αναμνήσεις του πρωταγωνιστή).

Το παιχνίδι είναι χωρισμένο σε έξι κεφάλαια, στα οποία ο παίκτης συλλέγει αντικείμενα τα οποία παίζουν καταλυτικό ρόλο στην εξέλιξη του παιχνιδιού, καθώς αυτό έχει τρία διαφορετικά τέλη που αφορούν την τύχη του ζωγράφου (δύο άσχημα για εκείνον και ένα στο οποίο λαμβάνει την αναγνώριση που τόσο πολύ επιθυμούσε να πετύχει).

Η ανάπτυξη του παιχνιδιού πραγματοποιήθηκε σε περιβάλλον Unity.

4. Η ΤΕΧΝΗ ΤΗΣ ΑΦΗΓΗΣΗΣ ΜΙΑΣ ΙΣΤΟΡΙΑΣ ΤΡΟΜΟΥ

Ο τρόμος είναι ένα από τα πιο δημοφιλή μέσα αφήγησης, καθώς με αυτό τον τρόπο στον αναγνώστη (ή και στο κοινό) δημιουργούνται έντονα συναισθήματα φόβου, απέχθειας και έκπληξης. Την ίδια στιγμη συγγραφείς όπως ο Δάντης και ο Όμηρος χρησιμοποιούσαν τον τρόμο, ώστε να εμβαθύνουν και να κάνουν πιο σκοτεινές τις ιστορίες τους. Οι ρίζες του ξεκινούν από την αρχαιότητα, είχαν τη μορφή ιστοριών από τη μυθολογία και περιλάμβαναν θεματολογία όπως ο θάνατος, η ζωή μετά από αυτόν, πλάσματα της νύχτας και το κακό που μπορεί να κρύβεται σε έναν άνθρωπο. Η Ευρωπαϊκή ιστορία τρόμου ξεκινά από την αρχαία Ελλάδα και από την Ρωμαϊκή αυτοκρατορία.

Ιστορίες όπως η μεταμόρφωση του Λυκάονος, βασιλιά της Αρκαδίας, σε λύκο εξαιτίας της ασεβειας του προς την κυριαρχία του Δία, και οι Βάκχες οι Ευριπίδη, είναι κάποια από τα παραδείγματα.

[Εικόνα 30]: Ο Δια μεταμορφώνει τον Λυκαων της Αρκαδίας σε Λύκο, γκραβούρα του Hendrik Goltzius

4.1 ΤΑ ΕΙΔΗ ΤΟΥ ΤΡΟΜΟΥ

4.1.1 CLASSICAL HORROR

Στην πρώτη κατηγορία έχουμε τον ‘κλασικό τρόμο’ στον οποίο περιλαμβάνονται πλάσματα της κλασικής λογοτεχνίας όπως ο Δράκουλας, το Φάντασμα της Όπερας κλπ ή της κλασικής κουλτούρας τρόμου του σινεμά, όπως είναι ο Jason και ο Freddy Krueger.

Συχνά ο κεντρικός ανταγωνιστής είναι ένα τέρας, ή μία απειλή που αντικατοπτρίζει τους φόβους της κοινωνίας την εκάστοτε εποχή. Το άτομο ή το πλάσμα το καλούν ως “άλλο”, ένας όρος που αναφέρεται σε κάποιον που τον φοβούνται επειδή είναι διαφορετικός.

4.1.2 COSMIC HORROR (H.P. LoveCraft)

Εδώ περιλαμβάνονται ιδέες τόσο ξένες/εξωγήινες για εμάς που δεν μπορούμε να τις κατανοήσουμε πλήρως. Ιδέες όπου οι αρχές τους είναι άμορφες, έχουν απεριόριστη έκταση ή υπεράνθρωπη δύναμη και αντικατοπτρίζουν την σκοτεινή πλευρά της ανθρωπότητας, τους περιορισμούς της γλώσσας και την των ορίων της ανθρώπινης ύπαρξης.

4.1.3 ΨΥΧΟΛΟΓΙΚΟΣ ΤΡΟΜΟΣ

Που εξερευνά στην ουσία τις πιο σκοτεινές γωνίες του ανθρώπινου νου και ερχόμαστε σε θέση να αντιμετωπίσουμε τις ανασφάλειές μας.

4.2 ΤΑ ΚΥΡΙΑ ΣΥΣΤΑΤΙΚΑ ΤΟΥ ΤΡΟΜΟΥ

4.2.1.ΈΚΠΛΗΞΗ

[Εικόνα 31]: Στιγμιότυπο από το παιχνίδι Silent Hill.

Το στοιχείο της έκπληξης, ή καλύτερα του ‘σοκ’, χρησιμοποιείται συνήθως σε τεχνικές ‘jumpscare’. Με τον όρο ‘jumpscare’ αναφερόμαστε στο σκηνοθετικό τέχνασμα που εκμεταλλεύεται το στοιχείο της έκπληξης, σε συνδυασμό με μία ανατριχιαστική εικόνα και έναν έντονο δυνατό ήχο, με σκοπό να τρομάξει τον θεατή.

Η πιο εύκολη μέθοδος στην υλοποίησή της, αν και φτηνή και πολυχρησιμοποιημένη. Ο λόγος επειδή το αντίκτυπο που αφήνει δεν διαρκεί πολύ, μόλις μερικά δευτερόλεπτα και μετά ο θεατής επανέρχεται πάλι στην πραγματικότητα, στο σημείο που ήταν. Ενεργοποιείται το αντανακλαστικό του "ή θα πολεμήσεις ή θα τρέξεις μακριά" που βρίσκεται μέσα σε κάθε άνθρωπο. Το ένστικτο της επιβίωσης. Δεν χρησιμοποιείται η φαντασία σου εναντίον του, παρέχεται κάτι έτοιμο και γι αυτό το λόγο ο εγκέφαλος μπορεί να το αντιμετωπίσει πολύ πιο εύκολα. Στην αντίθετη περίπτωση, όταν δεν του ‘δίνεται στο πιάτο' αυτό που πρέπει να αντιμετωπίσει, ο εγκέφαλος είναι αυτός που συμπληρώνει τα κενά και αφήνεται στο να αναρωτιέται τι μπορεί να συμβαίνει και για το τι μπορεί να κρύβεται πίσω από αυτό που βιώνει.

Αυτό μπορεί να παρατηρηθεί στο πρώτο παιχνίδι της σειράς Silent Hill. Συγκεκριμένα, με την ατμόσφαιρα που σου δημιουργεί (οι ήχοι που περιβάλλουν τον παίκτη, η ομίχλη) ο παίκτης είναι ανίδεος για το τι είναι αυτό που μπορεί να τον περιμένει, ή τι είναι αυτό που προκαλεί αυτά που ακούει. Αυτό έχει σαν αποτέλεσμα να φαντάζεται τα χειρότερα, να αναρωτιέται και να διστάζει να προχωρήσει. Στην αντίθετη περίπτωση, έχουμε το outlast, όπου ο παίκτης παίρνει το ρόλο ενός δημοσιογράφου σε ένα ψυχιατρικό άσυλο και καλείται να μαζέψει στοιχεία σχετικά με τα πειράματα που γίνονται εκεί, προσπαθώντας να αποφύγει ταυτόχρονα κάποιους από τους ψυχικά ασθενείς. Είναι αβοήθητος και το μόνο που έχει την δυνατότητα να κάνει στην ουσία, είναι να τρέξει και να κρυφτεί. Ο εχθρός είναι μπροστά σου, έχει υλική υπόσταση και γι αυτό το λόγο θες να τρέξεις μακριά ή να τον πολεμήσεις. Ακριβώς, επειδή όμως είσαι αβοήθητος, υπάρχει άλλη εναλλακτική, ώστε να επιλέξεις, το αίσθημα που σου δημιουργείται μοιάζει περισσότερο με αυτό του άγχους, παρά με αυτό του τρόμου.

Παρόλα αυτά, το jumpscare είναι σημαντικό εργαλείο που βοηθάει στην συρραφή μιας τρομακτικής εμπειρίας. Είναι η πιο εύκολη μέθοδος, αλλά χρειάζεται καλή διαχείριση του χρόνου και φινέτσας, ώστε ένα jumpscare να είναι πραγματικά τρομακτικό. Ο πραγματικός τρόμος είναι κάτι παραπάνω από αντανακλαστικό, είναι πιο προσωπικός και μένει περισσότερο στον παίκτη ακόμα και αν όταν έχει ολοκληρωθεί το παιχνίδι ή η θέαση της ταινίας. Με αυτό τον τρόπο έχουμε δύο ειδών εκπλήξεις, την αναμενόμενη και τη μη αναμενόμενη.

4.2.1.1. <u>ΑΝΑΜΕΝΟΜΕΝΗ ΕΚΠΛΗΞΗ</u>

Ο παίκτης γνωρίζει ότι κάτι τον περιμένει, αλλά πέρα από αυτό δεν παρέχεται καμία άλλη πληροφορία. Για να επιτευχθεί αυτό δεν χρειάζεται μόνο καλός συγχρονισμός, αλλά και η δημιουργία έντονου ‘suspense’, εξαιτίας της άγνοιάς του. Η μαγεία κείτεται στο χτίσιμο και όχι στον ίδιο το τρόμαγμα καθαυτό.

4.2.2.1 <u>ΜΗ ΑΝΑΜΕΝΟΜΕΝΗ ΕΚΠΛΗΞΗ</u>

Για να επιτευχθεί αυτό θα πρέπει να δημιουργηθούν πρώτα οι ιδανικές καταστάσεις ώστε ο παίκτης να νιώσει ασφάλεια, να κατεβάσει έστω και για λίγο την άμυνά του και στη συνέχεια το παιχνίδι να του τη στερήσει. Όταν ο παίκτης προσπαθεί κατά τη διάρκεια του παιχνιδιού να

επιβιώσει και να ανταπεξέλθει στα γεγονότα, βρίσκεται σε μία κατάσταση διαρκούς επαγρύπνησης, όπου στην ουσία, έστω και υποσυνείδητα, αναζητά εκείνη την μικρή στιγμή όπου θα ηρεμήσει, έστω και για λίγο. Το jumpscare δεν είναι μόνο γρήγορες εναλλασσόμενες εικόνες και δυνατοί ήχοι. Τα πιο πετυχημένα είναι αυτά που χτίζονται εν αγνοία του παίκτη και όταν αυτός το αντιλαμβάνεται, είναι αρκετά αργά. Ο μηχανισμός τους πρέπει να είναι συμπληρωματικός και όχι ο βασικός, μιας και μπορεί να καταντήσει βαρετό.

4.2.2 ΑΠΕΧΘΕΙΑ

Η απέχθεια μπορεί να χρησιμοποιηθεί με τον ίδιο τρόπο με το σοκ. Με βίαιο και με χοντροκομμένο τρόπο, αλλά μπορεί να χρησιμοποιηθεί με τρόπο που να εισβάλει στο ανθρώπινο νου και να του εξάγει κάτι πραγματικά τρομακτικό. Όταν αυτό γίνεται, τότε μπαίνουμε στα χωράφια του ψυχολογικού τρόμου.

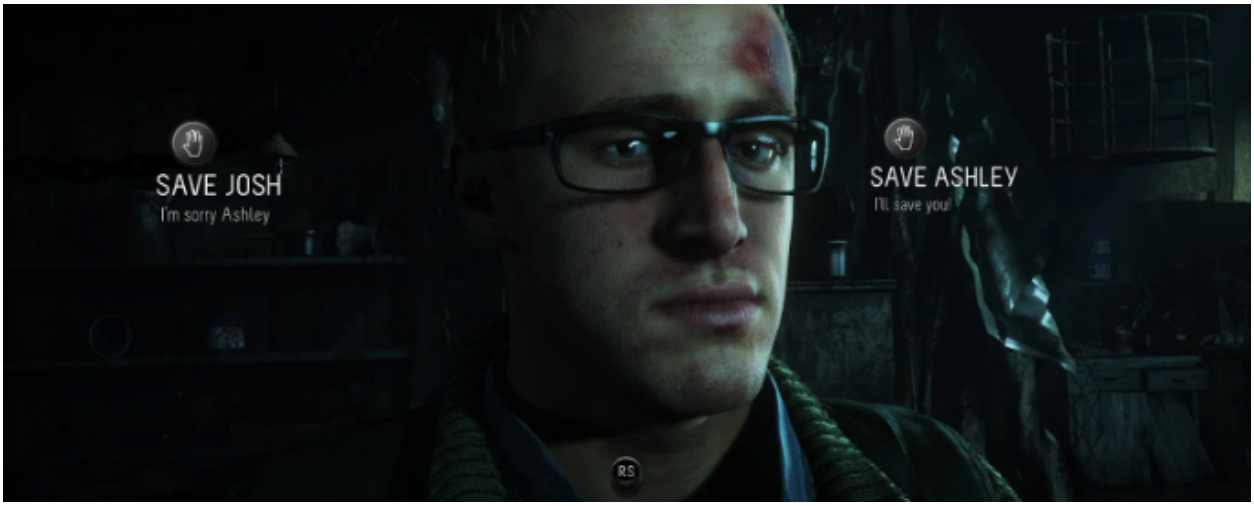
4.2.2.1. ΑΜΕΣΗ ΑΠΕΧΘΕΙΑ

Η άμεση απέχθεια μπορεί να δημιουργηθεί μέσα από γραφικά βίαιες εικόνες, άβολους ήχους κ.ο.κ. Το καλό με αυτή την τακτική είναι ότι δουλεύει από μόνη της. Δεν χρειάζεται να πραγματοποιηθεί κάποιο χτίσιμο ή επιπλέον δουλειά. Το μόνο που απαιτείται είναι να έρθει την κατάλληλη στιγμή μέσα στο παιχνίδι και θα έχει τα επιθυμητά αποτελέσματα.

Είναι κάτι που αποτυπώνεται απευθείας στην οθόνη, αλλά δεν χρειάζεται κατ΄ ανάγκη να προέρχεται από κάποιο τέρας. Μπορεί να πάρει τη μορφή ενός διαστημόπλοιου που τώρα είναι κατειλημμένο από κάποιο άλλο είδος ή μπορεί να έχει και τη μορφή ακόμα χαλασμένου φαγητού. Η απέχθεια ωστόσο, δεν χρειάζεται να προέρχεται από την αλληλεπίδραση του παίκτη με κάτι οφθαλμικά αποκρουστικό. Ο πιο αποτελεσματικός τρόπος είναι να εξερευνάται ακόμα και η πιο σκοτεινή γωνία του ανθρώπινου νου. Εδώ έρχεται και η έμμεση απέχθεια.

4.3.2.2. ΕΜΜΕΣΗ ΑΠΕΧΘΕΙΑ

Η έμμεση απέχθεια περιπαίζει με αυτά που θεωρείς ως δεδομένα και τα αλλοιώνει. Οικείες εικόνες μετατρέπονται σε ξένες. Καταστάσεις που νοιάζεσαι γι’ αυτές διαστρεβλώνονται σε σημείο που δεν είναι πια αναγνωρίσιμες.



[Εικόνα 32]:Στιγμιότυπο από το παιχνίδι Until Dawn

Στην παραπάνω εικόνα απεικονίζεται ένα χαρακτηριστικό παράδειγμα από το παιχνίδι Until Dawn που κυκλοφόρησε το 2015. Η Ashley και ο Josh, δύο από τους πρωταγωνιστές του παιχνιδιού είναι αλυσοδεμένοι σε ένα τοίχο και ο παίκτης καλείται, μέσα από τα μάτια του Chris, να επιλέξει ποιον θα θυσιάσει, ώστε να σώσει τον άλλον. Το τρομακτικό της κατάστασης είναι ο τρόμος του να βλέπεις κάποιον για τον οποίο νοιάζεσαι να πεθαίνει μπροστά στα μάτια σου και να είσαι αδύναμος να περάσεις το οτιδήποτε. Η ερώτηση είναι: χρειάζεται τέτοιου είδους απεικόνισης για να νιώσεις την απέχθεια; Η απάντηση είναι πως όχι, δεν χρειάζεται.

[Εικόνα 33]:Στιγμιότυπο από το παιχνίδι God’s Basement

Ένα παιχνίδι που πετυχαίνει άψογα αυτήν την ισορροπία είναι το 'God's Basement'. Εδώ παίρνεις το ρόλο ενός απρόσωπου χαρακτήρα που καταλήγει στο Καθαρτήριο και θα πρέπει να συμφιλιωθεί με το γεγονός ότι σκότωσε τη γυναίκα που τον μεγάλωσε. Συγκεκριμένα, την ίδια του τη γιαγιά η οποία έπασχε από άνοια. Εκείνη ισχυριζόταν ότι κάθε μέρα άκουγε έναν άντρα να της μιλά από το ράδιο, οπότε έγραψε μία κασέτα με οδηγίες για το πως να αυτοκτονήσει, την τοποθέτησε στο κασετόφωνο και περίμενε πίσω από την πόρτα.

Αξίζει να σημειωθεί, ότι ο τρόπος με τον οποίο χρησιμοποιείται η απέχθεια είναι πολυμορφικός και μπορεί να χρησιμοποιηθεί στα πλαίσια εκτός της βιομηχανίας του τρόμου. Για παράδειγμα σε ένα σημείο του παιχνιδιού spec ops online, απεικονίζονται οι τερατώδεις καταστάσεις του πολέμου. Συγκεκριμένα, ο πρωταγωνιστής καλείται να εξαλείψει μία υποθετική ομάδα από το αντίπαλο στρατόπεδο. Χρησιμοποιεί λευκό φώσφορο, ένα χημικό όπλο, ακραία τοξικό, το οποίο ψήνει τα θύματά του. Πηγαίνοντας όμως στον τόπο όπου έγινε η επίθεση, συνειδητοποιεί πως τα θύματά του φωσφόρου δεν ήταν μόνο ο αντίπαλος, αλλά και άμαχος πληθυσμός.

[Εικόνα 34]:Στιγμιότυπο από το παιχνίδι Agony

Όμως, όταν χρησιμοποιείται η μέθοδος της απέχθειας, καλό είναι να υπάρχει ένα μέτρο και να μην καταχράται. Στο παιχνίδι 'Agony' ο παίκτης καλείται να ξεφύγει από μία κόλαση φτιαγμένη από σάρκα και οστά. Στην αρχή, το όλο σκηνικό, αν και άρρωστο, μπορεί να περιγραφεί ως εκπληκτικό, εξαιτίας των αηδιαστικών μεν, ευφάνταστων δε, σκηνικών τα οποία παρέχουν μία αρκετά πιστευτή εικόνα της κόλασης. Επειδή όμως είναι συνεχές αυτό το σκηνικό, σύντομα τα πάντα φαντάζουν ίδια, μιας και δεν υπάρχει κάποιο σημείο στο παιχνίδι που να επιτρέπει στον παίκτη να αναπνεύσει και το αποτέλεσμα είναι μία συγκλονιστική αρχή και ένα ανιαρό τέλος.

4.2.3 ΦΟΒΟΣ

Ο φόβος, αντίθετα με την κοινή γνώμη, δεν είναι ένα απλό, αλλά ένα υψηλά πολύπλοκο συναίσθημα. **Φόβος**: Ένα δυσάρεστο συναίσθημα που προκαλείται από την απειλή του κινδύνου, του πόνου και του να σε βλάψει κάποιος.

Το πιο σημαντικό κομμάτι της πρότασης είναι η λέξη 'απειλή', μιας και υποδηλώνει ότι

δεν έχει συμβεί κάτι ακόμα. Αλλά ο κίνδυνος και ο πόνος μπορεί να πραγματοποιηθούν την οποιαδήποτε χρονική στιγμή. Το πρόβλημα είναι πως ο παίκτης στην ουσία, δεν είναι παρά ένα άτομο που απλά κάθεται σε μία καρέκλα και παρακολουθεί αυτό που γίνεται σε μία οθόνη. Δεν ανήκει με την κυριολεκτική έννοια στον κόσμο του παιχνιδιού, δεν μπορεί να τον βλάψει κάποιος, ούτε να βρεθεί σε κίνδυνο. Κι όμως, αρκετά παιχνίδια τρόμου καταφέρνουν με μία πληθώρα μηχανισμών να προκαλέσουν αυτά τα συναισθήματα σε όποιον επιλέξει να τα παίξει. Χρησιμοποιείται, όπως αναφέρθηκε και νωρίτερα, η φαντασία του παίκτη εναντίον του και αυτό επιτυγχάνεται μέσω της απομόνωσης και της έλλειψης πληροφορίας.

4.2.3.1. <u>ΑΠΟΜΟΝΩΣΗ</u>

Ο άνθρωπος από την φύση του είναι κοινωνικό ον και γι αυτό το λόγο χρειάζεται άλλους ανθρώπους τριγύρω του. Από την σκοπιά της εξέλιξης, βγάζει νόημα, μιας και από την αρχή του ανθρώπινου είδους, οι άνθρωποι ταξίδευαν σε ομάδες, ώστε να έχουν μεγαλύτερες ελπίδες επιβίωσης, αφού δεν είναι ούτε το πιο δυνατό ή το πιο γρήγορο ζώο. Για παράδειγμα, την εποχή που ο άνθρωπος έπρεπε να κυνηγήσει για την τροφή του, μία αρκούδα μπορούσε με μεγάλη ευκολία να κατατροπώσει έναν ενήλικα, όχι όμως μία ομάδα κυνηγών. Είναι έμφυτη η ανάγκη του ανθρώπου για συντροφιά και όταν αυτή η ανάγκη δεν ικανοποιείται, τότε νιώθει μοναξιά και είναι πιο ευάλωτος μπροστά σε επερχόμενο κίνδυνο. Γι' αυτό το λόγο άλλωστε τα τρομαχτικά παιχνίδια δεν εστιάζουν σε μία ομάδα ατόμων, αλλά στον ιδιώτη.

Ένα χαρακτηριστικό παράδειγμα είναι στην περίπτωση του Resident Evil 5 και 6, όπου ο παίκτης είναι μαζί με κάποιον άλλον που βρίσκεται στην ίδια κατάσταση με αυτόν, έχει κάποιον με τον οποίο μπορεί να συμπάσχει και αυτό αφαιρεί τον φόβο. Παρότι είναι αναγκαία η ύπαρξη ενός ασφαλούς και οικείου σημείου, το αίσθημα της βεβαιότητας δεν θα έπρεπε να είναι σταθερό σε ένα τρομακτικό παιχνίδι.

Ας θεωρήσουμε τώρα ότι ο παίκτης είναι ολομόναχος και είναι μάλιστα η μοναδική οντότητα σε ολόκληρο το παιχνίδι, αυτό από μόνο του δεν εγγυάται ότι απαραίτητα θα τον τρομάξει κιόλας. Σε παιχνίδια όπως το Super Mario, παρότι περιστρέφονται γύρω από τον βασικό χαρακτήρα, κανείς δεν έχει τρομάξει μ' αυτά. Απομονώνοντας τον κεντρικό χαρακτήρα δεν είναι αρκετό, γι' αυτό πρέπει να βρεθούν κι άλλοι τρόποι ώστε ο παίκτης να νιώσει μόνος του και εδώ έρχεται ο παράγοντας της παράξενης/ μυστηριώδους κοιλάδας.

4.2.3.2. UNCANNY VALLEY: ΤΟ ΦΑΙΝΟΜΕΝΟ ΤΗΣ ΠΑΡΑΞΕΝΗΣ ΚΟΙΛΑΔΑΣ

Η ορολογία της ‘Παράξενης Κοιλάδας' (uncanny valley στα αγγλικά) προήλθε από τον Ιάπωνα καθηγητή ρομπολογίας Μασαχίνο Μόρι και εξηγεί το περίεργο συναίσθημα που νιώθει κάποιος όταν κοιτά κάτι που μοιάζει με ανθρώπινο, αλλά δεν είναι, μιας και του λείπουν κάποια συγκεκριμένα ανθρώπινα χαρακτηριστικά, όπως για παράδειγμα ένα κέρινο ομοίωμα, τα ρομπότ με ανθρωπόμορφα χαρακτηριστικά και τα ζόμπι.

Οι βετεράνοι στη βιομηχανία του ‘χόρρορ’ γνωρίζουν τη δύναμη που έχει η ‘‘μυστηριώδης κοιλάδα’’ και τη χρησιμοποιούν συχνά στα έργα τους χωρίς να γίνεται αντιληπτό από τον θεατή. Υπάρχουν τρεις έμφυτοι φόβοι στον άνθρωπό. Φόβος του σκοταδιού, του ύψους και για κάτι που μοιάζει ανθρώπινο, αλλά δεν είναι. Αυτό έχει σχέση με την εξέλιξη του ανθρώπου και βρίσκεται μέσα σε όλους, είτε σε μικρό, είτε σε μεγάλο βαθμό. Πράγμα που κάνει αρκετούς να πιστεύουν ότι υπάρχει κάτι στην ανθρώπινη ιστορία της εξέλιξης του ανθρώπινου είδους που έμοιαζε με άνθρωπο αλλά δεν ήταν.

ΜΕΡΟΣ Β':

1. Εισαγωγή στις τεχνολογίες που χρησιμοποιήθηκαν

Για την ανάπτυξη ενός παιχνιδιού, όπως είναι λογικό χρειάζεται ένα αντίστοιχο εξειδικευμένο σύστημα λογισμικού για την ανάπτυξή του. Το λογισμικό αυτό ονομάζεται μηχανή παιχνιδιού (**game engine**) και υπάρχει πληθώρα αυτών στην αγορά, σχεδιασμένες για να δουλεύουν σε διάφορες κονσόλες, αλλά και σε αρκετά λειτουργικά συστήματα υπολογιστών (Windows, Linux, Mac OS X). Η βασική λειτουργία μιας μηχανής ανάπτυξης παιχνιδιού περιλαμβάνει μία μηχανή φωτο απόδοσης (ή όπως είναι πιο ευρέως διαδεδομένο με την ονομασία render - **renderer**) για την δημιουργία γραφικών σε περιβάλλοντα δύο και τριών διαστάσεων, μια μηχανή για φυσική (physics) ή εντοπισμό συγκρούσεων (collision detection), ήχο, ανάπτυξη script κώδικα, animation, τεχνητή νοημοσύνη, streaming, threading και έναν γράφο σκηνής (scene graph).

Αναλυτικότερα, μία μηχανή παιχνιδιού αποτελεί δομή για την ανάπτυξη παιχνιδιών. Αυτή η δομή μας βοηθά με πολλά βασικά στοιχεία που έχουν όλα τα παιχνίδια. Για λόγους απλούστευσης θα χρησιμοποιήσουμε τρία:

α) Γραφικά

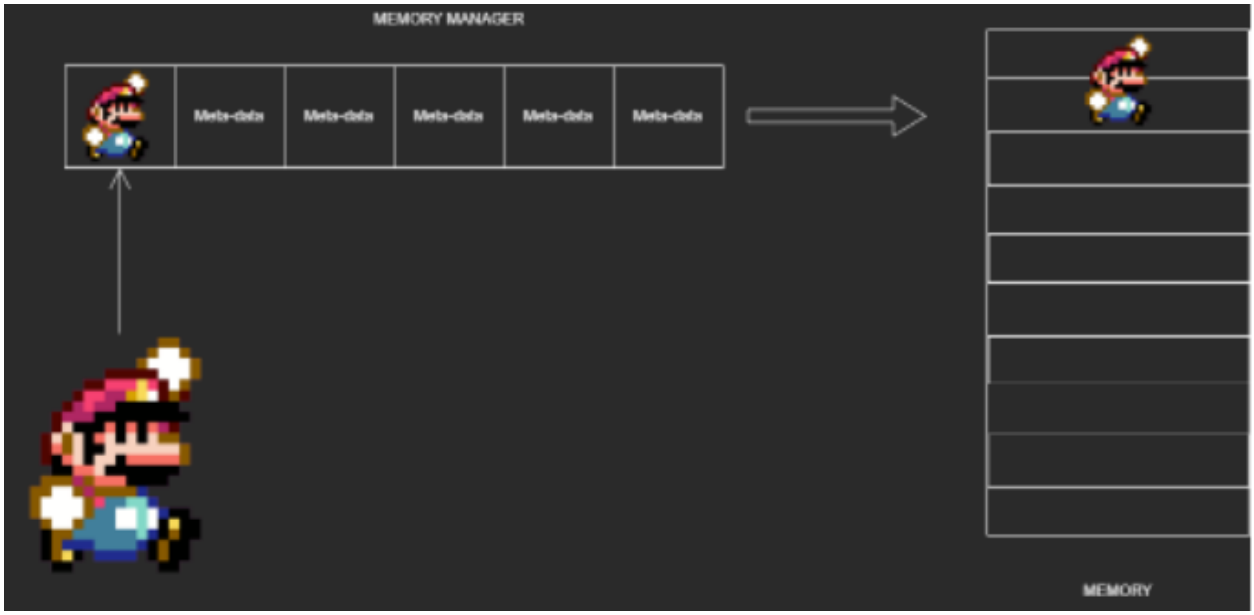
β) Ήχος

γ) Λογική

Αυτά τα τρία συστατικά βρίσκονται παντού, είτε σε μηχανές που χρησιμοποιούνται για την ανάπτυξη παιχνιδιών AAA (με τον όρο AAA ή Triple-A παιχνίδια, αναφερόμαστε σε αυτά που διαθέτουν μεγάλο προϋπολογισμό τόσο για την παραγωγή όσο και για το μάρκετινγκ) είτε για το Tetris. Οι μοντέρνες μηχανές έχουν γίνει πιο εξειδικευμένες και περιλαμβάνουν δομές για συστήματα φυσικής, τεχνητής νοημοσύνης, δικτύωσης, βελτιστοποίησης και ακόμα πολλά περισσότερα. Τι εννοούμε, όμως, με τον όρο δομή (**framework**) και πως βοηθάει;

Στα πρώιμα στάδια ανάπτυξης το παιχνιδιών τις περασμένες δεκαετίες, όλα τα παιχνίδια ήταν φτιαγμένα από την αρχή (from scratch). Για παράδειγμα, στο super Mario για την κονσόλα NES (Nintendo Entertainment System), υπάρχουν αρκετά αντικείμενα για την πρώτη πίστα, όπως

είναι ο πρωταγωνιστικός χαρακτήρας, να νομισματα, το έδαφος, οι πράσινες σωλήνες, οι εχθροί κ.ο.κ. Πρέπει να δοθεί εντολή στο NES να φορτώσει όλα αυτά τα αντικείμενα από την κασέτα μνήμης, στο κατάλληλο στίγμα συντεταγμένων μέσα στο χώρο όπου θα βρίσκονται. Επιπλέον, μόλις αυτά σταματήσουν να χρησιμοποιούνται πρέπει αντίστοιχα να δοθεί εντολή να τα διαγράφει, ώστε να απελευθερωθεί χώρος για τον επόμενο γύρο. Αυτό σημαίνει, πως κάθε φορά που δημιουργείται ένα νέο αντικείμενο, πρέπει να το εντοπιζεται και να διαγράφεται ένα προς ένα. Αυτό θα ήταν πιο εύκολο με την δημιουργία ενός συστήματος, μιας δομής δηλαδή, όπου θα παρακολουθεί αυτά τα αντικείμενα και θα τα 'απελευθερώνει' αυτόματα μόλις η πίστα θα τερματίζει και θα φορτώσει η επόμενη.



[Εικόνα 35]: Λογική της δομής διαχείρισης μνήμης

Αυτό το σύστημα θα χρειάζεται να δεσμεύει περισσότερη μνήμη, αλλά θα κερδίζει χρόνο στην ανάπτυξη του παιχνιδιού. Κάθε παιχνίδι έχει αντικείμενα που είναι οι εχθροί, ο χαρακτήρας, νομίσματα κ.ο.κ. Οπότε, όπως είναι λογικό, ένα σύστημα κοινό διαχείρισης μνήμης είναι αναγκαίο να δημιουργηθεί ώστε να μην χρειάζεται να γράφεται ο ίδιος κώδικας κάθε φορά. Για αυτό το λόγο αναφερόμαστε ότι μία μηχανή παιχνιδιού είναι ένα framework, καθώς εμπεριέχει εργαλεία και χαρακτηριστικά που χρειάζεται κάθε παιχνίδι, ώστε να λειτουργήσει. Με τη χρήση μίας μηχανής σημαίνει, ότι δεν χρειάζεται να ανακαλύπτουμε τον τροχό κάθε φορά που θέλουμε να

ξεκινήσουμε ένα νέο project. Ενώ τα καινούργια παιχνίδια μπορούν να διαχειριστούν την κατανάλωση πόρων σε πολύπλοκα συστήματα, στην πραγματικότητα τα περισσότερα παιχνίδια της παλιάς σχολής δεν ήταν σε θέση να ανταπεξέλθουν σε αυτή την ανταλλαγή, μιας και η διαθέσιμη μνήμη ήταν πολύ περιορισμένη.

Η ανάγκη για την χρήση μίας μηχανής παιχνιδιών αυξήθηκε με την διάδοση των παιχνιδιών τριών διαστάσεων. Παράλληλα, καθώς η τεχνολογία γινόταν όλο και πιο περίπλοκη, πολλά studio ξεκίνησαν να χρησιμοποιούν αυτές τις μηχανές, ώστε να κάνουν πιο γρήγορη τη διαδικασία της ανάπτυξης των παιχνιδιών, αν και τα περισσότερα ανέπτυξαν την δική τους μηχανή μέχρι, ένα φαινόμενο το οποίο παρατηρείται μέχρι και σήμερα. Για παράδειγμα, η CDprojekt ανέπτυξε την δική της μηχανή για όλα τα παιχνίδια της σειράς Witcher, την οποία ονόμασαν REDengine, με στόχο την δημιουργία μη-γραμμικών παιχνιδιών, κάτι που οι τότε μηχανές δεν παρείχαν αυτή τη δυνατότητα.

1.1 Unity

[Εικόνα 36]: Το επίσημο logo της Unity

Το Unity αποτελεί μία μηχανή ανάπτυξης παιγνίων, από την Unity Technologies, η οποία ανακοίνωσε την κυκλοφορία της τον Ιούνιο του 2005 στο Παγκόσμιο Συνέδριο Developer. Χρησιμοποιείται για την δημιουργία παιχνιδιών δύο και τριών διαστάσεων, αλλά και εικονικη (virtual reality) και επαυξημένης πραγματικότητας (augmented reality). Η Unity ιδρύθηκε στην Κοπεγχάγη από τον Νίκολας Φράνσις, Χοακίν Άντε και τον Ντέιβιντ Χέλγκασεν.

Η ιστορία της ξεκινά σε ένα openGL φόρουμ τον Μάιο του 2002. Η πρώτη της έκδοση δημοσιεύτηκε το 2005. Ο αρχικός της σκοπός είναι να κάνει την διαδικασία ανάπτυξης παιχνιδιών προσβάσιμη στο ευρύ κοινό. Με αυτό τον τρόπο, αρκετοί ανεξάρτητοι developers και άτομα με βασικές γνώσεις στον προγραμματισμό, κατάφεραν να υλοποιήσουν τις ιδέες τους, μιας και πριν, αν επιθυμούσε κάποιος να αναπτύξει ένα παιχνίδι θα έπρεπε να επιλέξει ανάμεσα σε λίγες και ακριβές μηχανές στην αγορά.

Το 2002 δημοσιεύτηκε η Unity 2.0 μαζί με 50 επιπλέον νέα χαρακτηριστικά, μέσα σε αυτά να περιλαμβάνονται: βελτιοποιημένη μηχανή για τον σχεδιασμό εδάφους, δυναμικές σκιές όπου απεικονίζονται σε πραγματικό χρόνο, κατευθυντικοί φωτισμοί, προβολείς φωτός και αναπαραγωγή βίντεο. Επιπλέον, πρόσθεσαν χαρακτηριστικά που διευκόλυναν την συνεργασία ανάμεσα σε δημιουργούς, περιλαμβάνοντας ένα ‘στρώμα’ δικτύου (networking), κάτι που επιτρέπει τη δημιουργία multiplayer παιχνιδιών χρησιμοποιώντας UDP πρωτόκολλο (user datagram protocol)

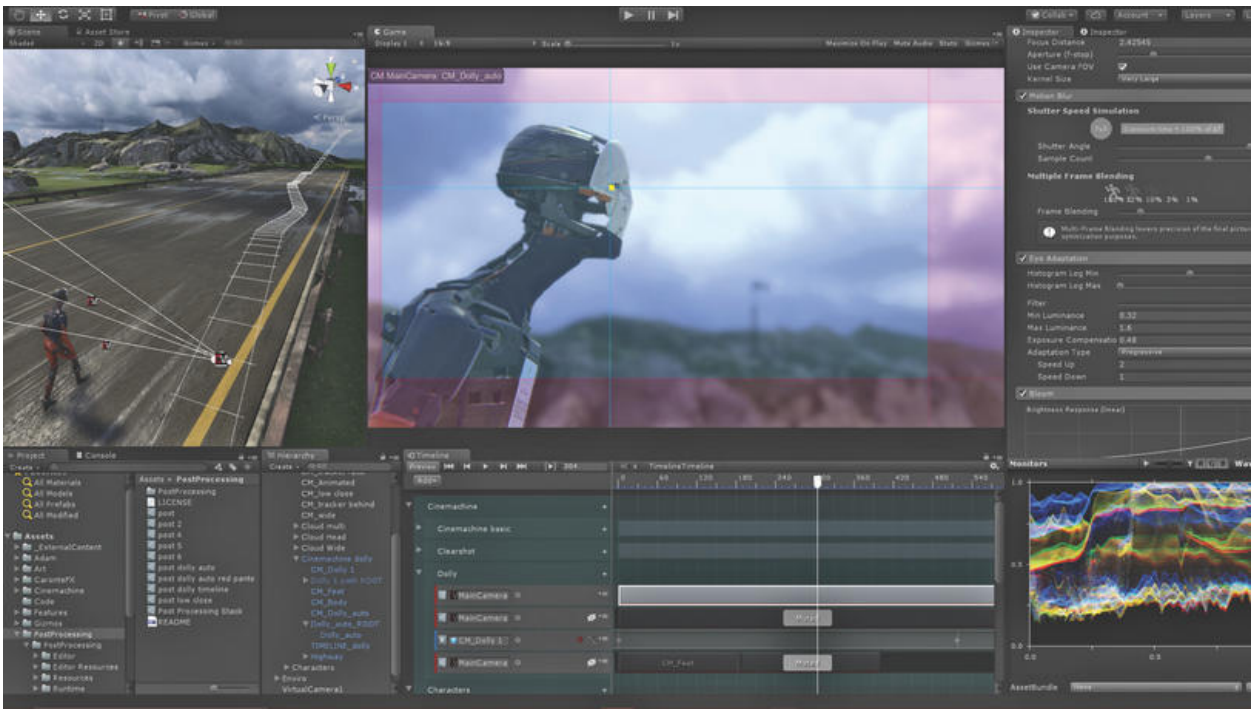
προσφέροντας network address translation, state synchronization και remote procedure call. Όταν η apple κυκλοφόρησε το app store το 2008, η Unity συμπεριέλαβε αμέσως υποστήριξη για ios λογισμικό. Για αρκετά χρόνια η κύρια χρήση της μηχανής ήταν κυρίως για ios εφαρμογές κινητής. Αυτή ήταν επίσης η αρχή στην κυριαρχία της στις εφαρμογές παιχνιδιών κινητής.

Η τρίτη της έκδοση, κυκλοφόρησε το 2010 με αρκετά βελτιωμένα γραφικά για υπολογιστές και κονσόλες. Ενώ το 2012, πάνω από 1.300.000 χρήστες χρησιμοποιούν το Unity για την δημιουργία Gee Whiz γραφικών. Στόχος της ήταν να αναπτύσσονται εφαρμογές χρησιμοποιώντας την, όπου θα τρέχουν σε όλες τις πλατφόρμες. Με την κυκλοφορία της 4ης έκδοσης της μηχανης, είχαμε την προσθήκη του DirectX 11, υποστήριξη για Adobe Flash, εργαλεία για animation, ομοναζόμενα "Mecanim" και πρόσβαση σε Linux.

Το 2015, με την κυκλοφορία της πέμπτης έκδοσης, η Unity έφτασε πολύ κοντά τον προαναφερόμενο αρχικό της στόχο. Είχε αρκετά πιο βελτιωμένα γραφικά, ρεαλιστικό φωτισμό και ακουστική. Με τη χρήση webGL, οι developers μπορούσαν να προσθέσουν το παιχνίδι τους "online", χωρίς να χρειάζεται ο παίκτης να εγκαταστήσει κάποιο πρόσθετο (plugin). Επιπλέον, πρόσθεσε τεχνικές φωτισμού που αντακοπρίκονταν σε πραγματικό χρόνο, προεπισκόπηση φωτοχαρτών, δικό της νέφος (cloud), σύστημα ήχου και τέλος το τότε νέο σύστημα φυσικής της NVidia, PhysX 3.3. Παράλληλα, προσέφερε ένα σύστημα σωματιδίων (**particle system** - χρησιμοποιείται για τη δημιουργία σκόνης, φωτιάς, χωνιού κ.ο.κ.) και βελτιστοποίηση των επιδόσεων της γενικότερα, ενώ επιπλέον προσθεσε υποστήριξη για τη Nintendo Switch φορητή κονσόλα και για το Facebook, google Daydream και έναν 4K player με την δυνατότητα να τρέχει 360 μοίρες βίντεο για εικονική και επαυξημένη πραγματικότητα. Ενώ η unity είχε καταφέρει πλέον να γίνει προσβάσιμη προς το ευρύ κοινό, πολλοί την επίπληξαν εξαιτίας της πληθώρας παιχνιδιών που παράγονταν χρησιμοποιώντας την και βρίσκονταν στο Steam από άπειρους developers.

Το 2016, η εταιρεία αποφάσισε να αλλάξει τον το τρόπο με τον οποίο χρησιμοποιούσε για ονομασει την κάθε έκδοση της μηχανής από αριθμητικό σε χρονολογικό. Έτσι από την 5.6 έκδοση, μεταφέρθηκε στην 2017, που κυκλοφόρησε την αντίστοιχη χρονιά. Σε αυτή, έχουμε την προσθήκη εργαλείου που επιτρέπει την ρεαλιστική απεικόνιση γραφικών σε πραγματικό χρόνο, την ανάπτυξη κόσμων, analytics και αναφορά των επιδόσεων. Η 2017.2 περιλάμβανε νέα εργαλεία, όπως είναι το timeline, ένα εργαλείο που έδινε την δυνατότητα στους χρήστες να προσθέσουν animation

πολύ πιο εύκολα με “drag and drop” και “cinemachine”, ένα έξυπνο σύστημα καμερών. Επιπλέον, ενσωμάτωσε υποστήριξη για αρχεία “autodesk”, “3ds” και “maya”, παρέχοντας τους developers μία πληρώρα assets για να χρησιμοποιήσουν στο παιχνίδι τους.



[Εικόνα 37]: Το εργαλείο Cinemachine της Unity

Η 2018 έκδοση, περιλάμβανε “render pipeline graphics tool”, δηλαδή εργαλείο που παρήγαγε γραφικά υψηλής ποιότητας και ρεαλισμού. Έχουμε την δημιουργία των εργαλείων: “high definition render pipeline” για υπολογιστές και κονσόλες και “lightweight rendering pipeline” για κινητές συσκευές, εικονική και επαυξημένη πραγματικότητα. Επιπροσθέτως, συμπεριέλαβε machine learning εργαλεία, όπως είναι η εκμάθηση από την μίμηση, όπου στην ουσία το παιχνίδι μαθαίνει μέσα από τις συνήθειες του παίκτη μέσα σε αυτό.

Ενώ στην αρχή ξεκίνησε ως μία μηχανή αποκλειστική για λειτουργικό Mac Os X, με το καιρό πρόσθεσε υποστήριξη και για άλλες πλατφόρμες, με τον αριθμό τους να φτάνει τις 25 το 2018. Είναι τόσο δημοφιλής σαν μηχανή, μιας και σύμφωνα με έρευνα του 2018, περίπου τα μισά παιχνίδια κινητής και το 60% των vr είναι αναπτυγμένα σε αυτή. Το 2020 ανακοίνωσε πως 2 δισεκατομμύρια ενεργοί χρήστες παίζουν παιχνίδια που έχουν αναπτυχθεί σε αυτή, ενώ 1,5

εκατομμύριο είναι οι developers που απασχολούνται με αυτή. Παρά το γεγονός, ότι η βασική της χρήση είναι για ανάπτυξη παιγνίων, το 2017 έστρεψε την προσοχή της και σε άλλους τομείς, όπως είναι ο κινηματογράφος και η αυτοκινητοβιομηχανία.

Σχεδιαστικά, η μηχανή είναι δομημένη εσωτερικά με C/C++, αν και η C# είναι η γλώσσα που αλληλεπιδρά ο χρήστης στην ανάπτυξη των εφαρμογών. Για αυτό το λόγο είναι αναγκαία η εξοικείωση με βασικές έννοιες ανάπτυξη κώδικα σε C#. Παρακάτω αναφέρεται αναλυτικότερα, πως η Unity, υλοποιεί .NET και C# και τους περιορισμούς που μπορεί να συναντήσει κάποιος κατά τον προγραμματισμό.

1.1.1 .NET

Το .NET Framework (προφέρεται ως "dot net") είναι ένα ιδιόκτητο πλαίσιο λογισμικού που αναπτύχθηκε από τη Microsoft και εκτελείται κυρίως σε Microsoft Windows. Ήταν η κυρίαρχη εφαρμογή της Κοινής Γλωσσικής Υποδομής (CLI) έως ότου αντικατασταθεί από το έργο cross-platform .NET. Περιλαμβάνει μια μεγάλη βιβλιοθήκη κλάσης που ονομάζεται Framework Class Library (FCL) και παρέχει γλωσσική διαλειτουργικότητα (κάθε γλώσσα μπορεί να χρησιμοποιήσει κώδικα γραμμένο σε άλλες γλώσσες) σε πολλές γλώσσες προγραμματισμού.

Η Unity χρησιμοποιεί την δομή ανοιχτού κώδικα, ώστε να διασφαλίσει ότι οι εφαρμογές που αναπτύσσεται με την μηχανή μπορε να τρέξει σε ποικίλες υλικο λειτουργικές (hardware) διαμορφώσεις, αυτό μιας και η .NET υποστηρίζει μεγάλο εύρος γλωσσών και API βιβλιοθηκών.

1.1.1.1 Scripting backends

Η Unity έχει δύο συστήματα υποστήριξης γραφής κώδικα (scripting backend), τα Mono και IL2CPP (ενδιάμεση γλώσσα για την C++), όπου κάθε μία χρησιμοποιεί διαφορετική τεχνική μεταγλώττισης.

Mono

Η Mono χρησιμοποιεί just-in-time (JIT) μεταγλώττιση του κώδικα, κατά παραγγελία κατά το χρόνο εκτέλεσης.

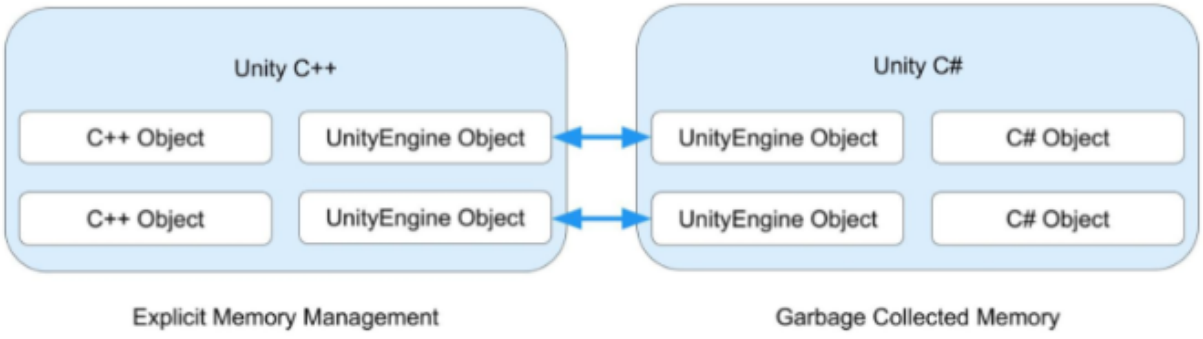
IL2CPP

Η IL2CPP χρησιμοποιεί ahead-of-time (AOT) μεταγλώττιση της εφαρμογής πριν την εκτέλεση. Το πλεονέκτημα χρήσης JIT στο backend είναι ότι ο χρόνος μεταγλώττισης είναι πιο γρήγορος σε σχέση με αυτό στον AOT, όπου είναι και πιο ανεξάρτητο σε σχέση με την πλατφόρμα που θα τρέξει. Ο editor της unity είναι βασισμένος σε JIT και χρησιμοποιεί Mono στο backend.

1.1.1.2 Garbage collection

Στον τομέα της επιστήμης των υπολογιστών ο **garbage collector** (στα ελληνικά ξεσκαρτάρισμα) είναι μία μορφή αυτόματης διαχείρισης μνήμης. Είναι υπεύθυνο για την αξιοποίηση της μνήμης που έχει κατανεμηθεί από το σύστημα, και σε περίπτωση που δεν χρησιμοποιείται, την αποδεσμεύει από εκείνο το σημείο και την κάνει ξανά διαθέσιμη για χρήση.

Η Unity χρησιμοποιεί Boehm garbage collector και για τα δύο backend (Mono και IL2CPP). Ο Boehm garbage collector, χρησιμοποιεί μία μέθοδο αλγορίθμου σημάδευσης-σκούπας (mark-sweep). Παρέχει σταδιακή και γενεαλογική συλλογή σε λειτουργικά συστήματα που παρέχουν το σωστό είδος υποστήριξης εικονικής μνήμης (συγκεκριμένα αυτή τη στιγμή περιλαμβάνει SunOS(45), IRIX, OSF/1, Linux και Windows). Επιτρέπει την επίκληση του κώδικα οριστικοποίησης όταν συλλέγεται ένα αντικείμενο. Μπορεί να εκμεταλλευτεί τις πληροφορίες τύπου για τον εντοπισμό δεικτών, εάν παρέχονται τέτοιες πληροφορίες, αλλά συνήθως χρησιμοποιείται χωρίς τέτοιες πληροφορίες.



[Εικόνα 38]: Απεικόνιση του garbage collector στη Unity

1.1.2 Επαναφόρτωση κώδικα στον Unity Editor

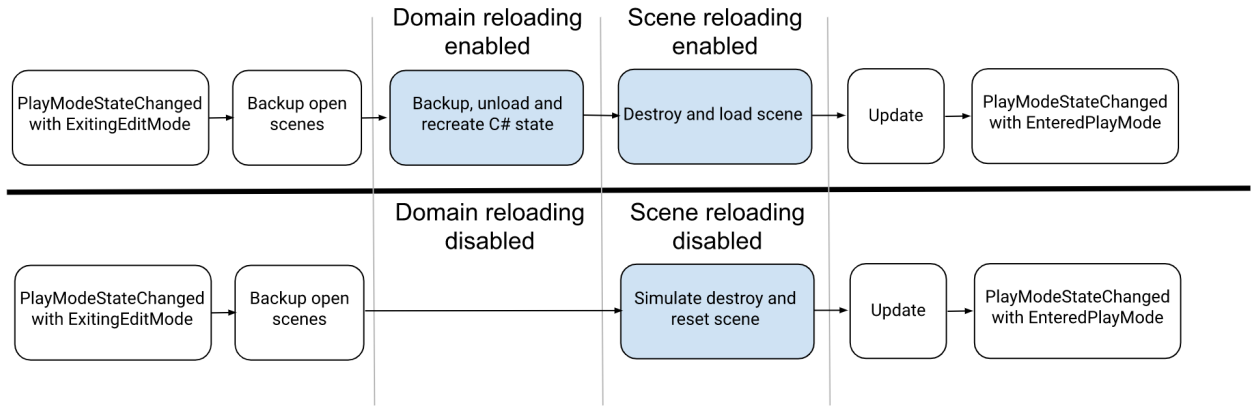
1.1.2.1 Enter Play Mode

Το Play Mode είναι ένα από τα θεμελιώδη χαρακτηριστικά του Unity. Επιτρέπει την εκτέλεση του project απευθείας από τον Editor μέσα από το κουμπί Play στο Toolbar. Με την εκκίνηση του play mode, το project ξεκινά και συμπεριφέρεται όπως αν είτε πρώτα κάνει build. Ό,τι αλλαγές πραγματοποιούνται κατά το Play Mode, επαναφέρονται όταν αυτό τερματιστεί. Κατά την είσοδο σε Play Mode, η Unity εκτελεί δύο σημαντικές ενέργειες, ώστε να διασφαλιστεί ότι το project ξεκινά στον Editor με τον ίδιο τρόπο όπως στο build.

- Επαναφέρει την κατάσταση δέσμης ενεργειών (αυτό ονομάζεται επίσης "Επαναφόρτωση τομέα")
- Επαναφέρει την σκηνή

Αυτές οι δύο ενέργειες χρειάζονται λίγο χρόνο για να εκτελεστούν και ο χρόνος αυξάνεται καθώς τα κομμάτια κώδικα και οι σκηνές γίνονται πιο σύνθετες.

Η δυνατότητα γρήγορης εισόδου και εξόδου από το Play Mode είναι ένας σημαντικός παράγοντας κατά την ανάπτυξη της εφαρμογής ή παιχνιδιού. Η ταχύτητα με την οποία εκεινάται και τερματίζεται το το Play Mode, επηρεάζει αντίστοιχα και την ταχύτητα με την οποία θα επέλθουν και οι αλλαγές / βελτιώσεις. Επειδή η γρήγορη ταχύτητα επανάληψης κατά την ανάπτυξη είναι σημαντική και επειδή ο χρόνος επαναφοράς της κατάστασης σκηνής και δέσμης ενεργειών μπορεί να αποτελέσει εμπόδιο σε αυτό, το Unity προσφέρει τη δυνατότητα διαμόρφωσης του τι συμβαίνει όταν εκτελεστεί η λειτουργία αναπαραγωγής, δίνοντάς την επιλογή να απενεργοποίησης, των ενεργειών "Επαναφόρτωση τομέα" (Domain Reload) και "Επαναφόρτωση σκηνής" (Scene Reload). Αυτές οι δύο επιλογές παρέχονται από το Configurable Enter Play Mode. Στο παρακάτω διάγραμμα παρουσιάζει τα αποτελέσματα της απενεργοποίησης των "Επαναφόρτωση τομέα" (Domain Reload) και "Επαναφόρτωση σκηνής" (Scene Reload).



[Εικόνα 39]: Αποτελέσματα της απενεργοποίησης των "Επαναφόρτωση τομέα" (Domain Reload) και "Επαναφόρτωση σκηνής" (Scene Reload)

1.1.2.1 Εκτέλεση του Editor Script Code κατά την εκκίνηση

Μερικές φορές είναι χρήσιμη η εκτέλεση κάποιου κομματιού κώδικα σε ένα project με το που εκκινηθεί η Unity, χωρίς να απαιτείται κάποια ενέργεια από τον χρήστη. Αυτό μπορεί να πραγματοποιηθεί με την εφαρμογή του χαρακτηριστικού InitializeOnLoad (αρχικοποίησε κατά την φόρτωση) σε μία κλάση που έχει static constructor.Παρακάτω παρατίθεται ένα παράδειγμα, με την χρήση του προαναφερόμενου attribute και σαν αποτέλεσμα εμφανίζει το μήνυμα “Up and running” στο console log του Unity:

```
[InitializeOnLoad]
public class Startup {
    static Startup() {
        Debug.Log("Up and running");
    }
```

```
}
```

Ένας static constructor είναι πάντα εγγυημένο ότι θα κληθεί πριν χρησιμοποιηθεί οποιαδήποτε στατική συνάρτηση ή στιγμιότυπο της κλάσης, ωστόσο το χαρακτηριστικό InitializeOnLoad διασφαλίζει ότι καλείται κατά την εκκίνηση του Editor.

1.1.3 Σειριοποίηση (Serialize Script)

Η σειριοποίηση αποτελεί μία αυτόματη διαδικασία μεταποίησης των δομών δεδομένων ή κατάστασης αντικειμένου σε μία μορφή όπου η Unity θα αποθηκεύσει και θα ανασχηματίσει αργότερα. Ορισμένες από τις ενσωματωμένες δυνατότητες του Unity χρησιμοποιούν σειριοποίηση, χαρακτηριστικά όπως αποθήκευση και φόρτωση, το παράθυρο του Inspector και τα Prefabs. Ο τρόπος με τον οποίο οργανώνονται τα δεδομένα στο Project του Unity επηρεάζει τον τρόπο με τον οποίο το Unity σειριοποιεί αυτά τα δεδομένα και μπορεί να έχει σημαντικό αντίκτυπο στην απόδοση του εκάστοτε Project.

1.2 C#

Όπως αναφέρθηκε και παραπάνω, το unity παρέχει τη δυνατότητα σε χρήστες να δημιουργήσουν παιχνίδια σε περιβάλλον δύο ή τριών διαστάσεων. Γι αυτό το λόγο, η μηχανή παρέχει βασικές λειτουργίες API σε γλώσσα C#, είτε με την μορφή προσθέτων, είτε απευθείας με την χρήση συγγραφής κώδικα στο Visual Code Studio.

Η C# ή αλλιώς C Sharp, είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, σχεδιασμένη από τον Anders Hejlsberg, προγραμματιστή της Microsoft to 2000. Δημιουργήθηκε με το σκεπτικό να αποτελεί μία ‘πιο εύκολη C’, καθώς είναι απλή, σύγχρονη. Είναι αρκετά παρεμφερής με τις γλώσσες της οικογένειάς της (C, C++), αλλά και με την Java ως προς στην αντικειμενοστρέφεια στη λογική της.

Συγκεκριμένα:

➔ Το ελληνικό ερωτηματικό για να υποδηλώσει το τέλος μιας γραμμής στον κώδικα.

➔ Αγκύλες ‘{}’ για τις απομαδοποιημένες γραμμές κώδικα, όπως είναι οι συναρτήσεις και οι κλάσεις.

➔ Τιμές σε μεταβλητές ανατίθενται με το ‘=’, ενώ για την σύγκριση τους με το διπλό ίσων.

➔ Αγκύλες ‘[]’ χρησιμοποιούνται σε πίνακες.

1.3 Vegas Pro

[Εικόνα 40]: Το επίσημο λογότυπο του Vegas Pro

Το Vegas Pro είναι πρόγραμμα επεξεργασίας βίντεο μη γραμμικής επεξεργασίας, αρχικά κυκλοφορημένο από την Sonic Foundry, ύστερα από την Sony Creative Software και τώρα από την Magix. Το λογισμικό αυτό τρέχει αποκλειστικά σε Windows. Αρχικά, αναπτύχθηκε ως λογισμικό επεξεργασίας ήχου, στη συνέχεια όμως προστέθηκε και η δυνατότητα χειραγώγησης βίντεο. Επιπλέον, ο χρήστης έχει την δυνατότητα να κάνει:

1) Προεπισκόπηση το έργο του σε πραγματικό χρόνο ανεξάρτητα των αριθμών tracks που

χρησιμοποιεί

2) Αλληλουχία βίντεο ανεξάρτητα της ανάλυσης που χρησιμοποιείται.

3) Περίπλοκα εφέ και εργαλεία σύνθεσης

Το Vegas περιλαμβάνει:

1) Υποστήριξη 24-bit/192 KHz ήχου

2) Μίξη Dolby Digital Surround

3) Διάθεση σε πραγματικό χρόνο πολυκάναλο βίντεο και ήχο

Επίσης δεν χρειάζεται κάποιες ειδικές προδιαγραφές για υπολογιστή για να τρέξει κατάλληλα, επιτρέποντάς το να λειτουργεί στον μέσο υπολογιστή με windows.

Αρχικά, η Sonic foundry εισήγαγε μία γρήγορη ματιά στο "Vegas Beta" τον Ιούνιο του 1999, με την ονομασία "Multitrack Media Editing Software", δηλαδή λογισμικό πολλαπλών track επεξεργασίας. Έπειτα η Sonic συνέχισε να εισάγει μεταγενέστερες εκδόσεις με το πρόγραμμα επεξεργασίας. Αυτή η αρκετά πρωτοποριακή έκδοση της 3.0 εισήγαγε νέα εφέ βίντεο, όπως είναι τα lens flare (μία τεχνική αντανάκλασης του φωτός που χρησιμοποιείται στον κινηματογράφο), color curves (τρεις καμπύλες σε άξονα x και y, όπου η κάθε μία αντιστοιχεί σε κάποιο από τα χρώματα rgb) και πολλά ακόμα. Η sonic συνέχισε να εκδίδει ένα πιο εκλεπτυσμένο λογισμικό επεξεργασίας βίντεο, αλλά αργότερα μέσα στην ίδια χρονιά η Sony αγόρασε την εταιρεία, κατα συνέπεια αποκτώντας και τα δικαιώματα για το πρόγραμμα. Το Vegas συνέχισε να κυκλοφορεί από την Sony, από την έκδοση 5.0 και μετά.

Η 8η έκδοση ήταν η πρώτη που είχε την ονομασία "Sony Vegas Pro", με την 8.1 να αποτελεί την πρώτη που υποστηριζόταν σε 64-bit συστήματα. Η 9.0 είναι η πρώτη με υποστήριξη 4k ανάλυσης για σινεμά. Η τελευταία έκδοση που αναπτύχθηκε εξ ολοκλήρου από την Sony Interactive και εισήγαγε νέα εργαλεία σε συνεργασία με άλλες εταιρείας που διευκόλυναν για επαγγελματίες δημιουργούς περιεχομένου. Τον Μάιο του 2016, αγόρασε το Vegas Pro από την Sony μαζί με το Movie Studio, με την έκδοση 14.0 και μετά να εκδίδονται εξ ολοκλήρου από εκείνη. Συγκεκριμένα, βελτιστοποίησε την επεξεργασία βίντεο σε αναλύσεις 4κ, με επίλυση

αρκετών μικρο προβλημάτων και μεγαλύτερη αύξηση στην μέγιστη ταχύτητα που μπορεί να φτάσει η ταχύτητα του βίντεο. Η Magix εξέδωσε και άλλες παραλλαγές του προγράμματος, με την 17.0 να καινοτομεί με την συμπερίληψη νέων λειτουργιών που στοχεύουν την βελτιστοποίηση του γενικότερων του προγράμματος. Για να λειτουργήσει σωστά το Vegas δεν απαιτείται κάποιο εξειδικευμένο υλικό, με αποτέλεσμα να λειτουργεί σε αρκετούς υπολογιστές με το αντίστοιχο λειτουργικό των windows.

1.4 Artbreeder

Το artbreeder είναι μία machine-learning ιστοσελίδα. Στην συγκεκριμένη σελίδα, οι χρήστες επιτρέπει στους χρήστες να πειραματιστούν και να τροποποιήσουν εικόνες προσώπων, τοπίων, ζωγραφιών και άλλων κατηγοριών. Πίσω από αυτή τη σελίδα, υπάρχει μία μεγάλη βάση δεδομένων από φωτογραφίες. Η διαδικασία αυτή πραγματοποιείται με την χρήση sliders, στην σελίδα της εκάστοτε εικόνας, αλλιώς γνωστά ως genes, δηλαδή γονίδια. Στα πορτραίτα, οι επιλογές ποικίλουν από την ρύθμιση του χρωματισμού, μέχρι και την τροποποίηση φύλου, μαλλιού και εθνικότητας. Επιπλέον, υπάρχει η δυνατότητα ανάμειξης δύο φωτογραφιών από την βάση και σε συνδυασμό με τα προαναφερθέντα sliders, επηρεάζουν το αποτέλεσμα - παιδί τους.

Η βασική τεχνολογία πίσω από το Artbreeder είναι το παραγωγικό αντιπαλικό δίκτυο (εν συντομία ΠΑΔ ή αλλιώς επίσης γνωστό ως Αντιπαλικό Δίκτυο). Το ΠΑΔ είναι μία τεχνική μηχανικής εκμάθησης, εφευρημένο από τον Ian Goodfellow. Σύμφωνα με αυτό, δύο νευρωνικά δίκτυα διαγωνίζονται σε ένα παίγνιο, με δεδομένο το σύνο εκπαίδευσης, η τεχνική αυτή, μαθαίνει να δημιουργεί νέα δεδομένα με τα ίδια στατιστικά στοιχεία. Είναι ένα φιλικό προς τον χρήστη σετ μοντέλων BigGAN που δημοσιοποιεί την ευκολία χρήσης του Data Visualization. Επιτρέπει στους χρήστες να δημιουργούν οπτικά αριστουργήματα από υπάρχοντα DNA σε μια διαδικασία συνεργασίας με άλλους χρήστες, καλλιτέχνες και θεατές, ακόμα κι αν δεν έχουν εκτεταμένες δεξιότητες Επιστήμης Δεδομένων. Το Artbreeder, λοιπόν, χρησιμοποιεί BigGAN και StyleGAN μοντέλα.

1.4.1 BigGAN

Το BigGan θεωρείται ως μία προσέγγιση που συγκεντρώνει μία σειρά από βέλτιστες

πρακτικές για την εκπαίδευση εικόνων και την κλίμά

1.4.2 StyleGAN

Τα ΠΑΔ γενικότερα είναι είναι αποτελεσματικά στη δημιουργία μεγάλων εικόνων υψηλής ποιότητας. Οι περισσότερες βελτιώσεις έχουν γίνει στα μοντέλα που κάνουν διάκριση σε μια προσπάθεια εκπαίδευσης πιο αποτελεσματικών μοντέλων γεννητριών, αν και λιγότερη προσπάθεια έχει καταβληθεί για τη βελτίωση των μοντέλων γεννήτριας.

Το Style Generative Adversarial Network, ή StyleGAN για συντομία, είναι μια επέκταση της αρχιτεκτονικής GAN που προτείνει μεγάλες αλλαγές στο μοντέλο της γεννήτριας. Παράλληλα χρησιμοποιεί ένα δίκτυο για τη χαρτογράφηση σημείων σε έναν ενδιάμεσο λανθάνοντα χώρο, για τον έλεγχο του στυλ σε κάθε σημείο του μοντέλου γεννήτριας και την εισαγωγή στο θόρυβο ως πηγή διακύμανσης σε κάθε σημείο του. Το μοντέλο που προκύπτει είναι ικανό όχι μόνο να δημιουργεί εντυπωσιακά φωτορεαλιστικές φωτογραφίες προσώπων υψηλής ποιότητας, αλλά προσφέρει επίσης έλεγχο του στυλ της παραγόμενης εικόνας σε διαφορετικά επίπεδα λεπτομέρειας μέσω της διαφοροποίησης των διανυσμάτων στυλ και του θορύβου.

2. Φέρνοντας στη ζωή τον πρωταγωνιστή

2.1 FPS Controller

Παίρνοντας έμπνευση από τα περισσότερα κλασικά παιχνίδια ψυχολογικού τρόμου, η αφήγηση της ιστορίας ορίστηκε ως πρωτοπρόσωπη, γι αυτό τον λόγο, προχώρησα στη δημιουργία ενός πρωτοπρόσωπου χειρισμού ή αλλιώς first person controller.

Οι λειτουργίες που θέλει να κάνει ο χαρακτήρας μου είναι οι βασικές.

α) Να μετακινείται στο χώρο με τα κουμπιά από το πληκτρολόγιο W,A,S,D (ή τα βελάκια)

β) να τρέχει πατώντας παρατεταμένα με το αριστερό shift

γ) να επηρεάζεται από τη βαρύτητα, δηλαδή άμα πέσει από κάποιο ύψος, να μην ‘‘παραμείνει’’ στην ίδια θέση

δ) να μην διέρχεται και κολλά σε αντικείμενα.

Στο περιβάλλον του Unity, η κίνηση μπορεί να πραγματοποιηθεί με δύο τρόπους:

α) Με Character controller component (περισσότερα παρακάτω)

β) rigidbody component.

Rigidbody είναι ο έλεγχος της θέσης ενός αντικειμένου μέσω προσομοίωσης φυσικής. Αυτό σημαίνει ότι με την προσθήκη ενός στοιχείου Rigidbody σε ένα αντικείμενο θα αρχίσει να χρησιμοποιείται η μηχανής φυσικής του Unity. Ακόμη και χωρίς την προσθήκη κώδικα, ένα αντικείμενο Rigidbody θα τραβηχτεί προς τα κάτω από τη βαρύτητα και θα αντιδράσει σε συγκρούσεις με εισερχόμενα αντικείμενα εάν υπάρχει αντίστοιχα και το σωστό στοιχείο Collider. Παράλληλα, το Rigidbody διαθέτει επίσης ένα API δέσμης ενεργειών που επιτρέπει την εφαρμογή δυνάμεων στο αντικείμενο, ώστε να ελέγχεται με φυσικό ρεαλιστικό τρόπο.

Ένας **Character Controller** επιτρέπει εύκολη κίνηση η οποία περιορίζεται από συγκρούσεις χωρίς να χρειάζεται να χρησιμοποιεί rigidbody. Ο Character Controller δεν επηρεάζεται από δυνάμεις και θα κινηθεί μόνο όταν κληθεί η συνάρτηση Move(). Θα πραγματοποιήσει την κίνηση αλλά θα περιοριστεί από συγκρούσεις (Collisions).

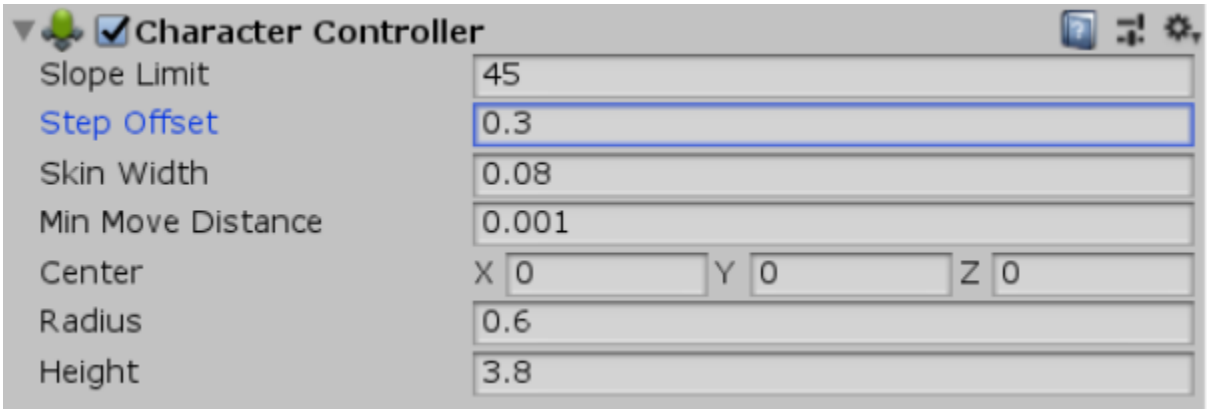
Και οι δύο έχουν τα πλεονεκτήματά τους, η επιλογή σε κάθε περίπτωση γίνεται με γνώμονα τι επιθυμούμε να υλοποιήσουμε στο παιχνίδι. Οι βασικές διαφορές τους φαίνονται στον

παρακάτω πίνακα:

CHARACTER CONTROLLER	RIGIDBODY
• Handles Steps	• Gravity built in
• Handles Slopes	• Drag built in
• Doesn't get stuck on walls	• Interacts with physics objects
• Easy to make snappy	

Λαμβάνοντας το παραπάνω υπόψη, επέλεξα ο χειρισμός του χαρακτήρα μου να υλοποιηθεί με character controller. Ξεκινώντας, δημιουργούμε στη unity ένα κενό αντικείμενο (Hierarchy > Create Empty), μετονομάζουμε το αντικείμενο σε FPS Controller και το μεταφέρουμε στο κέντρο της σκηνής μας, με drag and drop.

Στο παράθυρο του inspector προσθέτουμε τον component του character controller. Παρακάτω παρατίθενται οι ρυθμίσεις που όρισα στον χαρακτήρα μου. Στο step offset έβαλα 0.3, μιας και θέλω να πηγαίνει ο χαρακτήρας μου αργά όταν θα περπατάει κανονικά. Η προτίμηση αυτή είναι καθαρά υποκειμενική και μπορεί ο καθένας να ορίσει αυτό που προτιμά, ανάλογα και με το παιχνίδι που θέλει να αναπτύξει.



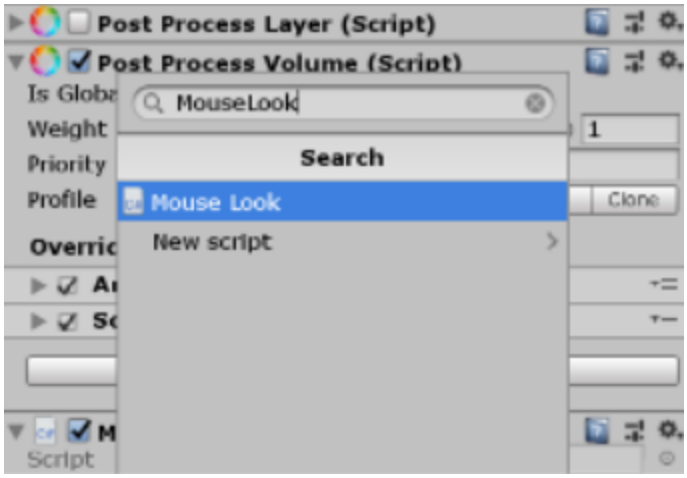
[Εικόνα 41]: Δήλωση τιμών στον Character Controller από τον Inspector

Έπειτα, στο αντικείμενό μου πρόσθεσα την κάμερα (FPS Controller > Right Click> Camera) και στο transform του inspect το μετακίνησα στον άξονα του y προς τα πάνω (θεωρητικά εκεί που θα βρίσκεται το κεφάλι), ώστε να λειτουργεί και κατά αυτό τον τρόπο. Η κάμερα αποτελεί τα μάτια του χαρακτήρα και κατά συνέπεια και του θεατή/παίκτη. Αυτό που πρέπει επίσης να προσέξουμε, είναι να μην είναι να μην βρίσκεται τελείως στην κορυφή, μιας και σε περίπτωση που προσθέτει μελλοντικά λειτουργία για jump και ο χώρος είναι μικρός, να μην περάσει η κάμερα από το ταβάνι.

Τώρα που δημιουργήθηκε ο χαρακτήρας στο χώρο, πρέπει να φτιάξουμε την λογική πίσω από αυτόν. Να κοιτά και να κινείται δηλαδή, μέσα σε αυτόν.

2.1.1.1 MouseLook Script

Η όραση πραγματοποιείται μέσα από την κίνηση του cursor του ποντικιού στον χώρο, στους άξονες x,y. Αν για παράδειγμα ο cursor κινηθεί στον άξονα του x (από πλευρά σε πλευρά δηλαδή), τότε θέλουμε περιστραφεί το σώμα του παίκτη στον άξονα του y. Αυτό μας επιτρέπει να κοιτάξουμε τριγύρω. Αντίστοιχα, αν θέλουμε να κοιτάξουμε πάνω-κάτω, πρέπει να κινηθεί η κάμερα στον άξονα του x, μέχρι τις 180 μοίρες. Έχοντας επιλεγμένη την κάμερα στο hierarchy του project μας, προσθέτουμε από το inspect έναν νέο component όπου το ονομάζουμε MouseLook (το όνομα του script μας). Πατώντας πάνω του δύο φορές, ανοίγει ένα κενό script με Visual Code Studio, όπου θα το επεξεργαστούμε.



[Εικόνα 42]: Add Script Component από τον Inspector

Από προεπιλογή, κάθε script έχει δημιουργημένες τις μεθόδους Start και Update. Η start καλείται μία φορά, όταν ξεκινά το παιχνίδι μας, ενώ η update κάθε φορά που ανανεώνεται η οθόνη μας, ανά frame (καρέ) δηλαδή. Για παράδειγμα αν ο ρυθμός ανανέωσης είναι 30fps, τότε η update θα κληθεί 30 φορές το δευτερόλεπτο.

Στην update, αποθηκεύουμε σε μία float μεταβλητή mouseX την κίνηση που λαμβάνουμε από τον άξονα x του ποντικιού, επί την ταχύτητα κίνησής του (mouse sensitivity). Το Mouse X που ορίζουμε μέσα στην GetAxis, είναι προκαθορισμένο από το ίδιο το Unity. Το mouseSensitivity είναι μία δικιά μας μεταβλητή, όπου έχουμε προκαθορισει με τιμή 100f. Το ίδιο κάνουμε και για τον άξονα του y. Τέλος, την mouseX και mouseY, τις πολλαπλασιάζουμε κάθε μία ξεχωριστά με το Time.DeltaTime. Το Time.DeltaTime είναι ο χρόνος που πέρασε από την στιγμή που κλήθηκε η update(). Αυτό το κανουμε, μιας και δεν θέλουμε, αν έχουμε υψηλότερο frame rate, να πηγαίνει η κάμερα πιο γρήγορα, από ότι σε ένα χαμηλότερο framerate.

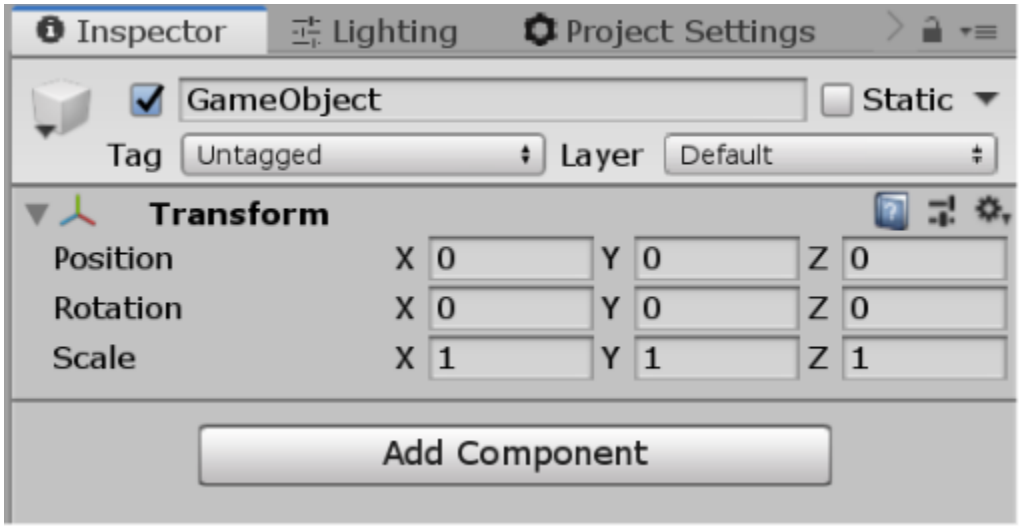
Για να περιστρεψουμε τον παίκτη στον άξονα του y με την κίνηση στον x, πρέπει να δημιουργηθεί μία public αναφορά για το σώμα/αντικείμενο του παίκτη. Δηλώνουμε πάνω-πάνω στον κωδικα μας μία public μεταβλητή playerBody και στην update, καταχωρούμε στο playerBody το Vector3.up (όπου είναι ο άξονας y) πολλαπλασιαζόμενο με το mouseX (το πόσο έχουμε κουνήσει το ποντίκι). Για να μετακινήσουμε την κάμερα πάνω-κάτω, δηλώνουμε μία float μεταβλητή, xRotation και στην update δηλώνουμε ότι άνα frame, η τιμή της μειώνεται όσο το

mouseY. Ο λόγος που μειώνουμε, αντί να αυξάνουμε, είναι διότι η περιστροφή στην δεύτερη περίπτωση αναστρέφεται από εκεί που θέλουμε. Έπειτα, για να εφαρμόσουμε την τιμή του xRotation καλούμε την κλάση transform.localRotation() και της καταχωρούμε την τιμή του Quaternion.Eyler(xRotation, 0f, 0f).

Σύμφωνα με το documentation της unity:

Η transform (class) αποτελεί έτοιμη κλάση (για c#) και component που μας παρέχεται από την unity. Με αυτή ορίζουμε την θέση, το μέγεθος και την περιστροφή του αντικειμένου μέσα στο χώρο. Κάθε αντικείμενο από default στα components του την transform, η οποία δεν γίνεται να αφαιρεθεί.

Παρακάτω παρατίθενται οι αρχικοποιημένες τιμές της transform με το που δημιουργείται ένα αντικείμενο.



[Εικόνα 43]: Transform component από τον Inspector

localRotation (method): Η περιστροφή του transform σε σχέση με την περιστροφή του transform του γονέα.

Quaternions (struct): Τα Quaternions (ή αλλιώς στα ελληνικά τεταρτημόρια) χρησιμοποιούνται

για να αναπαραστήσουν τις περιστροφές. Το Unity χρησιμοποιεί εσωτερικά τα Quaternions για να αναπαραστήσει όλες τις περιστροφές.

Euler (method): Επιστρέφει μία περιστροφή όπου περιστρέφει z μοίρες γύρω από τον άξονα του z, x μοίρες γύρω από τον άξονα του x και y μοίρε γύρω από τον άξονα του υ, εφαρμοσμένα με αυτή τη σειρά.

Ορίζουμε τα όρια όπου θα κινείται η κάμερα, το κεφάλι του χαρακτήρα μας, στον άξονα του y, ώστε να μην 'κοιτάξουμε' πίσω. Οι τιμές του xRotation, δηλαδη, να κυμαίνεται από τις -90 μοίρες, μέχρι τις 90 (αν 0, είναι το κέντρο μάς, εκεί όπου θα κοιτάει ευθεία). Αυτό γίνεται με την Mathf.Clamp(xRotation, -90f, 90f).

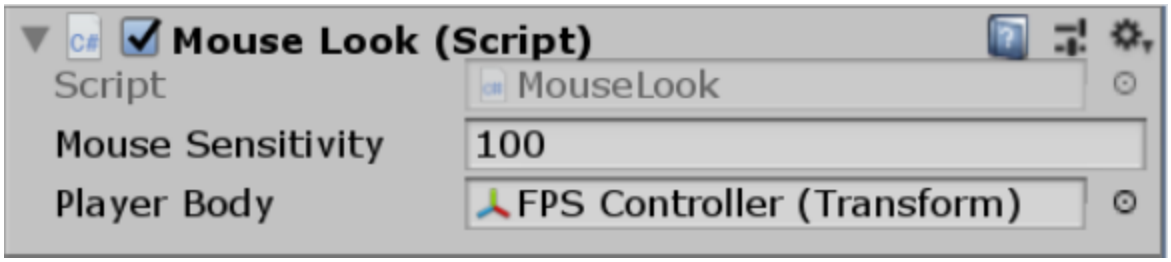
Mathf (struct): Μια συλλογή από κοινές μαθηματικές συναρτήσεις

Clamp (class): public static float Clamp(float value, float min, float max), επιστρέφει ένα float αποτέλεσμα ανάμεσα στην πιο χαμηλή και μέγιστη τιμή που της έχουν ορίσει.

Τέλος, στην Start() εκτελούμε την Cursor.lockState = CursorLockMode.Locked, ώστε να κρύψουμε και να κλειδώσουμε τον κέρσορα στο κέντρο της οθόνης.

Cursor (class) : Cursor API for για την ρύθμιση του κέρσορα (mouse pointer).
lockState : Καθορίζει εάν ο δείκτης υλικού είναι κλειδωμένος στο κέντρο της οθόνης, περιορισμένος στο παράθυρο ή δεν περιορίζεται καθόλου.
CursorLockMode (enumeration) : Καθορίζει πως ο κέρσορας πρέπει να συμπεριφέρεται
Locked: Κλείδωμα του κέρσορα στο κέντρο του παραθύρου του παιχνιδιού.

Μόλις ολοκληρωθεί το script και επιστρέψουμε πάλι στο περιβάλλον του unity, στα components της κάμερας, όπου έχουμε το script μας, ορίζουμε με drag and drop από την ιεραρχία, τον FPS Controller στο Player Body, όπως φαίνεται και στην εικόνα.



[Εικόνα 44]: Δήλωση μεταβλητών του script Mouse Look από τον Inspector

2.1.1.2 PlayerMovement Script

Επόμενο βήμα, είναι η προσθήκη κίνησης στον χαρακτήρα. Αυτό μπορεί να πραγματοποιηθεί μέσα από την συλλογή πληροφορίας στον άξονα x και z. Όπως υπάρχουν έτοιμες προκαθορισμένες τιμές για το Mouse X και Mouse Y, έτσι και στην συγκεκριμένη περίπτωση το unity, έχει ορίσει κάποια κλειδιά (keys) από το πληκτρολόγιο που μπορούν να χρησιμοποιηθούν για την κίνηση.

KEY	FUNCTIONALITY
W / ↑	Character moves forward. By default the value of the vertical = 1.
A / ←	Character moves left. By default the value of the horizontal = -1.
S / →	Character moves right. By default the value of the horizontal = 1.
D / ↓	Character moves backwards. By default the value of the vertical = -1.

Παράλληλα, θέλουμε ο χαρακτήρας μας να κινείται μπροστά, σε σχέση την κατεύθυνση που κοιτάει. Επιλέγουμε τον χαρακτήρα από την ιεραρχία του project, δηλαδή τον FPS Controller.

Προσθέτουμε, ένα νέο component, που θα ονομαστεί PlayerMovement. Αρχικά, δημιουργούνται δύο μεταβλητές x και z οι οποίες λαμβάνουν το input του χρήστη στον άξονα x και z αντίστοιχα.

Float x = Input.GetAxis(“Horizontal”).

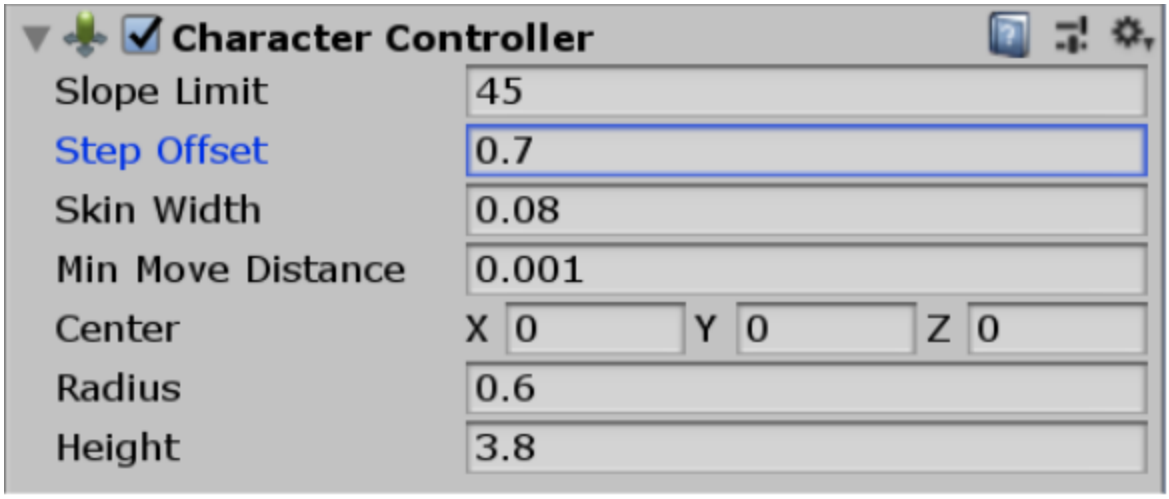
Input (class): Interface into the Input system. Use this class to read the axes set up in the Conventional Game Input, and to access multi-touch/accelerometer data on mobile devices. To read an axis use Input.GetAxis with one of the following default axes: "Horizontal" and "Vertical" are mapped to joystick, A, W, S, D and the arrow keys. "Mouse X" and "Mouse Y" are mapped to the mouse delta.

Δημιουργούμε μία μεταβλητή τύπου Vector3, ονομαζόμενη move. Εκεί θα οριστεί η τιμή transform.right * x (παίρνουμε την κατεύθυνση που κοιτάει ο παίκτης και κινειται προς τα δεξιά). Προσθέτουμε το transform.forward * z (παίρνουμε την κατεύθυνση που κοιτάει ο παίκτης και κινείται μπροστά). Δημιουργούμε μία αναφορά στον character controller, ώστε να μπορέσει να εφαρμοστεί η κίνηση κάπου και κάτω από την move εχουμε controller.Move(move). Δημιουργούμε και μία μεταβλητή speed, για την ταχύτητα με την οποία θα κινείται ο παίκτης. Από προεπιλογή, έχει την τιμή 5 και την πολλαπλασιάζουμε με το move στο controller.Move(move * speed) με επιπλέον μαζί * Time.Delta ώστε να μην επιρρεάζεται από το framerate του παιχνιδιού.

Vector3 (struct): Representation of 3D vectors and points. This structure is used throughout Unity to pass 3D positions and directions around. It also contains functions for doing common vector operations.

Move (method): Supplies the movement of a GameObject with an attached CharacterController component. The CharacterController.Move motion moves the GameObject in the given direction

Αλλάζουμε το stepOffset του CharacterController στον FPS Controller σε 0,7 , ώστε να μπορεί να ανέβει σκάλες.



[Εικόνα 45]: Δήλωση μεταβλητών του script CharacterController από τον inspector

Η μεταβλητή Slope Limit ορίζει την γωνία που θα ανέβει τις σκαλες, όπου αυτή τη στιγμή είναι ορισμένη στις 45 μοίρες. Ο χαρακτήρας πρέπει να επηρεάζεται από το φαινόμενο της βαρύτητας. Έχουμε το σκεπτικό ότι με την πάροδο του χρόνου η ταχύτητα με την οποία πέφτει ο χαρακτήρας, αυξάνεται. Δημιουργούμε μία Vector3 μεταβλητή με όνομα velocity, όπου velocity.y (η βαρύτητα ορίζεται στον άξονα του y) += gravity * Time.DeltaTime. Gravity είναι μία public μεταβλητή που έχουμε ορίζει έξω από την update. Από default, ορίσαμε την τιμή -9.81 και έχουμε Time.DeltaTime, γιατί είναι ανεξάρτητο του framerate.

```
Controller.Move(velocity * Time.DeltaTime)
```

Πολλαπλασιάζουμε πάλι με το Time.DeltaTime, καθώς εφαρμόζουμε την εξίσωση της ελεύθερης πτώσης που είναι $y = ½*g + t^2$.

Αφήνοντας το script ως έχει δημιουργείται το εξής πρόβλημα. Όσο ο χαρακτήρας πέφτει από κάποιο ύψος, θα αυξάνεται και η ταχύτητα με την οποία θα πέφτει κάθε φορά. Για αυτό το λόγο θα πρέπει να μηδενίζεται το velocity, κάθε φορά που αγγίζει το έδαφος. Στον FPS Controller δημιουργούμε ένα νέο κενό αντικείμενο και το ονομάζουμε groundCheck. Αυτή θα αποτελεί ένα αντικείμενο το οποίο ελέγχει κάθε φορά αν ο παίκτης στέκεται κάπου. Αυτό θα πραγματοποιηθεί με την χρήση μιας μεθόδου σφαίρας. Μετατρέπουμε δηλαδή το αντικείμενο μας σε σφαίρα,

έχοντας την λογική ότι κάθε φορά που εισέρχεται το ‘έδαφος’ στην ακτίνα της, τότε σημαίνει ότι το αγγίζουμε. Αυτό το αντικείμενο, το μετακινούμε στο τέλος του χαρακτήρα (είναι τα νοητά πόδια του).

Δημιουργούμε:

1) ένα reference στο script μας με το groundCheck
2) μία μεταβλητή τύπου LayerMask, την groundMask (ώστε να ελέγχουμε τι αγγίζει το αντικείμενο μας, αν είναι στο layer του εδάφους.)
3) Μία private boolean μεταβλητή isGrounded, για το αν έχουμε αγγίξει το έδαφος ή όχι.
4) Μία float μεταβλητή groundDistance που αποτελεί την ακτίνα της σφαίρας. Την θέτουμε αρχικά ως 0,4f. Την ορίζουμε επίσης ως public, για να μπορούμε να μεταβάλλουμε την τιμή της από το παράθυρο του inspector.

Στην αρχή της Update() ελέγχουμε αν είμαστε στο έδαφος. Αυτό το κάνουμε ως εξής:

isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask). Δημιουργεί μία σφαίρα με βάση το που βρίσκεται το αντικείμενο groundCheck (στα πόδια του παίκτη) με ακτίνα την τιμή του groundDistance.

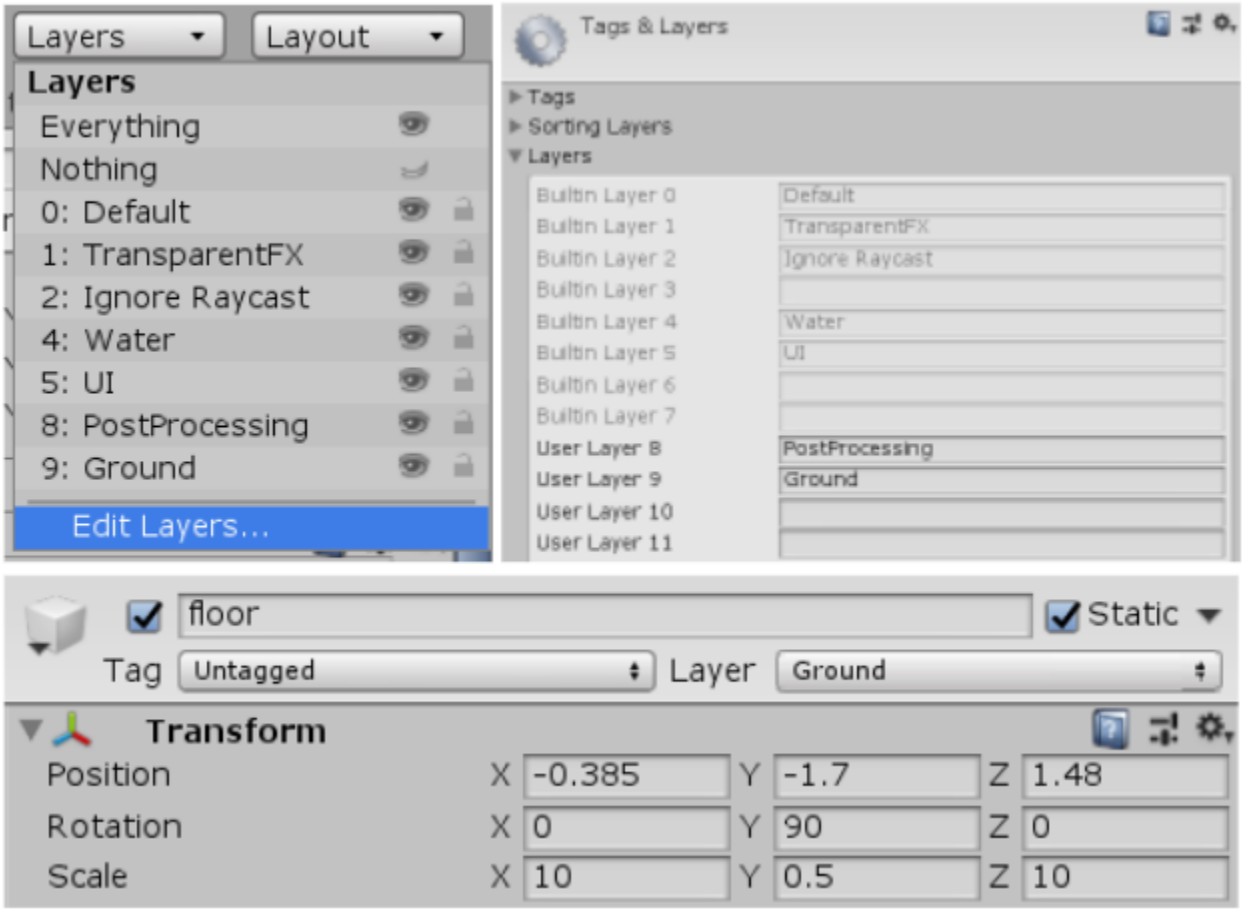
Έπειτα ελέγχουμε την τιμή της. Αν είναι true (αλλά και η ταχύτητα στον άξονα y είναι μεγαλύτερο του 0-velocity.y), τότε ο παίκτης έχει αγγίξει το έδαφος έπειτα από κάποια πτώση. Οπότε θέτουμε στο velocity.y έναν αρνητικό αριθμό. Στα layers του περιβάλλοντός μου, έχω δημιουργήσει ένα δικό μου, με όνομα Ground, όπου είναι το έδαφος. Παραλληλα, ότι αντικείμενα χρησιμοποιούνται ως ground, του έχω ορίσει ότι βρίσκονται στο αντίστοιχο layer.

LayerMask (struct) : Specifies Layers to use in a Physics.Raycast.

Physics.Raycast: Casts a ray, from point (origin), in direction, of length (maxDistance), against all colliders in the Scene

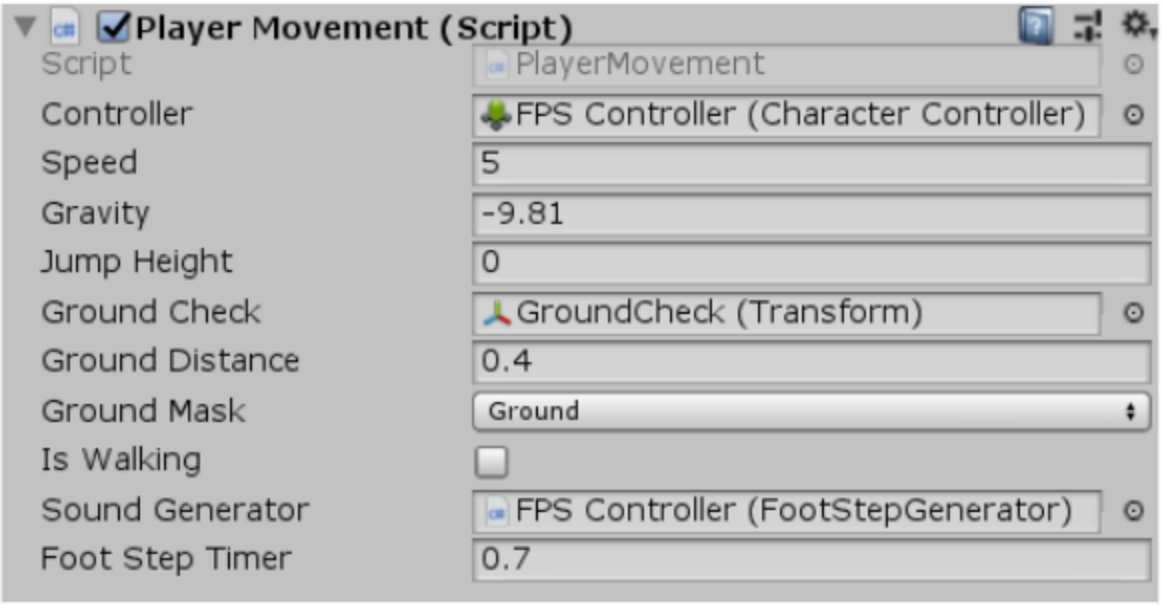
Physics (module) : implements 3D physics in Unity.

Physics.CheckSphere(): Returns true if there are any colliders overlapping the sphere defined by position and radius in world coordinates



[Εικόνα 46]:Δημιουργία καινούργιου Layer όπου θα αντιστοιχεί στο έδαφος

Οπότε στον inspector, ορίζουμε σαν GroundMask το Ground, το έδαφος.



[Εικόνα 47]: Δήλωση μεταβλητών του script PlayerMovement από τον inspector

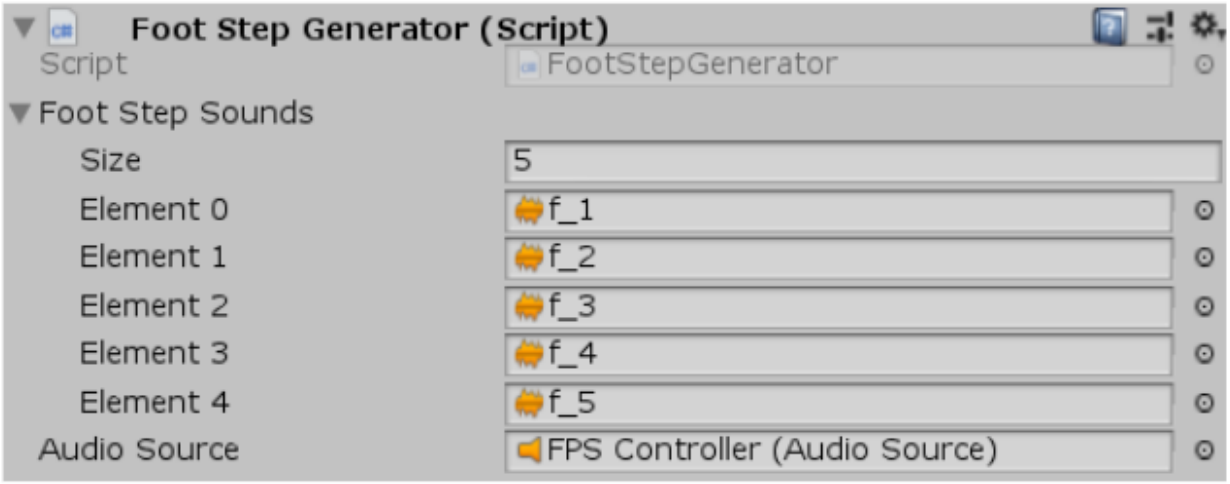
Αφού ο βασικός μας FPSController είναι έτοιμος, ήθελα να προσθέσω κάποιες λειτουργίες σε αυτόν. Όπως είναι να παράγεται ήχος κάθε φορά που κινείται και η δυνατότητα να μπορεί να τρέξει όσο πατάει παρατεταμένα το shift πλήκτρο.

Δημιουργούμε ένα νέο script, με όνομα FootStepGenerator. Μέσα σε αυτό δηλώνεται μία λίστα τύπου AudioClip με όνομα footStepSound.

> AudioClip (class): A container for audio data. An AudioClip stores the audio file either compressed as ogg vorbis or uncompressed. AudioClips are referenced and used by AudioSources to play sounds.

Με drag and drop μεταφέρουμε το script στον FPSController και στον inspector δηλώνουμε τους ήχους που θα χρησιμοποιηθούν για τους βηματισμούς του χαρακτήρα. Επέλεξα να μην χρησιμοποιήσω του ήχους που μου παρείχε ο default FPSController του Unity, μιας και μου φαινόταν αρκετά τυποποιημένος. Έτσι ανέτρεξα στο youtube και επέλεξα αυτό που με εξυπηρετούσε περισσότερο. Στο συγκεκριμένο αρχείο ήχου υπήρχε αρκετή ποικιλία, οπότε μόλις

το κατέβασα, το έκανα import στο Vegas και το έκοψα σε αρκετά, διαλέγοντας στο τέλος τα 5 που μου εξυπηρετούσαν περισσότερο. Σαν αποτέλεσμα αυτής της διαδικασίας, έχουν παραχθεί 5 αρχεία ήχου τύπου mp3, τα οποία στην συνέχεια, έγιναν import στο unity και δηλώθηκαν ως στοιχεία της λίστας του footStepGenetator.



[Εικόνα 48]: Δήλωση μεταβλητών του script Footstep generator από τον inspector

Στον FPSController δηλώνουμε τον τρεις μεταβλητές:

1) Την soundGenerator τύπου FootStepGenerator (αυτό που μόλις δημιουργήσαμε)
2) Μία boolean isWalking με αρχική τιμή false και
3) Μία public (για να μπορούμε να μεταβάλλουμε την τιμή της και από τον inspector) footStepTimer, που σχετίζεται με τα δευτερόλεπτα ανάμεσα σε κάθε βήμα.

Ξεκινώντας, στην Update() ελέγχουμε αν ο παίκτης κινείται στον άξονα x και z (ο άξονας y σχετίζεται με το ύψος). Αν αυτό ισχύει, τότε ελέγχουμε την τιμή του isWalking. Όσο το isWalking είναι false, τότε δεν παράγεται κάποιος ήχος. Αν η τιμή της είναι true, τότε καλούμε την PlayFootSound().

Η PlayFootSound() καλεί την μέθοδο StartCoroutine, η οποία παίρνει σαν παραμέτρους μία μέθοδο τύπου IEnumerator και έναν αριθμό, στην συγκεκριμένη περίπτωση τον PlayStepSound (περισσότερα παρακάτω) και footStepSound αντίστοιχα.

IEnumerator: the base interface for all non-generic enumerators

StartCoroutine(): Starts a Coroutine. The execution of a coroutine can be paused at any point using the yield statement. When a yield statement is used, the coroutine pauses execution and automatically resumes at the next frame.

Φτιάχνουμε την μέθοδο PlaySound τύπου IEnumerator. Σε αυτή καλούμε την Random.Range(0,5) η οποία επιλέγει έναν τυχαίο αριθμό από το 1 έως το 5 (όσοι είναι και οι ήχοι μας) και την καταχωρεί το αποτέλεσμα σε μία μεταβλητή randomIndex. Έπειτα:

1) παίζουμε τον ήχο από αντιστοιχίζεται στον πίνακα soundGenerator, με δείκτη τον αριθμό που δημιουργήσαμε μόλις.
2) Θέτουμε το isWalking σε true, αφού υπάρχει βηματισμός

Μέχρι στιγμής έχουμε έναν FPSController ο οποίος κινεί τον παίκτη και παράγεται ήχος με κάθε του βήμα. Η λογική με το τρέξιμο είναι απλή. Πρέπει να ελέγχουμε συνέχεια αν ο παίκτης έχει πατημένο το Shift στο αριστερό μέρος του πληκτρολογίου. Αν αυτό ισχύει, τότε η ταχύτητα με την οποία θα κινείται θα αυξάνεται. Οι τιμές που μας ενδιαφέρουν είναι οι speed και η footStepTimer, οποίες είναι αντιστρόφως ανάλογες. Αυτό επειδή όσο πιο γρήγορα κινείται ο παίκτης, ο χρόνος ανάμεσα στον ήχο που θα ακούγεται με κάθε βήμα, θα είναι όλο και πιο μικρός. Οπότε, στην update του FPSController εχουμε Input.GetKeyDown(KeyCode.LeftShift). Αν το αποτέλεσμα που επιστρέφει είναι true, τότε:

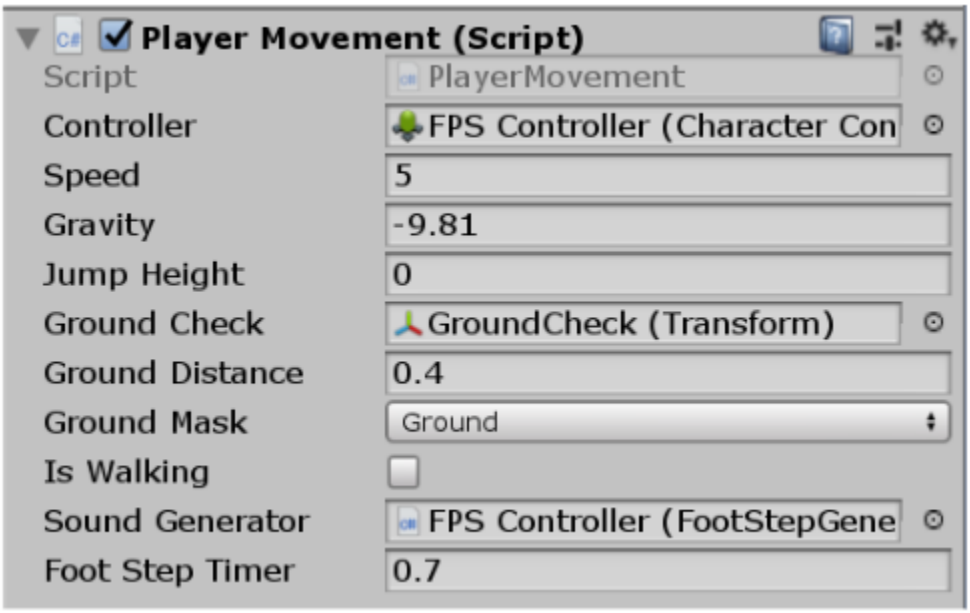
1) η τιμή της μεταβλητής speed αλλάζει σε 12f.
2) ο footStepTimer ορίζεται ως 0.3f.

Από κάτω ακριβώς ελέγχουμε αντίστοιχα έχουμε Input.GetKeyUp(KeyCode.LeftShift) για το αν ο παίκτης έχει αφήσει το πλήκτρο. Αν επιστρέψει true, τότε:

1) θέτουμε πάλι την τιμή του speed σε 5f.

2) ο footStepTimer ορίζεται ως 0.7f.

Input.getKeyDown(): Returns true during the frame the user starts pressing down the key identified by name. The state gets reset each frame. It will not return true until the user has released the key and pressed it again.



[Εικόνα 49]: Δήλωση μεταβλητών του Player Movement script από τον inspector

2.2. Πίσω από την κάμερα

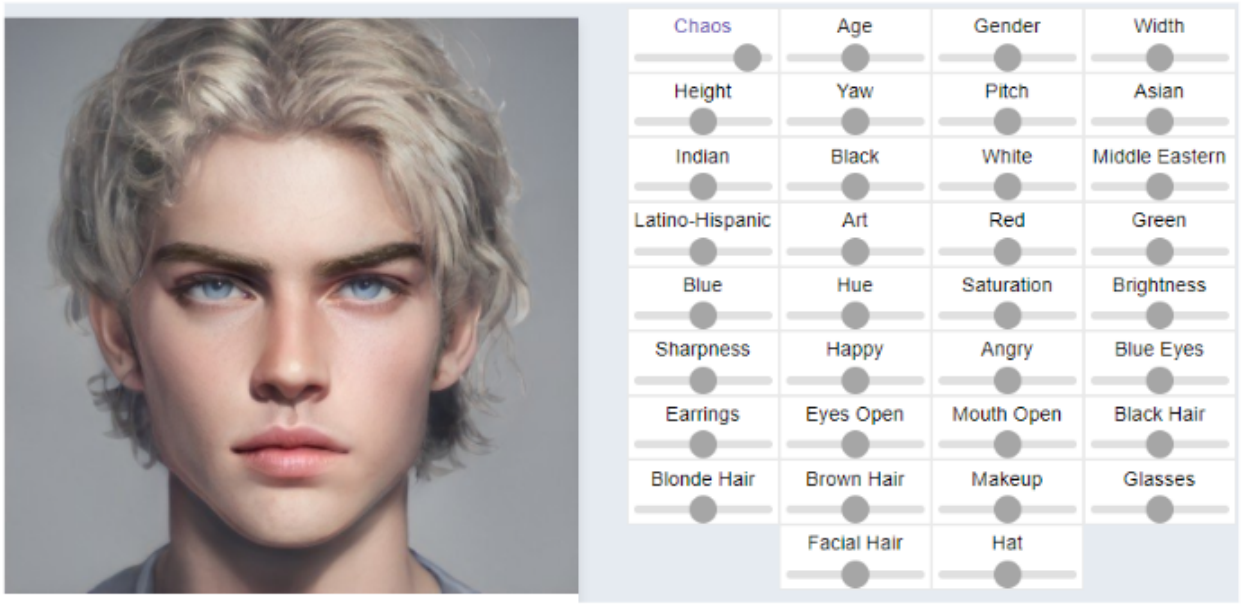
Ένα παιχνίδι ώστε να είναι ρεαλιστικό, όπως και με τα περισσότερα μέσα αφήγησης μιας ιστορίας, πρέπει να έχει βάθος, τόσο με τους χαρακτήρες της, όσο και με το περιβάλλον γύρω του (Character Building και World Building).

2.2 Character Building

Ξεκινώντας είχα στο πίσω μέρος του μυαλού μου δύο βασικά ερωτήματα.

1) Ποιος είναι ο παίκτης;
2) Ποια είναι η κινητήρια δύναμη που εξελίσσει τα γεγονότα;

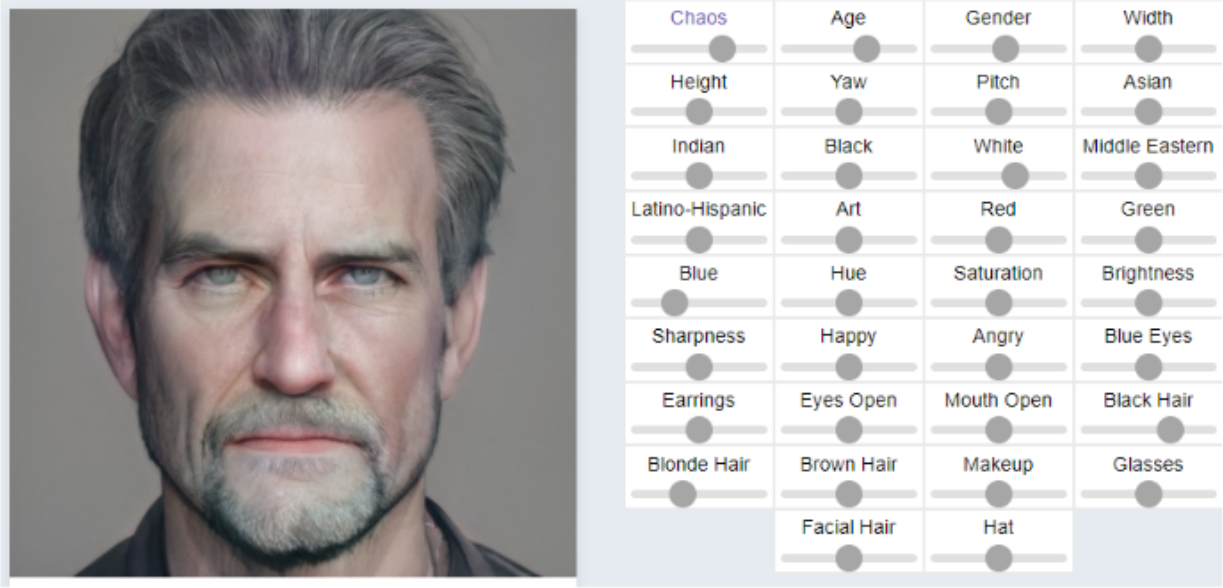
Εμπνευσμένη από το μυθιστόρημα του Fiodor Dostojewski, προχώρησα στην σκιαγράφηση του προφίλ του κεντρικού μου χαρακτήρα, τον Christopher Wilson. Για την οπτικοποίηση χρησιμοποίησα το Artbreeder. Η διαδικασία είναι αρκετά απλή και είναι η εξής: Αφού συνδεθώ στον λογαριασμό μου στην ιστοσελίδα, επιλέγω μία από τις εικόνες στην αρχική, κατά προτίμηση αυτή που είναι πιο κοντά στο επιθυμητό αποτέλεσμα που έχω σκοπό να πετύχω. Μιας τυχαία κάνει επίσης, μιας και με τα sliders που υπάρχουν, η εικόνα μπορεί να τροποποιηθεί.



[Εικόνα 50]: Το User Interface του Artbreeder για την δημιουργία προσώπου

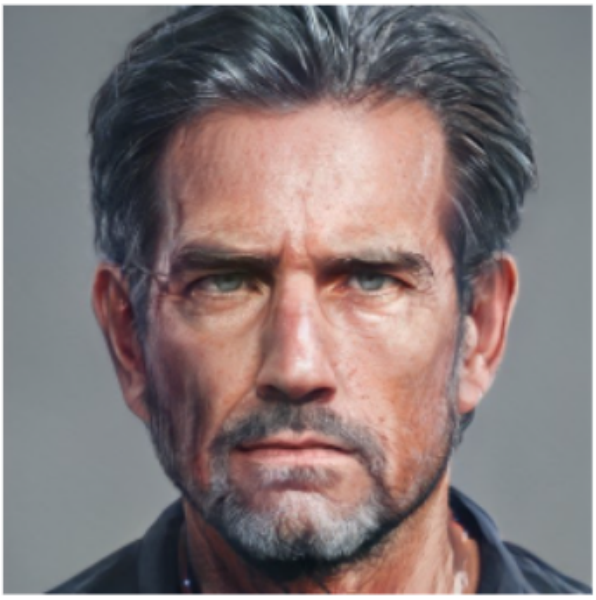
Κάθε slider από default βρίσκεται στο κέντρο της μπάρας και ανάλογα με τις προτιμήσεις, μετακινείται ανάλογα. Ο σκοπός μου, ήταν για αρχή να δημιουργήσω έναν άντρα, μεσήλικα, λευκό, με ευρωπαϊκά χαρακτηριστικά, να φαινεται έξυπνος και ψυχρός. Συνεπώς για αρχή

μετακίνησα το chaos (το οποίο είναι στις άκρες του έχει μία τυχαία γυναικεία εικόνα και μία ανδρική και ρυθμίζει το πρόσωπο ανάλογα), την ηλικία προς τα δεξιά, το φύλο, white, black hair από την ίδια κατεύθυνση, ενώ μείωσα το blonde hair, καταλίγοντας για αρχή στην παρακάτω



[Εικόνα 51]: Τροποποίηση των τιμών στα slider

Πειραματίστηκα ακόμα με τα sliders φτάνοντας σε ένα επιθυμητό αποτέλεσμα, το οποίο φαίνεται παρακάτω:



[Εικόνα 52]: Christopher Wilson

Σύμφωνα με τον Hirohiko Araki, συγγραφέα Manga και συγκεκριμένα της σειράς Jojo's Bizarre Adventure, στο βιβλίο του 'Manga in Theory and Practice: The Craft of Creating Manga' αναφέρει:

"When I'm creating a new character, before I do any drawing, I always write a character history [...] it's something akin to a chef's seasoning. It protects against inconsistencies in the character."

"Όταν δημιουργώ έναν καινούργιο χαρακτήρα, προτού κάνω κάποια ζωγραφιά, πάντα γράφω την ιστορία του χαρακτήρα [...] είναι κάτι παρεμφερές με τα μπαχαρικά που χρησιμοποιεί ένας σεφ. Προστατεύει ενάντια στις ασυνέπειες στον χαρακτήρα."

Στην συνέχεια, ο Araki παρουσιάζει έναν πίνακα τον οποίο χρησιμοποιεί για την ιστορία του χαρακτήρα, τον οποίο συμπλήρωσα και εγώ, για τον δικό μου χαρακτήρα.

Name/Nicknames	Christopher Wilson	Age	53

		Sex	Male		
Birthdate/Star Sign	06-14-1967 (Gemini)	Blood Type	A positive	Birthplace	Springfield, Illinois
Height	1,83	Weight	108	Hair Color	Gray-ish Black
				Eye Color	Blue
Eyesight / Glasses? / Colorblind?	Φοράει γυαλιά / Έχει κάποιους βαθμούς πρεσβυωπία	Handedness	Ναι, παρά το γεγονός ότι δεν είναι και αυτός 'άγιος'	Type of voice	Βαριά, ελαφρά Δανέζικη προφορά, μιας και είχε μείνει με την οικογένειά του κάποια χρόνια εκεί (όταν ήταν στο δημοτικό)
History of Surgeries/ Cavities / Illnesses	1) Σπασμένη επιγονατίδα. Είχε πέσει από το ποδήλατο στα 11. Νιώθει έντονους πόνους, σαν σουβλιές κάθε φορά που αλλάζει ο καιρός 2) Κοινωνιοπαθής, σύμπλεγμα του Θεού, ιδιαίτερα σε ζητήματα που σχετίζονται με την δουλειά του				
Scars, Burns/Skin Damage, Birthmarks, Tattoos	Ελαφριά ουλή στο αριστερό ζυγωματικό του, προκλήθηκε από το αυτοκινητιστικό. Λευκή κηλίδα στην αριστερή πλευρά των πλευρών του.				
Other Distinctive physical Characteristics / Nose & Eye Shape	Βαθούλωμα στο πηγούνι Κουκουλωμένα μάτια Σωματοτυπος δρομεα, ωστόσο λόγω κακών	Race	Αμερικανική	Religion	Άθεος

	διατροφικών συνηθειων εχει πλαδαρη κοιλιά.				
Criminal Record / Awards/ Education					
Formative Experiences as a infant/ young child (including who involved)	Πτυχίο στην ιατρική, πρώτος στο έτος του. Αρκετά βραβεία σε στίβου από το πανεπιστήμιο.				
Sexual Hisotry / Lovers / Thoughts Towards Marriage / Romance	Σεξιστής, αρομαντικός, έχει την πεποίθηση ότι το αντίθετο φύλο είναι μέσο για την ικανοποίηση σεξουαλικών επιθυμιών. Χωρισμένος και κατά διαστήματα έχει διαπράξει μοιχεία.				
People the subject looks up to / People the subject hates					
Dreams for the future	-				
Fears	Κλειστοφοβία				
Personality Traits / Favorite Sayings, habits	Τζόκινγκ τις πρωινές ώρες, αλκοολικός (μετά το ατύχημα), κάπνισμα				
Relationships/Family (incl. Behaviour)	 **Edith** (πρώην σύζυγος): Άσχημος χωρισμός. Σύντομο διάστημα μεταξύ γνωριμίας και γάμου. Επειδή δεν είχαν γνωριστεί καλά προτού, μόλις παντρεύτηκαν και άρχισαν να γνωρίζονται καλά, η σχέση τους είχε μία κατηφορική διαδρομή. Το ατύχημα που συνέβει, ήταν και το τελειωτικό χτύπημα, όπου μετά και χώρισαν. Είχε διαπράξει μοιχεία σε αρκετές περιπτώσεις, συνήθως μετά από έντονους τσακωμούς μεταξύ τους. Δεν του άρεσε να				

<table>
<tr><td></td><td colspan="5">φορά τη βέρα του. Η δικαιολογία του ήταν ότι εν ήθελε να την χάσει κατά τη διάρκεια χειρουργίου.

Annabeth (κόρη). Κατά τη διάρκεια της εγκυμοσύνης της Edith δεν μπορούσαν να μάθουν το φύλο επειδή, το παιδί είχε γυρισμένο την πλάτη, έχοντας σαν αποτέλεσμα να το μάθουν στην γέννα. Πίστευε ότι θα είχε γιο, μιας και από την μεριά της οικογένειάς του, η πλειοψηφία απαρτίζεται από αρσενικούς γόνους. Απογοητεύτηκε όταν είδε ότι είχε κόρη, ωστόσο όσο περνούσε ο καιρός άρχισε να την αγαπάει όλο και πιο πολύ. Το ατύχημα τον ισοπέδωσε ψυχολογικά, μιας και άρχισε να πίνει, για να ξεχαστεί, έχοντας σαν αποτέλεσμα να χειρουργεί μεθυσμένος και να λαμβάνει αποφάσεις κατά τη διάρκεια αυτού, όπου δεν ήταν ιδιαίτερα υπέρ των ασθενών. (<u>Σημείωση</u>: εξαιτίας των απόψεων που κατέχει σχετικά με το θηλυκό γένος, μόλις η κόρη του θα έφτανε την εφηβεία και θα γινόταν και αυτή, στην ουσία, σεξουαλικό αντικείμενο για κάποιον, θα αποκτούσε μία απάθεια/αδιαφορία προς το πρόσωπό της)
  </td></tr>
<tr><td>Employment/ School</td><td>Γιατρός/Χειρούργος</td><td>Economic Status/ Behaviour</td><td>Τυπική κατάσταση, μιας και συντηρεί μόνο ένα διαμέρισμα και τον ίδιο</td><td>Pets / Plants</td><td>-</td></tr>
<tr><td>Personality</td><td>Cheerful/Gloomy?

Humorous / Violent

Active? Sociable?

Intellectual? VIrtuous?

Expressive? Weakness/ Worries/ Unusual traits</td><td>What is distinctive about them?</td><td colspan="3">Ζοφερός, με μακάβριο χιούμορ, ζηλεύει εύκολα για πράγματα που δεν έχει, κτητικός για τις ιδιοκτησίες του (στις ιδιοκτησίες συμπεριλαμβάνοται και τα άτομα)</td></tr>
</table>

Hobbies / Recreations / Likes, Dislikes / Food, Clothing, Shelter / Habits, Favorite Phrases	Music/Newspapers, Books/ Magazines, Movies, Creative Pursuits Collections, Favorite/Least Favorite colour, Perfume / Cologne, Decor/Fashion, Location, People, Favorite Stores, brands, Favorite things, Wear necklace, rings? Taste / Drugs	Habits/Passions, Meaningless things	Τζόκινγκ, δοκιμή κρασιών, παλιά του άρεσε να επισκέπτεται εκθέσεις πινάκων. Αγαπημένος ζωγράφος:Φρανσίσκο Γκόγια. Αγαπημένο φαγητό: Κρεατόπιτα από μπριζόλα και συκώτι μόσχου Αγαπημένο χρώμα: Καφέ, πράσινο, κόκκινο Αγαπημένο κλασικό κομμάτι:Serenade for Strings in C Major op.48 του Τσαϊκόφσκι Ταινία: Μπλε βελούδο Απεχθάνεται: την αίσθηση του δέρματος σε επιπλα, ρουχα κλπ, κύκλους, σπίρες, γραβάτες.
Special Skills	Sports/Dance, Martial Arts, Guns, Driving, Language study / other certifications	Strengths and Weaknesses	Άριστη γνώση Δανέζικων, Γαλλικών και Γερμανικών

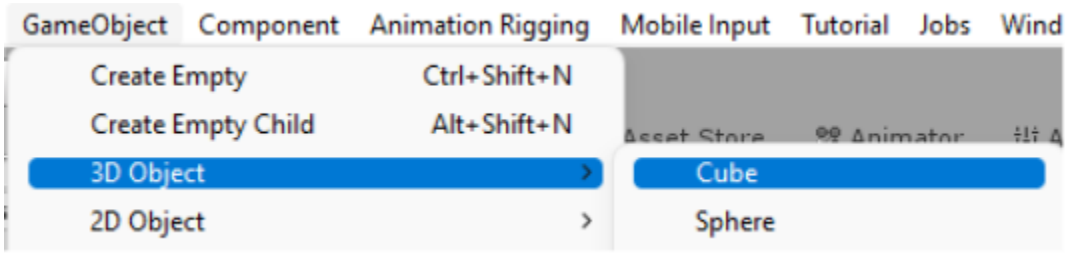
2.3 Σχεδίαση χώρου

Ο χώρος όπου διαδραματίζεται το παιχνίδι πρέπει να προσδίδει μία ιδιότητα ως προς τον παίκτη. Η αφετηρία της ιστορίας μας, αποτελεί παράλληλα, το μέσον όπου οι παίκτης θα γνωρίσει τον βασικό χαρακτήρα, αλλά και θα μάθει και τους χειρισμούς του παιχνιδιού.

2.3.1 Σπίτι

[Εικόνα 53 και 54]: Το καθιστικό στο διαμέρισμα του Christopher

Ένας χώρος αποτελείται από 3d αντικείμενα διαφόρων σχημάτων. Οι τοίχοι για παράδειγμα είναι κύβοι, οι οποίοι στους οποίους έχει τροποποιηθεί το μέγεθός τους. Για την εισαγωγή ενός κύβου στη σκηνή από το μενού: GameObject > 3D Object > Cude.



[Εικόνα 55]: Δημιουργία κύβου από το menu μέσα στο Unity

Ένας λευκός κύβος, απέχει αρκετά από το να μοιάζει με κανονικό τοίχο. Όπως όλα τα αντικείμενα στον χώρο, έτσι και αυτό πρέπει να διακατέχεται από κάποια συγκεκριμένα συστατικά:

Υλικό (Material):

Εκεί ορίζεται η εμφάνιση του αντικειμένου και επίσης αν θα είναι η επιφάνειά του αντανάκλαση και πόσο. Ένα material έχει παραμέτρους:

1) Albedo

Η συγκεκριμένη παράμετρος ελέγχει το βασικό χρώμα της επιφάνειας. Τις περισσότερες φορές είναι χρήσιμο ο ορισμός ενός συγκεκριμένου χρώματος., αλλά είναι πιο διαδεδομένη η χρήση ενός **texture (**αλλιώς ψηφιογραφική εικόνα πληροφορία υψής**)**, μιας υφής δηλαδή, σε αυτό το πεδίο. Αυτό αντιπροσωπεύει τα χρώματα της επιφάνειας του αντικειμένου. Αξίζει να σημειωθεί, ότι στην υψή που ορίζουμε στο Albedo, δεν θα πρέπει να περιέχει καθόλου φωτισμό, μιας και ο φωτισμός θα συμπεριληφθει σε αυτό ανάλογα με το γενικό πλαίσιο όπου βρίσκεται το αντικείμενο.

Επίσης στο Albedo μπορείς να ορίσεις πόσο διαφανές θα είναι ένα αντικείμενο, κάτι αρκετά χρήσιμο, σε περιπτώσεις που έχουμε γυάλινα ή πλαστικά αντικείμενα.

2) Metallic/Smooth

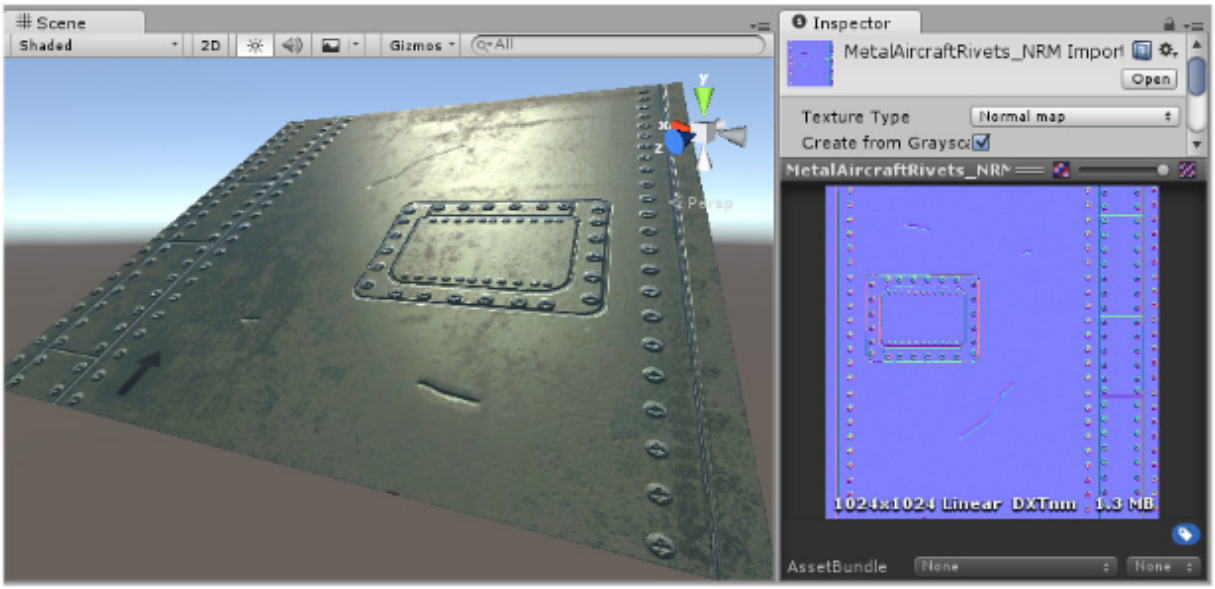
Πόσο αντανακλάσιμη θα είναι η επιφάνεια του αντικειμένου, δηλαδή. Η τροποποίηση του πραγματοποιείται με την μετακίνηση μιας μπάρας με τιμές που κυμαίνονται από το 0 ως το 1. Όσο πιο κοντά βρισκόμαστε στο 1, τόσο πιο πολύ αντανακλά το φως που υπάρχει στο περιβάλλον και το χρώμα που έχει ορισθεί το albedo γίνεται όλο και πιο εμφανές.

[Εικόνα 56]: Παράδειγμα υλοποίησης του material ενός ημιδιαφανούς αντικειμένου, του γυάλινου βάζου.

Στην παραπάνω εικόνα έχω επιλέξει το χρώμα του βάζου να είναι πράσινο. Για να το μετατρέψω σε διάφανο μετακίνησα τη μπάρα από το A χαμηλά φτάνοντας στην τιμή 73, όπου με ικανοποιούσε το αποτέλεσμα. Έπειτα, για να είναι η επιφάνειά μου αντακλάσιμη, αλλά όχι πολύ, ώστε θα φαίνεται καθαρά το χρώμα του, μετακίνησα την μπάρα του metallic λίγο λιγότερο κοντά από το κέντρο. Επίσης ήθελα να είναι αρκετά γυαλιστερή, οπότε πήγα στο smoothness περίπου σε ένα ποσοστό 85% της μπάρας.

3) Normal Map ή Bump Map (χαρτογράφηση προσκρούσεων)

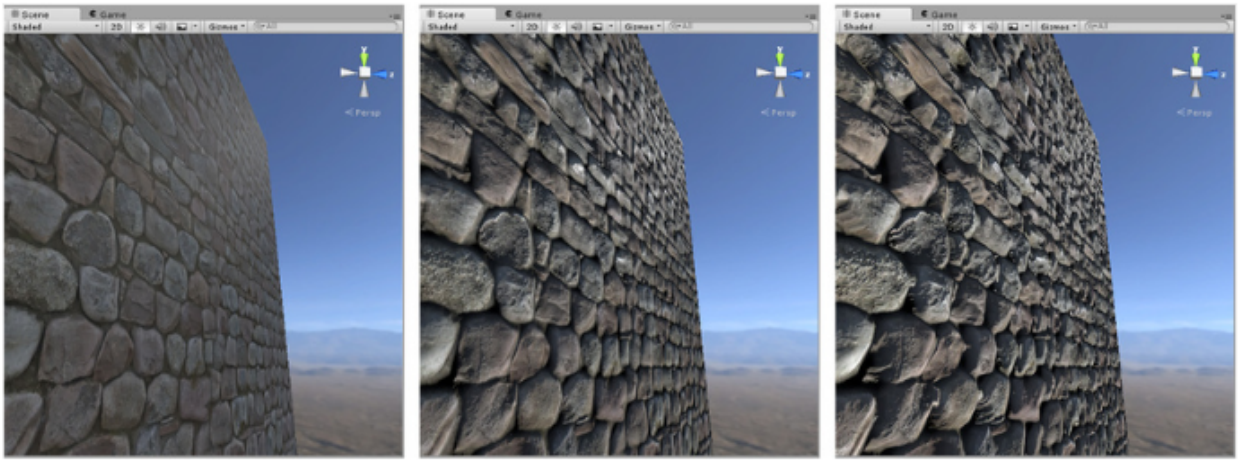
Ειδική τεχνική, όπου χρησιμοποιώντας την κατάλληλη εικόνα μπορούμε να προσομοιώσουμε χτυπήματα, ρυτίδες στην επιφάνεια του αντικειμένου. Αυτό συμβάλλει στο να φαίνεται το αντικείμενο πιο ρεαλιστικό, μιας και το φως που αντανακλάται σε αυτό φαίνεται έτσι σαν να υπάρχει κανονική γεωμετρία. Αν για παράδειγμα θέλουμε να φτιάξουμε ένα αεροπλάνο, όπου πάνω στα φτερά του έχει βίδες, αντί να κάτσει ο δημιουργός να σχεδιάζει τις βίδες με πολύγωνα, μία διαδικασία χρονοβόρα και δύσκολη, μπορεί να ορίσει ένα κατάλληλο texture στο normal map όπου θα έχει παρεμφερές αποτέλεσμα.



[Εικόνα 57]: Παράδειγμα χρήσης Normal Map για την μεγαλύτερη απόδοση λεπτομερειών

4) Height Map ή Parallax Mapping

Αποτελεί μία έννοια αρκετά παρεμφερής με αυτή του Normal Mapping, ωστόσο η τεχνική του είναι πιο περίπλοκη και κατά συνέπεια πιο δαπανηρή στην απόδοση. Τα height maps χρησιμοποιούνται συνήθως σε συνδυασμό με τα normal maps, προσφέροντας βάθος και λεπτομέρεια στο τελικό αποτέλεσμα. Ενώ τα normal maps τροποποιούν τον φωτισμο πάνω στην επιφάνεια, το height mapping, κάνει ένα βήμα παραπέρα και ουσιαστικά αλλάζουν την επιφάνεια του αντικειμένου. Όσο πιο κοντά βρισκόμαστε στην κάμερα, τόσο πιο έντονα θα είναι τα βαθουλώματα ή τα χτυπήματα, ενώ όσο απομακρυνόμαστε, τόσο θα μειώνονται.



[Εικόνα 58]: Διαφορές ως προς το βάθος των λεπτομερειών με την εξής σειρά: (1) Μόνο Albedo, (2) Σε συνδυασμό με Normal Map, (3) Σε συνδυασμό με Normal και Height Map

Στο πρώτο παράδειγμα έχουμε στο material μας μόνο το albedo. Στην δεύτερη εικόνα είναι το albedo σε συνδυασμό με το normal map. Οπως φαίνεται, υπάρχει περισσότερη λεπτομέρεια, μιας και οι σκιές είναι πιο έντονες, αλλά με την συμπερίληψη και του height map, βλέπουμε ότι τον τοίχο, ανάμεσα στα τούβλα υπάρχει βάθος.

5) Occlusion map

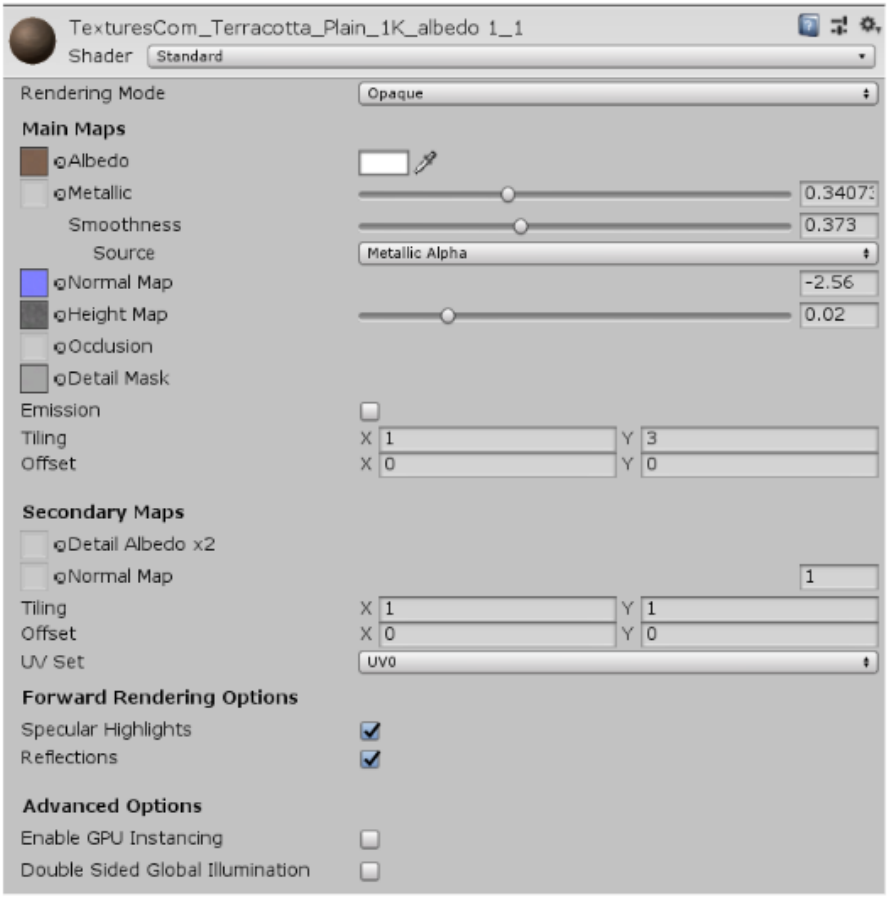
Χρησιμοποιείται ώστε να προσφέρει πληροφορία για το ποιες περιοχες στο μοντέλο θα πρέπει να λάβουν περισσότερο ή λιγότερο έμμεσο φωτισμό. Με την έννοια έμμεσο φωτισμό, εννοούμε το φως που προέρχεται από το περιβάλλον ή από άλλες αντανακλάσεις που υπάρχουν τριγύρω. Για παράδειγμα, μία κουκούλα στο εσωτερικό της πρέπει να έχει το δείκτη του occlusion όσο πιο χαμηλά γίνεται, καθώς δεν λαμβάνει σχεδόν καθόλου φως από το περιβάλλον.

6) Emission

Προσθετοντας emission σε ένα αντικείμενο το κάνουμε και να μοιάζει ως μία εμφανή πηγή φωτός μέσα στην σκηνή, το κάνουμε να φωσφορίζει δηλαδή. Αρκετά χρήσιμο σε περιπτώσεις που έχουμε μία ενεργή οθόνη

7) Secondary Maps (Detail Maps) & Detail Mask

Τα δευτερεύοντα maps επιτρέπουν στο να προσθέσεις ένα overlay ή ένα δεύτερο σετ από textures, προσδίδοντας μεγαλύτερη λεπτομέρεια. Στην ουσία, το σετ αυτό επαναλαμβάνεται σε πολύ μικρότερη κλίμακα πάνω στην επιφάνεια του αντικειμένου.



[Εικόνα 59]: Απόδοση λεπτομέρειας στην υφή του τοίχου, χρησιμοποιώντας Albedo, Normal Map, Height Map και Detail Mask

Βασική πηγή μου για την εύρεση textures αποτέλεσε η σελίδα textures.com, ενώ για τα αντικείμενα που υπάρχουν στο χώρο, τα βρήκα είτε μέσα από το Asset Store μέσα στο Unity, είτε μέσω της σελίδας Sketchfab.

BoxCollider:

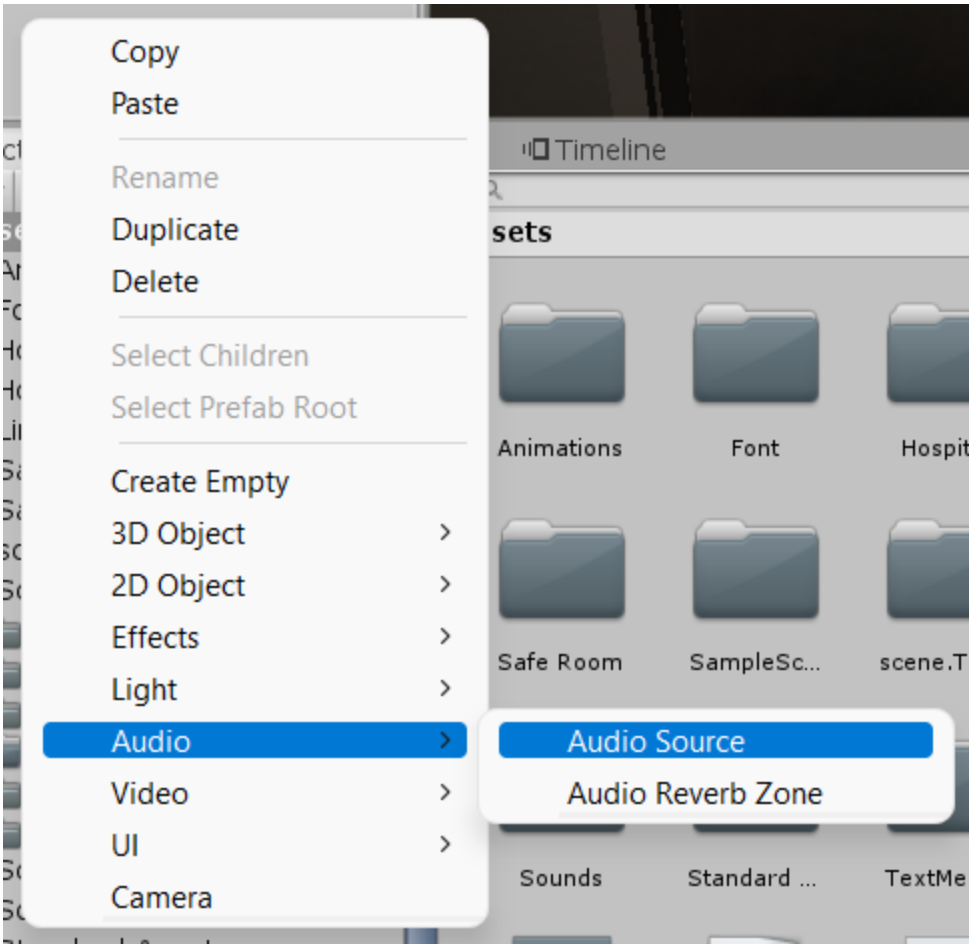
Αποτελεί ένα αόρατο αντικείμενο κυβικού μεγέθους όπου ο παίκτης συγκρούεται με αυτό και δεν του επιτρέπει να διέλθει από μέσα του. Χρησιμοποιείται ως επί το πλείστον σαν physics parameter σε αντικείμενα στο χώρο ή σε αόρατους κύβους οριοθετόντας τα όρια για το που μπορεί να κινηθεί ο παίκτης.

2.3.2 Ατμόσφαιρα και λειτουργίες

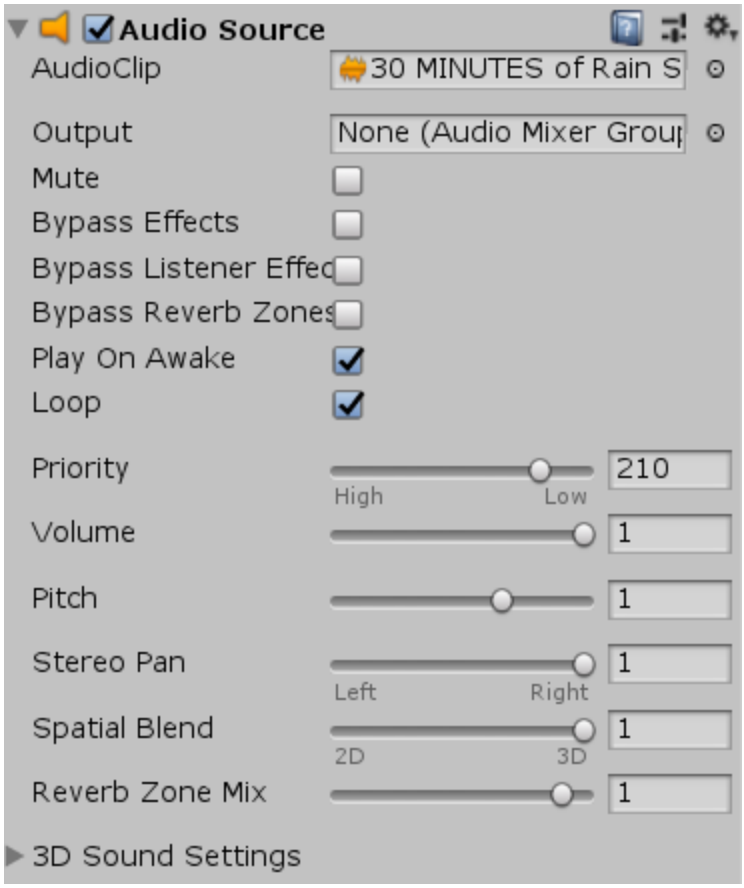
Ένα μέρος για να θεωρείται ρεαλιστικό πρέπει να βασίζεται στις λεπτομέρειες και στην αισθητική. Η αισθητική αποτελείται από τους περιβάλλοντες ήχους, τον φωτισμό, η σκόνη στην ατμόσφαιρα, το στήσιμο ενός χώρου. Στην συγκεκριμένη περίπτωση επιθυμούσα ο χώρος να έχει μία μουντή αισθητική. Για αυτό το λόγο συμπεριέλαβα βροχή να ρίχνει έξω από το παράθυρο, σε συνδυασμό με την αντίστοιχη ακουστική, μουσική να παίζει από το ράδιο, σωματίδια σκόνης να υπάρχουν στην ατμόσφαιρα κοκ.

Βροχή

Για την ακουστική, μέσα από την ιεραρχία δημιούργησα ένα αντικείμενο ήχου (Audio Object), με όνομα Rain Audio. Στον Inspector, στο κομμάτι του Audio Source, στο πεδίο AudioClip ορίζεται το αρχείο mp3 από το youtube. Κάτω, επέλεξα το Play on Awake, ώστε ο ήχος να αρχίζει μαζί με την εκκίνηση της σκηνής και loop. Σε αυτό το σημείο να σημειωθεί ότι ο ήχος αναπαράχθηκε μέχρι το τέλος του, ώστε να βεβαιωθώ ότι όταν ξεκινούσε από την αρχή, να είναι ομοιόμορφη η μετάβαση.



[Εικόνα 60]: Δημιουργία αντικειμένου ήχου μέσα από την ιεραρχία.



[Εικόνα 61]: Οι ρυθμίσεις του AudioSource στον Inspector

Το αντικείμενο αυτό, μετακινήθηκε μέσα στην σκηνή και τοποθετήθηκε σε δύο διαφορετικά σημεία. Το πρώτο, έξω από το παράθυρο της κουζίνας και το δεύτερο έξω από το αντικείμενο της μπαλκονόπορτας, εκεί όπου σε κανονικά θα ακουγόταν ο ήχος σε κανονικές συνθήκες.

Έπειτα, για να φαίνεται ότι όντως βρέχει έξω, προχώρησα στην δημιουργία shader όπου όταν θα είναι τοποθετημένο στο παράθυρο, θα φαίνονται σταγόνες βροχής. Δημιουργούμε ένα καινούργιο αντικείμενο unlit shader, όπου θα αντιστοιχεί στο αντικείμενο που θα δημιουργήσουμε.

Shader (class):

UV mapping: Η χαρτογράφηση UV είναι η διαδικασία τρισδιάστατης μοντελοποίησης προβολής μιας εικόνας 2D στην επιφάνεια ενός τρισδιάστατου μοντέλου για χαρτογράφηση υφής. Οι συντεταγμένες UV(W) είναι ένα κανονικοποιημένο (0 έως 1) δισδιάστατο σύστημα συντεταγμένων, όπου η αρχή (0,0) υπάρχει στην κάτω αριστερή γωνία του χώρου. Το UV είναι σε αυτή την περίπτωση συνώνυμο του XY, αλλά επειδή το XY χρησιμοποιείται στον τρισδιάστατο χώρο XYZ, χρησιμοποιούμε το UVW στον χώρο της υφής για να αποφύγουμε τη σύγχυση. Σημειώστε ότι υπάρχει ένα W που αντιστοιχεί στον άξονα Z, αλλά δεν θα χρησιμοποιηθεί σε ένα Texture2D καθώς είναι 2D.

frac(): στα μαθηματικά αφορά τη γραμμική συνάρτηση.

smoothstep(): Έτοιμη συνάρτηση της κλάσης Mathf. Παρεμβάλλει μεταξύ min και max με εξομάλυνση στα όρια. Αυτή η συνάρτηση παρεμβάλλεται μεταξύ min και max με παρόμοιο τρόπο με τον Lerp. Ωστόσο, η παρεμβολή θα επιταχυνθεί σταδιακά από την αρχή και θα επιβραδύνει προς το τέλος. Αυτό είναι χρήσιμο για τη δημιουργία κινούμενων εικόνων με φυσική εμφάνιση, το ξεθώριασμα και άλλες μεταβάσεις.

Πόρτα

Για την υλοποίηση της λειτουργίας της πόρτας πρέπει, όχι μόνο να υλοποιηθεί ένα στιγμιότυπο κώδικα που θα ανταποκρίνεται στην λειτουργία, αλλά θα υπάρχει αντίστοιχα και ένα animation, όπου θα φαίνεται ‘το άνοιγμα της πόρτας’. Για τη δημιουργία ενός animation, αρχικά επιλέγεται το επιθυμητό αντικείμενο στο οποίο θα πραγματοποιηθεί η λειτουργία. Για να υλοποιηθεί αυτό, θα πρέπει στο αντικείμενο της πόρτας που θα χρησιμοποιηθεί, να έχει τουλάχιστον δύο αντικείμενα να την αποτελεί: η πόρτα και το κούφωμα. Στην ιεραρχία των αντικειμένων μου, επιλέγουμε το αντικείμενο που αντιστοιχεί στην πόρτα:

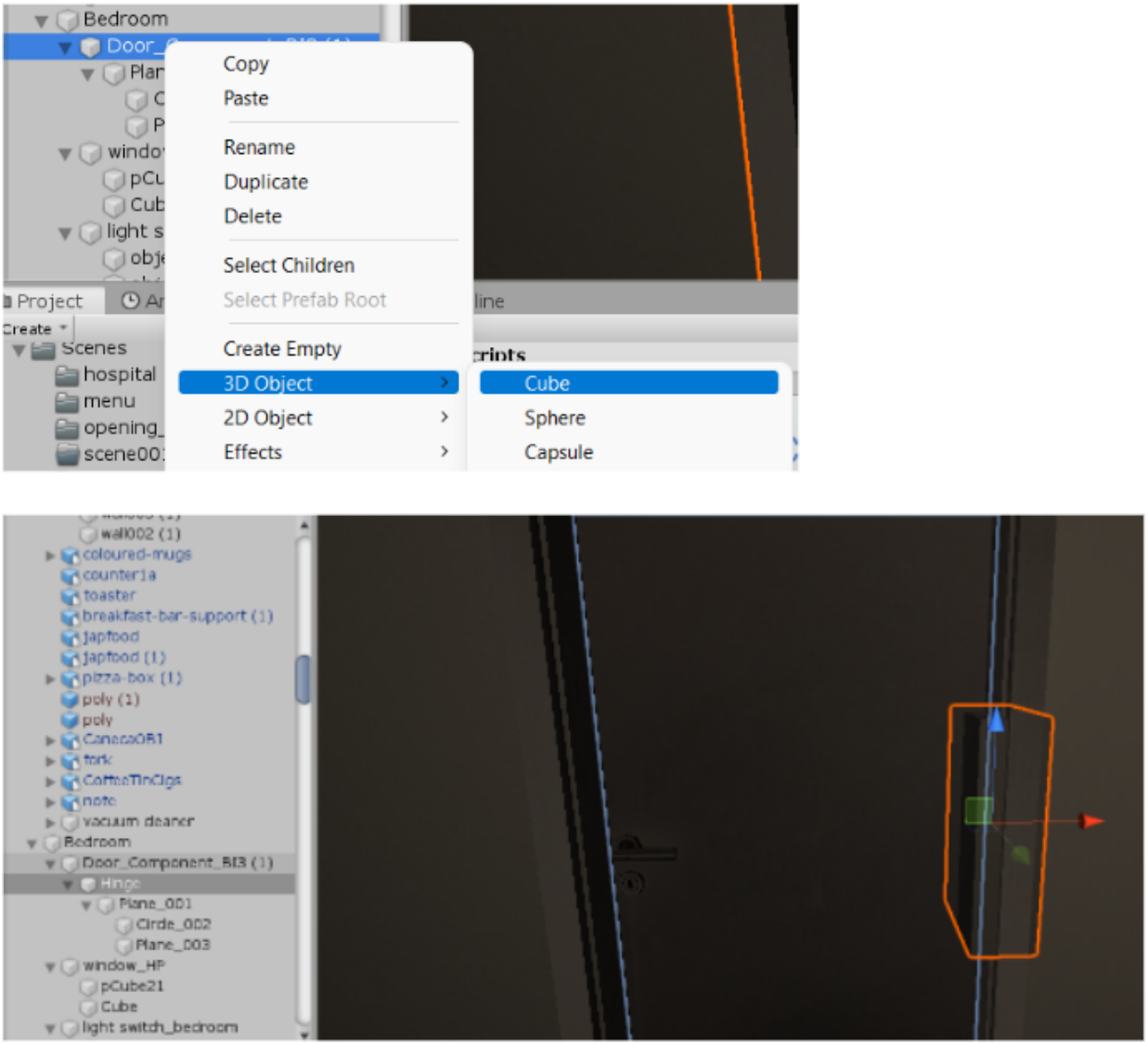
[Εικόνα 62]: Δημιουργία μεντεσέ για την πόρτα

Αρχικά δημιουργούμε ένα ένα αντικείμενο, έναν κύβο, όπου τον προσαρμόζουμε στο μέγεθος και τον μετακινούμε στην άκρη της πόρτας ώστε να λειτουργήσει μεντεσές. Ιεραρχικά, το μεταφέρουμε πάνω και μέσα σε αυτό μεταφέρουμε την πόρτα. Κάνουμε στην ουσία, την πόρτα παιδί του μεντεσέ, έτσι ώστε όταν θα μετακινηθεί ο μεντεσές στον άξονα του z, τότε θα ανοίγει και η πόρτα μαζί.

Μόλις είναι έτοιμο αυτό, έχοντας επιλεγμένο το αντικείμενο της του μεντεσέ, επιλέγουμε το tab του animation και create, ώστε να δημιουργήσουμε καινούργιο. Πατάμε το κόκκινο κουμπί που αντιστοιχεί στο record και στο inspector > transform (Component), στο πεδίο του rotation, στον άξονα του z, θέτουμε την τιμή 0. Με αυτό τον τρόπο ορίζουμε το σημείο αφετηρίας της κίνησης. Η κίνηση θα πραγματοποιηθεί σε διάστημα των δύο δευτερολέπτων, οπότε πάμε στο 120 frame και στο rotation θέτουμε την τιμή 90, όπου θα είναι πλήρως ανοιχτή.

Στη συνέχεια, αφού έχουμε δημιουργήσει την κίνηση, στον inspector, προσθέτουμε ένα νέο στοιχείο, animation, και στο πεδίο animation ορίζουμε αυτό που μόλις φτιάξαμε. Η λογική είναι η εξής, μόλις ο παίκτης θα βρίσκεται σε μία αρκετά κοντινή απόσταση από την πόρτα, θα του εμφανίζεται στο user interface μήνυμα να κάνει ενέργεια, ώστε να ανοίξει η πόρτα. Με την κατάλληλη είσοδο χρήστη (το κουμπί E) θα εκτελείται το animation και ο box collider που έχει η

πόρτα, θα απενεργοποιείται, ώστε να μπορεί να εισέλθει μέσα από αυτή. Η παραπάνω λογική υλοποιείται στο κομμάτι κώδικα DoorCallOpen.



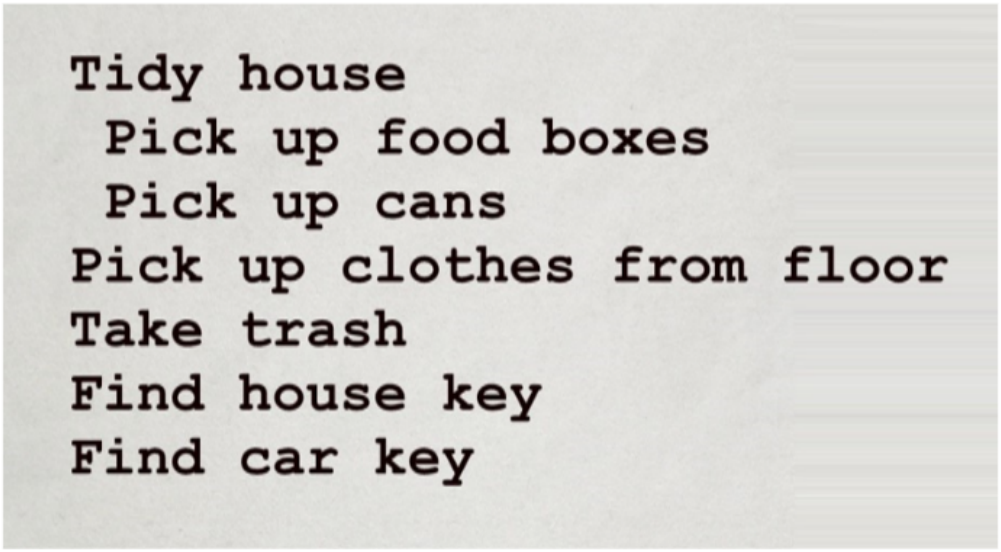
[Εικόνα 63]: Ιεράρχηση του μεντεσέ στην πόρτα.

Ωστόσο, προτού προχωρήσω στην υλοποίηση του, δημιούργησα ένα script με όνομα PlayerCasting το οποίο μετρά την απόσταση του παίκτη από την πόρτα σε κάθε καρέ που περνά χρονικά. Η λογική είναι σαν να ρίχνουμε ένα αόρατο λέιζερ πάνω σε ένα αντικείμενο και η τιμή που επιστρέφεται είναι η απόσταση του από αυτό.

Λίστα με τα αντικείμενα

Κατά την εκκίνηση του παιχνιδιού, με στόχο ο χρήστης να εξοικειωθεί με τους χειρισμούς του παιχνιδιού και με τον χώρο, αναγκαία αποτελεί η ύπαρξη ενός κομματιού, όπου υποσυνείδητα θα συντελέσει στην εκμάθηση αυτή, αλλά ταυτόχρονα θα προάγει και την ιστορία του παιχνιδιού. Για αυτό το σκοπό δημιούργησα ένα αντικείμενο σημείωμα. Από το Asset Store της Unity κατέβασα ένα τέτοιο αντικείμενο που εξυπηρετούσε τις ανάγκες μου. Στο συγκεκριμένο, ανέτρεξα στο φάκελο όπου έχει τα textures του και επεξεργάστηκα αυτό που αντιστοιχούσε στο albedo με το κείμενο που ήθελα να αποτυπώνεται στο χαρτί. Η επεξεργασία έγινε με τη χρήση του προγράμματος ζωγραφικής, που υπάρχει εγκατεστημένο στα windows. Στη λίστα αναγράφονται απλές λειτουργίες, μικροδουλειές, να κάνει ο χρήστης προτού αναχωρήσει από το χώρο του.

Συγκεκριμένα:



[Εικόνα 64]: Η λίστα των αντικειμένων

Η λογική είναι η εξής: όταν θα βρίσκεται πολύ κοντά θα εμφανίζει μήνυμα στην οθόνη του χρήστη, προτρέποντάς τον με το κατάλληλο κουμπί από το πληκτρολόγιο (keyboard input) να προεπισκοπήσει το περιεχόμενο του σημειώματος. Οταν θα απομακρύνεται αυτό το μήνυμα θα εξαφανίζεται. Επιπλέον σε κάθε ένα από τα αντικείμενα που θα χρειαστεί να αλληλεπιδράσει θα

εμφανίζεται αντίστοιχα ένα μήνυμα, αυτό το μήνυμα θα εμφανίζεται μονάχα όταν θα έχει κάνει πρώτα interact με το σημείωμα.

Πρώτα, αφότου ετοίμασα το αντικείμενό μου εμφανισιακά, το τοποθέτησα στον πάγκο της κουζίνας. Έπειτα, στον inspector του αντικειμένου, αντικατέστησα τον box collider με έναν sphere collider. Με το 

σύμβολο, επεξεργάστηκα τον collider μου, μεγαλώνοντας τα όριά του και το checkbox για το αν είναι trigger ή όχι ο collider του αντικειμένου μου, τον επέλεξα ώστε να είναι θετικό. Αυτό διότι, θέλω όταν θα εισέρχεται ο παίκτης μέσα στην ακτίνα της σφαίρας, τότε θα εμφανίζεται το μήνυμα στο user interface.

Σε κώδικα αυτό μεταφράζεται ως εξής:

Δημιούργησα μερικές global μεταβλητές στο script μου.

- noteSelected: μία boolean μεταβλητή, όπου η τιμή της αντιστοιχεί αν το σημείωμα έχει επιλεχθεί ή όχι από τον χρήστη
- onRange: boolean μεταβλητή της με την τιμή της να αντιστοιχεί στο αν ο χρήστης βρίσκεται μέσα στα όρια του αντικειμένου. Από προεπιλογή, η τιμή της είναι true.
- _inTrigger: boolean μεταβλητή με την τιμή της να αντιστοιχεί στο αν ο χρήστης έχει μπει στα όρια του sphere collider
- noteContent: τύπου gameObject, όπου ανταποκρίνεται στο κείμενο που θα εμφανιστεί στο user interface. Από προεπιλογή, έχουμε ορίσει από τον inspector το αντικείμενο να είναι μη ενεργό, να μην φαίνεται δηλαδή το κείμενο.
- pickUpPaper: τύπου AudioSource, όπου ανταποκρίνεται στο ήχο που θα κάνει το αντικείμενο όταν θα αλληλεπιδρά ο χρήστης με αυτό. Συγκεκριμένα, ακούγεται ο ήχος χαρτιού που σηκώνεται.

Το sphere collider attribute, προσδίδει επιπλέον συναρτήσεις που μπορούν να επεξεργαστούν. Υπάρχουν τρεις βασικές συναρτήσεις που σχετίζονται με τις περιπτώσεις αλληλεπίδρασης με τον collider, onTriggerEnter, onTriggerStay και onTriggerExit.

- Η onTriggerEnter εκτελείται μία φορά, με το εισέλθει ο παίκτης στα όρια του collider
- Η onTriggerStay, εκτελείται συνέχεια όσο βρίσκεται μέσα στα όρια του collider
- Η onTriggerExit, εκτελείται με το που αποχωρήσει ο παίκτης από τον collider

Συγκεκριμένα, έξω από την update στο script μας επικαλούμαστε τις συναρτήσεις onTriggerEnter και onTriggerExit. Μόλις, κληθεί η OnTriggerEnter, η τιμή της _inTrigger γίνεται αληθής. Στην update, ελέγχουμε για κάθε frame αν η _inTrigger είναι true. Αν είναι true, τότε κάνουμε ενεργό το αντικείμενο noteContent με την συνάρτηση SetActive(true). Έπειτα, με GetComponent<Text>().text θέτουμε το περιεχόμενο του κειμένου ως "[E] Open note". Με το συγκεκριμένο, προτρέπουμε τον παίκτη να πατήσει το κουμπί 'E' από το πληκτρολόγιο, ώστε να διαβάσει το περιεχόμενο του σημειώματος. Μετά κάνουμε έλεγχο αν έχει πατηθεί το κουμπί E από το πληκτρολόγιο. Στην περίπτωση που γίνει αυτό, τότε το περιεχόμενο του κειμένου αλλάζει σε αυτό που έχει και η εικόνα, η noteSelected γίνεται true και παίζουμε τον ήχο pickUpPaper με την Play(). Τέλος, με την onTriggerExit, μόλις ο παίκτης απομακρυνθεί από το σημείωμα, τότε παίζουμε πάλι τον ήχο και την noteSelected την κάνουμε false.

Play(): Σε ένα κομμάτι ήχου, με την Play(). Με την Play() πραγματοποιείται αναπαραγωγή αυτού του ήχου.

onTriggerExit(): Καλείται όταν ο Collider other έχει σταματήσει να αγγίζει τον trigger.

onTriggerEnter(): Η onTriggerEnter καλείται με το που το αντικείμενο εισέλθει στον trigger, μία φορά

SetActive(): Ενεργοποιεί/απενεργοποιεί ένα GameObject, ανάλογα με τη δεδομένη τιμή true ή false.

Για το κάθε αντικείμενο που υπάρχει στη λίστα πρέπει να υπάρχει ένα script που θα ελέγχει αντίστοιχα αν ο παίκτης βρίσκεται κοντά σε αυτό (στα όρια του sphere collider), θα εμφανίζει το μήνυμα για να κάνει αλληλεπιδράσει και με το κατάλληλο user input, αυτό θα

εξαφανίζεται. Παράλληλά, για λόγους αλληλουχίας με το σημείωμα, αν δεν έχει επιλεχθεί πρώτα αυτό (ώστε να γνωρίζει ο χρήστης τι θα κάνει), δεν θα εμφανίζει το prompt message.

Όπως και με το note, έτσι και με το κάθε αντικείμενο της λίστας, έχει αντικατασταθεί με τον ίδιο τρόπο ο box collider με έναν sphere collider και έχουν μεγαλώσει τα όρια της σφαίρας. Στο script, έχουμε τις εξής μεταβλητές:

- isSelected: boolean μεταβλητή, όπου η τιμή της αντιστοιχεί στο αν έχει επιλεχθεί το αντικείμενο ή όχι
- _inTrigger: boolean μεταβλητή με την τιμή της να αντιστοιχεί στο αν ο χρήστης έχει μπει στα όρια του sphere collider
- textContent: αντιστοιχεί στο αντικείμενο του κειμένου

Αρχικά, για να ελέγξουμε αν έχει επιλεχθεί το σημείωμα, κάνουμε αναφορά σε αυτό, από το στιγμιότυπο του κώδικα noteInteract και η τιμή της μεταβλητής που μας αφορά, είναι η noteSelected. Αν η τιμή της είναι αληθής (true), τότε θα εκτελεστεί το συγκεκριμένο script. Με τον ίδιο τρόπο που είχαμε και στο script του σημειώματος, έτσι και εδώ ελέγχουμε αν ο χρήστης έχει μπει στην ακτίνα της σφαίρας μέσα από την συνάρτηση OnTriggerEnter. Στην περίπτωση που ανταποκρίνεται σε αυτή τη συνθήκη, τότε η inTrigger γίνεται true και εμφανίζουμε μήνυμα στο χρήστη να "σηκώσει το αντικείμενο" με το κουμπί E. Μόλις πατήσει το κουμπί, τότε εξαφανίζουμε το αντικείμενο με την SetActive(false) και απενεργοποιούμε το boxCollider του με το GetComponent<BoxCollider>().enabled = false.

Σκοπός του πρώτου κομματιού του παιχνιδιού είναι να κάνει κάποιες ενέργειες (αυτές που αναγράφει η λίστα) και μόλις ολοκληρωθούν όλα αυτά, τότε να μπορεί να φύγει από το σπίτι. Συνεπώς, παράλληλα με το κάθε αντικείμενο έχουμε έναν counter, όπου κάθε φορά που αλληλεπιδρά με τα αντικείμενα, ο counter αυξάνεται κατά ένα.

UI (User Interface)

Με τον όρο UI (διεπαφή χρήστη) εννοούμε το σημείο αλληλεπίδρασης και επικοινωνίας ανθρώπου-υπολογιστή σε μια συσκευή. Είναι επίσης ο τρόπος με τον οποίο ένας χρήστης

αλληλεπιδρά με μια εφαρμογή ή έναν ιστότοπο. Ο σκοπός της διεπαφής χρήστη στα παιχνίδια είναι να επιτρέψει σε έναν χρήστη να εκτελέσει μια εργασία στο χώρο του παιχνιδιού είτε μέσω άμεσης εισαγωγής (direct input), είτε μέσω μιας ενέργειας σε μια οθόνη Heads Up (HUD). Η διεπαφή χρήστη είναι το τμήμα ενός προγράμματος που επιτρέπει αλληλεπιδράσεις ανθρώπου-υπολογιστή (HCI). Η ύπαρξη User Interface σε ένα παιχνίδι είναι αναγκαία στα περισσότερα από αυτά, μιας και με αυτό τον τρόπο ο χρήστης γνωρίζει τι του γίνεται και πιο εύκολα μπορεί να αλληλεπιδράσει σε αυτό.

Δημιουργία Menu

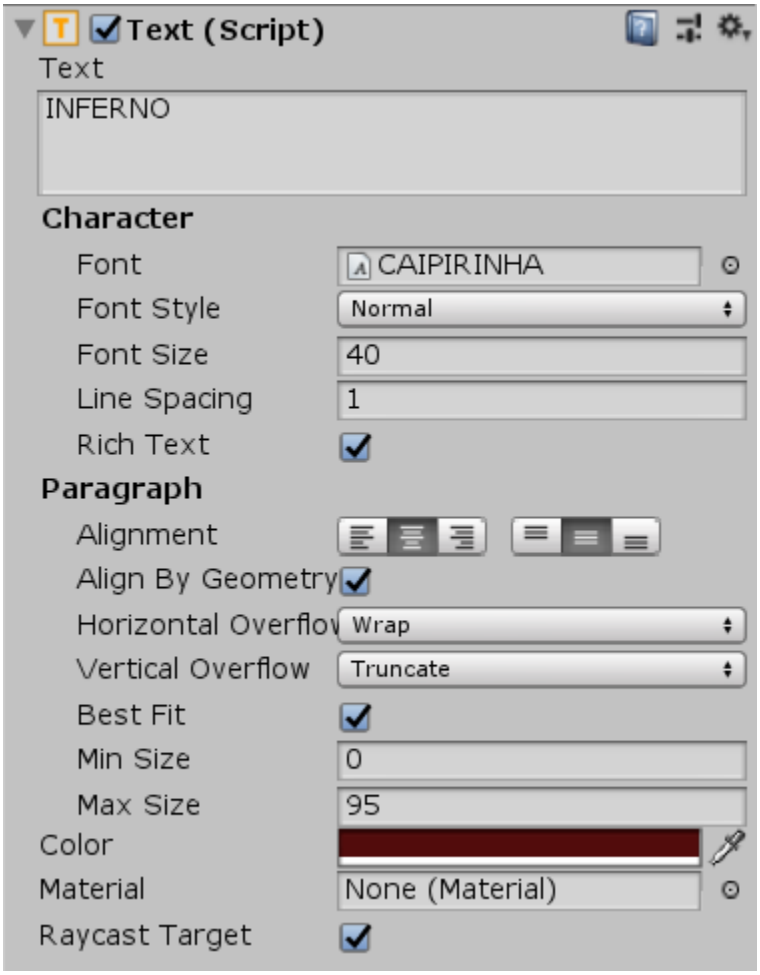
Ένα βασικό Menu ένα παιχνίδι πρέπει να υλοποιεί τις εξής λειτουργίες:

1) Να εκκινεί το παιχνίδι,
2) Να μπορεί να τροποποιεί κάποιες βασικές λειτουργίες όπως είναι η αυξομείωση ήχου ή επιλογή γραφικών
3) Να μπορεί να το κλείσει εντελώς.

Για την υλοποίηση του μενού, χρησιμοποίησα δύο σκηνές. Η μία αποτελεί εκεί όπου θα βρίσκονται οι προαναφερόμενες επιλογές και η δεύτερη θα είναι η πρώτη σκηνή του παιχνιδιού μου. Στην πρώτη σκηνή, πήγα στην ιεραρχία, δεξί κλικ > UI > Panel. Με αυτό τον τρόπο δημιουργεία ένα Canvas (καμβάς) αντικείμενο όπου σε αυτό υπάρχει μέσα το panel που μόλις δημιούργησα.Έχοντας επιλεγμένο το αντικείμενο, στον inspector, κάτω από την επιλογή του image, αντικατέστησα το αρχείο εικόνας που ήταν προεπιλεγμένο και επέλεγα μία από αυτές που υπήρχαν ήδη στο unity. Ήθελα μία transparent εικόνα, όπου θα μπορεί ο χρήστη παράλληλα να βλέπει πίσω στη σκηνή. Έπειτα, το color, στην επιλογή του Alpha Channel, άλλαξα την τιμή που, επιλέγοντας μία κάπου στη μέση της μπάρας, ώστε να η εικόνα μου να είναι transparent, αλλά όχι τελείως. Αυτό αποτελεί το background, η βάση του μενού μου. Άλλαξα το όνομά του στην ιεραρχία από panel σε background.

Για να προσθέσω κείμενο, πρώτα κατέβασα από το Unity Asset Store το πακέτο TextMeshPro, το οποίο ήταν και δωρεάν. Έπειτα, έχοντας επιλεγμένο τον Καμβά, πάτησε δεξί κλικ > UI > TextMeshPro. Ο λόγος που επέλεξα το συγκεκριμένο αντί για αυτό που μου παρείχε

ήδη το Unity από μόνο του είναι επειδή το TextMeshPro έχει καλύτερη ποιότητα μετά στο Render και περισσότερες επιλογές κατά την επεξεργασία.



[Εικόνα 65]: Ανάθεση τιμών στο αντικείμενο κειμένου, ώστε να πάρει την παρούσα μορφή.

Συνεχίζοντας, για την υλοποίηση των τριών λειτουργιών που προανεφερα, αναγκαία είναι η ύπαρξη κουμπιών και όχι απλού κειμένου, μιας και στο απλό κείμενο, δεν υπάρχει η δυνατότητα δημιουργίας script. Με την ίδια λογική που δημιούργησα το κείμενο, δεξί κλικ στο επιλεγμένο αρχείο του καμβά > UI > Button. Αυτή η ενέργεια πραγματοποιήθηκε τρεις φορές.

Το πρώτο κουμπί είναι το Play. Στην Ιεραρχία μπορεί να παρατηρήσει κάποιος ότι το κουμπί έχει δύο αντικείμενα: το ίδιο το κουμπί και ένα θυγατρικό, τύπου Text. Άλλαξα το όνομα του κουμπιού, μέσω του αντικειμένου Text στον Inspector, κάτω από το πεδίο Text (Script) σε

Play. Πατώντας το κουμπί Play, πρέπει να φορτώνει η πρώτη σκηνή του παιχνιδιου, την οποία έχω ονομάσει Opening Scene.

Το δεύτερο κουμπί είναι το Options. Με αυτό, όταν ο χρήστης το επιλέγει θα εμφανίζει ένα δεύτερο μενού, με τις επιλογές που θα μπορεί να τροποποιήσει. Με αυτό τον τρόπο, υπάρχουν δύο μενού. Το πρώτο που είναι το βασικό (Main Menu) και το δεύτερο που είναι το Options Menu. Για λόγους ευκολίας, στην Ιεραρχία δημιούργησα ένα κενό αντικείμενο (Δεξί Κλικ μέσα στον Καμβά > Create Empty) και μέσα σε αυτό, έσυρα τα δύο κουμπιά που έχω δημιουργήσει μέχρι στιγμής, εκεί αργότερα θα βρίσκεται και το τρίτο, το Quit, το οποιο ονόμασα Main Menu.

Έπειτα, δημιούργησα ένα δεύτερο κενό αντικείμενο, με όνομα Options Menu και προχώρησα στην δόμηση του. Ξεκινώντας, απενεργοποίησα το Main Menu, ώστε να βρίσκεται μόνο το αντικείμενο Options Menu on display. Δημιούργησα ένα κουμπί, με όνομα back, όπου με αυτό ο χρήστης θα επιστρέψει στο αρχικό μενού. Για τη μπάρα αυξομείωσης του ήχου, έχουμε ένα κείμενο που λέει Volume και δίπλα του ένα slider. Για το Slider έχουμε δεξί κλικ μέσα στο αντικείμενο Options Menu > UI > Slider

Τέλος έχουμε το Quit κουμπί, όπου με αυτό θα κλείνει το παιχνίδι και κατά συνέπεια και το παράθυρο του.

Αφού έχουμε δημιουργήσει όλα τα κουμπιά, προχώρησα στη συγγραφή script, όπου θα είναι προσκολλημένο στο αντικείμενο του Main Menu και με αυτό θα εκτελεί τις λειτουργίες για κάθε μία από τις επιλογές αντίστοιχα. Έχοντας επιλεγμένο το Main Menu αντικείμενο, στον Inspector προσθέτουμε νέο component (Add Component) και στη μπάρα αναζήτησης γράφουμε mainMenu. Με αυτό τον τρόπο δημιουργείται εύκολα και γρήγορα ένα νέο script με το όνομα που έχουμε δώσει και είναι στο αντικείμενο που θέλουμε να το συσχετίσουμε.

Μέσα στο script δημιουργούμε μία νέα συνάρτηση που θα καλούμε όταν θα επιλέγουμε κάποιο από τα κουμπιά του μενού με όνομα PlayGame(). Μέσα σε αυτή καλούμε την SceneManager.LoadScene() και σαν παράμετρο δηλώνουμε το όνομα της σκηνής που θέλουμε να φορτώσουμε, σε αυτή την περίπτωση ‘Opening Scene’. Στη συνέχεια, επιλέγουμε το Play κουμπί

από την Ιεραρχία και στον Inspector, στο κομμάτι που αναγράφει onClick, με το (+) προσθέτουμε την συνάρτηση που μόλις δημιουργήσαμε. Σε αυτό το σημείο μας ζητά να προσθέσουμε ένα αντικείμενο, στην προκειμένη περίπτωση, το αντικείμενο στο οποίο βρίσκεται το script μας, δηλαδή το MainMenu. Δίπλα από εκεί που δηλώσαμε το αντικείμενο, στο dropdown menu που υπάρχει, επιλέξουμε τη συνάρτηση από το script που, το PlayGame().

Για το κουμπί Quit ακολουθούμε την ίδια λογική. Στο MainMenu script δημιουργούμε μία νέα συνάρτηση με όνομα QuitGame(). Σε αυτή τη συνάρτηση, τα πράγματα είναι πιο απλά. Το μόνο που γράφουμε είναι: Application.Quit(). Με αυτό τον τρόπο κλείνουμε το πρόγραμμα.

Για το Options κουμπί, στην onClick, κάνουμε αναφορά κατευθείαν στο αντικείμενο OptionsMenu και σαν συνάρτηση επιλέγουμε Game Object > Set Active. Έπειδή θέλουμε μόλις επιλέγουμε το Options από το κεντρικό μενού να απενεργοποιεί το MainMenu και να ενεργοποιεί το OptionsMenu, το SetActive σε αυτό που έχουμε ήδη προσθέσει το αφήνουμε ως έχει (ενεργό), δημιουργούμε μία νέα ενέργεια με το (+) κουμπί, σαν αντικείμενο δηλώνουμε το MainMenu και σαν συνάρτηση επιλέγουμε το SetActive πάλι, με το checkbox μη επιλεγμένο. Κάνουμε ανάλογη δουλειά στο κουμπί back στο options με την μόνη διαφορά ότι στο MainMenu και στο OptionsMenu έχουμε αντίστροφες τις τιμές στο SetActive.

SceneManager: Η διαχείριση σκηνών κατά την εκτέλεση.
LoadScene(): έτοιμη συνάρτηση του SceneManager. Φορτώνει τη σκηνή χρησιμοποιώντας το όνομά της ή τον δείκτη στο Build Settings.
Build Settings: Το Unity για να δημιουργήσει την εφαρμογή, ώστε να είναι συμβατή με διαφορετικές πλατφόρμες και με διαφορετικές ρυθμίσεις.
Application.Quit(): Κλείνει την εφαρμογή. Τερματίζει την εφαρμογή που εκτελείται. Η κλήση Application.Quit αγνοείται στον Editor.

Ξεκινώντας την εφαρμογή, οι επιλογές του μενού είναι επιλέξιμες και με τον κέρσορα και με τα βελάκια από το πληκτρολόγιο.

ΣΥΜΠΕΡΑΣΜΑ

Το παιχνίδι αποτελεί ένα από τα σημαντικότερα μέσα ψυχαγωγίας του ατόμου, εδώ και πολλούς αιώνες. Παραλληλα, έχει αποτελέσει αντικείμενο έρευνας πολλών διαφορετικών επιστημόνων. Όσο η τεχνολογία συνεχίσει να αναπτύσσεται με ραγδαίους ρυθμούς.

Η παρούσα πτυχιακή εργασία εστίασε το ενδιαφέρον της στην ανάλυση και σχεδίαση ενός τρομακτικού παιγνίου. Παράλληλα, μέσα από αυτό, έγινε εξερευνηση στα είδη του τρόμου, στα παιχνίδια που προσδιόρισαν την συγκεκριμένη κατηγορία, αλλά και πραγματοποιήθηκε αναδρομή τόσο στα γραφικά των παιχνιδιών όσο και στο υλικοτεχνικό κομμάτι των κονσολών. Αναλύοντας τα χαρακτηριστικά των videogames και των τρομακτικών παιχνιδιών, μέσω της έρευνας πεδίου και χαρακτηριστικών παραδειγμάτων εντοπίζονται κοινά στοιχεία όσον αφορά τις μεθόδους και τα βήματα σχεδιασμού και ανάπτυξης.

Στο δεύτερο μέρος, πραγματοποιήθηκε η υλοποίηση του στο περιβάλλον του Unity. Ξεκινώντας πρώτα από το σχεδιασμό του χαρακτήρα, ενός βασικού first person controller και προχωρώντας στην ανάπτυξη του περιβάλλοντος τριγύρω του τόσο λειτουργικά, όσο και αισθητικά. Όλα αρχίζουν με την βασική ιδέα, πάνω στην οποία αναπτύσσεται το παιχνίδι και η ατμόσφαιρα όπου πρέπει να του δοθεί. Το περιβάλλον μέσα στο οποίο θα βρίσκεται θα πρέπει να είναι όσο το δυνατόν πιο ρεαλιστικό γίνεται σχεδιαστικά και αρχιτεκτονικά, ώστε ο παίκτης να απορροφηθεί πιο εύκολα σε αυτό και να επενδύσει ψυχολογικά.

ΜΕΡΟΣ Γ’:

1. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]Πρότυπα εμφάνισης γραφικών, http://www.it.uom.gr/project/mycomputer/v_card/prot_emf.html.

[2] “Artbreeder: How Artificial Intelligence Changes Our Perception of Art.” *SDS Club*, 6 July 2021, https://sdsclub.com/artbreeder-how-artificial-intelligence-changes-our-perception-of-art/.

[3] “Best AAA games - Everything you should know about Triple A.” *G2A*, 21 February 2022, https://www.g2a.com/news/features/best-aaa-games/.

[4] Brownlee, Jason. “A Gentle Introduction to StyleGAN the Style Generative Adversarial Network.” *Machine Learning Mastery*, 19 August 2019, https://machinelearningmastery.com/introduction-to-style-generative-adversarial-network-stylegan/.

[5] Churchville, Fred. “What is user interface (UI)? Definition from SearchAppArchitecture.” *TechTarget*, https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI.

[6] Cronenberg, David. “E3 2001: Silent Hill 2 Interview.” *IGN*, 17 May 2001, https://www.ign.com/articles/2001/05/17/e3-2001-silent-hill-2-interview.

[7] “A garbage collector for C and C++.” *Hans-J. Boehm*, https://www.hboehm.info/gc/#details.

[8] Hejlsberg, Anders. “C Sharp (programming language).” *Wikipedia*, https://en.wikipedia.org/wiki/C_Sharp_(programming_language).

[9] “Horror in the Making: How Red Barrels outlasted Outlast.” *Game Developer*, 29 January 2015, https://www.gamedeveloper.com/design/horror-in-the-making-how-red-barrels-outlasted-i-outlast-i-.

[10] “Layers of Fear.” *Wikipedia*, https://en.wikipedia.org/wiki/Layers_of_Fear.

[11] “Manual: Overview of .NET in Unity.” *Unity - Manual*, https://docs.unity3d.com/Manual/overview-of-dot-net-in-unity.html.

[12] “Manual: Unity architecture.” *Unity - Manual*, https://docs.unity3d.com/Manual/unity-architecture.html.

[13] “Monochrome & Hercules Graphics Aspect Ratio and Scaling.” *Nerdly Pleasures*, 20 February 2014, http://nerdlypleasures.blogspot.com/2014/02/monochrome-hercules-graphics-aspect.html.

[14] “.NET Framework.” *Wikipedia*, https://en.wikipedia.org/wiki/.NET_Framework.

[17] “PT | Silent Hill Wiki | Fandom.” *Silent Hill Wiki*, https://silenthill.fandom.com/wiki/P.T.

[18] Rothwell, Richard, et al. “Horror fiction.” *Wikipedia*, https://en.wikipedia.org/wiki/Horror_fiction.

[19] Scott, Jonathan. “What is Horror? Definition and Examples in Film.” *StudioBinder*, 24 May 2020, https://www.studiobinder.com/blog/what-is-horror-definition/.

[20] “Spacewar!” *Wikipedia*, https://en.wikipedia.org/wiki/Spacewar!

Takahashi, Dean. “Unity (game engine).” *Wikipedia*, https://en.wikipedia.org/wiki/Unity_(game_engine).

[21] “Unity - Scripting API: CharacterController.” *Unity - Manual*, https://docs.unity3d.com/ScriptReference/CharacterController.html.

[22] “Unity - Scripting API: Rigidbody.” *Unity - Manual*, https://docs.unity3d.com/ScriptReference/Rigidbody.html.

2. ΚΑΤΆΛΟΓΟΣ ΕΙΚΌΝΩΝ

Enter the survival horror