ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

"Ανάπτυξη στην Unity Engine εφαρμογής / βιντεοπαιχνιδιού τριών διαστάσεων για το συσχετισμό διαφορετικών ρυθμίσεων ομίχλης, φωτεινότητας και αντίθεσης με την πρόκληση του συναισθήματος αβεβαιότητας και τη σύνδεση με την εμπειρία."

"Development in Unity Engine of a 3D application / videogame to relate different fog, brightness and contrast settings to the feeling of uncertainty and connection to the experience."

Διπλωματική Εργασία

Νίκος Σφήκας

Εξεταστική επιτροπή

Καθ. Μιχαήλ Γ. Λαγουδάκης (επιβλέπων)
Καθ. Αικατερίνη Μανιά (επιτροπή)
Καθ. Κωνσταντίνος – Αλκέτας Ουγγρίνης (επιβλέπων, Σχολή ΑΡΜΗΧ)

Μάιος 2022

Contents

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει την ανάπτυξη ενός ολοκληρωμένου τρισδιάστατου βιντεοπαιχνιδιού για την πλατφόρμα των Windows, με το όνομα “Drainage”. Πρόκειται για παιχνίδι είδους πρώτου προσώπου/περιπέτειας/γρίφων σε περιβάλλον υπονόμου και υλοποιήθηκε με χρήση της μηχανής γραφικών Unity Engine. Το συγκεκριμένο ολοκληρωμένο παιχνίδι αποτελεί τμήμα ενός μεγαλύτερου project του TUC TIE LAB που ερευνά τη σύνδεση διαφορετικών χώρων με διαφορετικά ανθρώπινα συναισθήματα, ο υπόνομος στην συγκεκριμένη περίπτωση με το συναίσθημα της αβεβαιότητας. Για την ολοκλήρωση του συγκεκριμένου παιχνιδιού το περιβάλλον των πέντε διαφορετικών επιπέδων (χώροι και αρκετά από τα τρισδιάστατα μοντέλα) δημιουργήθηκε στο πρόγραμμα σχεδίασης τρισδιάστατων γραφικών Blender, τα textures στα εργαλεία Awesome Bump και Materialize, ενώ για επεξεργασία εικόνας χρησιμοποιήθηκε η εφαρμογή ανοιχτού λογισμικού GIMP και ο απαραίτητος κώδικας αναπτύχθηκε στην γλώσσα προγραμματισμού C# και CG/HLSL. Ο παίκτης εξερευνώντας το περιβάλλον του παιχνιδιού και διαδρώντας με αντικείμενα θα πρέπει να ανακαλύψει τα απαραίτητα στοιχεία, έτσι ώστε να φτάσει στην επίλυση των γρίφων και να προχωρήσει στη συνέχεια στον τερματισμό. Με την ειδική επιλογή “Play Experiment” οι δεκαπέντε συμμετέχοντες χωρίστηκαν σε τρεις ομάδες και συμμετείχαν στο πείραμα που διεξήχθη με σκοπό να απαντήσουν σε ειδικό ερωτηματολόγιο, έτσι ώστε να αντληθούν πληροφορίες σχετικά με το πώς διαφορετικές ρυθμίσεις φωτεινότητας, contrast και ομίχλης επηρεάζουν ή/και εντείνουν το συναίσθημα αβεβαιότητας του χρήστη και την εμβύθισή του στην εμπειρία.

Abstract

This thesis presents the development of a complete 3D video game for the Windows platform, called “Drainage”. It is a first person / adventure / puzzle game in a sewer environment and was created using the Unity graphics engine. This novel video game is part of a larger project of TUC TIE LAB that explores the connection of different places with different human emotions, sewer in this case, with the feeling of uncertainty. To complete this game, the environment of five different levels, game spaces and several of the 3D models, were created in Blender 3D graphic design program, the textures in Awesome Bump and Materialize tools, while GIMP open source application was used for image processing and the necessary code was developed in the C# and CG/HLSL languages. The player by exploring the game environment and interacting with objects, will have to discover the necessary elements, so as to reach the solution of the puzzles and then proceed to the finish. With the special option “Play Experiment”, fifteen participants were divided into three groups and participated in the experiment conducted, with the purpose of answering a special questionnaire, in order to obtain information on how different brightness, contrast and fog settings in the game, affect and/or intensify the user’s sense of uncertainty and immersion in the experience.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή κ. Ουγγρίνη για τις πολύτιμες συμβουλές του και την καθοδήγηση από την αρχή της υλοποίησης της διπλωματικής εργασίας καθώς για τον χρόνο του, τον υποψήφιο διδάκτορα κ. Χριστουλάκη για τις πολύτιμες συμβουλές του όσον αφορά στην Unity Engine και τον χρόνο που αφιέρωσε για βοήθεια και συμβουλές, τον κ. Λαγουδάκη για τον χρόνο που διέθεσε για διορθώσεις του κειμένου και την βοήθεια του έτσι ώστε να αυτό να πάρει την τελική του μορφή, καθώς και την κ. Μανιά για τον χρόνο που θα αφιερώσει στην ανάγνωση της διπλωματικής μου εργασίας. Επίσης τον Γεώργιο Προβατόπουλο που συνέθεσε την μουσική του παιχνιδιού και τον Μιχάλη Ζυμβραγουδάκη που σχεδίασε το logo του παιχνιδιού.
Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την στήριξη, αλλά και την υπομονή τους αυτά τα χρόνια.

Κεφάλαιο 1: Εισαγωγή και το συναίσθημα της αβεβαιότητας

1.1 Εισαγωγή

Στόχος της παρούσας διπλωματικής εργασίας ήταν η υλοποίηση ενός ολοκληρωμένου τρισδιάστατου βιντεοπαιχνιδιού πρώτου προσώπου για την πλατφόρμα των Microsoft Windows και είδους γρίφων/περιπέτειας (puzzle/adventure). Τα παιχνίδια του συγκεκριμένου είδους προσφέρουν συνήθως στον παίκτη τη δυνατότητα για εξερεύνηση ενός τρισδιάστατου περιβάλλοντος. Μέσω της διάδρασης με αντικείμενα, αλλά και της παρατήρησης στοιχείων που έχει επιλέξει ο δημιουργός του παιχνιδιού, ο παίκτης έχει σαν σκοπό την επίλυση γρίφων, έτσι ώστε να συνεχίσει την πορεία του προς τις επόμενες πίστες, να έλθει σε επαφή με νέες προκλήσεις, χώρους και προβλήματα λογικής με απώτερο σκοπό την ολοκλήρωση του παιχνιδιού και τον τερματισμό, αλλά και την βίωση μίας εμπειρίας. Πιο συγκεκριμένα, το περιβάλλον του παρόντος παιχνιδιού τοποθετείται σε υπόνομο κι ένας από τους βασικούς πυλώνες της διπλωματικής εργασίας ήταν να ερευνηθεί κατά πόσο διαφορετικές ρυθμίσεις φωτεινότητας, αντίθεσης αλλά και ποσότητας ομίχλης/περιβαλλοντικές συνθήκες μπορούν να επηρρεάσουν το συναίσθημα της αβεβαιότητας στον χρήστη και τον βαθμό εμβύθισής του στην εμπειρία.

1.2 Το συναίσθημα της αβεβαιότητας

Πολλές φορές στην καθημερινή μας ζωή, ερχόμαστε αντιμέτωποι με το άγνωστο, το αβέβαιο, την αδυναμία πρόβλεψης για το τι θα συμβεί στο μέλλον. Συνηθισμένα καθημερινά γεγονότα μπορούν να πυροδοτήσουν συναισθήματα, όπως ο φόβος και η αγωνία [20]. Υπάρχει επίσης η περίπτωση να αντιμετωπίσουμε δυσκολίες στην λήψη αποφάσεων ή στην εύρεση μιας ισορροπίας που αναζητούμε για εμάς ή μία λύση σε κάποιο πρόβλημα. Ένας από τους ορισμούς της αβεβαιότητας θα μπορούσε να είναι η "απουσία του βέβαιου, η έλλειψη σιγουριάς και βάσης", μία κατάσταση στην οποία μπορεί να περιέλθει ένας ανθρώπος και χαρακτηρίζεται από την απουσία στοιχείων, δεδομένων, γνώσεων και βασικών πληροφοριών [1] για ένα γεγονός που έχει συμβεί ή και για κάτι που δεν έχει ακόμα συμβεί ή δεν γνωρίζουμε αν θα συμβεί. Όπως φαίνεται, σύμφωνα με τον παραπάνω ορισμό, η αβεβαιότητα περιλαμβάνει δύο βασικές πτυχές. Απ' τη μία πλευρά έχουμε την απουσία πληροφοριών [1] και δεδομένων και από την άλλη έχουμε την ολική ή μερική αίσθηση άγνοιας σχετικά με το τι πρόκειται να συμβεί.

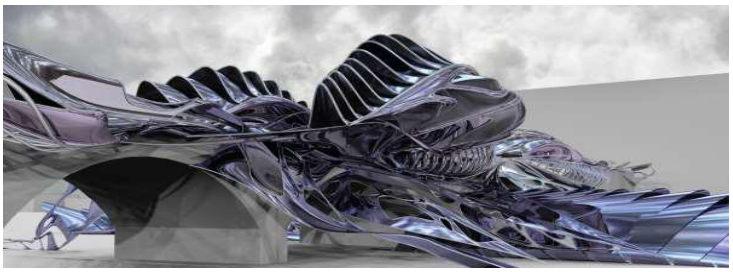


Figure 1: Structure of uncertainty, Chinese golden fish -animal analogy, liquid architecture, Zeng, H. https://www.researchgate.net/figure/Structure-of-uncertainty-Chinese-golden-fish-animal-analogy-liquid-architecture-Zeng_fig3_303213906

Από τη φύση του ο άνθρωπος επιζητεί να ελέγχει τις καταστάσεις γύρω του και αναζητά τη σταθερότητα και το οικείο. Όταν λοιπόν εμφανίζεται η αβεβαιότητα ως αποτέλεσμα της έλλειψης γνώσης και του ελέγχου, ο άνθρωπος μπορεί να αισθανθεί άγχος και ανασφάλεια. Κάποιες νέες μελέτες επιπλέον

συνδέουν το συναίσθημα της αβεβαιότητας με συγκεκριμένες περιοχές του εγκεφάλου που σχετίζονται με την ευχάριστη ή μη-ευχάριστη αίσθηση της γεύσης και της οσμής [3]. Το λεξικό Merriam-Webster δίνοντας τον δικό του ορισμό για την αβεβαιότητα την χαρακτηρίζει ως "την κατάσταση του να είσαι αβέβαιος" [4] και χρησιμοποιεί μία πληθώρα όρων για να περιγράψει ακόμα καλύτερα τι σημαίνει να είναι κανείς αβέβαιος: απροσδιόριστο, προβληματικό, αναξιόπιστο, αμφίβολο, μη σαφώς προσδιορισμένο ή καθορισμένο, μεταβλητό, αόριστο. Ο συγκεκριμένος ορισμός τονίζει ιδιαίτερα αυτή την σημαντική περιοχή συμφωνίας μεταξύ διαφορετικών ορισμών. Την έννοια της αβεβαιότητας δηλαδή σαν μία, στη βάση της, κατάστασης της ανθρώπινης ψυχής, μία γνωστική εμπειρία που επιδίδεται διαφορετικής "μετάφρασης" και βίωσης από το κάθε ξεχωριστό ανθρώπινο ον και δεν αποτελεί κομμάτι του υλικού κόσμου, τον οποίο πάνω-κάτω αντιλαμβανόμαστε με όμοιο τρόπο οι περισσότεροι άνθρωποι λόγω της αντικειμενικής του φύσης. Στο επίκεντρο μίας εμπειρίας τέτοιου τύπου, έχουν συνήθως την θέση τους λέξεις και έννοιες όπως "άγνοια" και "απουσία γνώσης". Είναι ιδιαίτερης σημασίας το γεγονός πως η αβεβαιότητα δεν ισοδυναμεί με απλή άγνοια, αλλά θα μπορούσε να χαρακτηριστεί και ως η συνειδητή επίγνωση ή η υποκειμενική εμπειρία της άγνοιας. Είναι μια μεταγνώση ανώτερης τάξης, δηλαδή αντιπροσωπεύει ένα συγκεκριμένο είδος ρητής γνώσης – μια αναγνώριση αυτού που δεν γνωρίζει κανείς, αλλά και ότι δεν το γνωρίζει.

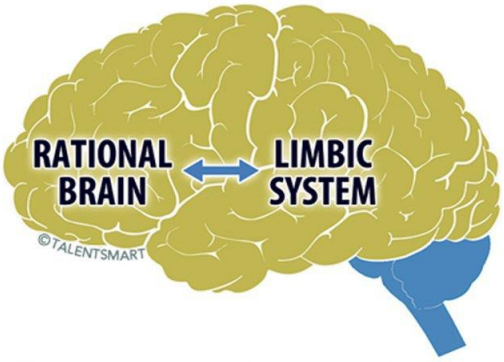


Figure 2: Rational Brain vs. Limbic System

Κεφάλαιο 2: Σχετική Έρευνα

2.1 Τι είναι ένα βιντεοπαιχνίδι;

Ο όρος βιντεοπαιχνίδι αναφέρεται σε "ένα ηλεκτρονικό παιχνίδι το οποίο περιλαμβάνει αλληλεπίδραση με μια διεπαφή χρήστη για την παραγωγή οπτικής ανάδρασης σε μία συσκευή βίντεο." [18] Το κομμάτι "βίντεο" στο βιντεοπαιχνίδι αναφέρεται στην προβολή raster, με την χρήση μίας συσκευής οπτικής διεπαφής χρήστη, όπως για παράδειγμα είναι μία οθόνη. Το "Raster" ουσιαστικά, είναι μία εικόνα που αποτελείται από ένα σύνολο εικονοστοιχείων (pixels) τα οποία βρίσκονται κατανεμημένα σε ένα ορθογώνιο πλέγμα (grid). Πρέπει να σημειώσουμε πως οι περισσότερες φωτογραφίες αποθηκεύονται στον υπολογιστή σαν αρχεία μορφής raster, όπως είναι τα πολύ γνωστά png, jpeg και gif. Μία δομή δεδομένων τύπου raster στηρίζεται κυρίως στην ορθογώνια ή τετραγωνική ψηφοθέτηση του δισδιάστατου επιπέδου σε κελιά, καθένα εκ των οποίων αντιστοιχείται σε μία αριθμητική τιμή. Ο όρος "βιντεοπαιχνίδι" μπορεί να εξηγηθεί απλά, ως μία συσκευή ή εφαρμογή ικανή (να απεικονίσει) και υπεύθυνη για την απεικόνιση δισδιάστατων ή τρισδιάστατων γραφικών σε κάποια οθόνη, που επιπλέον αυτή καθεαυτή η εφαρμογή δίνει τη δυνατότητα στον χρήστη να διαδράσει με στοιχεία της και να παίξει, αποτελεί λοιπόν ένα είδος ηλεκτρονικού παιχνιδιού, παίρνοντας έτσι αποστάσεις από την υλικότητα των παραδοσιακών παιδικών παιχνιδιών (ξύλινων για παράδειγμα) ή άλλων παιχνιδιών που παίζονται μακριά από ψηφιακές οθόνες και συστήματα και δεν αποτελούνται από ηλεκτρικά κυκλώματα και δηφία ή κάποιου είδους γλώσσα μηχανής και δισδιάστατα ή τρισδιάστατα γραφικά, αλλά έχουν μία κάπως πιο υλική υπόσταση (board games για παράδειγμα, τα γνωστά επιτραπέζια). Τέτοια ηλεκτρονικά συστήματα, που χρειάζεται κάποιος για να μπορέσει να παίξει βιντεοπαιχνίδια, ονομάζονται "πλατφόρμες" [18] , όπως μπορούν να θεωρηθούν οι προσωπικοί υπολογιστές και οι παιχνιδομηχανές – κονσόλες. Αυτού του είδους τα συστήματα μπορούν να είναι μεγάλα σε όγκο όπως ένας υπολογιστής ή μία παιχνιδομηχανή-κονσόλα τύπου PlayStation ή αρκετά μικρά σε μέγεθος όπως τα handheld devices, σαν το Nintendo GameBoy ή το Nintendo Switch.

Figure 3: Retro Space Harrier Arcade - SEGA

2.2 Ιστορία των βιντεοπαιχνιδιών

Τα πιο πρώιμα βιντεοπαιχνίδια (θα μπορούσαμε να τα χαρακτηρίσουμε ίσως και proto-videogames) ήταν ουσιαστικά ηλεκτρονικές συσκευές, με τις οποίες ο χρήστης μπορούσε να διαδράσει μέσω κάποιου τύπου διεπαφής. Η παλαιότερη συναντάται στο 1947, γνωστή και ως "Συσκευή σωλήνα καθοδικών ακτίνων ψυχαγωγίας" [16] (Cathode-Ray Tube Amusement Device), η οποία κατατέθηκε για ένα δίπλωμα ευρεσιτεχνίας στις 25 Ιανουαρίου 1947, από τον Thomas T. Goldsmith Jr. και Estle Ray Mann και εκδόθηκε στις 14 Δεκεμβρίου 1948, από την US Patent [16]. Εμπνευσμένη από τις γνωστές στους περισσότερους από εμάς, οθόνες των ραντάρ, αποτελούνταν από μία αναλογική συσκευή, που επέτρεπε στον χρήστη να ελέγχει μία κουκίδα για να προσομοιώσει την ρίψη βλημάτων με στόχο [16], σε σταθερά σχέδια όμως στην οθόνη. Κάποια άλλα πρώιμα βιντεοπαιχνίδια ήταν τα: Nimrod Computer (1951) που ζύγιζε πάνω από έναν τόνο, OXO (1952) που ουσιαστικά ήταν ένα παιχνίδι TIC-TAC-TOE, Tennis For Two (1958) και το Spacewar! (1961) το οποίο μπορούσε να παιχτεί αποκλειστικά και μόνο σε ένα μηχάνημα το οποίο κόστιζε πάνω από 100.000$. Κανένα όμως εξ αυτών των πρώιμων βιντεοπαιχνιδιών δεν πωλήθηκε στο ευρύ κοινό, καθώς ήταν είτε πολύ βαριά και ογκώδη είτε πολύ ακριβά για να βγουν από το εργαστήριο και να επιτραπεί η πρόσβαση απλού κόσμου σε αυτά.
Όλα όμως άλλαξαν το 1972 όταν ο Ralph Baer κοιτώντας την οθόνη της τηλεόρασής του αναρωτήθηκε σχετικά με το πώς αλλιώς θα μπορούσε αυτή να χρησιμοποιηθεί. Αυτή η απορία του Ralph Baer ο οποίος θεωρείται ο "πατέρας" των βιντεοπαιχνιδιών, τελικά οδήγησε το 1972 στη δημιουργία μιας κονσόλας βιντεοπαιχνιδιών με το όνομα "Odyssey" η οποία επέτρεψε στο ευρύ κοινό να παίξει παιχνίδια στην τηλεόρασή του.
Λίγο πριν, το 1971, οι Nolan Bushnell και Ted Dabney δημιουργούσαν το Computer Space, το οποίο ήταν το πρώτο βιντεοπαιχνίδι (παιχνιδομηχανή) που πωλήθηκε στο εμπόριο και λειτουργούσε με κέρματα (coin-op), το πρώτο δηλαδή arcade. Χρησιμοποιούσε μία ασπρόμαυρη τηλεόραση για την προβολή, καθώς και ένα σύστημα ηλεκτρονικών υπολογιστών, το οποίο ήταν κατασκευασμένο από 74 σειρές τσιπ TTL. Οι Bushnell και Dabney ίδρυσαν την εταιρεία η οποία κυκλοφόρησε το πρώτο βιντεοπαιχνίδι στο οποίο είχε πρόσβαση το ευρύ κοινό, το Pong(Arcade) το 1972. Αργότερα το 1974 κυκλοφόρησε μία Home έκδοση του ίδιου παιχνιδιού. «Η εμπορική επιτυχία του παιχνιδιού Pong οδήγησε πολλές άλλες εταιρείες να αναπτύξουν Pong clones και για τα δικά τους συστήματα, με αποτέλεσμα η γέννηση της βιομηχανίας των βιντεοπαιχνιδιών να ξεκινήσει» [16] . Όμως όλες αυτές οι κυκλοφορίες clones τελικά οδήγησαν στη συντριβή του original παιχνιδιού το 1977, η οποία ήρθε στο τέλος της με την επικράτηση του shooter παιχνιδιού της Midway, "Space Invaders" [16] . Το παιχνίδι αυτό σηματοδότησε την έναρξη της χρυσής εποχής του arcade, ενέπνευσε δεκάδες κατασκευαστές να εισέλθουν στην αγορά [16] και είχε σαν αποτέλεσμα εμπορικά κέντρα, εστιατόρια, καφέ και άλλα καταστήματα να γεμίσουν με αυτές τις κάπως ογκώδεις παιχνιδομηχανές, ήταν επίσης το πρώτο video game το οποίο είχε τη δυνατότητα να αποθηκεύσει high score ανάλογα με την πρόοδο των παικτών που το έπαιζαν και να δημιουργήσει έτσι κλίμα ανταγωνισμού ανάμεσα στους παίκτες. Πολυάριθμες εφημερίδες και περιοδικά γέμισαν με σχετικά άρθρα και ιστορίες, ενώ την τηλεόραση άρχιζαν να γεμίζουν διαφημίσεις βιντεοπαιχνιδιών τα οποία ήταν πια μία νέα ταχέως αναπτυσσόμενη μέθοδος ψυχαγωγίας για όλες τις ηλικίες.
Το Space Invaders σύντομα πήρε άδεια για την home κονσόλα Atari VCS (αργότερα γνωστό ως Atari 2600) και αυτό είχε σαν αποτέλεσμα τον τετραπλασιασμό στις πωλήσεις της κονσόλας, γεγονός που βοήθησε την εταιρεία Atari να ανακάμψει από τις προηγούμενες απώλειές της και με τη σειρά του το Atari VCS να αναβιώσει την αγορά των video games κατά τη διάρκεια της δεύτερης γενιάς κονσόλων[16]. Η βιομηχανία βιντεοπαιχνιδιών αναζωογονήθηκε ιδιαίτερα και από την ευρεία επιτυχία της κονσόλας Nintendo Entertaintment System (γνωστής και ως NES), η οποία αποτέλεσε ουσιαστικά το έναυσμα για την κυριαρχία της βιομηχανίας των Ιαπωνικών videogames στις Ηνωμένες Πολιτείες της Αμερικής, κατά της διάρκεια της τρίτης γενιάς κονσολών [16] . Συν τοις άλλοις, γνωστές μέχρι και σήμερα μεγάλες εταιρίες της βιομηχανίας δημιουργήθηκαν τότε στις αρχές του 1980, όπως η Sega, Electronic Arts και Activision και μπήκαν κι αυτές στον χορό των κερδών.

Το 1980 επίσης σηματοδότησε και την άφιξη έγχρωμων βιντεοπαιχνιδιών(τα μόνα βιντεοπαιχνίδια που είχαν κυκλοφορήσει μέχρι τότε ήταν ασπρόμαυρα) με πρώτο εξ' αυτών να είναι το Pacman που κυκλοφόρησε από την Namco το 1980 και με το Donkey Kong(1981) της Nintendo να είναι ένα από αυτά που ακολούθησαν.
Από το 1977 έως σήμερα έχουν κυκλοφορήσει πάρα πολλά μοντέλα κονσολών από διάφορους κατασκευαστές, ας αναφέρουμε λοιπόν επιγραμματικά τις πιο σημαντικές από αυτές με χρονολογική σειρά:

- Magnavox Odyssey (1977)
- Atari 2600 (1977)
- Commodore 64 (1982)
- Coleco Vision (1982 NA / 1983 EU)
- Nintendo Entertainment System – NES (1983)
- Sega Master System (1985)
- Sega Genesis / Mega Drive (1988)
- Nintendo Game Boy (1989) Handheld
- Super Nintendo Entertainment System – SNES (1990)
- Sega Game Gear (1990 Japan / 1991 North America and Europe) Handheld
- Atari Jaguar (1993)
- Sony PlayStation (1994)
- Sega Saturn (1994)
- Nintendo 64 (1996)
- Sega Dreamcast (1998)
- Sony PlayStation 2 (2000)
- Nintendo GameCube (2001)
- Nintendo Game Boy Advance (2001) Handheld
- Microsoft Xbox (2001)
- Nintendo DS (2004 NA+JP / 2005 EU) Handheld
- Microsoft Xbox 360 (2005)
- Sony PlayStation 3 (2006)
- Nintendo Wii (2006)
- Nintendo Wii U (2012)
- Sony PlayStation 4 (2013 NA + PAL / 2014 Japan)
- Microsoft Xbox ONE (2013 World / 2014 Japan)
- Nintendo Switch (2017) Hybrid Console: TV and Handheld
- Sony PlayStation 5 (2020)
- Microsoft Xbox Series X (2020)

Figure 4: Gaming Consoles

Με το πέρασμα των χρόνων, πέρα από τις κονσόλες και τα arcades, το PC gaming όπως και τα βιντεοπαιχνίδια σε κινητά τηλέφωνα γνώρισαν κι αυτά μεγάλη ανάπτυξη. Επίσης τα τελευταία χρόνια έχουν αρχίσει να αναπτύσσονται νέες τεχνολογίες/προτάσεις στον χώρο, όπως τα γυαλιά εικονικής πραγματικότητας - VR Oculus Rift [16] , HTC Vive, Oculus Quest, Oculus Quest 2 κτλ. Η βιομηχανία των βιντεοπαιχνιδιών έχει πια γιγαντωθεί και οι μεγάλες gaming εκθέσεις αποτελούν σημαντική βιτρίνα και τρόπο διαφήμισης. Η ετήσια έκθεση της E3 στο Λος Άντζελες (ΗΠΑ) είναι η μεγαλύτερη Expo του κόσμου για video games, όπως και η Gamescom στην Κολωνία (Γερμανία) [16] , Tokyo Game Show (Ιαπωνία), καθώς και το ετήσιο συνέδριο Game Developers Conference. Ορισμένοι εκδότες φιλοξενούν επίσης τις δικές τους εκθέσεις, με χαρακτηριστικά παραδείγματα να αποτελούν οι BlizzCon, QuakeCon.

2.3 Γνωστά Είδη βιντεοπαιχνιδιών

2.3.1 Sandbox

Βιντεοπαιχνίδια όπου ο παίκτης έχει αρκετά μεγάλη ελευθερία κινήσεων, μπορεί να επιδείξει δημιουργικότητα στον τρόπο με τον οποίο μπορεί να παίξει το παιχνίδι και συνήθως ο στόχος δεν είναι τόσο συγκεκριμένος, το gameplay είναι μη-γραμμικό, ενώ τα περιβάλλοντα στα οποία έχει τη δυνατότητα να περιπλανηθεί ο παίκτης, είναι συνήθως αρκετά μεγάλα σε έκταση. Γνωστά sandbox βιντεοπαιχνίδια: Minecraft, Sims, Grand Theft Auto.

Figure 5: Minecraft

2.3.2 Real-time strategy (RTS)

Τα συγκεκριμένα βιντεοπαιχνίδια αποτελούν υποείδος των παιχνιδιών στρατηγικής, όπου το παιχνίδι δεν εξελίσσεται σταδιακά σε γύρους (δηλαδή δεν είναι turn-based). Οι παίκτες τοποθετούν και ελίσσουν μονάδες και κατασκευές υπό τον έλεγχο τους σε ασφαλείς/εμφανείς περιοχές ενός χάρτη και καταστρέφουν πόρους του αντιπάλου. Σε ένα τυπικό RTS, είναι δυνατόν να δημιουργηθούν πρόσθετες μονάδες(πχ. στρατιώτες ή χωρικοί) και κατασκευές (πχ. πυργίσκοι ή φάρμες) κατά τη διάρκεια του παιχνιδιού, δαπανώντας συσσωρευμένους πόρους. Ο παίκτης έχει τη δυνατότητα να συγκεντρώσει αυτούς τους πόρους από συγκεκριμένες περιοχές-σημεία του χάρτη (για παράδειγμα δάση-ξυλεία, ορυχεία-χρυσός, φάρμες-κρέας) δίνοντας εντολή μέσω της διεπαφής σε συγκεκριμένες μονάδες, υπεύθυνες γι' αυτή τη λειτουργία (χωρικοί-ξυλοκόποι-μεταλλωρύχοι και τα λοιπά). Πιο συγκεκριμένα, το τυπικό παιχνίδι είδους Real-time strategy χαρακτηρίζεται από τη συγκέντρωση πόρων, κατασκευή βάσης, εντός παιχνιδιού τεχνολογική ανάπτυξη και έμμεσο έλεγχο των μονάδων. Γνωστά RTS παιχνίδια: Age Of Empires, Sid Meier's Civilization, Starcraft, Into The Breach.

Figure 6: Age Of Empires II: Definitive Edition

2.3.3 First Person Shooter (FPS)

Τα βιντεοπαιχνίδια βολών πρώτου προσώπου είναι βιντεοπαιχνίδια που εκτελούνται σε οπτική πρώτου προσώπου. Κάτι που σημαίνει ότι μέσα σε αυτά, ο παίκτης βλέπει τον κόσμο του παιχνιδιού από τα "μάτια" του ήρωα του παιχνιδιού, ουσιαστικά με μία κάμερα εφαρμοσμένη στο ύψος του κεφαλιού του μοντέλου του ήρωα, κατά την ανάπτυξη του παιχνιδιού. Στα παιχνίδια του είδους ο παίκτης χρησιμοποιεί οχι φυσικά-αληθινά όπλα, αλλά τα ψηφιακά-ψεύτικα όπλα του χαρακτήρα του παιχνιδιού, τα οποία απεικονίζονται στην οθόνη και είναι ικανά να ρίξουν εικονικές σφαίρες-βλήματα

(projectiles) προς τους εχθρούς. Αυτά τα όπλα μπορεί να είναι τυφέκια, καραμπίνες, αλυσσοπρίονα ή ακόμα και γυμνά χέρια. Τα παιχνίδια του συγκεκριμένου είδους συνήθως διαθέτουν τη δυνατότητα παιχνιδιού για έναν παίκτη αλλά και παιχνίδι μεταξύ πολλαπλών παικτών μέσω διαδικτύου. Γνωστά παιχνίδια του είδους: Quake, Wolfenstein 3D, Doom, Unreal Tournament, Medal Of Honour, Crysis, Call Of Duty, Half-Life, Soldier Of Fortune.

Figure 7: Wolfenstein 3D – id Software

2.3.4 Multiplayer Online Battle Arena (MOBA)

Τα συγκεκριμένα βιντεοπαιχνίδια (Διαδικτυακά βιντεοπαιχνίδια μάχης αρένας πολλαπλών παικτών) αποτελούν υπογενές των Real-Time Strategy (στρατηγική σε πραγματικό χρόνο) όπου δύο ομάδες παικτών ανταγωνίζονται μεταξύ τους. Κάθε παίκτης ελέγχει έναν χαρακτήρα μέσω διεπαφής. Η διαφορά είναι ότι δημιουργούνται συνήθως αυτόματα μονάδες - "creeps" από την βάση της κάθε ομάδας και οι παίκτες ελέγχουν μόνο τον χαρακτήρα τους. Άλλο ένα χαρακτηριστικό είναι το "farming", διαδικασία κατά την οποία ο παίκτης προσπαθεί να σκοτώσει πιο αδύναμους εχθρούς που δημιουργούνται κι ελέγχονται από τον υπολογιστή, με σκοπό να βελτιώσει τα στατιστικά των δυνατοτήτων του και να κερδίσει πόντους εμπειρίας (Experience Points - XP). Καταλήγουν έτσι λοιπόν τα συγκεκριμένα παιχνίδια να αποτελούν συνδυασμό μεταξύ των βιντεοπαιχνιδιών στρατηγικής σε πραγματικό χρόνο και των βιντεοπαιχνιδιών δράσης με έμφαση στην συνεργασία μεταξύ παικτών και του ομαδικού gameplay και RPG στοιχείων, με τελικό στόχο συνήθως, να καταστραφεί το φρούριο της αντίπαλης ομάδας. Γνωστά MOBA βιντεοπαιχνίδια: DOTA, League Of Legends.


Figure 8: League Of Legends – Riot Games

2.3.5 Role-playing games (RPG)

Βιντεοπαιχνίδια ρόλων χαρακτηρίζονται τα βιντεοπαιχνίδια στα οποία η ιστορία εξελίσσεται με την εξερεύνηση από τον παίκτη, του κόσμου του παιχνιδιού, ενώ οι μάχες διαδραματίζονται με μορφή εντολών και πολλαπλές επιλογές επόμενης κίνησης, μέσω της διεπαφής χρήστης του βιντεοπαιχνιδιού. Ο παίκτης έχει τη δυνατότητα παράλληλου χειρισμού ενός ή περισσότερων χαρακτήρων σε έναν κόσμο δημιουργημένο από τη φαντασία του σχεδιαστή του παιχνιδιού. Η βασική επιρροή των συγκεκριμένων βιντεοπαιχνιδιών είναι τα παραδοσιακά επιτραπέζια παιχνίδια ρόλων, με τη διαφορά ότι μέσω της εξελλικτικής διαδικασίας και της ψηφιοποίησης, εμπλουτίστηκαν με γραφικό περιβάλλον, βίντεο, animations, κείμενο και ήχο και μετατράπηκαν σε πλούσιες, πότε δισδιάστατες και πότε τρισδιάστατες πολυμεσικές εμπειρίες. Τα παιχνίδια του συγκεκριμένου είδους έχουν την ίδια ονοματολογία, παραμέτρους και mechanics με τα πρώιμα παιχνίδια ρόλων, όπως το Dungeons & Dragons. Άλλες ομοιότητες μπορούν να είναι το ανεπτυγμένο σενάριο, η αφήγηση της ιστορίας, η ανάπτυξη των χαρακτήρων και η πολυπλοκότητά τους καθώς και η αξία έναρξης του παιχνιδιού ξανά από την αρχή (replayability). Ο παίκτης χειρίζεται τον κεντρικό χαρακτήρα ή πολλαπλούς χαρακτήρες με το σύστημα εντολών, ενώ η αποτελεσματικότητα του καθενός εξαρτάται από τη στατιστική ανάπτυξή του μέσω της διαδικασίας του leveling, με την οποία στο τέλος της μάχης ο κάθε ένας λαμβάνει πόντους εμπειρίας (experience) που ανεβάζουν τα στατιστικά της δύναμης, της άμυνας, της επιδεξιότητας, τύχης και άλλα. Ο κάθε χαρακτήρας μπορεί να διαθέτει όπλα και αντικείμενα που ενισχύουν τις δυνάμεις και τα χαρακτηριστικά του, ενώ μπορεί να ανήκει σε κάποια κατηγορία επαγγέλματος ή ιδιότητας, όπως μάγος, κλέφτης, βάρδος, πολεμιστής και άλλα και συνήθως εκτελεί αποστολές (quests). Γνωστά RPG βιντεοπαιχνίδια: Final Fantasy, Disco Elysium, World Of Warcraft, Lineage, Persona, Might and Magic, Dragon Quest, Diablo, Yakuza: Like A Dragon.

Figure 9: Disco Elysium – ZA/UM

2.3.6 Racing Games

Τα βιντεοπαιχνίδια αγώνων είναι είδος βιντεοπαιχνιδιών που εκτελούνται σε τρίτο ή πρώτο πρόσωπο (βλέποντας δηλαδή ή όχι το τρισδιάστατο μοντέλο του οχήματος) και στα οποία ο παίκτης λαμβάνει μέρος σε αγώνες με οποιοδήποτε τύπο οχήματος ξηράς, θαλάσσης ή άλλου τύπου οχήματος (Star Wars Episode I: Racer, διαστημικά οχήματα στη συγκεκριμένη περίπτωση). Αποτελούν αναπαράσταση επίσημων διοργανώσεων αγώνων (για παράδειγμα Formula 1) ή φανταστικών αγώνων (καταδιώξεις ή αγώνες με περίτεχνα οχήματα και φανταστικούς οδηγούς – ήρωες – Need For Speed και Twisted Metal αντίστοιχα, ως παραδείγματα). Χαρακτηριστικά παραδείγματα του

είδους: Gran Turismo, Forza Horizon, Need For Speed, Microsoft Flight Simulator, Twisted Metal, Carmageddon, Wipeout64, OutRun.

Figure 10: OutRun - SEGA

2.3.7 Sports Games

Αποτελούν προσωμοίωση πραγματικών αθλημάτων όπως το ποδόσφαιρο, η αντισφαίριση και η καλαθοσφαίριση. Είναι από τα πιο παλιά είδη βιντεοπαιχνιδιών και οι πιο πρόσφατες εκδόσεις τους έχουν εξελιχθεί σε υπερθετικό βαθμό σε σχέση με τους προκατόχους τους. Περιλαμβάνουν δοκιμασίες που βασίζονται στο πόσο καλή τεχνική έχει αποκτήσει ο παίκτης στον ιδιαίτερο χειρισμό του κάθε παιχνιδιού και γίνεται προσπάθεια μίμησης και μοντελοποίησης ρεαλιστικών χαρακτηριστικών, όπως η ταχύτητα, η ακρίβεια, η επιτάχυνση, η δύναμη των αθλητών ή/και της μπάλας, ενώ τα τελευταία χρόνια χρησιμοποιούνται αισθητήρες σε πραγματικούς αθλητές για να αποδοθεί πιο ρεαλιστικά η ανθρώπινη κίνηση μέσα στο βιντεοπαιχνίδι σε σημείο πλέον που τα γραφικά και το animation δεν απέχουν πολύ από την παρακολούθηση ενός πραγματικού αγώνα στην τηλεόραση. Γνωστά βιντεοπαιχνίδια του είδους: Pro Evolution Soccer, Fifa, NBA2K, Madden NFL, Pong.

Figure 11: FIFA 2000 – EA Sports

2.3.8 Puzzle Adventure Games

Αποτελούν υποκατηγορία των Adventure games. Τα παιχνίδια του συγκεκριμένου είδους μπορεί να είναι πολύ απλά με απόλυτα μινιμαλιστικό γραφικό περιβάλλον-διεπαφή χρήστη, όπως το γνωστό Minesweeper (Ναρκαλιευτής) ή πιο immersive (εμβυθιστικά), σύνθετα, με πλήρως ανεπτυγμένα τρισδιάστατα ή δισδιάστατα περιβάλλοντα, περιβαλλοντικούς ήχους και υβριδικό gameplay. Το βιντεοπαιχνίδι Portal είναι ένα χαρακτηριστικό παράδειγμα, όπου τα puzzles παρουσιάζονται πότε με έναν συμβατικό τρόπο και άλλοτε με έναν αντισυμβατικό τρόπο σε μορφή δοκιμασιών για τον παίκτη προκειμένου να προχωρήσει στη συνέχεια του παιχνιδιού και σε επόμενες δοκιμασίες χρησιμοποιώντας συγκεκριμένα mechanics. Χαρακτηριστικά video games του είδους: Portal, Talos Principle, The Witness, Superliminal, Antichamber, Limbo-Inside-Tamashii (Puzzle Platform Games).

Figure 12: Portal 2 - Valve

2.3.9 Action-adventure Games

Τα βιντεοπαιχνίδια του συγκεκριμένου είδους αποτελούν υβρίδιο παιχνιδιών δράσης και παιχνιδιών περιπέτειας-γρίφων. Εστιάζουν σε ένα συνδυασμό αφήγησης-εξιστόρησης, mechanics μάχης και επίλυσης γρίφων. Σαν αποτέλεσμα αυτού του συνδυασμού, πολλά βιντεοπαιχνίδια εμπίπτουν στην συγκεκριμένη κατηγορία όπως οι κυκλοφορίες της γνωστής σειράς Legend Of Zelda. Άλλα χαρακτηριστικά παραδείγματα: Tomb Raider, Assasin's Creed, Sekiro, Yakuza.

Figure 13: Rise Of The Tomb Raider – Square Enix

2.3.10 Adventure Games

Στο συγκεκριμένο είδος βιντεοπαιχνιδιών ο παίκτης αναλαμβάνει το ρόλο ενός πρωταγωνιστή σε μία διαδραστική ιστορία με κυρίαρχα στοιχεία την αφήγηση, την εξερεύνηση, την συλλογή αντικειμένων και την επίλυση γρίφων (puzzle solving). Το γεγονός ότι τα παιχνίδια του συγκεκριμένου είδους εστιάζουν στο "ξετύλιγμα" μιας ιστορίας (storytelling), τους επιτρέπει να αντλήσουν στοιχεία και από άλλα αφηγηματικά μέσα (media), όπως οι κινηματογραφικές ταινίες και η λογοτεχνία. Τα περισσότερα είναι σχεδιασμένα αποκλειστικά για παιχνίδι ενός και μόνο παίκτη, καθώς η έμφαση που δίνεται στην ιστορία και τον χαρακτήρα, καθιστά αδύνατο τον σχεδιασμό παιχνιδιού για πολλούς παίκτες. Τα πρώτα adventure games της δεκαετίας του 1970 και των αρχών της δεκαετίας του 1980 ήταν βασισμένα σε κείμενο (text based adventures), χωρίς δηλαδή να περιέχουν γραφικό περιβάλλον και χρησιμοποιούσαν text parsers (συντακτικούς αναλυτές) για να μεταφράσουν το κείμενο που εισήγαγε ο παίκτης σε in-game εντολές (εντολές μέσα στο παιχνίδι) οι οποίες πυροδοτούν γεγονότα. Με την πάροδο του χρόνου και όσο τα συστήματα των προσωπικών υπολογιστών γίνονταν πιο ισχυρά με ανεπτυγμένες δυνατότητες στην απόδοση γραφικών, έγινε εφικτή και η ανάπτυξη adventure games με πιο εμβυθιστικά γραφικά πραγματικού χρόνου για τον παίκτη ή τρισδιάστατους pre-rendered χώρους, ακόμα και Full Motion Video (FMV), δηλαδή προ-βιντεοσκοπημένες καταγραφές σε ποιότητα ταινίας ή τηλεόρασης αντί για γραφικά στοιχεία (sprites), διανύσματα ή τρισδιάστατα μοντέλα για να απεικονιστεί η δράση στο παιχνίδι. Μία πολύ γνωστή αλλά και κλασική υποκατηγορία των adventure games, συνδεμένη με την "χρυσή" εποχή του είδους είναι τα Point and Click adventures. Στα συγκεκριμένα βιντεοπαιχνίδια, ο παίκτης ελέγχει τον χαρακτήρα του μέσω μίας point and click διεπαφής, χρησιμοποιώντας ουσιαστικά ένα ποντίκι υπολογιστή. Ο παίκτης κλικάρει για να μετακινήσει τον χαρακτήρα του στο περιβάλλον, για να αλληλεπιδράσει με άλλους χαρακτήρες και να ξεκινήσει διαλόγους με αυτούς, για να εξετάσει (examine) αντικείμενα στο περιβάλλον ή στο inventory του που αποτελεί ουσιαστικά την λίστα με τα αντικείμενα που έχει περισυλλέξει. Αρκετά συχνά τα παιχνίδια του συγκεκριμένου είδους συνοψίζονται στη συλλογή "χρήσιμων" αντικειμένων από το περιβάλλον και χρήση τους την κατάλληλη στιγμή για επίλυση προβλημάτων/γρίφων. Ο παίκτης θα πρέπει να χρησιμοποιήσει στοιχεία που έχει συλλέξει από την εξερεύνηση και την παρατήρηση του οπτικού μέρους του παιχνιδιού, τις περιγραφές των διαφόρων αντικειμένων και τους διαλόγους με τους υπόλοιπους χαρακτήρες, έτσι ώστε να ανακαλύψει πότε είναι η κατάλληλη στιγμή ή τοποθεσία για να χρησιμοποιήσει το εκάστοτε αντικείμενο (item). Πολύ γνωστά Point and Click Adventures: Mystery House, Machinarium, Monkey Island, Grim Fandango, Broken Sword, Myst, Yuppie Psycho (επιρροές από point and click και στοιχεία από άλλα είδη, πχ. Κίνηση χαρακτήρα με WASD πέρα απ' το κλικ για διάδραση με αντικείμενα, Resources, healing items κτλ.)

Figure 14: Grim Fandango – Lucas Arts

2.3.11 Survival Horror Games

Τα βιντεοπαιχνίδια επιβίωσης και τα βιντεοπαιχνίδια τρόμου έχουν αρκετά χαρακτηριστικά που επικαλύπτονται, κάτι που οδήγησε στο να δημιουργηθεί ένα ξεχωριστό υποείδος (survival horror). Τα βασικά χαρακτηριστικά ενός βιντεοπαιχνιδιού επιβίωσης είναι η διαχείριση πόρων, που συχνά περιλαμβάνει σύστημα crafting, δηλαδή τη δυνατότητα του παίκτη να δημιουργήσει εικονικά αντικείμενα μέσα στο παιχνίδι χρησιμοποιώντας πόρους που έχει περισυλλέξει, έτσι ώστε να μπορέσει ο χαρακτήρας του μέσα στο παιχνίδι να επιβιώσει/να διατηρηθεί ζωντανός (παρόμοια στοιχεία "επιβίωσης" μπορούν να βρεθούν και σε Action/Adventure παιχνίδια, όπως ο γνωστός τίτλος Rise Of The Tomb Raider της εταιρείας Crystal Dynamics). Τα βιντεοπαιχνίδια τρόμου αποτελούν μία ακόμη πιο ευρεία κατηγορία που αδιαμφισβήτητα περιλαμβάνει δεκάδες τίτλους βιντεοπαιχνιδιών επιβίωσης. Σχεδόν οποιοδήποτε παιχνίδι με zombies, μετα-αποκαλυπτική υπόθεση και στιγμές που κάποιος ή κάτι ξεπετάγεται απρόσμενα με σκοπό να τρομάξει τον παίκτη (να προκαλέσει jump scares δηλαδή), θεωρείται βιντεοπαιχνίδι τρόμου. Οι δημιουργοί των συγκεκριμένων βιντεοπαιχνιδιών επιχειρούν με ψυχολογικά τρικ, οπτικά ή ηχητικά, να προκαλέσουν ένταση, φόβο, τρόμο και αγωνία στον παίκτη και να τον εμβυθίσουν με αυτόν τον τρόπο στη συγκεκριμένη εμπειρία. Κάποιοι χαρακτηριστικοί τίτλοι horror βιντεοπαιχνιδιών είναι οι Amnesia: The Dark Descent, Alien: Isolation και Blair Witch, όμως οι πιο δημοφιλής και με αρκετά μεγάλο fan base και ιδιαίτερα ευρύ κοινό, τίτλοι τρόμου εμπίπτουν στην κατηγορία Survival Horror (επιβίωσης τρόμου), όπως είναι τα βιντεοπαιχνίδια των franchises Resident Evil, Silent Hill και Evil Within, στα οποία είναι και πιο έντονο το χαρακτηριστικό στοιχείο των περιορισμένων πόρων (first-aid sprays, περιορισμένο ammo για παράδειγμα), σχεδιαστική επιλογή που προφανώς προσδίδει χαρακτήρα επιβίωσης στην όλη εμπειρία για τον παίκτη σε συνδυασμό με τις ορδές των εχθρών που έχουν σκοπό να εμποδίσουν την πορεία του παίκτη προς την ολοκλήρωση του παιχνιδιού.

Figure 15: Silent Hill - Konami

2.3.12 Platformer Games

Αποτελούν αναμφίβολα ένα είδος βιντεοπαιχνιδιών που η έννοιά τους δεν έχει μεταβληθεί ιδιαίτερα με το πέρασμα των χρόνων. Το είδος των παιχνιδιών πλατφόρμας καλύπτει μία τεράστια ποικιλία βιντεοπαιχνιδιών που παρουσιάζουν τις ρίζες τους και τις επιρροές τους από τα πιο πρώιμα δισδιάστατα παιχνίδια πλευρικής κύλισης (side-scrollers). Στα Platformers ο ελεγχόμενος από τον παίκτη κεντρικός ήρωας δύναται να τρέχει, να πηδάει και να σκαρφαλώνει ενώ ταυτόχρονα

εξερευνάει πίστες και μετέχει σε δοκιμασίες. Ένα χαρακτηριστικό των δισδιάστατων παιχνιδιών πλατφόρμας είναι η πλευρική άποψη (side view) με το Donkey Kong να αποτελεί ένα από τα πρώτα δημοφιλή παραδείγματα του είδους. Το συγκεκριμένο βιντεοπαιχνίδι έδωσε την σκυτάλη στο Super Mario Bros. της Nintendo και ύστερα στο Sonic The Hedgehog της Sega. Μερικά χρόνια αργότερα το Crash Bandicoot της Naughty Dog απέκτησε το δικό του κοινό φανατικών υποστηρικτών, με νέα γωνία κάμερας (head-on), τρισδιάστατα γραφικά, side-scrolling (πλευρική κύλιση) στιγμές που αποτελούσαν φόρο τιμής στο νοσταλγικό παρελθόν του είδους και τη δική του ξεχωριστή γοητεία. Με την πάροδο των χρόνων τα βιντεοπαιχνίδια πλατφόρμας διασταυρώθηκαν με άλλα είδη (RPG, Puzzle κτλ.) κάτι που οδήγησε στη δημιουργία μερικών πολύ ιδιαίτερων crossover τίτλων. Γνωστά platformers: Mario 64, Banjo Kazooie, Donkey Kong Country, A Hat In Time, Psychonauts, Cuphead, Alex Kidd.

Figure 16: Alex Kidd In Miracle World - SEGA

2.3.13 Fighting Games

Τα βιντεοπαιχνίδια μάχης είναι βιντεοπαιχνίδια που επικεντρώνονται στην πάλη μεταξύ ζευγαριών μαχητών, με κυρίαρχα στοιχεία το μπλοκάρισμα επιθέσεων (blocking), λαβές (grappling), αντεπιθέσεις (counter-attacking) και αλυσίδες επιθέσεων (chaining attacks) με μορφή συνδυασμού κινήσεων και χτυπημάτων (combos). Στα παιχνίδια του συγκεκριμένου είδους έχουμε κατά κύριο λόγο απεικόνιση των μαχητών σε πλευρική άποψη, ενώ ακόμα και στα τρισδιάστατα παιχνίδια του συγκεκριμένου στυλ η κίνηση περιορίζεται στο δισδιάστατο επίπεδο συμπεριλαμβάνοντας πλάγιους βηματισμούς (sidestepping) σε βάθος ή προς την κάμερα, για την ψευδαίσθηση τρισδιάστατης κίνησης, αλλά και για την αποφυγή επιθέσεων ή έναυσμα νέων combos. Τα συγκεκριμένα παιχνίδια εστιάζουν κυρίως σε διαπληκτισμούς του δρόμου, αλλά και αγώνες επίδειξης με τεχνικές γνωστών ή και άγνωστων πολεμικών τεχνών μεταξύ των αντιπάλων. Οι χαρακτήρες δίνουν μάχη μεταξύ τους μέχρι ο ένας να νικήσει τον άλλο (εξάντληση της μπάρας ζωής του καθενός – health bar) ή μέχρι να εξαντληθεί ο προκαθορισμένος χρόνος. Οι αγώνες αποτελούνται από συγκεκριμένο αριθμό γύρων σε μία πίστα-αρένα, ενώ ο κάθε χαρακτήρας έχει τις δικές του ξεχωριστές ιδιότητες, αλλά και ο καθένας από αυτούς είναι το ίδιο εφικτό να επιλεγεί απ' τον εκάστοτε παίκτη, κάτι που αποτελεί booster στην competitive φύση των συγκεκριμένων παιχνιδιών. Κάποια από τα πιο γνωστά Fighting Games: Street Fighter, Mortal Kombat, Super Smash Bros., Tekken, Virtua Fighter.

Figure 17: Street Fighter 2 Arcade - CAPCOM

2.4 Μηχανές Παιχνιδιών

Για να αναπτυχθεί ένα βιντεοπαιχνίδι είναι αναγκαία η ύπαρξη και του κατάλληλου λογισμικού. Συνήθως οι εταιρείες που αναπτύσσουν βιντεοπαιχνίδια πριν ξεκινήσουν να τα υλοποιούν, δημιουργούν τέτοια λογισμικά έτσι να διευκολύνουν τη διαδικασία σχεδίασης.[8] Μία σειρά παιχνιδιών μπορεί να αναπτυχθεί λοιπόν με την χρήση του ίδιου λογισμικού, μέχρι η εκάστοτε εταιρεία να το αντικαταστήσει γιατί φτάνει σε σημείο πια να θεωρείται απαρχαιωμένο με την πάροδο των χρόνων και την εξέλιξη της τεχνολογίας. Ένα λογισμικό τέτοιου τύπου, κατάλληλο για τη δημιουργία και σχεδίαση βιντεοπαιχνιδιών ονομάζεται Μηχανή Παιχνιδιών (Game Engine). Αποτελεί ουσιαστικά ένα ολοκληρωμένο περιβάλλον ανάπτυξης, μια έτοιμη δηλαδή σουίτα λογισμικού, βασικών οπτικών εργαλείων και έτοιμων συστημάτων που μπορούν να επαναχρησιμοποιηθούν και ο προγραμματιστής δεν είναι υποχρεωμένος να σπαταλήσει χρόνο για να τα υλοποιήσει από την αρχή, κερδίζοντας έτσι χρόνο για την ανάπτυξη αυτού καθεαυτού του βιντεοπαιχνιδιού, κάτι το οποίο στηρίζεται στο γεγονός πως τα περισσότερα παιχνίδια έχουν παρόμοιες βασικές λειτουργικότητες. Κάποια από τα έτοιμα συστήματα τα οποία δεν χρειάζεται να αναπτύξει ή να γράψει κώδικα για τη δημιουργία τους εξαρχής ο game developer και του παρέχονται έτοιμα για χρήση μετά την αρχική εγκατάσταση της μηχανής παιχνιδιών στον υπολογιστή του, είναι μια μηχανή φωτοαπόδοσης (renderer) για είτε δισδιάτατα είτε τρισδιάστατα γραφικά, μία μηχανή για υπολογισμούς φυσικής και εντοπισμό και προσομοίωση συγκρούσεων (physics engine-collision detection), ένα σύστημα ήχου με αρκετές παραμέτρους και διαφορετικές ρυθμίσεις για την οργάνωση και απόδοση των διαφορετικών ήχων του παιχνιδιού (ηχητικά effects, μουσική, περιβαλλοντικοί ήχοι, ήχοι μενού), σύστημα scripting και μεταγλωττιστή για τη μεταγλώττιση πηγαίου κώδικα ο οποίος είναι γραμμένος σε κάποια γλώσσα υψηλού επιπέδου όπως η C# για παράδειγμα, σε γλώσσα μηχανής και εντολές που μπορούν να είναι κατανοητές για την κεντρική μονάδα επεξεργασίας (CPU), έτοιμο σύστημα τεχνητής νοημοσύνης και pathfinding για τον καθορισμό ή περιορισμό κίνησης πρακτόρων σε περίπλοκους χώρους ή λαβυρίνθους (για παράδειγμα NavMesh Agent – Unity), έτοιμο σύστημα διαχείρισης μνήμης (memory management) για τον καταμερισμό και τη διαχείριση – δέσμευση – αποδέσμευση και την αυτοματοποιημένη ή χειροκίνητη διαγραφή των references στη managed memory (πιο συγκεκριμένα GC.alloc – GC.collect, μιλώντας συγκεκριμένα για τη μηχανή της Unity) κατά το runtime. Αυτές οι έτοιμες για χρήση μηχανές παιχνιδιών συχνά καλούνται “game middleware”, καθώς παρέχουν μια ευέλικτη και επαναχρησιμοποιήσιμη πλατφόρμα λογισμικού η οποία παρέχει όλη την κεντρική λειτουργικότητα που απαιτείται, άμεσα- “out of the box”, για την ανάπτυξη μιας εφαρμογής παιχνιδιού, ενώ ταυτόχρονα μειώνουν το κόστος, την πολυπλοκότητα, την δυσκολία της ανάπτυξης της εφαρμογής και τον χρόνο μέχρι την κυκλοφορία του παιχνιδιού στην αγορά – όλοι κρίσιμοι παράγοντες στην υψηλά ανταγωνιστική βιομηχανία των βιντεοπαιχνιδιών. Ας μιλήσουμε όμως με λίγα λόγια για μερικές γνωστές μηχανές παιχνιδιών.

2.4.1 Reach For The Moon Engine- RE Engine

Figure 18: RE Engine logo

Η Reach For The Moon Engine, ή αλλιώς RE Engine, είναι μία μηχανή παιχνιδιών που αναπτύχθηκε από την Capcom. Υλοποιήθηκε αρχικά, το 2014, ειδικά για το παιχνίδι Resident Evil 7: Biohazard και κατά την έναρξη της ανάπτυξής του, κι από τότε έχει χρησιμοποιηθεί και για τη δημιουργία ποικίλων άλλων τίτλων της Capcom, όπως το Devil May Cry 5 και το Monster Hunter Rise. Η συγκεκριμένη μηχανή παιχνιδιών αποτελεί τον διάδοχο της μηχανής MT Framework, της προηγούμενης μηχανής παιχνιδιών που είχε δημιουργηθεί από την Capcom και μέσω της οποίας επίσης είχαν αναπτυχθεί αρκετοί γνωστοί τίτλοι της. Η συγκεκριμένη μηχανή σχεδιάστηκε από τους προγραμματιστές της, οι οποίοι είχαν κατά νου τη γραμμική φύση του παιχνιδιού που είχαν σκοπό να σχεδιάσουν και ήθελαν να δημιουργήσουν το συγκεκριμένο game engine για να καταστήσουν τη διαδικασία ανάπτυξης του παιχνιδιού πιο αποτελεσματική. Ο λόγος που δεν χρησιμοποίησαν κάποια άλλη μη-εξειδικευμένη μηχανή παιχνιδιών άλλου κατασκευαστή ήταν όπως είπαν, ότι "μία μη-εξειδικευμένη μηχανή παιχνιδιών ανεπτυγμένη από κάποια άλλη εταιρεία δεν θα ήταν κατάλληλη για την ανάπτυξη ενός παιχνιδιού όπως το Resident Evil 7" [8]. Δεν ήταν δυνατή η χρήση της MT Framework, λόγω των αργών της εργαλείων ανάπτυξης, οπότε ήταν αναγκαίο η Capcom να σχεδιάσει μια νέα μηχανή για να πραγματοποιηθεί η "asset-based" ανάπτυξη του παιχνιδιού, δηλαδή ανάπτυξη με κυρίαρχα στοιχεία τα γραφικά και τα τρισδιάστατα μοντέλα. Η RE Engine παρουσιάζει ποικίλες βελτιώσεις σε σχέση με την MT Framework, που περιλαμβάνουν νέο σύστημα anti-aliasing (τεχνική εξομάλυνσης ορίων) και ογκομετρικού φωτισμού (volumetric lighting). Η εν λόγω μηχανή υποστηρίζει επιπλέον την χρήση φωτογραμμετρίας για τη δημιουργία μοντέλων και γραφικών υψηλότερης ποιότητας και πιστότητας καθώς και βελτιωμένη υποστήριξη για εικονική πραγματικότητα σε σχέση με την προκάτοχό της (MT Framework), που επιτρέπει υψηλότερο ρυθμό απόδοσης πλαισίων (framerate), απαραίτητου για να αποφευχθεί το motion sickness, ένα από τα βασικά προβλήματα που αναδύονται κατά την ανάπτυξη εφαρμογών VR. Επιπλέον περιλαμβάνει εργαλεία που κάνουν εφικτή την ταχύτερη δημιουργία animation (κινούμενη εικόνα), με τεχνικές όπως το modular rigging, το motion matching, το procedural animation και το motion retargeting. Άλλο ένα εκ των πλεονεκτημάτων της RE Engine είναι ότι διαθέτει διάφορες νέες επιλογές για προσωμοιώσεις φυσικής που επιτρέπουν την πιο ρεαλιστική προσωμοίωση θραυσμάτων.

2.4.2 Source + Source 2 Engine

Figure 19: Source Engine logo

Η Source είναι μία 3D μηχανή παιχνιδιών, η οποία αναπτύχθηκε από την Valve. Έκανε το ντεμπούτο της τον Ιούνιο του 2004 ως διάδοχος της GoldSrc[8], μίας τροποποιημένης εκδοχής της Quake Engine, με το παιχνίδια Half Life: Source και ακολούθως με τα παιχνίδια Counter-Strike: Source και Half Life 2 τον Νοέμβριο του ίδιου χρόνου και από τότε είναι σε ενεργή ανάπτυξη με συνεχή updates τουλάχιστον μέχρι να δώσει την σκυτάλη στη διάδοχο μηχανή παιχνιδιών Source 2 από τα τέλη της δεκαετίας του 2010. Η Source δημιουργήθηκε με σκοπό να αναβαθμίζεται σταδιακά μαζί με τις εξελίξεις της τεχνολογίας, σε αντίθεση με άλλες ανταγωνιστικές μηχανές παιχνιδιών που τα version jumps τους "έσπαγαν" τη συμβατότητα και δεν μπορούσαν να χρησιμοποιηθούν με παλιά games: διαφορετικά συστήματα που αντιπροσωπεύονταν από διαφορετικά τμήματα της Source engine μπορούσαν να αναβαθμιστούν ξεχωριστά. Υπήρξαν βέβαια κάποιες εκδόσεις της μηχανής που "έσπαγαν" κι αυτές την αλυσίδα της συμβατότητας. Οι κυκλοφορίες του Half-Life 2: Episode One και του The Orange Box ουσιαστικά θέσπισαν νέες εκδόσεις της Source, οι οποίες δεν μπορούσαν να χρησιμοποιηθούν για να τρέξουν παλιότερα παιχνίδια ή mods, χωρίς οι προγραμματιστές να χρειαστεί να επέμβουν και να κάνουν αλλαγές στον κώδικα της μηχανής. Κάτι που και πάλι απαιτούσε λιγότερη δουλειά απ' την πλευρά των developers σε σύγκριση με αντίστοιχες περιπτώσεις παιχνιδιών, στημένων σε άλλες μηχανές παιχνιδιών. Οι τρεις βασικές εφαρμογές, συστατικά στοιχεία του Source SDK, του software development kit δηλαδή της Source Engine που έγινε δωρεάν για όλους τους χρήστες του Steam μετά την κυκλοφορία του Team Fortress 2, είναι το Hammer Editor, το Model Viewer και το Face Poser. Το Model Viewer tool επιτρέπει ουσιαστικά την οπτική επισκόπηση των τρισδιάστατων μοντέλων στη μηχανή, για τον χρήστη. Οι developers μπορούν να χρησιμοποιήσουν το συγκεκριμένο εργαλείο για προβολή των μοντέλων και των animations τους, καθώς και των συστατικών στοιχείων-components αυτών και ούτω καθεξής. Το Face Poser είναι ένα εξειδικευμένο εργαλείο για πρόσβαση στις εκφράσεις προσώπων των χαρακτήρων, καθώς και στα συστήματα χορογραφίας και κίνησης. Επιτρέπει την επεξεργασία των εκφράσεων προσώπου, χειρονομιών και κίνησης των χαρακτήρων καθώς και τον συντονισμό των χειλιών με τον ήχο της ομιλίας και επιπλέον δίνει τη δυνατότητα προεπισκόπησης τού πώς όλα αυτά θα οπτικοποιηθούν από την μηχανή γραφικών κατά το runtime. Το Hammer Editor είναι ο επίσημος level editor (και όχι μόνο) της μηχανής Source. Άλλο ένα γνωστό εργαλείο της Source είναι το Source Filmmaker, χρήσιμο για τη δημιουργία βίντεο και trailers παιχνιδιών του engine. Ο διάδοχος της Source, η μηχανή παιχνιδιών Source 2 ανακοινώθηκε το 2015 και το πρώτο παιχνίδι που την χρησιμοποίησε ήταν το Dota 2, με μεταφορά του στη συγκεκριμένη μηχανή από την προηγούμενη έκδοση που είχε αναπτυχθεί (την Source). Τα σχέδια για τον διάδοχο της αρχικής Source είχαν ξεκινήσει ήδη από το 2007 με την κυκλοφορία του Half-Life 2: Episode Two. Το πρώτο tech demo για την Source 2 κυκλοφόρησε το 2010. Το 2015 η Valve ανακοίνωσε στο Game Developers Conference πως η νέα της μηχανή θα παρείχε support για το Vulkan Graphics API (διεπαφή προγραμματισμού εφαρμογών που χρησιμοποιείται κυρίως για τρισδιαστάτα γραφικά βιντεοπαιχνιδιών), καθώς και μία νέα ενσωματωμένη physics engine, την Rubikon που θα αντικαθιστούσε τα third-party εργαλεία για physics της Havok. Αργότερα η Source 2 έγινε συμβατή με Android και iOS. Η μηχανή επιπλέον είναι συμβατή με VR κι έχει χρησιμοποιηθεί στο Robot Repair tech demo που περιέχεται στο The Lab, καθώς και στο πολύ γνωστό Half-Life: Alyx. Κάποια εργαλεία της Source 2 που σχεδιάστηκαν ειδικά για τη δημιουργία περιεχομένου αποκλειστικά για το παιχνίδι Half-Life: Alyx κυκλοφόρησαν

τον Μάιο του 2020.

2.4.3 Unreal Engine

Figure 20: Unreal Engine logo

Η Unreal Engine είναι μία μηχανή παιχνιδιών που αναπτύχθηκε απ' την Epic Games και παρουσιάστηκε για πρώτη φορά το 1998 στο παιχνίδι βολών πρώτου προσώπου Unreal. Αρχικά αναπτύχθηκε συγκεκριμένα για τη δημιουργία παιχνιδιών βολών πρώτου προσώπου και από τότε έως και σήμερα έχει χρησιμοποιηθεί για τη δημιουργία πάρα πολλών τρισδιάστατων παιχνιδιών πολλών διαφορετικών ειδών, η χρήση της όμως έχει υιοθετηθεί και από άλλες βιομηχανίες, όπως ο κινηματογράφος και η τηλεόραση. Προγραμματισμένη στην γλώσσα C++, η εν λόγω μηχανή παιχνιδιών χαρακτηρίζεται από την σε μεγάλο βαθμό ευελιξία της, καθώς υποστηρίζει και είναι συμβατή με μία μεγάλη γκάμα από πλατφόρμες (υπολογιστές γραφείου, κινητά, κονσόλες, εικονική πραγματικότητα). Απ' το 2015 η λήψη της συγκεκριμένης μηχανής παιχνιδιών είναι δωρεάν και ο πηγαίος της κώδικας είναι διαθέσιμος μέσω GitHub. Η Epic Games επιτρέπει την χρήση της Unreal Engine σε εμπορικά προϊόντα, κρατώντας το 5% του συνόλου των εσόδων πωλήσεων, εκτός και αν το παιχνίδι κυκλοφορήσει στο Epic Games Store [8]. Στις 13 Μαϊου του 2020, η Epic ανακοίνωσε ότι παραιτείται των συγκεκριμένων δικαιωμάτων επί των εσόδων αρκεί οι πωλήσεις του παιχνιδιού να έχουν αποδόσει έσοδα λιγότερα από ένα εκατομμύριο δολλάρια. Η επόμενη έκδοση της δημοφιλούς αυτής μηχανής παιχνιδιών, η Unreal Engine 5 έχει σχεδιαστεί να κυκλοφορήσει στις αρχές του 2022. Η πρώτη έκδοση της Unreal Engine υλοποιήθηκε απ' τον Tim Sweeney, τον ιδρυτή της Epic Games. Μερικά από τα χαρακτηριστικά της ήταν η ανίχνευση συγκρούσεων (collision detection), ο χρωματιστός φωτισμός και μία μορφή φιλτραρίσματος υφών (texture filtering) περιορισμένων δυνατοτήτων. Η μηχανή είχε επιπλέον τον δικό της level editor, τον UnrealEd. Η επόμενη έκδοση της μηχανής, Unreal Engine 2, έκανε το ντεμπούτο της το 2002 και παρουσίαζε μάλιστα μεγάλη πρόοδο σε σχέση με την προκάτοχό της όσον αφορά σε θέματα απόδοσης γραφικών (rendering) καθώς και πλειάδα ενσωματωμένων εργαλείων. Ήταν ικανή να "τρέχει" πίστες με 100 φορές περισσότερη λεπτομέρεια απ' ότι η αρχική Unreal ενώ επιπροσθέτως παρείχε δικό της σύστημα σωματιδίων (particle system), plug-ins για εξαγωγή μοντέλων σε Studio Max και Maya καθώς και σύστημα για skeletal animation. Η επόμενη έκδοση της μηχανής, Unreal Engine 3, ήταν βασισμένη στην πρωταρχική έκδοση της μηχανής, αλλά φυσικά είχε νέα χαρακτηριστικά. Η βασική αρχιτεκτονική ορατή στους προγραμματιστές παρέμεινε, καθώς και ο αντικειμενοστραφής σχεδιασμός. Τα στοιχεία των παιχνιδιών όμως τα οποία ήταν ορατά στους τελικούς χρήστες/ gamers (τα γραφικά και ο φωτισμός, το σύστημα physics, το σύστημα ήχου και τα νέα εργαλεία) παρουσίαζαν δραματική βελτίωση και ήταν πολύ πιο ισχυρά απ' τα αντίστοιχα της Unreal Engine 2, ενώ υπήρχε επιπλέον η δυνατότητα για υπολογισμό φωτισμού και σκιών ανά εικονοστοιχείο (per pixel) και όχι ανά κορυφή τριγώνου (per vertex), κάτι που είχε φυσικά σαν αποτέλεσμα πολύ πιο εντυπωσιακά γραφικά. Η επόμενη "γενιά" της μηχανής, η Unreal Engine 4, πέρα από τις νέες βελτιώσεις στα γραφικά, περιελάμβανε και το σύστημα visual scripting "Blueprints", το οποίο επιτρέπει την ταχεία ανάπτυξη λογικής παιχνιδιού χωρίς να απαιτείται κώδικας, καθώς και μια

μεγάλη γκάμα από υποστηριζόμενες πλατφόρμες. Η τελευταία γενιά, τέλος, της μηχανής παιχνιδιών, η Unreal Engine 5 είχε αποκαλυπτήρια σε μορφή early access στις 26 Μαΐου του 2021 και με την επίσημη κυκλοφορία να υπολογίζεται στις αρχές του 2022. Το πιο βασικό νέο feature της μηχανής είναι το Nanite[8], ένα εργαλείο που επιτρέπει την εισαγωγή φωτογραμμετρικού υλικού υψηλής λεπτομέρειας και συγχρόνως είναι ικανό να διαχειρίζεται τα επίπεδα λεπτομέρειας (LODs) των εισαγόμενων μοντέλων, επιτρέποντας έτσι στους σχεδιαστές βιντεοπαιχνιδιών να μην σπαταλάνε επιπλέον χρόνο στην κατασκευή λεπτομερών μοντέλων. Άλλο ένα χαρακτηριστικό εργαλείο της Unreal Engine 5 είναι το Lumen, που εξουδετερώνει πια την ανάγκη για προϋπολογισμό του φωτισμού και επιτρέπει τον δυναμικό υπολογισμό του φωτισμού και τον σκιών σε πραγματικό χρόνο, καθώς και το Chaos σε ρόλο physics engine. Με τη δυνατότητα απεικόνισης δεκάδων δισεκατομμυρίων πολυγώνων ταυτόχρονα σε 4k ανάλυση, η νέα γενιά της Unreal Engine στοχεύει στις επερχόμενες υψηλής ταχύτητας αποθηκευτικές μεθόδους των κονσολών νέας γενιάς και στο υλικό τους που βασίζεται στο συνδυασμό χρήσης SSD δίσκων και μνήμης τυχαίας προσπέλασης (RAM). Άλλο ένα σημαντικό εργαλείο από τα πιο ισχυρά επερχόμενα features της μηχανής, είναι το MetaHuman Creator, εργαλείο που επιτρέπει την ταχεία δημιουργία ανθρώπινων χαρακτήρων για χρήση στο engine.

2.4.4 Unity Engine

Figure 21: Unity Engine Logo

Η Unity Engine είναι μία εκ των πιο διαδεδομένων μηχανών παιχνιδιών και χρησιμοποιείται για την ανάπτυξη (κυρίως) βιντεοπαιχνιδιών, αλλά και άλλων εφαρμογών για PC, κονσόλες, mobile και websites. Ανακοινώθηκε και κυκλοφόρησε πρώτη φορά το 2005 στο Worldwide Developers Conference της Apple Inc. ως μία μηχανή παιχνιδιών ειδικά για το λειτουργικό σύστημα Mac OS, στη συνέχεια όμως άρχισε να υποστηρίζεται και από μία μεγάλη γκάμα από άλλες πλατφόρμες. Η μηχανή διατίθεται και δωρεάν ενώ το UI της εφαρμογής είναι ιδιαίτερα φιλικό προς τον χρήστη και αυτό έχει σαν αποτέλεσμα την ιδιαίτερη αγάπη των indie developers, αλλά και απλού κόσμου που δεν είχε τις απαραίτητες γνώσεις από πριν, ανά τον πλανήτη που επιλέγουν να αναπτύξουν σε αυτή την μηχανή. Ένα επιπλέον ατού της είναι ότι πλέον είναι ανεξάρτητη πλατφόρμας (cross platform), οπότε μπορεί να τρέξει σε διαφορετικά λειτουργικά συστήματα και πλατφόρμες υλικού. Διαθέτει το δικό της Asset Store, διευκολύνοντας έτσι τους developers και δίνοντας τους πρόσβαση σε μία μεγάλη ποικιλία από free ή paid assets, επιταχύνοντας τα στάδια της ανάπτυξης, αλλά ταυτόχρονα και ένα τεράστιο community από χρήστες και γνώστες της μηχανής οι οποίοι έχουν τη διάθεση να βοηθήσουν ακόμα και τους πιο αρχάριους χρήστες να μάθουν τα βασικά και ν' αρχίσουν να κάνουν τα πρώτα τους βήματα στη δημιουργία παιχνιδιών. Ακόμα και μεγάλες εταιρείες ανάπτυξης παιχνιδιών που δεν έχουν αναπτύξει τις δικιές τους μηχανές, την επιλέγουν λόγω της ευελιξίας της, καθώς και της συμβατότητάς της με μια πραγματική μεγάλη γκάμα από διαφορετικές πλατφόρμες, καθώς και του εξαιρετικού support στην επαγγελματική έκδοση της μηχανής. Η γλώσσα

προγραμματισμού που χρησιμοποιείται στην Unity είναι η C#, υπάρχει όμως διαθέσιμο και το δωρεάν εργαλείο οπτικού scripting **Bolt** για τους χρήστες που δεν είναι εξοικειωμένοι με κάποια γλώσσα προγραμματισμού. Η μηχανή μπορεί να χρησιμοποιηθεί για τη δημιουργία δισδιάστατων και τρισδιάστατων παιχνιδιών, καθώς και για διαδραστικές προσωμοιώσεις και άλλες εμπειρίες. Η μηχανή έχει επίσης χρησιμοποιηθεί και από άλλες βιομηχανίες πέραν αυτής των βιντεοπαιχνιδιών, όπως ο κινηματογράφος, η αυτοκινητοβιομηχανία, η αρχιτεκτονική, η μηχανική, οι κατασκευές και από τις ένοπλες δυνάμεις των Ηνωμένων Πολιτειών της Αμερικής. Καθώς ήταν η μηχανή παιχνιδιών που χρησιμοποιήσαμε, θα προχωρήσουμε με περισσότερες λεπτομέρειες στη συνέχεια.

Κεφάλαιο 3: Άλλες εφαρμογές που χρησιμοποιήθηκαν

3.1 Πρόλογος

Για να ολοκληρωθεί ένα παιχνίδι συνήθως δεν είναι αρκετή μόνο και μόνο η ύπαρξη της μηχανής παιχνιδιών (η οποία δεν μπορεί να μας παρέχει τα πάντα ανά πάσα στιγμή), αλλά απαιτείται η χρήση και άλλων εργαλείων – εφαρμογών για να καλυφθούν οι επιπλέον ανάγκες κατά το στάδιο της ανάπτυξης. Τέτοια εργαλεία μπορούν να είναι εφαρμογές για 3d modeling (πχ. 3ds max, blender, maya), εφαρμογές για επεξεργασία εικόνας (π.χ. Adobe Photoshop, GIMP), εφαρμογές για επεξεργασία ήχου (πχ Audacity, Ableton Live), εφαρμογές για δημιουργία διαφορετικών texture maps και materials (Materialize, AwesomeBump, CrazyBump).

3.2 Εφαρμογές για 3d Modeling

Ένα από τα βασικότερα στοιχεία ενός τρισδιάστατου βιντεοπαιχνιδιού είναι τα γραφικά του και τα τρισδιάστατα μοντέλα του. Στην επιστήμη των υπολογιστών, με τον όρο "γραφικά" εννοούμε "την εικαστική παρουσίαση δεδομένων (data) οπτικού περιεχομένου", ενώ με τον όρο "3D μοντέλο" εννοούμε τη μαθηματική αναπαράσταση ενός πραγματικού ή φανταστικού τρισδιάστατου αντικειμένου. Τα 3D μοντέλα αποτελούν ανεκτίμητο συστατικό και άλλοτε αναπόσπαστο κομμάτι πολλών διαφορετικών τομέων, όπως η ιατρική, ο βιομηχανικός σχεδιασμός, η αρχιτεκτονική, ο κινηματογράφος, τα βιντεοπαιχνίδια κ.α. Ειδικά τα τελευταία χρόνια με την εμφάνιση και αυξημένη χρήση των εκτυπωτών 3D, τα 3D μοντέλα έγιναν ακόμα πιο σημαντικά καθώς είναι απαραίτητα για την εκτύπωση (πραγματικών) αντικειμένων. Για να τα κατασκευάσουμε στον υπολογιστή μας, χρειαζόμαστε ειδικά εργαλεία τρισδιάστατου σχεδιασμού. Τα πιο γνωστά είναι τα 3ds max, blender και maya. Εμείς χρησιμοποιήσαμε τις εκδόσεις 2.65 και 2.9 του Blender.

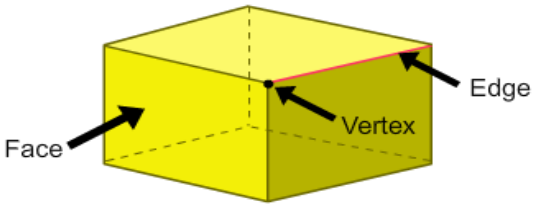


Figure 22: Face, Vertex and Edge of a 3d model

3.2.1 Η εφαρμογή Blender

Figure 23: Blender logo

Το Blender είναι ένα πανίσχυρο δωρεάν πρόγραμμα σχεδίασης τρισδιάστατων γραφικών, είναι ελεύθερο λογισμικό και διανέμεται υπό την άδεια GNU General Public License. Χρησιμοποιείται για modeling, sculpting, texturing, rigging, προσωμοιώσεις νερού, υφάσματος και φυσικής, animation, rendering, αλλά και για τη δημιουργία αλληλεπιδραστικών τρισδιάστατων εφαρμογών, όπως τα βιντεοπαιχνίδια [27], σαν game engine δηλαδή. Οι δυνατότητές του πλησιάζουν εκείνες κορυφαίων προγραμμάτων της αγοράς που κοστίζουν πολλές χιλιάδες ευρώ, ενώ διαθέτει πολλά δωρεάν εργαλεία και extensions.

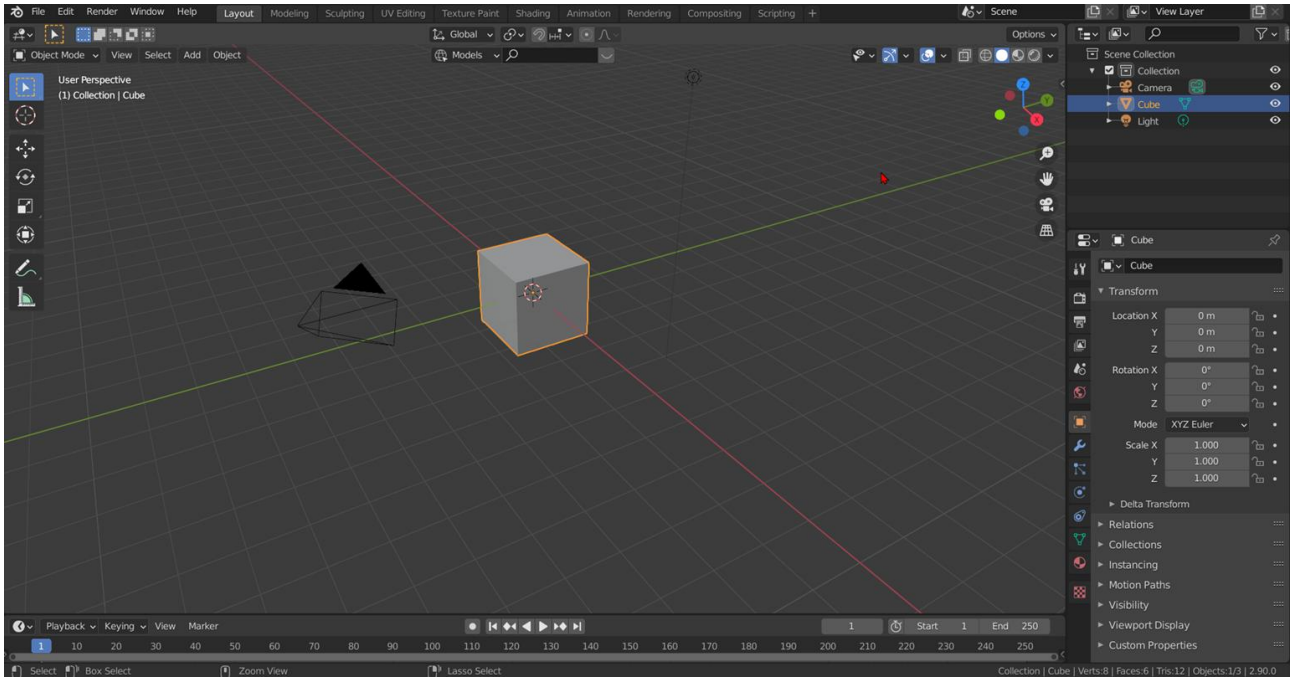


Figure 24: Blender UI

3.2.2 Ποια τμήματα του βιντεοπαιχνιδιού δημιουργήθηκαν στο Blender

Χρησιμοποιήσαμε το blender για τον σχεδιασμό όλων των διαφορετικών χώρων [14] του περιβάλλοντος του παιχνιδιού και των καμπυλωτών τούνελ, για γλυπτική σε τρισδιάστατα μοντέλα και δημιουργία των μοντέλων λάσπης, για τη δημιουργία άλλου τύπου μοντέλων λάσπης με την χρήση liquid simulation και animation, για τη δημιουργία συμπληρωματικών μοντέλων, όπως οι σωλήνες και τα στηρίγματά τους, για την δημιουργία υφασμάτων και σκισμένων υφασμάτων (με χρήση εργαλείων προσωμοίωσης υφάσματος και άλλα εξειδικευμένα extensions), για τα κάγκελα και τα συρμάτινα πλέγματα που βρίσκονται διάσπαρτα στο βιντεοπαιχνίδι, για πόρτες και τσιμεντένια barriers, επαναλαμβανόμενες κολώνες με χρήση array modifiers, κομμάτια χαρτιού με τσάκιση, για

το sculpting του fatberg (ενός είδους «παγόβουνου», μίας στερεής μάζας ουσιαστικά, που δημιουργείται από τα λίπη νοικοκυριών ή επιχειρήσεων, τα οποία καταλήγουν στους υπονόμους), τα μεταλλικά πατώματα, τις σκάλες, τρισδιάστατα μοντέλα σκουπιδιών, vertex painting [17], texture painting, δημιουργία materials με την χρήση του node editor, εξαγωγή μοντέλων (κυρίως σε αρχεία με κατάληξη fbx και obj), εξαγωγή textures, επεξεργασία ήδη έτοιμων δωρεάν μοντέλων και μείωση του αριθμού πολυγώνων τους με χρήση του decimate modifier κ.α. Μία άλλη σημαντική διαδικασία που επιτεύχθηκε με την χρήση του Blender ήταν και το UV Unwrapping μία απαραίτητη (συνήθως μη-αυτοματοποιήμενη, αν μιλάμε για πολύπλοκα μοντέλα που ξεφεύγουν από τα βασικά τρισδιάστατα γεωμετρικά σχήματα - primitives) διαδικασία που καθιστά εφικτή την κάλυψη ενός τρισδιάστατου μοντέλου με μία δισδιάστατη εικόνα για σκοπούς ουσιαστικά χρωματισμού του μοντέλου, μέσω της αντιστοίχησης συγκεκριμένων pixel της δισδιάστατης φωτογραφίας με συγκεκριμένες κορυφές (κορυφή - vertex) του τρισδιάστατου μοντέλου. Το συγκεκριμένο πρόγραμμα [17] ήταν απολύτως απαραίτητο για τη δημιουργία των γραφικών του παιχνιδιού και χωρίς αυτό, το παιχνίδι δεν θα μπορούσε να έχει την μορφή που έχει σήμερα, εφόσον το υβριδικό εργαλείο (Probuilder) της Unity Engine για δημιουργία 3d μοντέλων και level design δεν παρέχει το τεράστιο σύμπαν εργαλείων και δυνατοτήτων του Blender. Ακολουθούν στιγμιότυπα από τη διαδικασία.

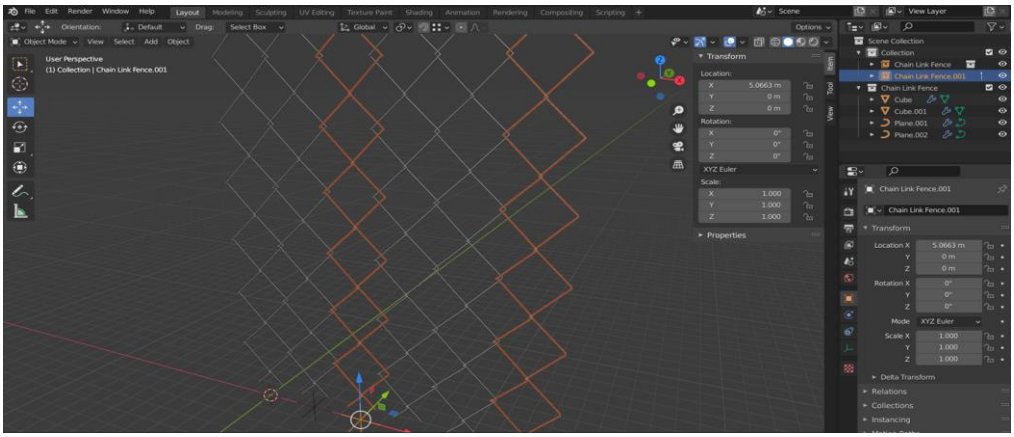


Figure 25: Chainlink fence modelling using vertex relocation and array modifier

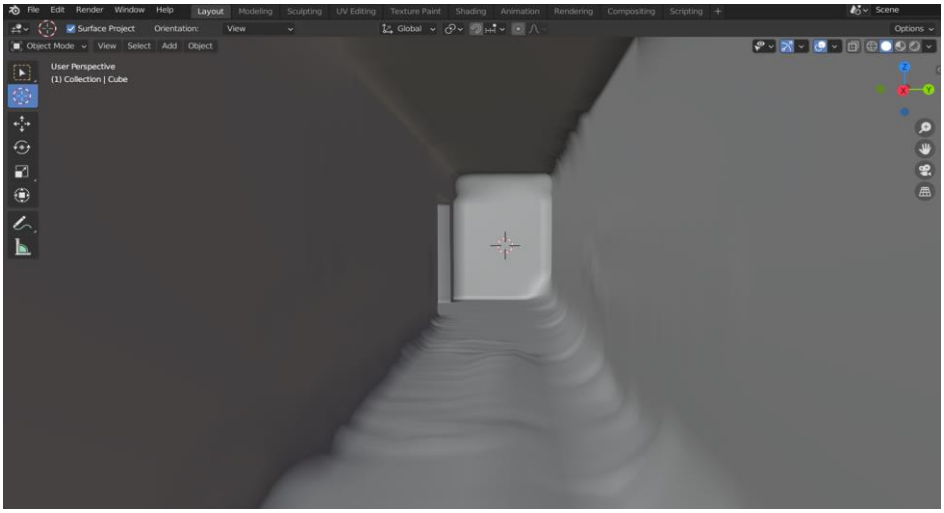


Figure 26: Level 1 extension sculpted prototype

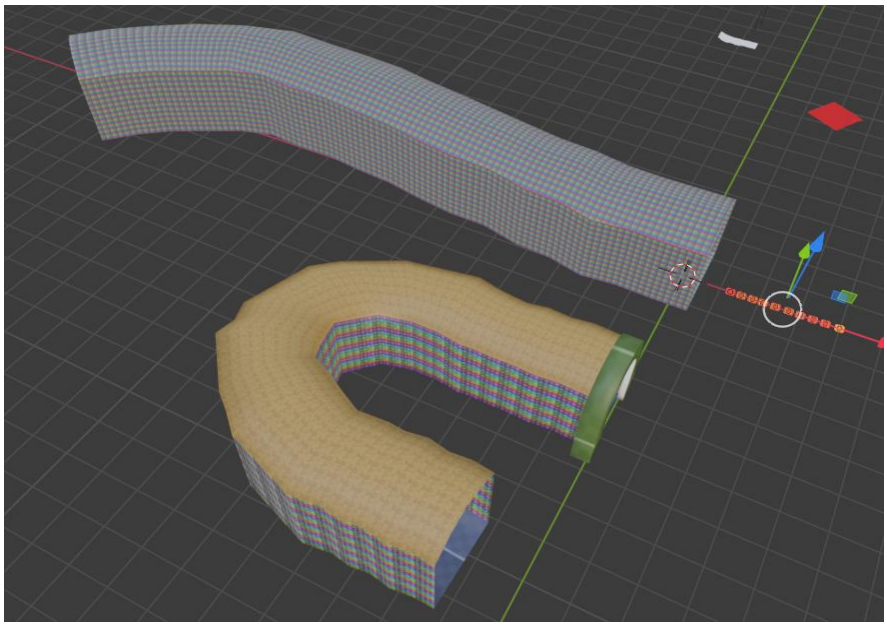


Figure 27: Early tunnel prototypes

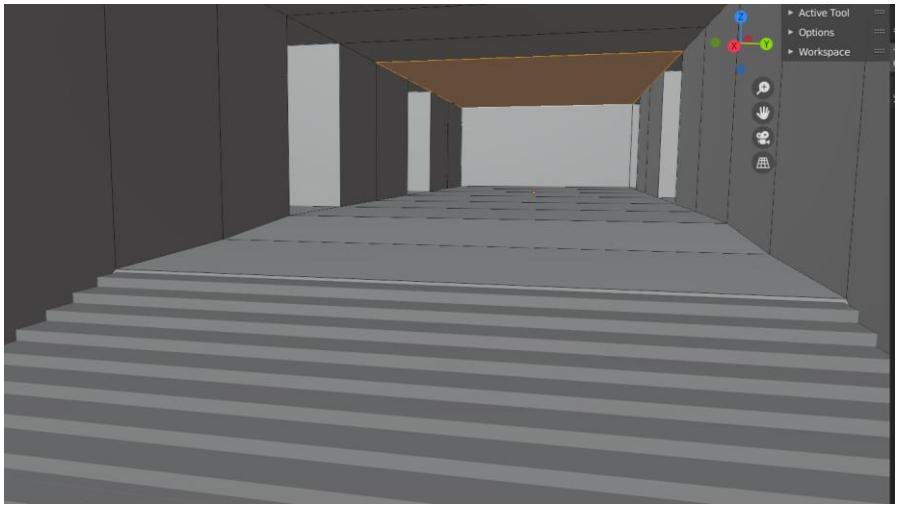


Figure 28: Level 2 crossroad prototyping in Blender

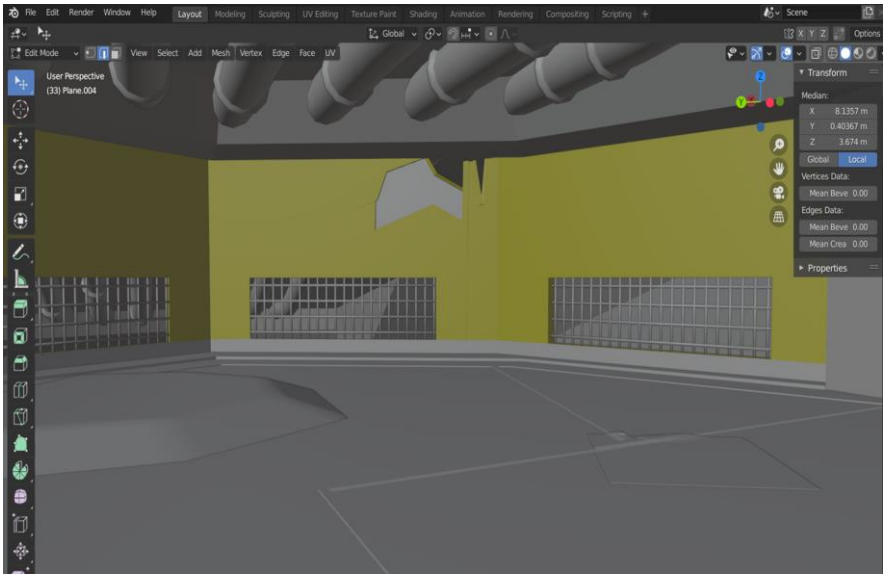


Figure 29: Level 2 "big room" prototyping

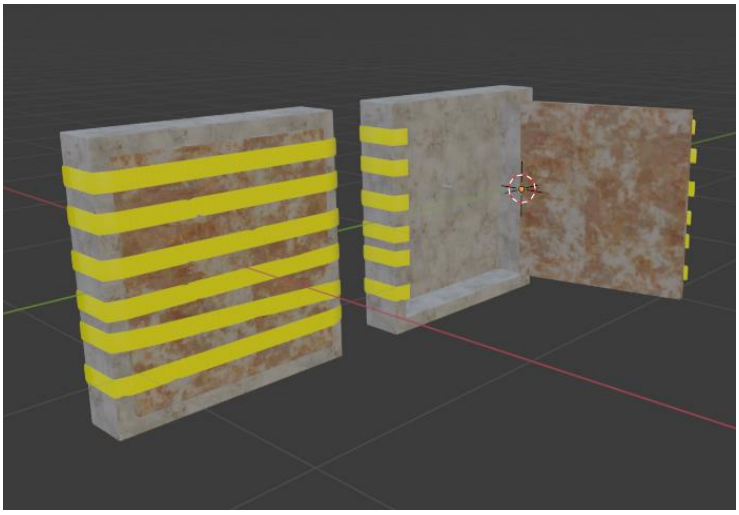


Figure 30: Modelling of two different versions of a wrapped box(closed-open) for a short stop motion animation

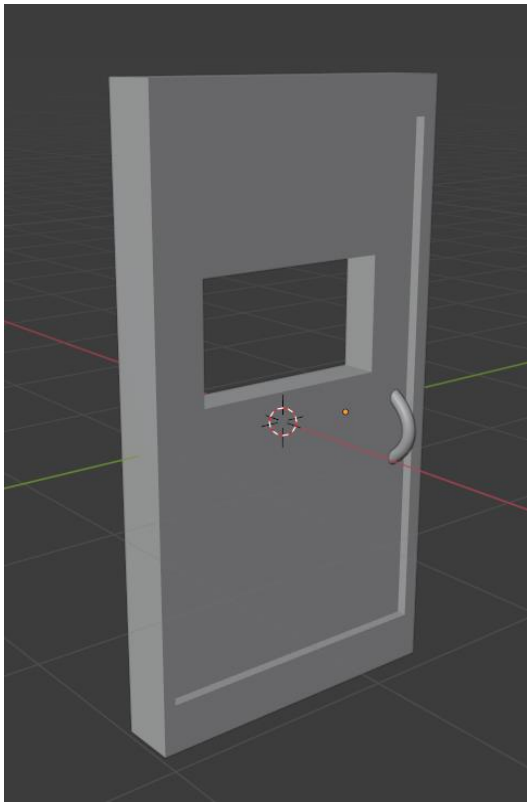


Figure 31: Simple door for Level 4 made in Blender

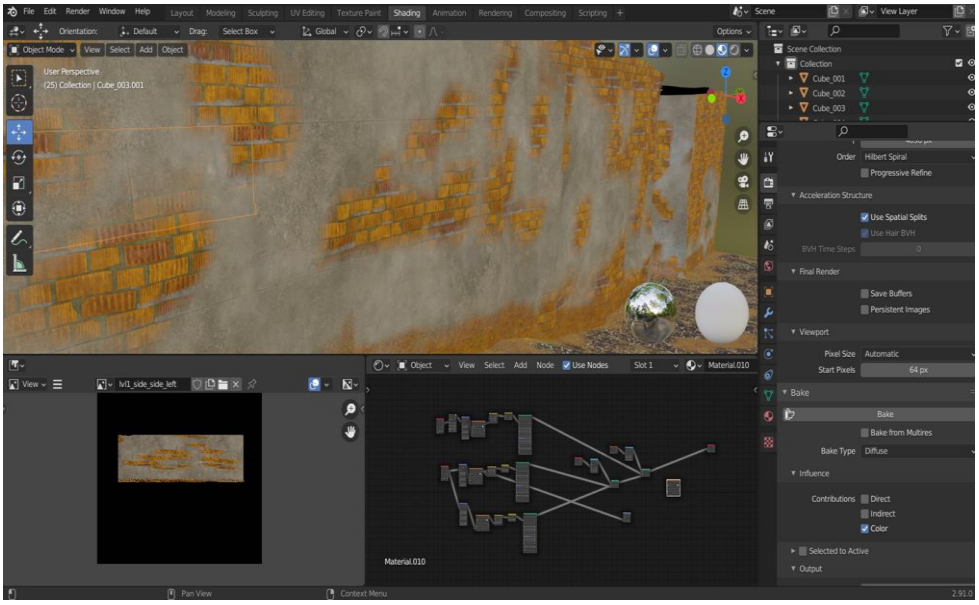


Figure 32: Vertex painting in Level 1 using Blender's shader nodes and baking textures for exporting

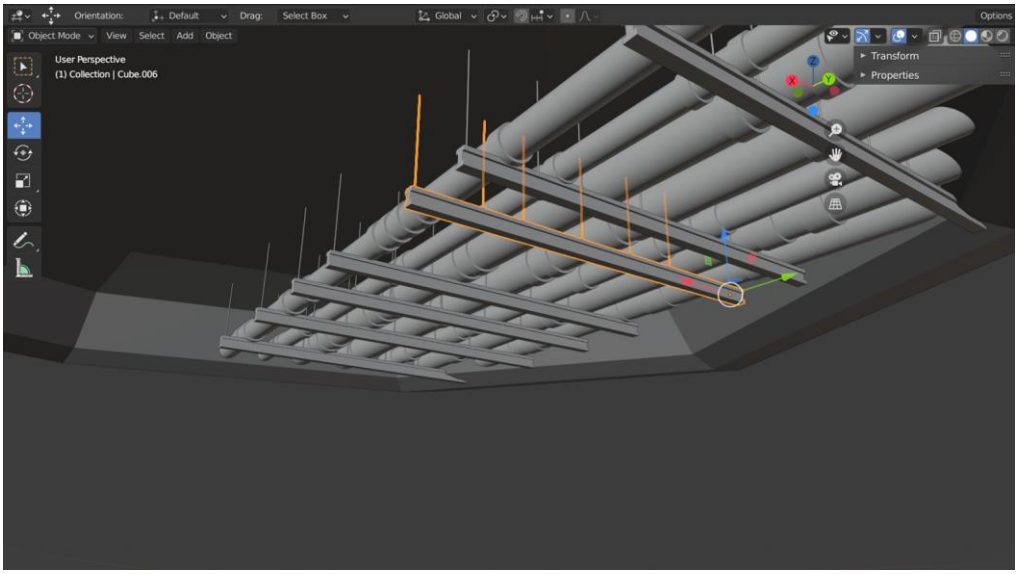


Figure 33: Creation of a pipe ceiling extension

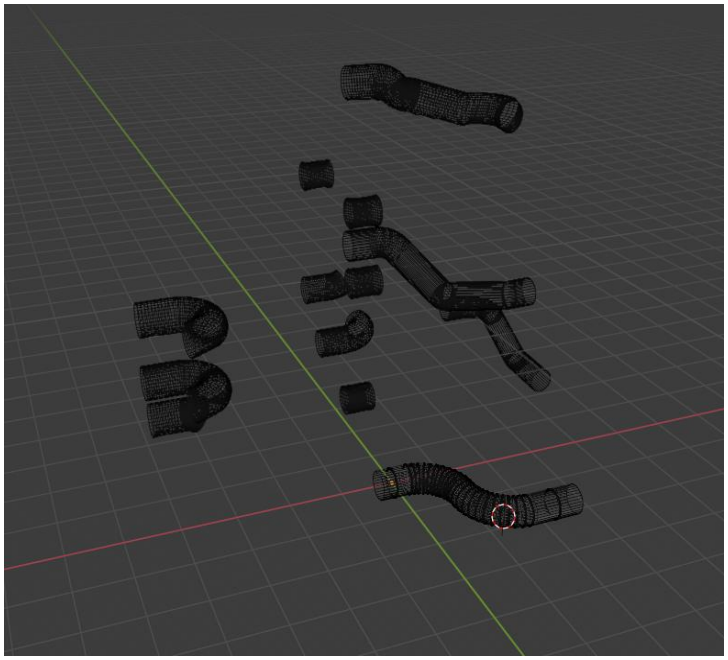


Figure 34: Pipe bending – modular approach

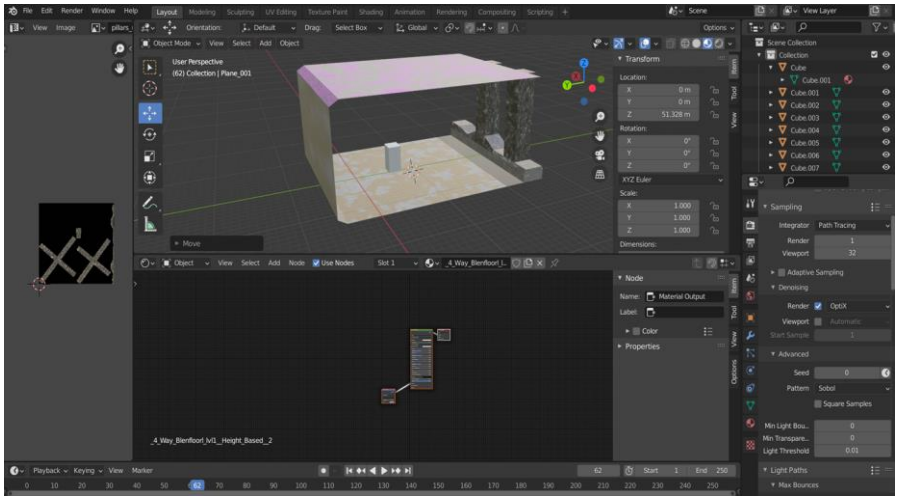


Figure 35: Early Level 3 modular extension experiment

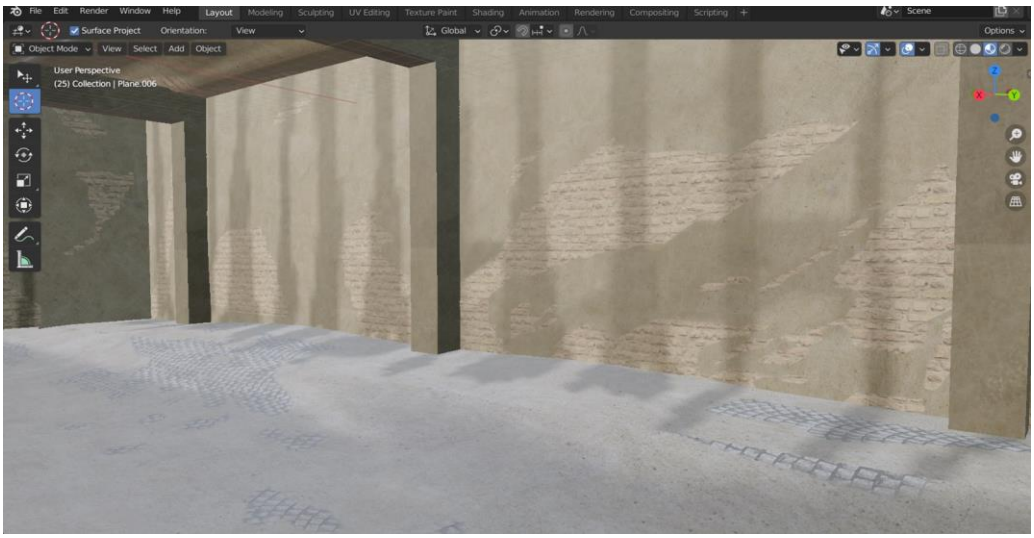


Figure 36: 3-way vertex painting of Level 3's main room for mold effect

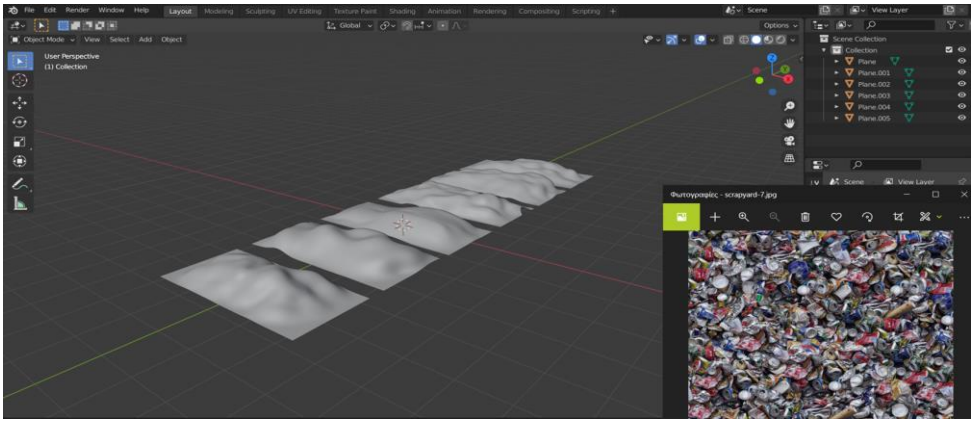


Figure 37: Sculpted planes to be combined with trash textures to create scattered trash throughout the game's environment

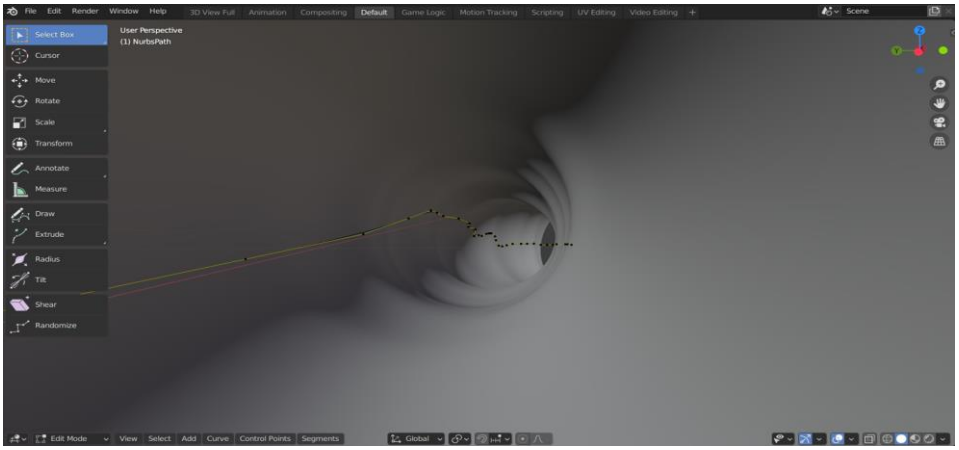


Figure 38: Early tunnel prototype

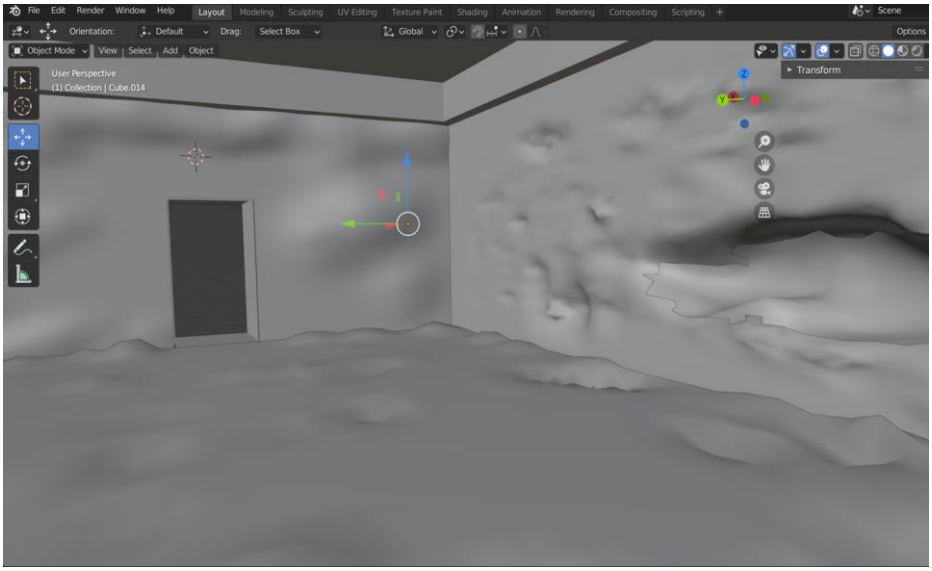


Figure 39: Really early prototype of an old Level 2 room

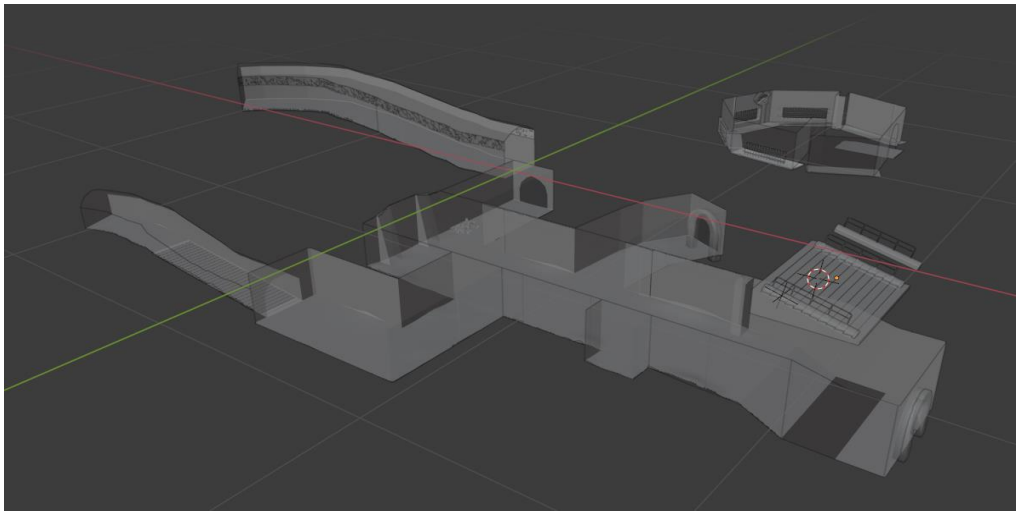


Figure 40: Modelling of the final version of Level 2

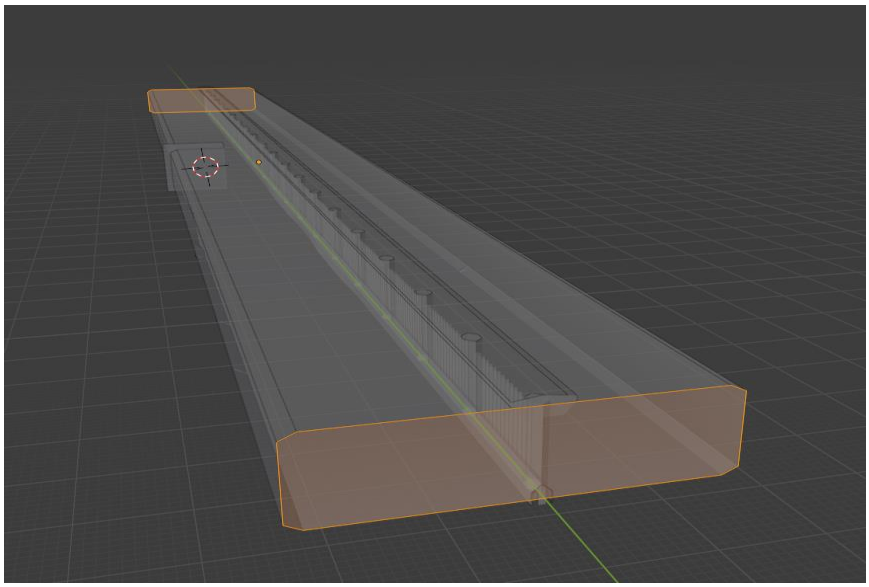


Figure 41: Making of Level 3's long extension

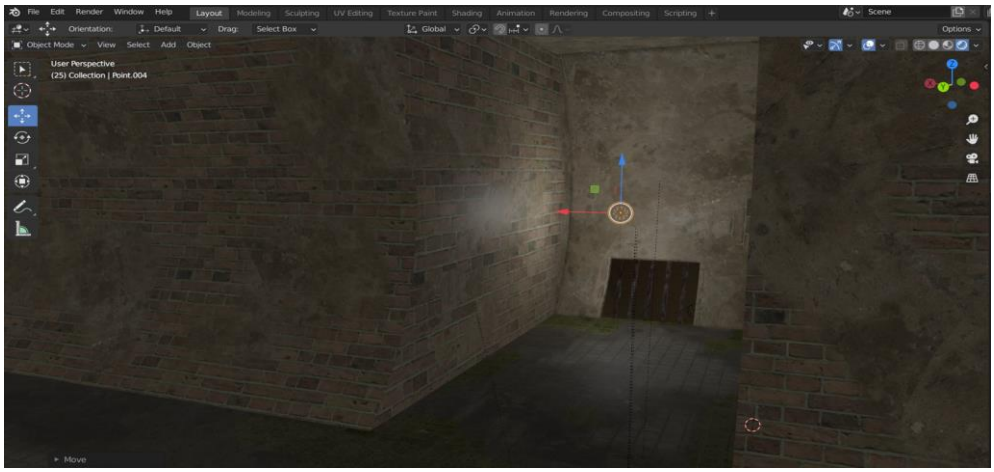


Figure 42: Eevee engine render of a previous Level 4 version

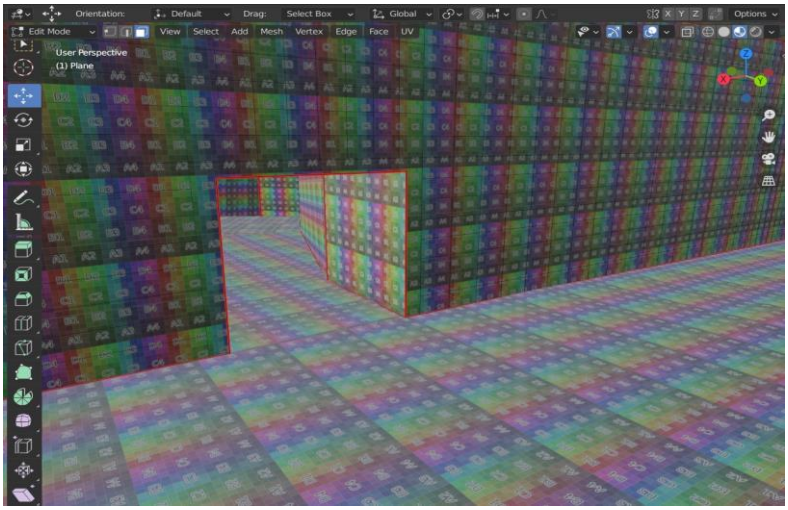


Figure 43: During the UV unwrapping procedure of a Level 4 protype

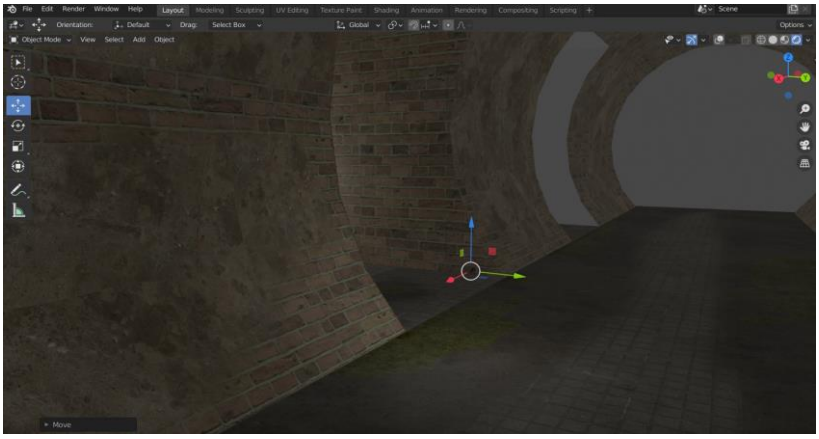


Figure 44: Earlier Level 4 render

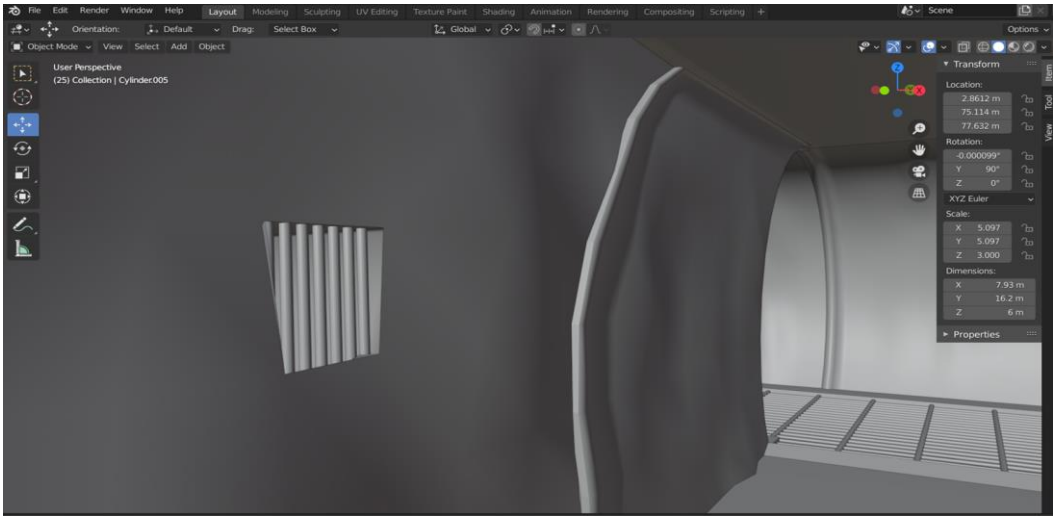


Figure 45: Modelling the final version of Level 4

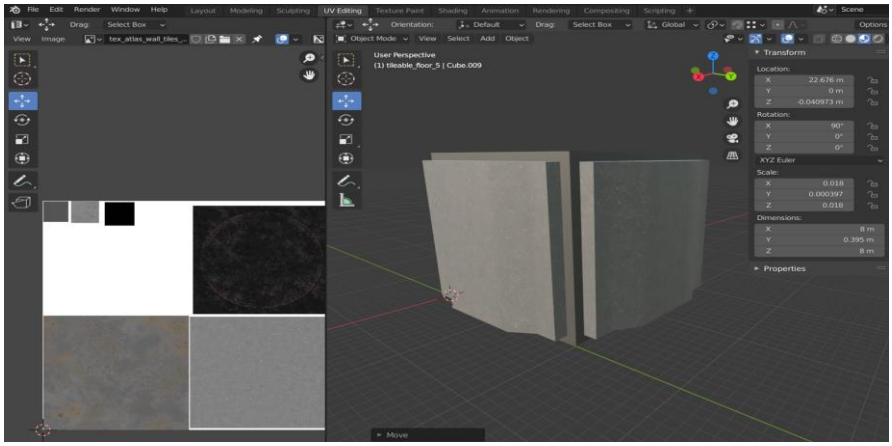


Figure 46: Making of a modular wall with its texture atlas

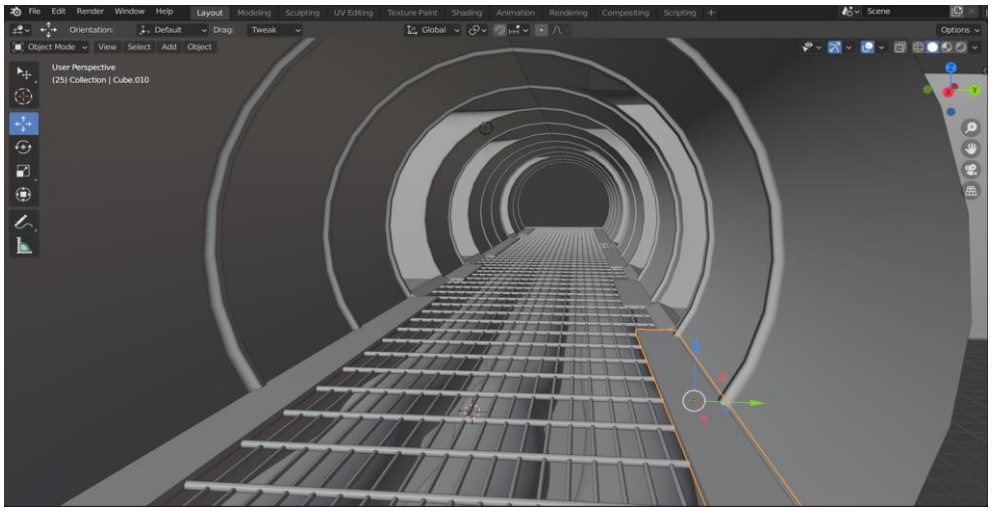


Figure 47: Final version of Level 4 prototyping

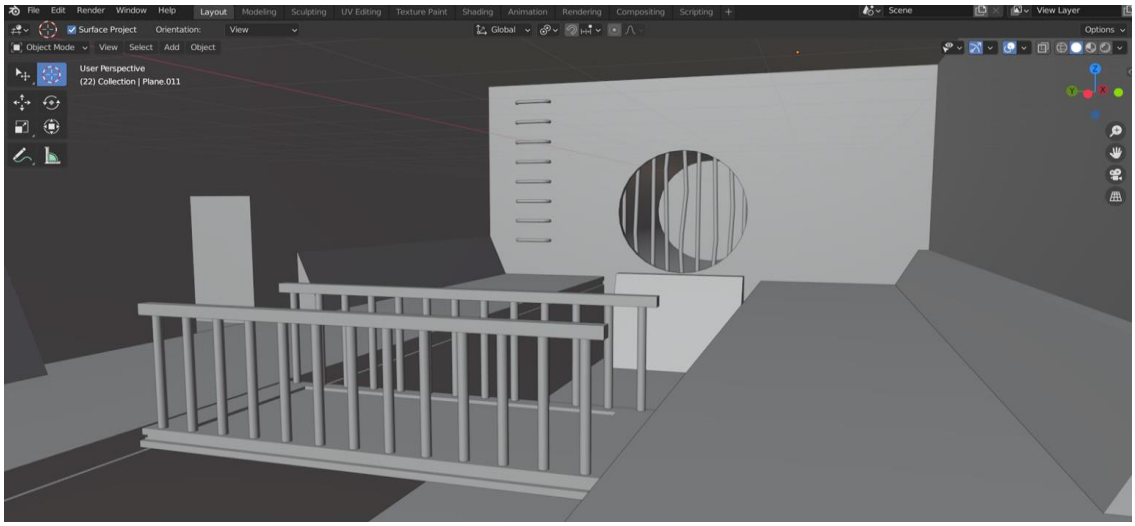


Figure 48: Level 5 prototyping

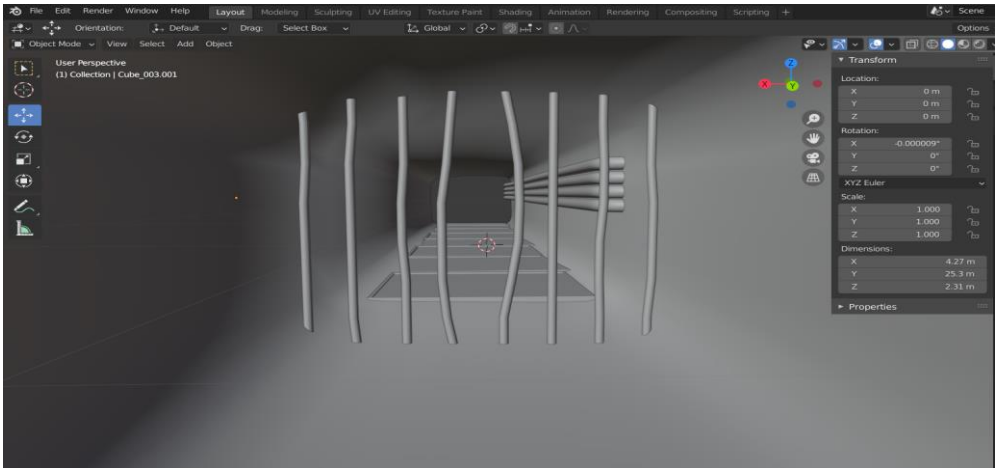


Figure 49: Level 5's entrance

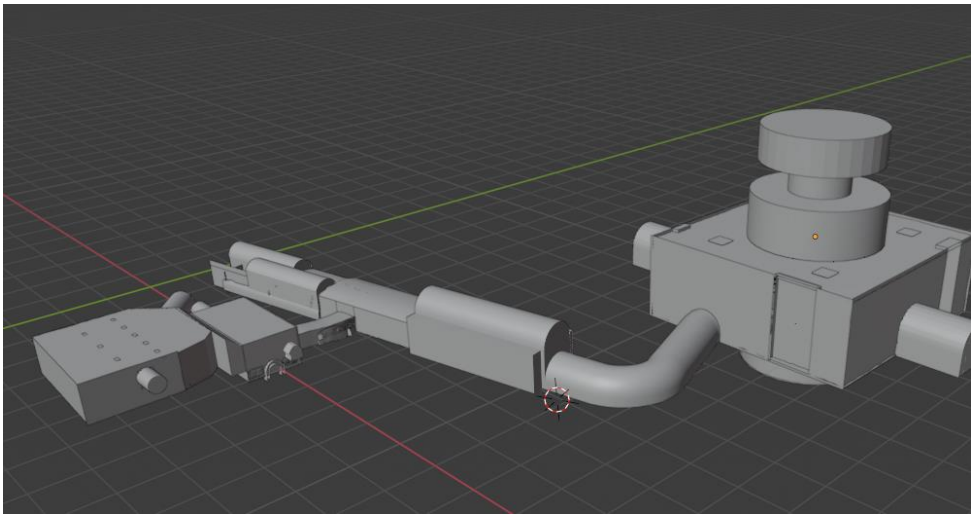


Figure 50: Level 5 final version with all its rooms and tunnels

3.3 Εφαρμογές για επεξεργασία εικόνας

Ένα πρόγραμμα επεξεργασίας εικόνας, όπως το Photoshop ή το GIMP, επιτρέπει στον χρήστη να δημιουργήσει ή να επεξεργαστεί διαδραστικά εικόνες στην οθόνη του υπολογιστή και μετά να τις

αποθηκεύσει σε τύπο αρχείου μορφής εικόνων bitmap, όπως JPEG, PNG, GIF κτλ. Κάποιες από τις πιο σημαντικές δυνατότητες αυτών των προγραμμάτων είναι οι εξής:

- Επιλογή συγκεκριμένης περιοχής της εικόνας για επεξεργασία.
- Ο χρήστης δύναται να σχεδιάσει γραμμές με τεχνητά brushes διαφορετικού χρώματος, μεγέθους, σχήματος και πίεσης.
- Ο χρήστης μπορεί να γεμίσει μία περιοχή φωτογραφίας με ένα χρώμα ή με ένα texture.
- Ο χρήστης μπορεί να επιλέξει ένα χρώμα χρησιμοποιώντας διαφορετικά μοντέλα χρώματος όπως RGB, HSV ή χρησιμοποιώντας το color picker tool για να επιλέξει ένα χρώμα από μία ήδη υπάρχουσα φωτογραφία.
- Ο χρήστης μπορεί να χρησιμοποιήσει ένα crop tool για να "κόψει" τμήμα εικόνας.
- Ο χρήστης μπορεί να προσθέσει κείμενο σε εικόνα με γραμματοσειρά της αρεσκείας του.
- Ο χρήστης μπορεί να αφαιρέσει ατέλειες από φωτογραφίες.
- Ο χρήστης μπορεί να κάνει σύνθετη επεξεργασία εικόνας χρησιμοποιώντας layers.
- Είναι δυνατή η χρήση φίλτρων για αλλαγή της όψης της εικόνας (π.χ blurring – artistic effects).
- Μετατροπή μεταξύ διαφορετικών τύπων αρχείου εικόνας.

Εφαρμογές σαν κι αυτές λοιπόν είναι με την σειρά τους απαραίτητα εργαλεία για τον σχεδιαστή γραφικών κατά τη διάρκεια της ανάπτυξης ενός βιντεοπαιχνιδιού. Εμείς χρησιμοποιήσαμε την έκδοση 2.10.22 του GIMP.

3.3.1 Η εφαρμογή GIMP

Figure 51: GIMP logo

Το GIMP (GNU Image Manipulation Program) είναι ένα δωρεάν και ελεύθερο λογισμικό επεξεργασίας γραφικών τύπου raster [25]. Ένα εργαλείο που επικεντρώνεται στην επεξεργασία εικόνας και είναι διαθέσιμο σε προσαρμοσμένες εκδόσεις για τα λειτουργικά συστήματα των Microsoft Windows, MAC OS X και GNU/Linux. Το GIMP μπορεί να φέρει εις πέρας βασικές εργασίες επεξεργασίας εικόνας, όπως η αλλαγή μεγέθους και η περιστροφή, η επεξεργασία εικόνας και το "κόψιμο" αυτής (cropping), η πιο σύνθετη επεξεργασία με layers, η χρήση φίλτρων και το φωτομοντάζ, είναι επιπλέον όμως διαθέσιμα και plug-ins (G'MIC-Qt) που επεκτείνουν ακόμα περισσότερο τις δυνατότητες του προγράμματος.

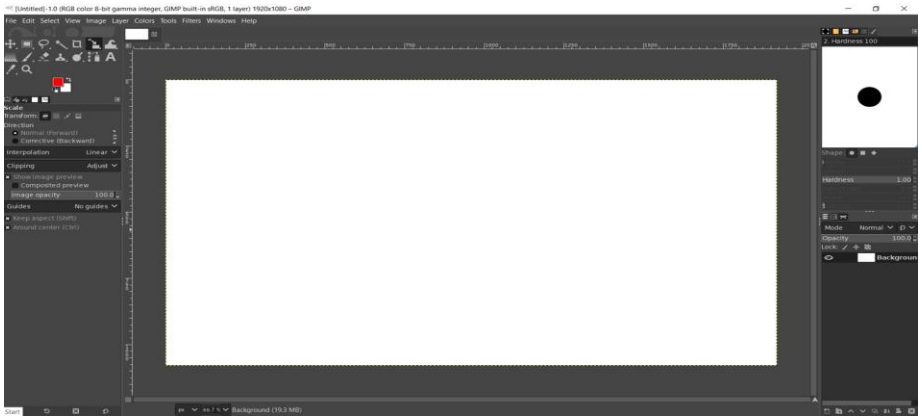


Figure 52: GIMP UI

3.3.2 Πώς χρησιμοποιήσαμε το GIMP για το βιντεοπαιχνίδι

Χρησιμοποιήσαμε το GIMP κυρίως για cropping (κόψιμο) φωτογραφιών από τον πραγματικό κόσμο και μεταφοράς τους στο βιντεοπαιχνίδι. Graffiti, tags, ταμπέλες, οπτικές ενδείξεις και πολλά απορρίματα για την διακόσμηση και εμπλουτισμό του περιβάλλοντος υπονόμου του παιχνιδιού, αποτελούν προϊόν επεξεργασίας αληθινών φωτογραφιών που τραβήξαμε στους δρόμους ή στο σπίτι και με cropping (με χρήση του lasso tool ή magic wand) στο GIMP και επικόλληση πάνω σε transparency, μπορούσαν να εισαχθούν στην Unity Engine και να τα προσθέσουμε σε τοίχους και πατώματα με χρήση του Cutout rendering mode του Standard Shader της Unity πάνω σε ένα απλό μοντέλο plane ή quad. Το GIMP χρησιμοποιήθηκε επίσης για την επεξεργασία έτοιμων textures (περιστροφή, αλλαγή στην ανάλυση ή αλλαγή των χρωμάτων) καθώς και για τα φίλτρα του, ακόμα και για ζωγραφική με τα brushes σε συγκεκριμένες περιπτώσεις που θέλαμε να έχουμε ένα πιο "βρώμικο" οπτικό αποτέλεσμα. Επιπλέον για τη δημιουργία emission texture maps για συγκεκριμένα αντικείμενα. Ακολουθούν κάποια στιγμιότυπα:

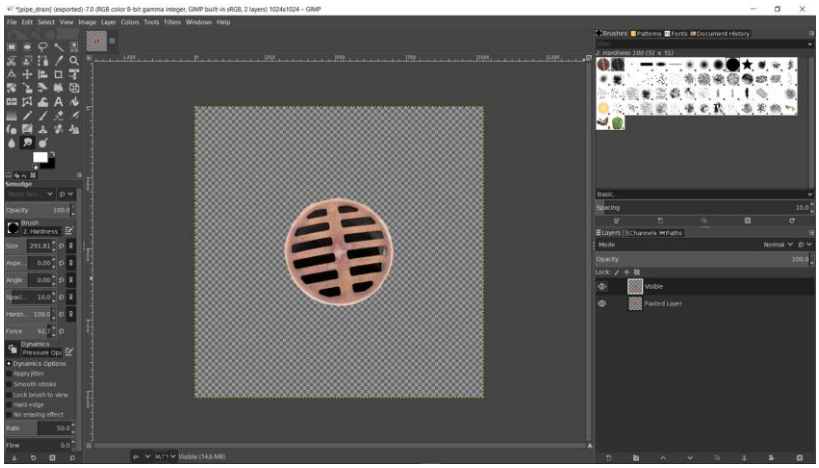


Figure 53: Creation of a drain pipe texture

Figure 54: Trash decals made in Gimp from top view photos of real objects

3.4 Εφαρμογές για επεξεργασία ήχου

Ένα πρόγραμμα για επεξεργασία ήχου επιτρέπει την επεξεργασία, αλλά και την παραγωγή ηχητικών δεδομένων. Οι περισσότερες από αυτές χρησιμοποιούνται για την επεξεργασία μουσικών αρχείων, ενώ επιτρέπουν την εφαρμογή εφέ και φίλτρων, ηχογράφηση και αναπαραγωγή μουσικής, εισαγωγή κι εξαγωγή ηχητικών αρχείων, δυνατότητα αλλαγής ρυθμού-τόνου (pitch-tempo), crop ηχητικών αρχείων, μίξη πολλών διαφορετικών καναλιών ήχου, αφαίρεση θορύβου, ρύθμιση καναλιών stereo κτλ. Εμείς χρησιμοποιήσαμε για το βιντεοπαιχνίδι την έκδοση 2.4.2 του Audacity.

3.4.1 Η εφαρμογή Audacity

Figure 55: Audacity logo

Το Audacity είναι ένα πρόγραμμα ψηφιακής επεξεργασίας ήχου και ηχογράφησης, που κυκλοφορεί ως ελεύθερο λογισμικό και είναι ανεξάρτητο πλατφόρμας, διαθέσιμο για Windows, MAC OS X, Linux και BSD. Δημιουργήθηκε από τον Dominic Mazzoni όταν ήταν φοιτητής στο πανεπιστήμιο Carnegie Mellon [26] , ο οποίος παρόλο που εργάζεται πλέον στην Google αποτελεί τον κύριο συντηρητή του προγράμματος με τη βοήθεια και στήριξη πολλών ανθρώπων ανά τον κόσμο.

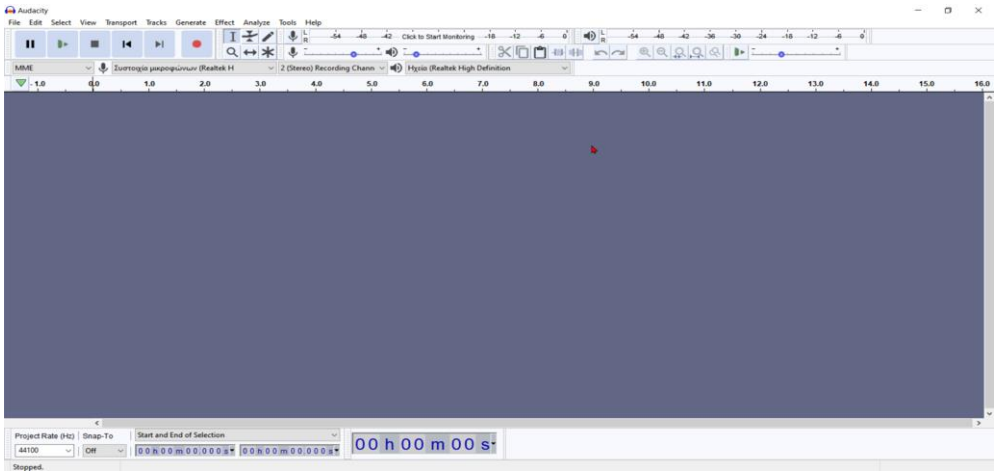


Figure 56: Audacity UI

3.4.2 Γιατί χρησιμοποιήσαμε το Audacity για το βιντεοπαιχνίδι

Για τον ήχο του βιντεοπαιχνιδιού ήταν απαραίτητη η επεργασία αρχείων ήχου. Επιλέξαμε την πλειοψηφία των περιβαλλοντικών ήχων, βημάτων κτλ. από την online βιβλιοθήκη δωρεάν ήχων freesound.org. Χρειάστηκε όμως να κάνουμε crop σε ενιαία ηχητικά αρχεία επαναλαμβανόμενων βηματισμών και σταγόνων για να δημιουργήσουμε μικρά αρχεία ξεχωριστών βημάτων και σταγόνων για μεγαλύτερο ρεαλισμό (σε αντίθεση με την περίπτωση που θα χρησιμοποιούσαμε το ίδιο ηχητικό αρχείο βήματος για όλα τα βήματα), χρήση των macros του Audacity για ταυτόχρονη επεξεργασία πολλών αρχείων μαζί, για να επιτύχουμε ομοιογένεια κι εξομάλυνση στα επίπεδα volume διαφορετικών αρχείων βημάτων και για την χρήση εφέ fade-in και fade-out στα ξεχωριστά αρχεία της μουσικής του βιντεοπαιχνιδιού, έτσι ώστε να μην χρειάζεται περαιτέρω scripting για κάτι αντίστοιχο στη μηχανή της Unity.

3.5 Εφαρμογές για τη δημιουργία διαφορετικών texture maps

Για να προσδώσουμε περισσότερη λεπτομέρεια σε ένα τρισδιάστατο μοντέλο χωρίς να επέμβουμε στην γεωμετρία του προσθέτοντας περισσότερα πολύγωνα με χρήση κάποιου προγράμματος 3d modeling, μπορούμε απλά να χρησιμοποιήσουμε διαφορετικά αρχεία εικόνας τα οποία αποκαλούνται texture maps. Κάθε ένα από αυτά έχει την δική του ξεχωριστή λειτουργία και σκοπό. Ας αναφέρουμε τα πιο βασικά από αυτά:

Albedo map

Με την χρήση albedo map ουσιαστικά επικαλύπτουμε το κενό από χρώματα μοντέλο, επιτυγχάνοντας απεικόνιση χρωμάτων πάνω σε αυτό.

Figure 57: Model without (up-left) and with (down-left) albedo texture (right).

Normal (Bump) map

Με την χρήση normal map στο material του τρισδιάτατου μοντέλου μπορούμε να προσδώσουμε λεπτομέρεια (όπως εξογκώματα, αυλάκια και γρατζουνιές) στην επιφάνεια του μοντέλου, χωρίς να επέμβουμε στο σχήμα του ή την γεωμετρία του μοντέλου αυτού καθαυτού, ενώ οι συγκεκριμένες λεπτομέρειες αλληλεπιδρούν με την κατεύθυνση του φωτός με αποτέλεσμα την διαφορετική, πιο ρεαλιστική σκίαση και την ψευδαίσθηση λιγότερο επίπεδης γεωμετρίας.

Figure 58: With and without normal mapping

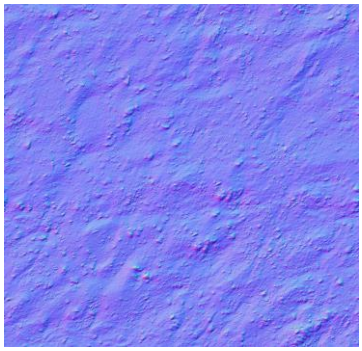


Figure 59: Sand normal map

Metallic map

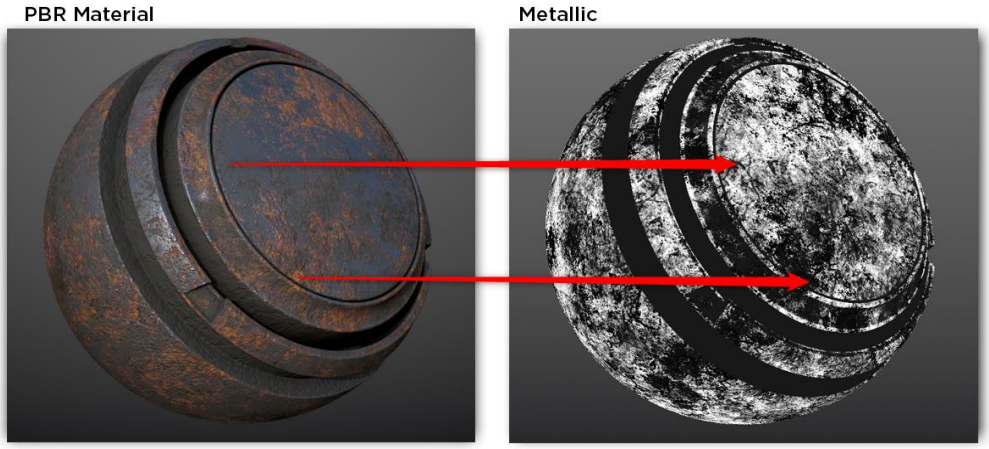


Figure 60: PBR - metallic

Το metallic texture αποτελεί ουσιαστικά ένα grayscale αρχείο εικόνας, το οποίο καθορίζει ποιες περιοχές του μοντέλου είναι μεταλλικές και ποιες όχι όσον αφορά και πάλι στην αλληλεπίδραση του αντικειμένου με το φως. Οι λευκές περιοχές του metallic map ορίζουν τις μεταλλικές και γυαλιστερές περιοχές του μοντέλου, ενώ οι μαύρες περιοχές του metallic map τις καθόλου μεταλλικές περιοχές του μοντέλου, όσον αφορά στην οπτική του απόδοση κατά τη διάρκεια του rendering.

Ambient Occlusion map

Το ambient occlusion (ao) texture είναι ένα ασπρόμαυρο texture με την χρήση του οποίου μπορούμε να προσδώσουμε περισσότερη λεπτομέρεια σκίασης στο τρισδιάστατο αντικείμενο [22].

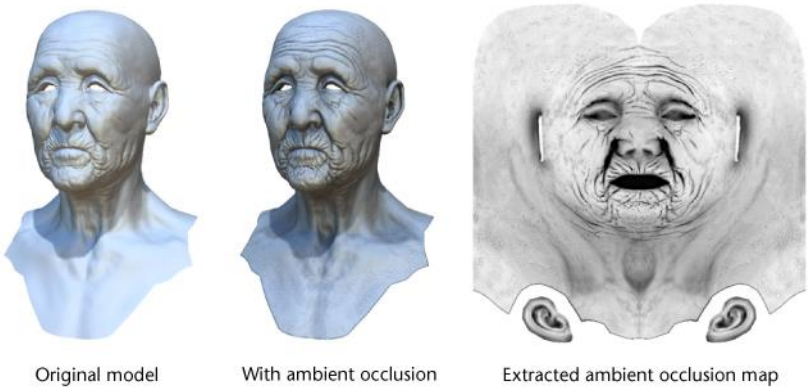


Figure 61: Original model – model with ambient occlusion – ambient occlusion map

Height/Displacement map:

Το height map χρησιμοποιείται συχνά ως συμπλήρωμα του normal map και περιέχει δεδομένα για πραγματική μετατόπιση (ανύψωση) της γεωμετρίας (παραδείγματος χάριν, αν επιθυμούμε να δημιουργήσουμε ένα terrain με αυξομειώσεις στο ύψος χρησιμοποιώντας ένα απλό επίπεδο 3d plane, μπορούμε απλά με την χρήση ενός tessellation shader με height map, να πετύχουμε αυξομειώσεις στα ύψη της γεωμετρίας). Όπως ήδη προαναφέραμε, το συγκεκριμένο map αποτελεί μία grayscale εικόνα της οποίας οι εντελώς λευκές περιοχές αναπαριστούν την μέγιστη ανύψωση της γεωμετρίας ενώ οι εντελώς μαύρες την ελάχιστη ανύψωση.

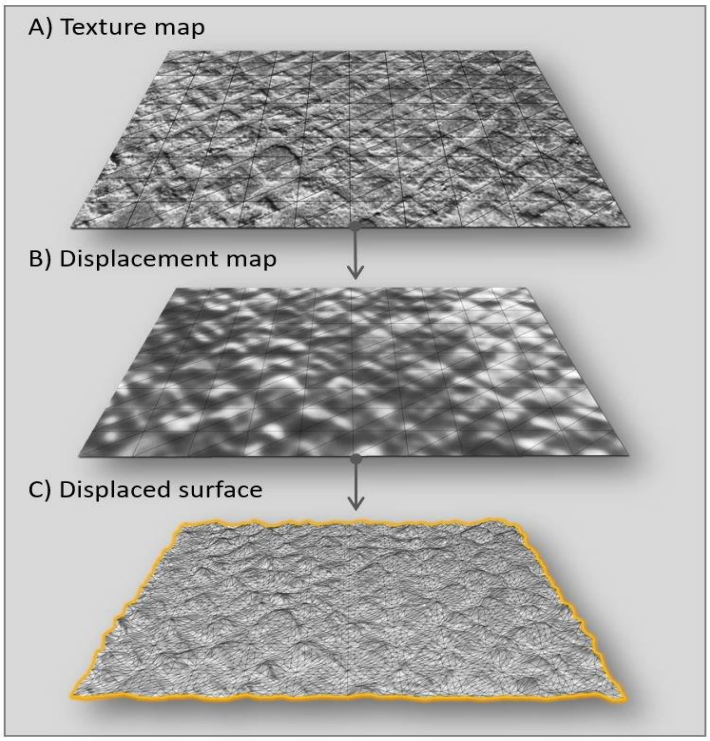


Figure 62: Height/displacement map effect on the geometry of mesh

Για την παραγωγή αυτών των διαφορετικών τύπων textures με στόχο τη δημιουργία εντύπωσης διαφορετικών υλικών των αντικειμένων για τον παίκτη και την επένδυση των τρισδιάστατων μοντέλων, απαιτείται η χρήση ειδικών προγραμμάτων. Κάποια από αυτά είναι το CrazyBump, το

AwesomeBump και το Materialize. Εμείς χρησιμοποιήσαμε για το βιντεοπαιχνίδι την έκδοση 1.78 του Materialize και την έκδοση 5.0 του Awesome Bump.

3.6 Λοιπές χρήσιμες έννοιες στα τρισδιάστατα γραφικά

Vertex

Αποτελεί ένα μοναδικό σημείο στο τρισδιάστατο επίπεδο (πληθυντικός: vertices).

Edge

Μία ευθεία γραμμή που συνδέει δύο ξεχωριστά vertices.

Face

Μία επίπεδη επιφάνεια περιφραγμένη από edges (αλλιώς polygon-triangle).

Mesh

Μία συλλογή από vertices, edges και faces που χαρακτηρίζει και καθορίζει το σχήμα ενός τρισδιάστατου μοντέλου.

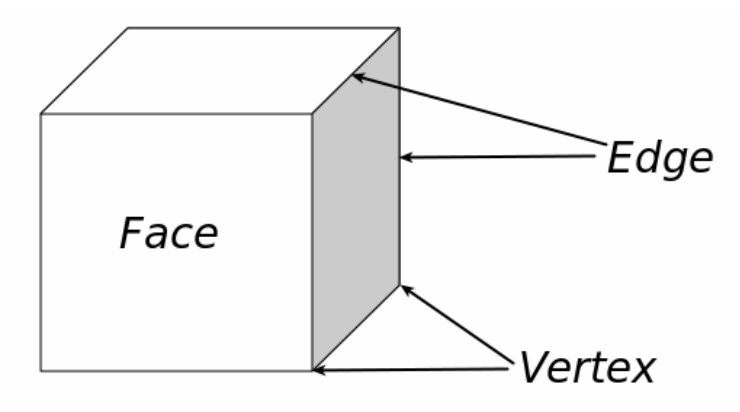


Figure 63: Ένα παράδειγμα Mesh και τα συστατικά του στοιχεία.

Rendering

Ονομάζεται η διαδικασία της ρεαλιστικής οπτικής απόδοσης των μοντέλων και περιβαλλόντων με την χρήση χρωμάτων, υφών, φωτισμού και σκιάσεων. Ο απαιτούμενος χρόνος για την απόδοση αυτή είναι ανάλογος της πολυπλοκότητάς των παραπάνω. Το πρόγραμμα που χρησιμοποιείται για την παραπάνω διαδικασία ονομάζεται renderer.

Shader

Ο Shader (σκιαστής) είναι ένα κομμάτι κώδικα γραμμένο από τον προγραμματιστή γραφικών – technical artist σε γλώσσα προγραμματισμού (CG-HLSL στην περίπτωση της Unity Engine) που μπορεί να καταλάβει ο επεξεργαστής γραφικών (GPU) και έτοιμο να τρέξει σε κάποιο στάδιο της

διοχέτευσης του επεξεργαστή της κάρτας γραφικών. Καθορίζει τον τρόπο με τον οποίο γίνεται rendered το κάθε εικονοστοιχείο (pixel) του αντικειμένου που χρησιμοποιεί τον συγκεκριμένο shader. Τα εν λόγω προγράμματα αποκαλούνται Shaders (σκιαστές) [9] επειδή καθορίζουν τον τρόπο με τον οποίο φωτίζονται και σκιάζονται τα μοντέλα, χωρίς όμως αυτό να αποκλείει ότι μπορούν να χρησιμοποιηθούν και για τη δημιουργία άλλων ειδικών εφέ όπως Vertex Manipulation – Mesh Displacement (για παράδειγμα Hair Shader, Jelly Shader, Grass Shader, Water Shader, Flag Shader, Cracked Ice Shader [12] δίνοντας στα τρισδιάστατα αντικείμενα, στη γεωμετρία τους, χρώμα και άλλες ιδιότητες της επιφάνειάς τους όταν διαδρά με το τεχνητό φως (glossiness, transparency για παράδειγμα), τις αντίστοιχες ιδιότητες σε πραγματικό χρόνο κατά το runtime και κατά τη διαδικασία του rendering από την κάρτα γραφικών).

Κεφάλαιο 4: Unity Engine

4.1 Εισαγωγή

Για την ανάπτυξη του παιχνιδιού “Drainage” χρησιμοποιήσαμε την μηχανή παιχνιδιών της Unity, καθώς διαθέτει αρκετά πλεονεκτήματα έναντι άλλων μηχανών παιχνιδιών. Ένα από αυτά είναι ότι οι προγραμματιστές και οι καλλιτέχνες δεν χρειάζεται στην αρχή να εστιάσουν στην δημιουργία εργαλείων, καθώς ήδη πολλά είναι έτοιμα και προσφέρονται από την Unity “out of the box” ή μέσω του Asset Store. Αντίθετα μπορούν να ξεκινήσουν κατευθείαν να δημιουργούν παιχνίδια. Το συγκεκριμένο γεγονός δίνει επιπλέον ελευθερία στους creators, καθώς δεν χρειάζεται να σπαταλήσουν πολύτιμο χρόνο δημιουργώντας την δική τους μηχανή παιχνιδιών εκ του μηδενός. Βέβαια, ο κάθε χρήστης έχει τη δυνατότητα να δημιουργήσει και τα δικά του εργαλεία για πιο εξειδικευμένες λειτουργίες ή να έχει πρόσβαση στο Asset Store της Unity, ώστε να βρει αυτό που αναζητά. Ένα άλλο πλεονέκτημα της μηχανής είναι ότι διατίθεται και σε δωρεάν Personal έκδοση, αν τα κέρδη από την ανάπτυξη εφαρμογών με αυτή είναι λιγότερα από 100.000$ τον χρόνο, άρα είναι το κατάλληλο εργαλείο για indie/solo, αλλά και αρχάριους developers (λόγω της ευχρηστίας της και της απλότητας του γραφικού περιβάλλοντος της), που ευελπιστούν να εντρυφήσουν στην δημιουργία βιντεοπαιχνιδιών ή να ξεκινήσουν άμεσα να φτιάχνουν το παιχνίδι τους. Σε περίπτωση που τα κέρδη από τις πωλήσεις είναι περισσότερα από τις 100.000$, η μηχανή διατίθεται επί πληρωμή σε κλιμακωτά πακέτα με περισσότερα εργαλεία, προσθήκες και φυσικά εξαιρετική τεχνική υποστήριξη από υπαλλήλους της Unity. Πέρα από τις λύσεις επι-πληρωμή, η κοινότητα χρηστών της Unity είναι ιδιαίτερα πολυπληθής και τα μέλη της έχουν την διάθεση να βοηθήσουν ανά πάσα στιγμή όσον αφορά στην επίλυση προβλημάτων, με πολύτιμες συμβουλές και προτάσεις μέσω του forum. Η επίλυση ενός προβλήματος που εμφανίζεται κατά την ανάπτυξη του παιχνιδιού μπορεί να είναι τόσο κοντά, όσο μια απλή αναζήτηση στην αγαπημένη μας μηχανή αναζήτησης. Επιπλέον, το οπτικό εργαλείο Bolt που διατίθεται δωρεάν με την Unity επιτρέπει ακόμα και σε χρήστες χωρίς γνώσεις προγραμματισμού να προγραμματίσουν απλή λογική ή πιο σύνθετες καταστάσεις. Ένα άλλο πλεονέκτημα της μηχανής είναι η φορητότητά της: Οποιοδήποτε παιχνίδι ή εφαρμογή αναπτύσσεται στην Unity μπορεί να γίνει ported σε μία μεγάλη πλειάδα από πλατφόρμες (XBOX, PS5, Nintendo Switch, Android, MACOSX, PC, iOS κτλ.) με μεγάλη ευκολία (πολλές φορές απλά με ένα κλικ).

Figure 64: Unity logo

4.2 Βασικές Έννοιες της Unity Engine

4.2.1 GameObject

Τα GameObjects αποτελούν θεμελιώδη αντικείμενα στην Unity Engine και αντιπροσωπεύουν χαρακτήρες, κάμερες, props, lights, audio sources, σκηνικά, κ.α. Οτιδήποτε μπορούμε να φανταστούμε ως περιεχόμενο του παιχνιδιού μας, πρέπει και οφείλει να είναι ένα GameObject. Από μόνα τους βέβαια δεν έχουν κάποια λειτουργικότητα (Empty GameObjects), αποτελούν όμως δοχεία για τα Components[10] τα οποία είναι αυτά που ουσιαστικά υλοποιούν την πραγματική λειτουργικότητα των αντικειμένων.

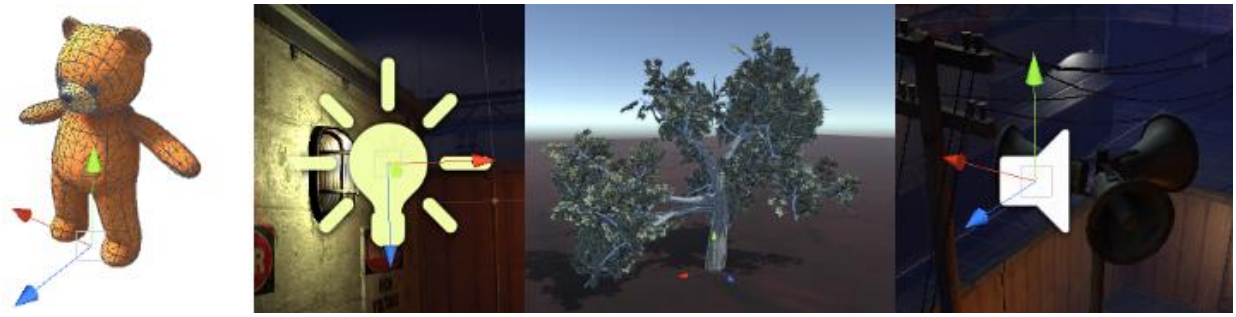


Figure 65: 4 διαφορετικοί τύποι GameObjects (από τα αριστερά προς τα δεξιά, ένα τρισδιάστατο μοντέλο-χαρακτήρας, μία πηγή φωτός, ένα δέντρο, μία ηχητική πηγή.

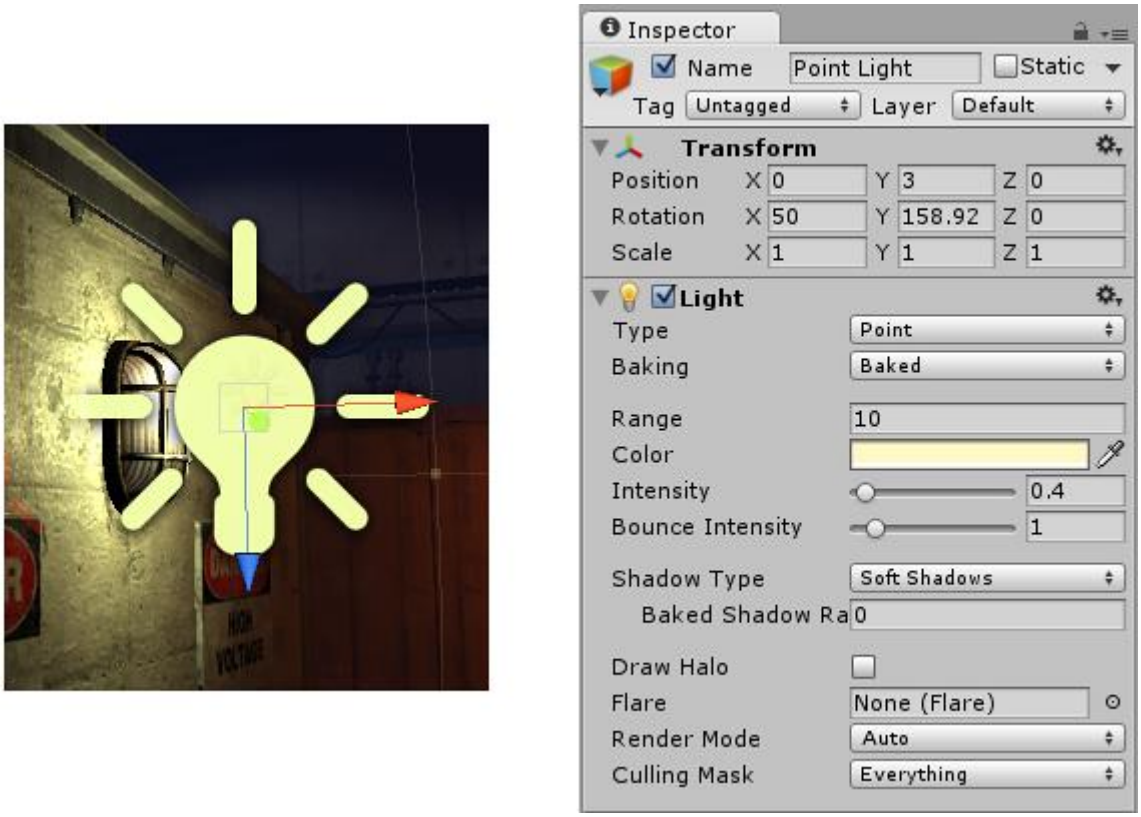


Figure 66: Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα Light Object αν επισυνάψουμε ένα Light Component στο GameObject μας μέσω του Inspector Window.

4.2.2 Components

Τα Components καθορίζουν την συμπεριφορά ενός GameObject και μπορούμε να τα βρούμε στο Inspector Window. Το Transform είναι βασικό Component κάθε GameObject, δεν μπορεί να αφαιρεθεί και καθορίζει το Position, το Rotation και το Scale ενός GameObject. Το Mesh Filter Component δίνει την πληροφορία στην κάρτα γραφικών σχετικά με το είδος Mesh που πρέπει να απεικονίσει. Το Mesh Renderer Component παρέχει πληροφορία στην κάρτα γραφικών σχετικά με το πώς ακριβώς θα απεικονίσει αυτό το Mesh. Στο Mesh Renderer μας δίνεται η δυνατότητα να επιλέξουμε το υλικό (material) του αντικειμένου, καθώς και άλλες λεπτομέρειες, όπως για παράδειγμα αν το αντικείμενο θα παράγει ή αν θα δέχεται σκιές. Το Collider Component δίνει πληροφορίες στην Physics Engine για το πώς το GameObject που έχουμε επιλέξει θα συγκρούεται ή θα διαδρά με άλλα GameObject που βρίσκονται στην σκηνή. Μπορούμε να προσθέσουμε Components σε κάποιο GameObject μέσω του πλήκτρου “Add Component” στον Inspector.

4.2.3 Scenes

Τα Scenes είναι τα κύρια assets στα οποία εργαζόμαστε με content στην Unity και περιέχουν το σύνολο ή κάποιο τμήμα ενός παιχνιδιού/εφαρμογής. Αποτελούν δοχεία για GameObjects και μπορούν να χρησιμοποιηθούν για να δημιουργήσουμε ένα κεντρικό μενού ή ξεχωριστές πίστες του παιχνιδιού. Για παράδειγμα, μπορούμε να φτιάξουμε ένα απλό παιχνίδι και να περιέχεται στην ολότητά του, μόνο σε μία σκηνή, ενώ για ένα πιο σύνθετο παιχνίδι μπορούμε να χρησιμοποιήσουμε μία σκηνή ανά πίστα, η κάθε μία με τα δικά της περιβάλλοντα, χαρακτήρες, εμπόδια, διακοσμήσεις, ήχους και διεπαφή χρήστη. Μπορούμε να δημιουργήσουμε μία νέα σκηνή για να εργαστούμε, μέσω της επιλογής File->New Scene. Με αυτόν τον τρόπο δημιουργείται μία νέα σκηνή η οποία περιέχει μόνο μία κάμερα και ένα Directional Light.

4.2.4 Prefabs

Τα Prefabs είναι ειδικός τύπος component που επιτρέπει την αποθήκευση, ταυτόχρονη παραμετροποίηση και μεταβολή ρυθμίσεων σε ένα αντικείμενο και απόθηκευση των ρυθμίσεων σε όλα τα instances αυτού του αντικειμένου στη σκηνή μας. Κάτι τέτοιο είναι ιδιαίτερα χρήσιμο για αντικείμενα που πρόκειται να επαναχρησιμοποιηθούν πολλές φορές (για παράδειγμα πλατφόρμες, εμπόδια, enemies, τοίχοι κτλ.). Αυτό ακριβώς είναι και ένα μεγάλο πλεονέκτημα των Prefabs, ότι αποτελούν ουσιαστικά διασυνδεμένα αντίγραφα του πρωτότυπου prefab που βρίσκεται στο project μας, οπότε οποιαδήποτε νέα παραμετροποίηση σε αυτό, ισχύει και για τα υπόλοιπα instances του Prefab (override changes), χωρίς να χρειάζεται η μεταβολή των ρυθμίσεων σε κάθε ένα από αυτά ξεχωριστά, σπαταλώντας πολύτιμο χρόνο. Υπάρχει για παράδειγμα η περίπτωση να έχουμε τοποθετήσει 1000 αντίγραφα του ίδιου prefab στην ιεραρχία της σκηνής μας και θα ήταν τρομερά χρονοβόρο να χρειαζόταν να ρυθμίσουμε το κάθε ένα από αυτά ξεχωριστά στην περίπτωση που θέλαμε να προχωρήσουμε σε αλλαγές/παραμετροποιήσεις γι' αυτή την ομάδα αντικειμένων. Τα Prefabs δημιουργούνται αυτόματα, αν σύρουμε ένα GameObject από το Hierarchy στο Project Window (βλ. 4.3.3).

4.3 Η διεπαφή χρήστη (User Interface) του Unity Editor

Το User Interface του Unity Editor αποτελείται από πέντε βασικά panels τα οποία κάνουν το project μας εύκολα διαχωρίσιμο και είναι τα εξής:

4.3.1 Scene Window

Στο συγκεκριμένο παράθυρο περιέχεται οπτικοποιημένη η διαδραστική όψη του περιβάλλοντος/σκηνής παιχνιδιού που δημιουργούμε. Εδώ μπορούμε να τοποθετήσουμε όπου εμείς θέλουμε τα αντικείμενα του παιχνιδιού (Game Objects) όπως σκηνικά, χαρακτήρες, φώτα, κάμερες, ηχητικές πηγές, να τα περιστρέψουμε ή να μεταβάλλουμε το μέγεθός τους και να στήσουμε οπτικά το περιβάλλον του παιχνιδιού μας με όλα τα επιμέρους στοιχεία που το αποτελούν.

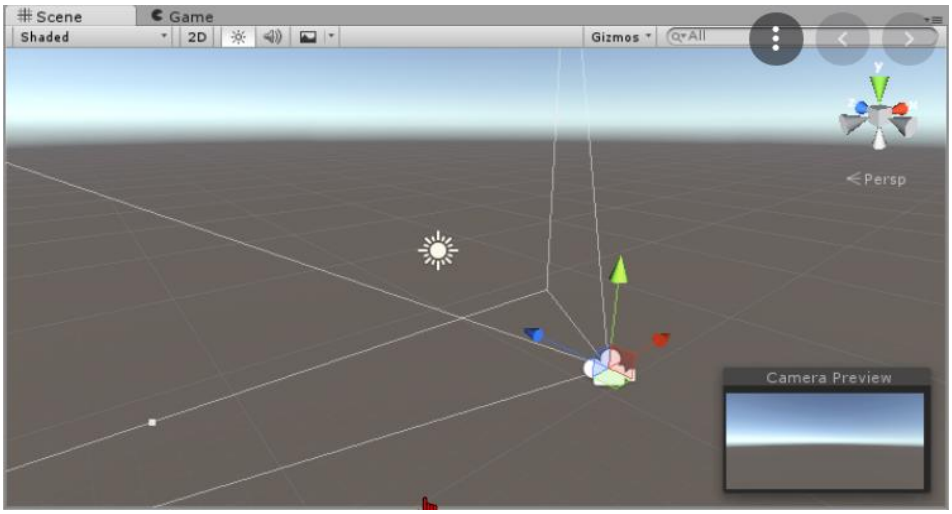


Figure 67: Το Scene Window του Unity Editor.

4.3.2 Game Window

Στο συγκεκριμένο παράθυρο βλέπουμε την άποψη της κάμερας, δηλαδή οτιδήποτε «κοιτάζει» η κάμερα που έχουμε στήσει στο παιχνίδι μας. Πατώντας το πλήκτρο "Play", μπορούμε ανά πάσα στιγμή να «τρέξουμε» μία προεπισκόπηση του παιχνιδιού και να το δοκιμάσουμε-τεστάρουμε, να χειριστούμε τον χαρακτήρα μας για παράδειγμα, κάτι που μας δίνει τη δυνατότητα αποσφαλμάτωσης χωρίς όμως πρώτα να χρειαστεί να δημιουργήσουμε build της εφαρμογής/παιχνιδιού μας.

4.3.3 Project Window

Εδώ βρίσκονται συγκεντρωμένα όλα τα assets που έχουμε διαθέσιμα για χρήση στο παιχνίδι μας. Τα assets είναι αρχεία που βρίσκονται στο σκληρό μας δίσκο στην τοποθεσία που έχουμε αποθηκεύσει το project μας . Το Project Window μας δίνει τη δυνατότητα εύκολης και άμεσης πρόσβασης και αναζήτησης σε αυτά τα αρχεία, τα οποία μπορεί να είναι scripts, textures, τρισδιάστατα μοντέλα, ηχητικά clips, animation clips και άλλα αρχεία χρήσιμα για το παιχνίδι μας. Τα Scenes που δημιουργούμε είναι επίσης assets και μπορούμε να τα αποθηκεύσουμε ή να έχουμε πρόσβαση σε αυτά μέσω του Project Window. Για να κάνουμε import κάποιο asset στο Project Window, έτσι ώστε να γίνει διαθέσιμο για το project μας, επιλέγουμε "Import new asset" ή απλά σύρουμε το αρχείο από τον File Explorer στο Project Window.

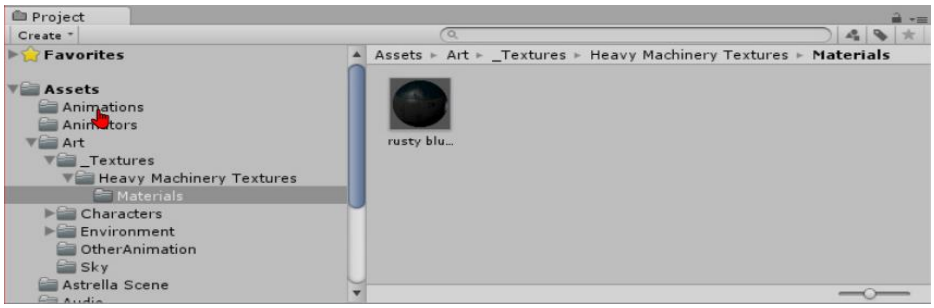


Figure 68: Το Project Window του Unity Editor

4.3.4 Hierarchy Window

Το Hierarchy Window είναι ουσιαστικά ένα είδος inventory που αποτελεί την λίστα με όλα τα GameObjects, τα οποία χρησιμοποιούνται στην τρέχουσα σκηνή μας. Τα συγκεκριμένα αντικείμενα παιχνιδιού μπορούν να μετακινηθούν και να αναδιαταχθούν στην λίστα, ανάλογα με τις προτιμήσεις μας. Μπορούμε επιπλέον να ομαδοποιήσουμε τα συγκεκριμένα GameObjects στο Hierarchy Window και να δημιουργήσουμε οικογένειες από GameObjects. Το GameObject στην κορυφή κάθε Hierarchy Group είναι ο γονέας (parent), ενώ τα GameObjects που βρίσκονται ομαδοποιημένα μέσα στο Parent GameObject είναι τα children GameObjects.

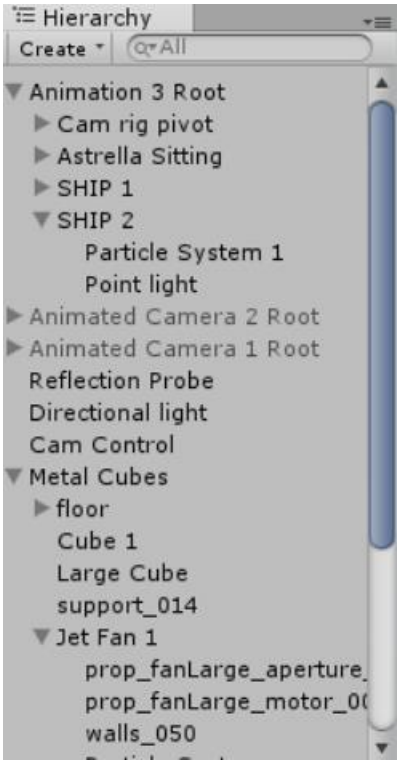


Figure 69: Το Hierarchy Window του Unity Editor

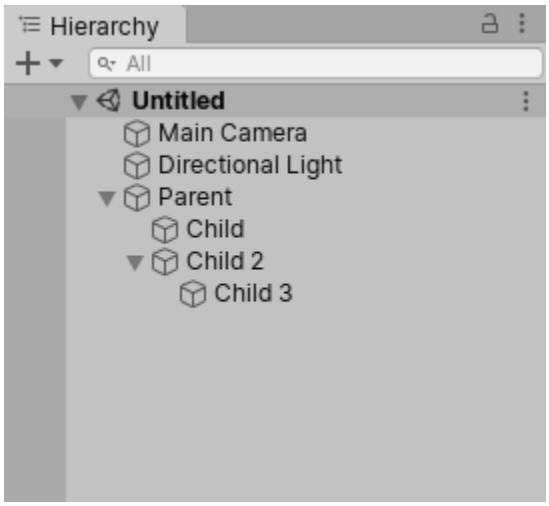


Figure 70: Hierarchy Parenting

4.3.5 Inspector Window

Το Inspector Window περιέχει όλες τις ιδιότητες, λεπτομέρειες και χαρακτηριστικά του GameObject ή Asset που έχουμε επιλέξει στο Scene/Hierarchy Window ή στο Project Window αντίστοιχα. Μπορούμε να το φανταστούμε σαν έναν μεγεθυντικό φακό που κάνει zoom στο αντικείμενο που

έχουμε επιλέξει και στα ιδιαίτερα συστατικά του. Στο συγκεκριμένο παραθύρο μας δίνεται η δυνατότητα να μεταβάλλουμε τιμές, να μετακινήσουμε sliders, να επέμβουμε στα επιμέρους συστατικά του αντικειμένου μας, να προσθέσουμε ή να αφαιρέσουμε components (συστατικά) του αντικειμένου μας κ.α.

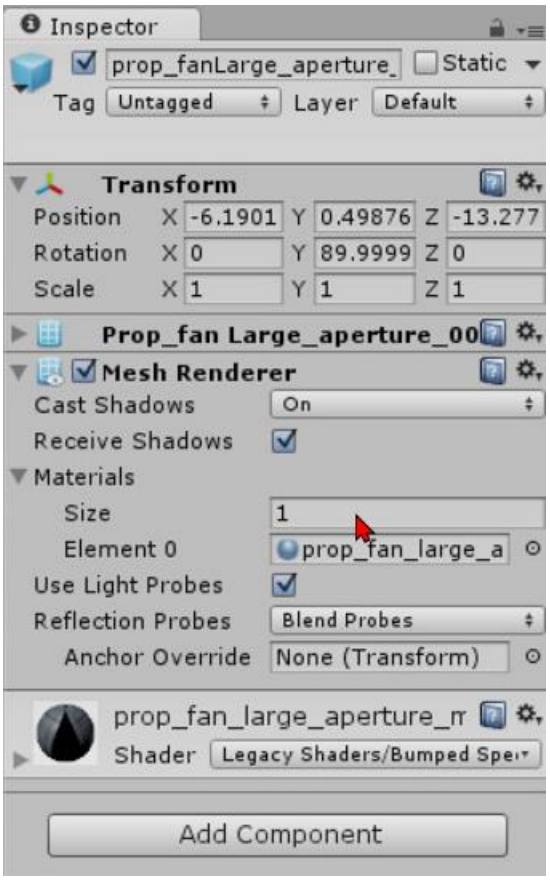


Figure 71: Το Inspector Window του Unity Editor

4.4 Φωτισμός (Lighting)

Για τον φωτισμό των σύγχρονων παιχνιδιών γίνεται εκτεταμένη χρήση του Global Illumination. Το Global Illumination ή "GI" είναι ένας όρος που χρησιμοποιείται για να περιγράψει ένα σύνολο αλγορίθμων που χρησιμοποιείται για τη ρεαλιστική φωτοαπόδοση στα τρισδιάστατα γραφικά υπολογιστών. Η ακριβής προσομοίωση του φωτισμού είναι σίγουρα μία πρόκληση για το υλικό και μπορεί να είναι ιδιαίτερα δαπανηρή υπολογιστικά. Γι' αυτό το λόγο, τέτοιου είδους συστήματα δίνουν επιπλέον τη δυνατότητα προϋπολογισμού (baking) του φωτός και της σκίασης για στατικά αντικείμενα παιχνιδιού (τα οποία δεν θα χρειαστεί να μετακινηθούν κατά το runtime δηλαδή), έτσι ώστε να επιτύχουμε πιο ταχείς χρόνους rendering κατά το runtime, γεγονότος ιδιαίτερα σημαντικού για την ποιότητα της επιθυμητής, smooth εμπειρίας για τον χρήστη που παίζει το παιχνίδι.
Σε γενικές γραμμές, ο φωτισμός στη Unity μπορεί να θεωρηθεί είτε "σε πραγματικό χρόνο" (Real-time), είτε "προϋπολογισμένος" (Baked), όμως και οι δύο αυτές τεχνικές μπορούν να χρησιμοποιηθούν σε συνδυασμό για να πετύχουμε φωτισμό εξαιρετικής ποιότητας στην σκηνή μας.

4.4.1 Real-time Lighting (Φωτισμός πραγματικού χρόνου)

Εξ ορισμού, τα φώτα στη Unity είναι πραγματικού χρόνου. Αυτό σημαίνει ότι όλοι οι υπολογισμοί του εικονικού φωτός διεξάγονται κατά το runtime και σε κάθε frame, κάτι που ανάλογα με την πολυπλοκότητα της σκηνής, των shaders και της ισχύος της συσκευής (πχ. low – end devices) που χρησιμοποιούμε, μπορεί να αποβεί μοιραίο για την απόδοση της εφαρμογής μας. Καθώς τα φώτα και τα GameObjects εντός της σκηνής κινούνται, ο φωτισμός των αντικειμένων ενημερώνεται αμέσως και οι υπολογισμοί συντελούνται εκείνη τη στιγμή. Αυτό μπορεί να γίνει αντιληπτό τόσο κατά την προεπισκόπηση στο Scene View, όσο και κατά το runtime της τελικής εφαρμογής μας ή κατά το Play mode στον editor. Ο realtime φωτισμός είναι ο πιο βασικός τρόπος φωτισμού κινούμενων αντικειμένων-χαρακτήρων στη σκηνή του παιχνιδιού μας. Δυστυχώς, δεν μπορούμε να έχουμε αναπήδηση των ακτίνων φωτός (indirect lighting) από τα φώτα πραγματικού χρόνου της Unity, οπότε για να προσδώσουμε ρεαλισμό στον φωτισμό μας, πρέπει επιπλέον να ενεργοποιήσουμε προϋπολογισμένο φωτισμό και να κάνουμε “bake” πληροφοριών φωτισμού σε lightmaps, πριν και όχι κατά τη διάρκεια του runtime.

4.4.2 Baked GI Lighting (“Ψημένος” προϋπολογισμένος φωτισμός)

Κατά το “ψήσιμο” (baking) ενός “lightmap”, υπολογίζονται τα αποτελέσματα του φωτός σε στατικά (static) αντικείμενα στη σκηνή και τα αποτελέσματα γράφονται σε υφές (textures) που επικαλύπτουν τη γεωμετρία της σκηνής για να δημιουργήσουν εφέ φωτισμού. Τα lightmaps πέρα από την πληροφορία για το άμεσο φως (direct lighting) σε μία επιφάνεια, μπορούν να περιλαμβάνουν και επιπρόσθετες πληροφορίες σχετικά με την αναπήδηση, τα διαφορετικά-διαδοχικά bounces των ακτίνων του φωτός, και πιο ποιοτικές σκιές, δίνοντάς μας έτσι ένα πιο ρεαλιστικό, πειστικό οπτικό αποτέσμα. Η συγκεκριμένη υφή φωτισμού (lightmap) μπορεί να χρησιμοποιηθεί μαζί με άλλες πληροφορίες επιφάνειας, όπως το χρώμα (albedo map) και το ανάγλυφο (normal map) από τον σκιαστή (Shader) που σχετίζεται με το υλικό (Material) ενός αντικειμένου. Με τον προϋπολογισμένο φωτισμό, οι συγκεκριμένες υφές που περιέχουν πληροφορίες για τον φωτισμό της σκηνής (lightmaps) δεν μπορούν να μεταβληθούν κατά τη διάρκεια του Play mode και έτσι αναφέρονται ως “στατικές”. Τα φώτα πραγματικού χρόνου μπορούν να λειτουργήσουν επιπρόσθετα πάνω σε σκηνές που έχουν επικαλυφθεί με υφές προϋπολογισμένου φωτός, αλλά δεν μπορούν να αλλάξουν τις υφές αυτές καθεαυτές. Με αυτήν την προσέγγιση, ανταλλάσσουμε τη δυνατότητα να κινούμε τα φώτα μας στο παιχνίδι με μια πιθανή αύξηση της απόδοσης, κάτι που είναι αρκετά χρήσιμο όταν στοχεύουμε σε λιγότερο ισχυρό υλικό, όπως για παράδειγμα οι κινητές συσκευές.

Εμείς για λόγους απόδοσης χρησιμοποιήσαμε κυρίως την τεχνική του προϋπολογισμένου φωτισμού (Baked GI Lighting) και ελάχιστα φωτισμό πραγματικού χρόνου, κι αυτό μόνον όπου υπήρχε η ανάγκη για Volumetric Lighting effect (εφέ ογκομετρικού φωτισμού και προσομοίωση του φωτός, όταν διαδρά για παράδειγμα με μικροσωματίδια σκόνης) ή speculars σε αντικείμενα.

Figure 72: Volumetric Lighting

4.4.3 Types of Light

Point lights

Ένα point light τοποθετημένο σε κάποιο σημείο της σκηνής, στέλνει το φως προς όλες τις κατευθύνσεις εξίσου. Το φως ξεκινάει από το κεντρικό σημείο του point light στο world space της σκηνής μας και παράγει ακτίνες με κατεύθυνση και ισόποσες εντάσεις μέχρι ένα προκαθορισμένο range. Όσο απομακρύνουμε τα αντικείμενα του παιχνιδιού μας από το κέντρο του point light, η ένταση αυτού μειώνεται (όσον αφορά στο πόσο ακριβώς φωτίζονται τα αντικείμενά μας) και φτάνει στο μηδέν μετά το προκαθορισμένο range που αναφέραμε προ λίγου. Το πόσο έντονο είναι το φως σε κάποιο σημείο στο world space, είναι αντιστρόφως ανάλογο του τετραγώνου της απόστασης αυτού του σημείου από το κέντρο της πηγής του φωτός μας, χαρακτηριστικό γνωστό ως "inverse square law", που περιγράφει κατά κάποιον τρόπο και τη συμπεριφορά του φωτός στην πραγματικό, υλικό κόσμο.
Στις μηχανές γραφικών χρησιμοποιούμε τα point lights για να προσομοιώσουμε φως σαν αυτό που παράγουν οι κλασικοί λαμπτήρες (light bulbs).

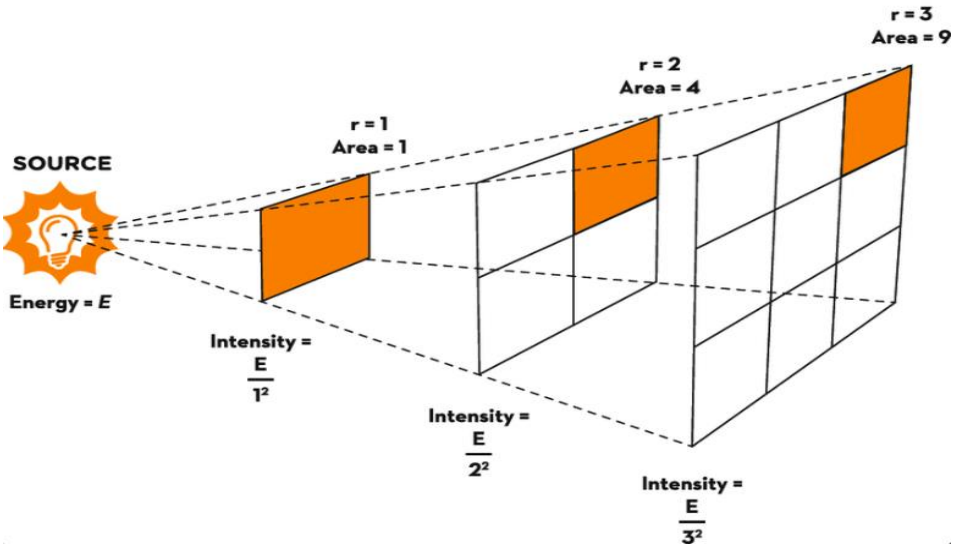


Figure 73: Inverse square law

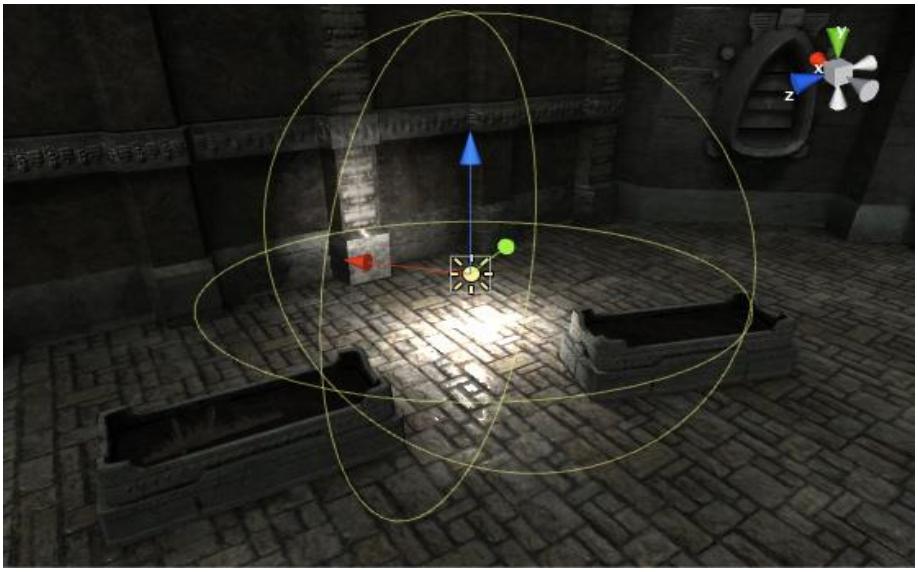


Figure 74: Effect of a point light in a scene

Spot lights

Τα spot lights διαχέουν το φως στη σκηνή σε σχήμα κώνου και όχι προς όλες τις κατευθύνσεις- σφαιρικά όπως τα point lights και το φως που παράγουν, έχει επίσης ένα καθορισμένο κέντρο, εύρος, αλλά και γωνία. Όσο πλησιάζουμε τις άκρες του κώνου, σε σχέση πάντα με το εύρος και την ένταση που έχουμε καθορίσει στο light component, μειώνεται και η ένταση του φωτός, όσον αφορά στη διάδραση με τα αντικείμενα, το φως ξεθωριάζει και το φως ομοιάζει όλο και περισσότερο με ημίφως, όσο το αντικείμενο που φωτίζεται απομακρύνεται από το κέντρο της πηγής του spot light.
Τα spot lights χρησιμοποιούνται για να αναπαραστήσουν τεχνητές πηγές φωτός, όπως φακούς, προβολείς αυτοκινήτων και προβολείς γενικότερα [10].

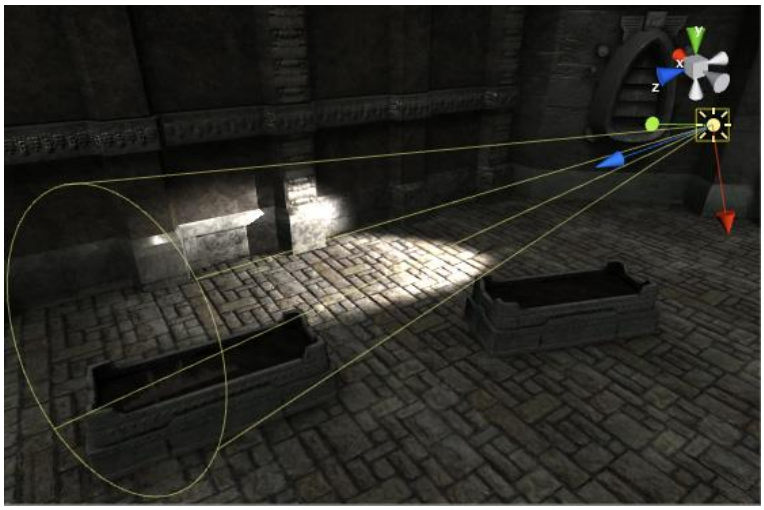


Figure 75: Effect of a Spot Light in a scene

Directional lights

Τα Directional Lights χρησιμοποιούνται για την προσομοίωση μίας πολύ μεγάλης και πολύ μακρινής πηγής φωτός, όπως είναι το φως του ηλίου ή της σελήνης και δεν έχει καμία σημασία η θέση που θα τα τοποθετήσουμε στη σκηνή, εφόσον θεωρείται ότι βρίσκονται σε απείρως μακρινή απόσταση, αλλά κυρίως σημασία έχει η ένταση και η περιστροφή του φωτεινού αντικειμένου που έχουμε προκαθορίσει στον inspector. Καθώς η θέση του directional light δεν διαδραματίζει κάποιο ρόλο, η ένταση του φωτός δεν μειώνεται, αλλά είναι ίση σε όλες τις αποστάσεις από την πηγή.

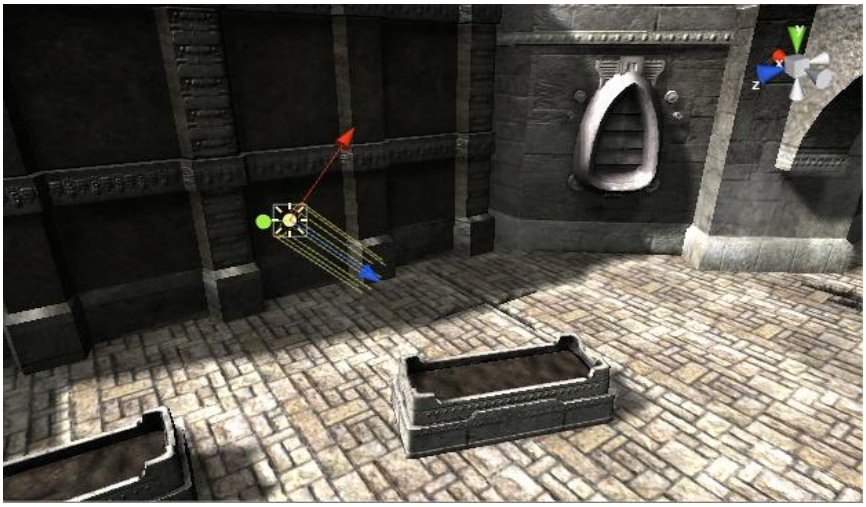


Figure 76: Effect of a Directional light in a scene.

4.5 Physics, Colliders, Rigidbodies

Για να είναι αληθοφανές ένα κινούμενο αντικείμενο μέσα στο παιχνίδι, πρέπει να επηρεάζεται από νόμους της φυσικής, να έχει κάποια μάζα, να επιταχύνει σωστά, να επηρρεάζεται από τη βαρύτητα, να μπορεί να συγκρουστεί με άλλα αντικείμενα. Ευτυχώς, η Unity μας παρέχει out-of-the-box έτοιμα συστήματα φυσικής για δισδιάστατα και τρισδιάστατα παιχνίδια που χειρίζονται την προσομοίωση υπολογισμών φυσικής, χρήσιμων για την εφαρμογή μας.

Colliders (Συγκρουστές)

Ο συγκρουστής ενός αντικειμένου, ο οποίος είναι αόρατος για τον τελικό χρήστη, αλλά μπορούμε να τον δούμε στον editor κατά την ανάπτυξη του βιντεοπαιχνιδιού, είναι ουσιαστικά ένα component που προσθέτουμε στο gameObject μας και καθορίζει το σχήμα του, όχι οπτικά, αλλά με στόχο την ανίχνευση συγκρούσεων (από το σύστημα Physics), με άλλα αντικείμενα, τα οποία έχουν επίσης τους δικούς τους συγκρουστές, αλλά και την άσκηση δυνάμεων σε αυτό και δεν χρειάζεται να έχει ακριβώς το ίδιο σχήμα με το mesh του αντικειμένου. Αντίθετα μία πιο πρόχειρη προσέγγιση είναι συχνά πιο αποτελεσματική και απαιτεί μικρότερη χρήση πόρων κατά το runtime, οπότε και έχει καλύτερη απόδοση. Οι απλούστεροι και λιγότερο δαπανηροί όσον αφορά στους πόρους του επεξεργαστή είναι οι primitive τύποι συγκρουστών. Μιλώντας για ένα τρισδιάστατο παιχνίδι, αυτοί είναι οι Box Collider, Sphere Collider, Wheel Collider (collider σε σχήμα τροχού, για τους τροχούς ένος οχήματος για παράδειγμα), Capsule Collider (συνήθως το χρησιμοποιούμε σαν collider του χαρακτήρα μας). Επιπλέον υπάρχουν οι Mesh Colliders (colliders που παίρνουν αυτόματα το σχήμα του mesh του αντικειμένου μας, ανάλογα με τη θέση των ξεχωριστών vertices, edges, faces αυτού, αλλά είναι πιο δαπανηροί για τον επεξεργαστή). Μπορούμε να προσθέσουμε περισσότερους από έναν εξ αυτών (primitive colliders) σε κάποιο αντικείμενο με σκοπό να δημιουργήσουμε πιο σύνθετους συγκρουστές, ώστε να προσεγγίσουμε αρκετά καλά το σχήμα ενός αντικειμένου (χωρίς την χρήση ενός δαπανηρού mesh collider) και ταυτόχρονα να διατηρήσουμε το χαμηλό κόστος για τον επεξεργαστή. Στο Collider component του GameObject υπάρχει η επιλογή Is Trigger. Με τη συγκεκριμένη επιλογή γίνεται εφικτή η εκτέλεση ενός script και η "πυροδότηση" γεγονότων, όταν ένας collider αγγίξει, βγει ή μείνει μέσα στον Collider, τον οποίο έχουμε θέσει ως trigger, μέσω των μεθόδων OnTriggerEnter, OnTriggerExit και OnTriggerStay αντίστοιχα. Μπορούμε επίσης προαιρετικά να έχουμε πρόσβαση στον Collider που ήρθε σε επαφή με τον trigger collider μας.

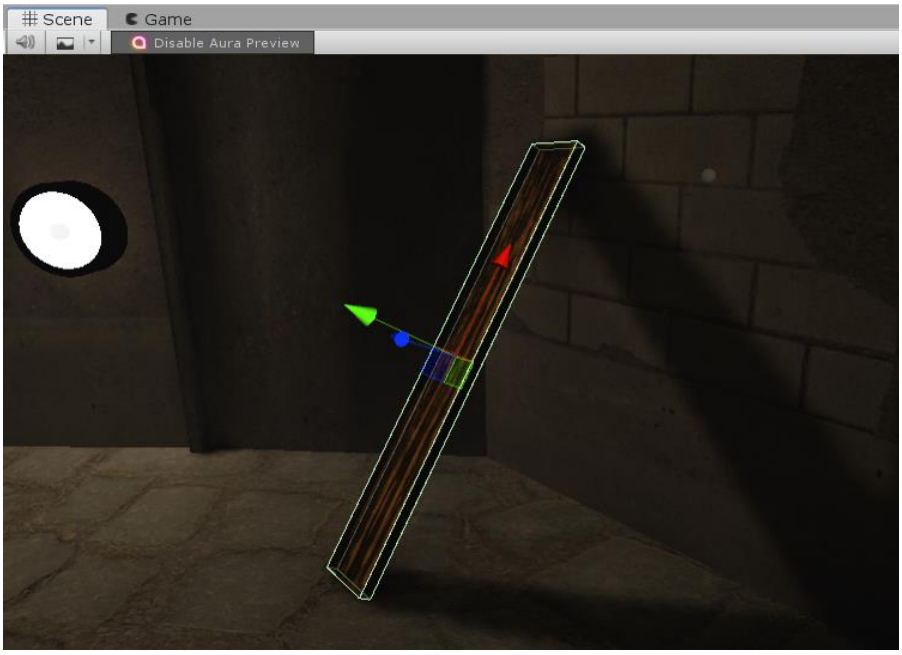


Figure 77: A piece of wood and its box collider (green outline).

Rigidbodies

Το Rigidbody είναι ένα κύριο συστατικό ενός GameObject και επιτρέπει τη φυσική συμπεριφορά του. Με προσαρτημένο ένα RigidBody το αντικείμενο ανταποκρίνεται αμέσως στη βαρύτητα. Αν το επιθυμούμε, ένα αντικείμενο παιχνιδιού μπορεί να έχει κάποιο rigidbody component, αλλά η κίνησή του να μην επηρρεάζεται από δυνάμεις της φυσικής, άρα και να μην ελέγχεται από το σύστημα φυσικής της Unity, αλλά να το μετακινούμε κατά βούληση με κώδικα (για παράδειγμα με την έτοιμη συνάρτηση Rigidbody.MovePosition). Για παράδειγμα, μπορεί να επιθυμούμε να ελέγχουμε τον χαρακτήρα μας απευθείας μέσω συναρτήσεων στον κώδικά μας, αλλά να εξακολουθούν να ανιχνεύονται οι επαφές του αντικειμένου και η είσοδος αυτού σε triggers. Τότε λέμε ότι το rigidbody μας είναι “kinematic” (isKinematic toggle – rigidbody component στον Inspector). Είναι δυνατό να μεταβάλλουμε την τιμή της boolean μεταβλητής IsKinematic από ένα script για να επιτρέψουμε την ενεργοποίηση και την απενεργοποίηση της φυσικής για ένα αντικείμενο, κάτι το οποίο μας χρειάστηκε στην υλοποίηση του παιχνιδιού και συγκεκριμένα στο Level 3, που επιθυμούσαμε να ισχύει η δύναμη της βαρύτητα για κάποια αντικείμενα, όχι πριν, αλλά από την χρονική στιγμή της πυροδότησης ενός event και μετά.

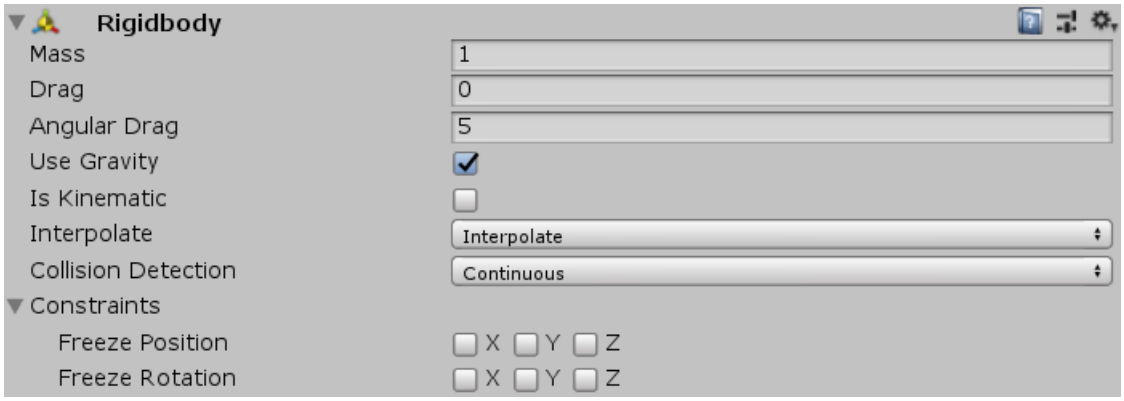


Figure 78: A rigidbody component.

4.6 Particle System

Η Unity Engine διαθέτει ένα ιδιαίτερα ισχυρό σύστημα κίνησης σωματιδίων, με την χρήση του οποίου, μπορούμε να δημιουργήσουμε ειδικά εφέ, όπως κινούμενα υγρά, καπνό, σύννεφα, φλόγες, μαγικά ξόρκια, σπίθες, σταγόνες και μία σειρά από άλλα εφέ. Μία σχετικά μικρή χρήση particle effects μπορεί να δώσει ζωντάνια στο παιχνίδι μας και να κάνει το περιβάλλον να φαίνεται λιγότερο αδιάφορο. Παλιότερα, χρειαζόταν η γνώση προγραμματισμού γραφικών για να δημιουργήσουμε έστω ένα μικρό κομμάτι καπνού. Ευτυχώς, το Unity κάνει τη δημιουργία συστημάτων σωματιδίων αρκετά απλή με ένα αρθρωτό σύστημα σωματιδίων που ονομάζεται Shuriken, το οποίο είναι εύκολο στην εκμάθηση, αλλά μας επιτρέπει να δημιουργήσουμε και πιο πολύπλοκα εφέ. Τα particle effects φτιαγμένα στο Shuriken Particle System "τρέχουν" στη CPU, ενώ τα εφέ του μεταγενέστερου Particle System (VFX Graph) στην GPU.

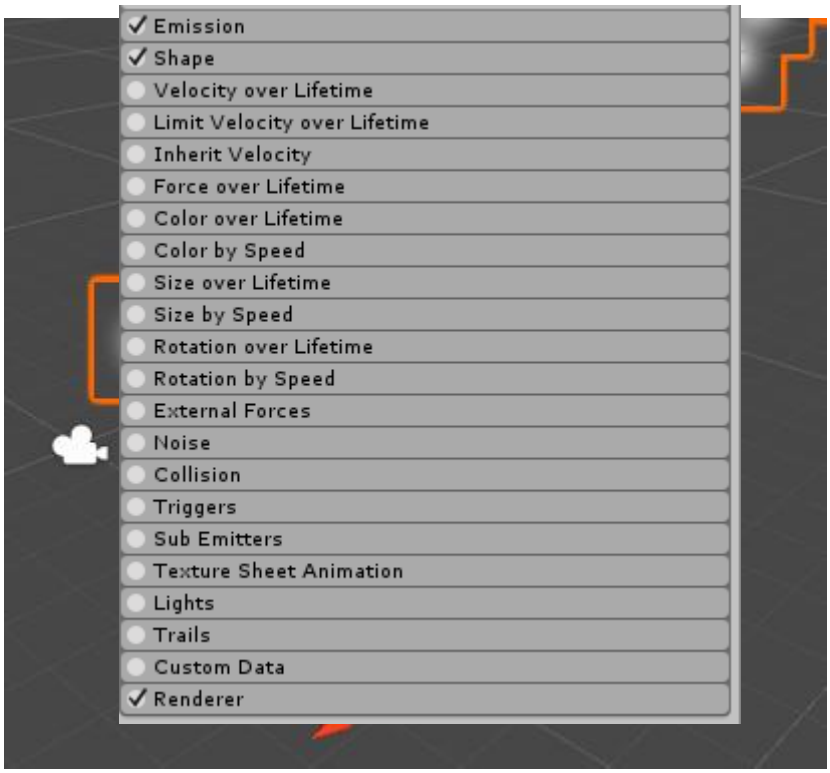


Figure 79: Particle System in the Unity Scene. Modules of a Particle System .

4.7 Animation System

Η Unity μας παρέχει ένα έτοιμο σύστημα για δημιουργία κίνησης και animation (γνωστό και ως “Mecanim”. Χαρακτηριστικά του είναι τα εξής:

- Απλότητα στη δημιουργία κίνησης μέσω της μετάβασης από μία κατάσταση του αντικειμένου σε μία άλλη, εντός ενός προκαθορισμένου χρονικού διαστήματος, για όλα τα αντικείμενα της Unity και τις παραμέτρους αυτών.
- Δυνατότητα χρήσης και παραμετροποίησης curves, για πιο ομαλή μετάβαση μεταξύ διαφορετικών καταστάσεων.
- Υποστήριξη για εισαγωγή clip κινούμενων εικόνων και κινουμένων σχεδίων που δημιουργούνται στη Unity, αλλά και σε άλλες εφαρμογές.
- Εξειδικευμένη κίνηση για ανθρωποειδή μοντέλα.
- Ιδιαίτερα ξεκάθαρη προεπισκόπηση των animation clips και προεπισκόπηση των μεταβάσεων μεταξύ διαφορετικών καταστάσεων. Κάτι τέτοιο, επιτρέπει στους δημιουργούς να εργάζονται ανεξάρτητα από τους προγραμματιστές, να δημιουργούν πρωτότυπα και να κάνουν προεπισκόπηση των κινούμενων εικόνων τους, πριν αυτές διασυνδεθούν με τον κώδικα παιχνιδιού κατά το build της εφαρμογής.
- Απλοποιημένη ροή εργασίας για τη δημιουργία animations, αλλά και μηχανή καταστάσεων για τη μετάβαση σε διαφορετικά animation states με τον καθορισμό και χρήση παραμέτρων.

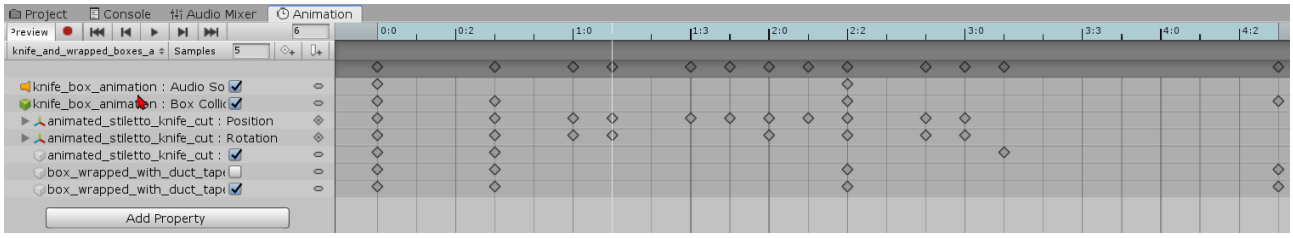


Figure 80: The animation window where we can prepare our animation clip and preview it.

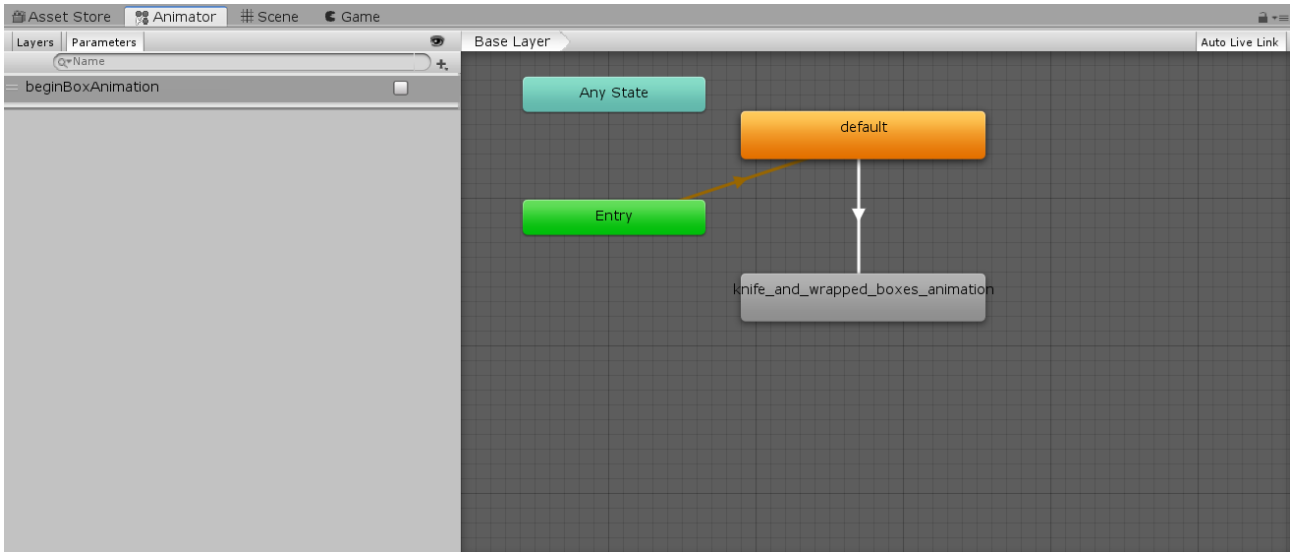


Figure 81: The animator window with its state machine and parameters.

4.8 Audio System

Βασικό κομμάτι κάθε παιχνιδιού είναι φυσικά και ο ήχος του, είτε μιλάμε για ειδικά εφέ, περιβαλλοντικούς ήχους, μουσική και ομιλίες, είτε για ήχους διάδρασης του παίκτη με το User Interface του παιχνιδιού. Η Unity Engine διαθέτει ένα ιδιαίτερα ευέλικτο και ταυτόχρονα ισχυρό σύστημα ήχου. Ο χρήστης μπορεί να εισάγει τις περισσότερες γνωστές μορφές αρχείων ήχου, ενώ το σύστημα διαθέτει εξελιγμένα χαρακτηριστικά για την αναπαραγωγή τους στον τρισδιάστατο χώρο, προαιρετικά ηχητικά εφέ, όπως echo, filters, reverb, compressors κ.α., καθώς και audio mixer για την καλύτερη οργάνωση των ήχων σε groups.
Εκτός βιντεοπαιχνιδιού, στην καθημερινή ζωή, οι ήχοι εκπέμπονται από πηγές ήχου και ο ακροατής τους αντιλαμβάνεται μέσω της ακοής. Αυτή είναι και η λογική που χρησιμοποιείται και στο Audio System της Unity. Υπάρχουν τα Audio Source components που προσδίδουν στο GameObject χαρακτηριστικά πηγής ήχου, ενώ το Audio Listener component μπορεί να μετατρέψει τον παίκτη σε ακροατή των ήχων του παιχνιδιού. Ο ακροατής μπορεί να πει κατά προσέγγιση από ποια κατεύθυνση προέρχεται ένας ήχος και μπορεί επίσης να νιώσει και μία αίσθηση της απόστασης από την ηχητική πηγή, την ένταση και την ποιότητα του ήχου αυτού καθεαυτού. Μία πηγή ήχου που βρίσκεται σε κατάσταση γρήγορης κίνησης (όπως ένα διερχόμενο αυτοκίνητο για παράδειγμα) θα αλλάξει τόνο καθώς κινείται, ως αποτέλεσμα του φαινομένου Doppler. Το περιβάλλον επίσης θα επηρρεάσει τον τρόπο με τον οποίο αντανακλάται ο ήχος, έτσι μια φωνή ή ο ήχος μίας σταγόνας σε μία σπηλιά θα έχει ηχώ, άλλα ο ίδιος ήχος σε υπαίθριο χώρο όχι. Για να προσομοιωθεί σωστά η θέση της πηγής του ήχου σε σχέση με την ακροατή, τα αντικείμενα που επιθυμούμε να παράγουν ήχο μέσα στο παιχνίδι, πρέπει να έχουν προσαρτημένο ένα Audio Source component. Οι ήχοι που εκπέμπονται συλλαμβάνονται στη συνέχεια από έναν ακροατή ήχου (audio listener), που συνήθως αποτελεί συστατικό της κύριας κάμερας του παίκτη. Η Unity μπορεί στη συνέχεια να προσομοιώσει το αποτέλεσμα της απόστασης και της θέσης μιας πηγής από τον αντικείμενο-ακροατή και να τα αναπαράγει ανάλογα στον χρήστη. Η σχετική ταχύτητα των αντικειμένων-ηχητικών πηγών μπορεί επίσης να χρησιμοποιηθεί για την προσομοίωση του φαινομένου Doppler για πρόσθετο ρεαλισμό. Η

ηχώ δεν μπορεί να προσομοιωθεί αυτόματα από τη μηχανή, λαμβάνοντας υπόψιν της γεωμετρίας της σκηνής, αλλά μπορούμε να πετύχουμε ένα τέτοιο εφέ προσθέτοντας εμείς οι ίδιοι αντίστοιχα φίλτρα και effects στο Audio Mixer [10].
Σε περιπτώσεις όπου τα αντικείμενα μπορούν να μετακινηθούν μέσα και έξω από ένα μέρος με έντονη ηχώ, μπορούμε να προσθέσουμε μια ζώνη αντήχησης στη σκηνή. Το Audio Mixer μας επιτρέπει την ηχητική μίξη διαφορετικών καναλιών ήχου στη σκηνή μας, την ομαδοποίηση των ήχων μας σε διαφορετικά Audio Mixer groups, την χρήση φίλτρων και εφέ στα ξεχωριστά κανάλια μας, την έκθεση παραμέτρων (exposed parameters), όπως η ένταση του ήχου ή μεταβλητές των φίλτρων και εφέ, έτσι ώστε να τις μεταβάλλουμε (SetFloat) μέσω κώδικα και να μπορούμε να κάνουμε ηχητικό mastering και audio arrangement.

Figure 82: Audio sources and audio listener.

4.9 Scripting In Unity

Μια γλώσσα σεναρίων (scripting language) είναι μία γλώσσα προγραμματισμού που επιτρέπει τον έλεγχο μίας ή περισσότερων εφαρμογών. Βασικά, οι "γλώσσες σεναρίων" είναι ένα είδος γλωσσών προγραμματισμού. Οι scripting languages συνήθως ερμηνεύονται ("interpreted") αντί να μεταγλωτίζονται ("compiled"). Για παράδειγμα, ένα πρόγραμμα C πρέπει πρώτα να μεταγλωτιστεί, πριν εκτελεστεί, ενώ μια γλώσσα δέσμης ενεργειών δεν χρειάζεται να μεταγλωττιστεί.
Σκοπός του μεταγλωττιστή και του μεταφραστή είναι να μετατρέψουν τον πηγαίο κώδικα που είναι γραμμένος σε γλώσσα υψηλού επιπέδου (High Level language) που μπορεί να καταλάβει ευκολότερα ο άνθρωπος, σε γλώσσα μηχανής που μπορεί να καταλάβει ο υπολογιστής. Αν συναντήσει κάποιο λάθος, ο compiler εκτυπώνει το κατάλληλο μήνυμα error στην κονσόλα. Αν δεν συναντήσει λάθη, μετατρέπεται σε κώδικα μηχανής και συνδέει (linking) τα διαφορετικά αρχεία του κώδικα σε ένα εκτελέσιμο αρχείο για να εκτελεστεί από τον χρήστη (πχ. ένα exe αρχείο). Ο interpreter δημιουργεί το πρόγραμμα, αλλά δεν δημιουργεί links μεταξύ των αρχείων του κώδικα, ούτε μετατρέπει τον πηγαίο κώδικά μας σε κώδικα μηχανής, ενώ οι δηλώσεις κώδικα εκτελούνται γραμμή-γραμμή κατά την εκτέλεση του προγράμματος. Προφανώς ένα πρόγραμμα που έχει μεταγλωττιστεί, άρα έχει μετατραπεί σε κώδικα μηχανής, "τρέχει" πιο γρήγορα κατά το runtime, αλλά έχει το μειονέκτημα ότι αν θέλουμε να κάνουμε αλλαγές στο πρόγραμμα, πρέπει να ανατρέξουμε πάλι στον πηγαίο κώδικα και να τον μεταγλωττίσουμε ξανά.

Differences between Interpreter and Compiler	
Interpreter translates just one statement of the program at a time into machine code.	Compiler scans the entire program and translates the whole of it into machine code at once.
An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.
Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.
Interpreters are used by programming languages like Ruby and Python for example.	Compliers are used by programming languages like C and C++ for example.

Figure 83: Διαφορές μεταξύ μεταφραστή και μεταγλωτιστή.

Για να κληθεί ένα script που έχουμε γράψει από τη Unity κατά το runtime, θα πρέπει πρώτα να προστεθεί ως component σε κάποιο GameObject της σκηνής μας και φυσικά θα πρέπει να είναι γραμμένο σε μια συγκεκριμένη γλώσσα υψηλού επιπέδου, την οποία θα μπορεί να καταλάβει η συγκεκριμένη μηχανή παιχνιδιών, αφού πρώτα το script γίνει compiled και μετατραπεί σε γλώσσα μηχανής και εντολές που θα εκτελεστούν στην CPU. Γράφοντας πηγαίο κώδικα σε αυτή τη γλώσσα λοιπόν μπορούμε να επικοινωνήσουμε με τη μηχανή, να γράψουμε "σενάρια", tasks δηλαδή που εμείς έχουμε επιλέξει ως απαραίτητα για την εφαρμογής μας, τα οποία θα εκτελεστούν κυρίως στο main thread στην αρχή της εκτέλεσης της εφαρμογή μας ή σε άλλα threads που θα επιλέξει το main thread να δημιουργηθούν στη συνέχεια και να τρέχουν παράλληλα με το main, έτσι ώστε να μην έχουμε καθυστερήσεις (stalls) και να μην μπλοκαριστεί το main thread.
Η γλώσσα υψηλού επιπέδου που χρησιμοποιείται για επικοινωνία του χρήστη με τη Unity Engine ονομάζεται C# και είναι αντικειμενοστραφής scripting language. Όπως κάθε γλώσσα έτσι και οι scripting γλώσσες, έχουν τη δική τους σύνταξη και γραμματική καθώς επίσης τα κύρια μέρη τους είναι οι μεταβλητές, οι συναρτήσεις και οι κλάσεις.

- Οι *μεταβλητές* στη C# χρησιμοποιούνται για να αποθηκεύσουν πληροφορία και διατηρούν τιμές και αναφορές σε αντικείμενα. Μπορούμε να τις διαχωρίσουμε σε δύο τύπους, value type και reference type. Οι μεταβλητές οι οποίες είναι τύπου value περιέχουν ένα στιγμιότυπο του συγκεκριμένου type, ενώ οι τύπου reference περιέχουν μία αναφορά σε ένα στιγμιότυπο του συγκεκριμένου τύπου. Όταν καλείται μία συνάρτηση, οι τιμές των παραμέτρων της αποθηκεύονται σε μία περιοχή μνήμης δεσμευμένης ακριβώς γι' αυτόν το σκοπό. Τύποι που αποθήκευονται απευθείας όταν περνάμε παραμέτρους, ονομάζονται value types (structs της Unity, όπως Vector3 και Color, καθώς και ακέραιοι, δεκαδικοί και Booleans, είναι value types). Τύποι οι οποίοι δεσμεύονται στον σωρό (heap) και έχουν ως αποτέλεσμα memory allocations κατά το runtime (γεγονός που συνήθως θέλουμε να αποφύγουμε) είναι reference types, ενώ μπορούμε να έχουμε πρόσβαση σε αυτούς μέσω ενός δείκτη. Στιγμιότυπα μίας κλάσης, πίνακες (arrays), λίστες και συμβολοσειρές (strings) είναι reference types.
- Οι *συναρτήσεις* είναι ένας τρόπος να ομαδοποιήσουμε κώδικα που είναι υπεύθυνος για την εκτέλεση συγκεκριμένων tasks κατά το runtime και υπάρχει η σύμβαση να ξεκινούν με κεφαλαίο γράμμα. Είναι καλό να οργανώνουμε τον κώδικά μας σε συναρτήσεις έτσι ώστε αυτές να μπορούν να επαναχρησιμοποιηθούν πολλές φορές σε διαφορετικά μέρη του προγράμματος, χωρίς να χρειάζεται να ξαναγράψουμε όλο τον κώδικα της συνάρτησης, αλλά απλά να κάνοντας μία κλήση στο όνομα αυτής της συνάρτησης. Μπορούν να επιστρέφουν τιμές ή και όχι (type 'void' ως return type).
- Οι *κλάσεις* είναι ένας τρόπος να δομήσουμε τον κώδικά μας και να δημιουργήσουμε ισχυρές συσχετίσεις μεταξύ αντικειμένων και οντοτήτων, που ως έννοιες ομοιάζουν και περιστρέφονται γύρω από οντότητες του πραγματικού κόσμου. Η «κλάση» και το αντικείμενο αποτελούν βασικές έννοιες του οντοκεντρικού προγραμματισμού. Μία κλάση (base class) ορίζεται από τον χρήστη και αποτελεί ένα προσχέδιο για να δημιουργηθούν αντικείμενα αυτής της κλάσης ή άλλες κλάσεις που κληρονομούν από αυτή τα χαρακτηριστικά και τις μεθόδους της, αλλά είναι πιθανό να υλοποιούν επιπρόσθετες μεθόδους και να έχουν επιπρόσθετα και πιο εξειδικευμένα χαρακτηριστικά.

Στην Unity, τα scripts ουσιαστικά είναι Components που έχουν σκοπό να επεκτείνουν τις ήδη υπάρχουσες λειτουργίες της μηχανής και των αντικειμένων. Προστίθενται στα GameObjects και μπορούμε να χρησιμοποιήσουμε όσα θέλουμε για να ελέγξουμε τη φυσική συμπεριφορά των αντικειμένων, να προσθέσουμε ειδικές λειτουργίες για τον χαρακτήρα μας και την κίνηση του ή διάδραση με αντικείμενα, να δημιουργήσουμε εφέ και ακολουθίες γεγονότων, να δημιουργήσουμε score system ή γεωμετρία, συγκεκριμένα χαρακτηριστικά σε αντικείμενα και γενικά το τι μπορούμε να κάνουμε με αυτά, μπορεί να σταματήσει μόνο εκεί που φτάνει η φαντασία μας. Η MonoBehaviour είναι η base class από την οποία κληρονομούν όλα τα scripts της Unity. Κάποιες από τις πιο γνωστές μεθόδους που χρησιμοποιούνται στη Unity, είναι οι εξής:

- Η Awake() καλείται πριν την Start() στην έναρξη του runtime της εφαρμογής μας και θα κληθεί ακόμα κι αν το script component που την περιέχει είναι απενεργοποιημένο.
- Η Start(), όπως και η Awake, καλείται όταν ένα GameObject είναι ενεργό, αλλά μόνο εάν το script component που την περιέχει είναι ενεργοποιημένο. Χρονικά, η εκτέλεσή της ακολουθεί αυτήν της Awake() και αυτές οι δύο συναρτήσεις αποτελούν έναν βολικό τρόπο να χωρίσουμε τα start-up tasks της εφαρμογής ή της σκηνής παιχνιδιού μας, σε δύο διαδοχικά βήματα. Για να τρέξει η Start() θα πρέπει επιπλέον το script component που την περιέχει, να είναι ενεργοποιημένο.
- Η Update() καλείται μία φόρα σε κάθε frame. Αν για παράδειγμα, η εφαρμογή μας τρέχει στα 120 frames ανά δευτερόλεπτο, τότε η Update() θα τρέξει 120 φορές. Εδώ βάζουμε κώδικα για να ορίσουμε λογική που εκτελείται συνεχώς, π.χ όταν ο player controller δέχεται inputs ή για να δημιουργήσουμε ένα χρονόμετρο ή σε περίπτωση που μέρη του παιχνιδιού πρέπει να ενημερώνονται συνεχώς, σε κάθε frame. Είναι γενικά καλό για την απόδοση της εφαρμογής

μας, να μην εκτελούμε περίπλοκους υπολογισμούς μέσα στην Update(), σε κάθε frame δηλαδή, αλλά να ακολουθούμε μία περισσότερο event-based λογική και να εκτελούμε τέτοιου τύπου υπολογισμούς, μόνο όταν πραγματικά μας χρειάζεται ή μετά από συγκεκριμένο αριθμό frames (όχι σε κάθε frame).

- Η FixedUpdate() καλείται σε συγκεκριμένες χρονικές στιγμές, σε προκαθορισμένα-σταθερά χρονικά διαστήματα, αντίθετα με την Update() που καλείται σε κάθε frame (τα frames συνήθως κατά το runtime, δεν είναι σταθερά, οπότε ούτε οι χρονικές αποστάσεις μεταξύ των στιγμών που καλείται η Update()). Χρησιμοποιούμε συνήθως την FixedUpdate() για υπολογισμούς φυσικής, όταν απαιτείται ακρίβεια και δεν θέλουμε να έχουμε περίεργα αποτελέσματα. Το default fixed timestep της Unity ισούται με 0.02, κάτι που σημαίνει ότι η FixedUpdate() εκτελείται ακριβώς 50 φορές το δευτερόλεπτο και μπορούμε να αυξήσουμε ή να μειώσουμε αυτή την τιμή από τις ρυθμίσεις Time των Project settings, αν θέλουμε λιγότερη ή περισσότερη, αντίστοιχα, ακρίβεια στους υπολογισμούς μας.
- Η LateUpdate() καλείται και αυτή σε κάθε frame, με τη διαφορά ότι εκτελείται αφού έχουν τελειώσει οι υπολογισμοί της Update. Είναι ουσιαστικά ένας δεύτερος βρόχος που μας είναι ιδιαίτερα χρήσιμος. Χρονικά, όταν ξεκινάει μια σκηνή στη Unity, εκτελούνται πρώτοι οι υπολογισμοί φυσικής, οπότε εκτελείται πρώτη η FixedUpdate(), ακολουθεί η Update() και στο τέλος του frame και μετά την Update(), η LateUpdate().
- Η OnEnable() καλείται, όταν ενεργοποιούμε στην ιεραρχία της σκηνής μας, το gameObject που περιέχει script component με αυτή την συνάρτηση. Τρέχει πριν την Start() και την χρησιμοποιούμε για να εκτελέσουμε λογική, όταν ενεργοποιούμε το αντικείμενο στη σκηνή.
- Η OnDisable() καλείται, όταν απενεργοποιούμε στην ιεραρχία της σκηνής μας, το gameObject που περιέχει script component με αυτή τη συνάρτηση και τρέχει μετά την LateUpdate(). Την χρησιμοποιούμε για να εκτελέσουμε λογική, όταν απενεργοποιούμε το αντικείμενο στη σκηνή.

Για το παιχνίδι μας χρειάστηκε να γράψουμε αρκετά scripts σε C# για τη δημιουργία ειδικών εφέ, για πυροδότηση animations, για να δημιουργήσουμε level loaders, για την λειτουργικότητα του μενού διεπαφής χρήστη, για την αναπαραγωγή των ήχων και της μουσικής, για τα mechanics των γρίφων και για άλλες ειδικές λειτουργίες.

Κεφάλαιο 5: Περιγραφή παιχνιδιού

5.1 Πίστες και βασικά mechanics του παιχνιδιού

Το παιχνίδι αποτελείται από πέντε διαφορετικές πίστες/τοποθεσίες με τα δικά τους χαρακτηριστικά, περιβάλλοντα και γρίφους. Για να έχει τη δυνατότητα πρόσβασης στο επόμενο επίπεδο, ο παίκτης θα πρέπει πρώτα προχωρήσει στην επίλυση γρίφων και να βρει τον τρόπο να συνεχίσει, περνώντας από ξεχωριστές δοκιμασίες. Έχουν χρησιμοποιηθεί οι απαραίτητοι colliders, έτσι ώστε ο χαρακτήρας να μην μπορεί να περάσει μέσα από τοίχους ή από μεγάλα σε μέγεθος αντικείμενα, κάτι που προσδίδει ρεαλισμό στην εμπειρία και μιμείται την υλικότητα του ανθρώπινου σώματος και πως αυτό διαδρά ή συγκρούεται με τα όρια του εκάστοτε χώρου.

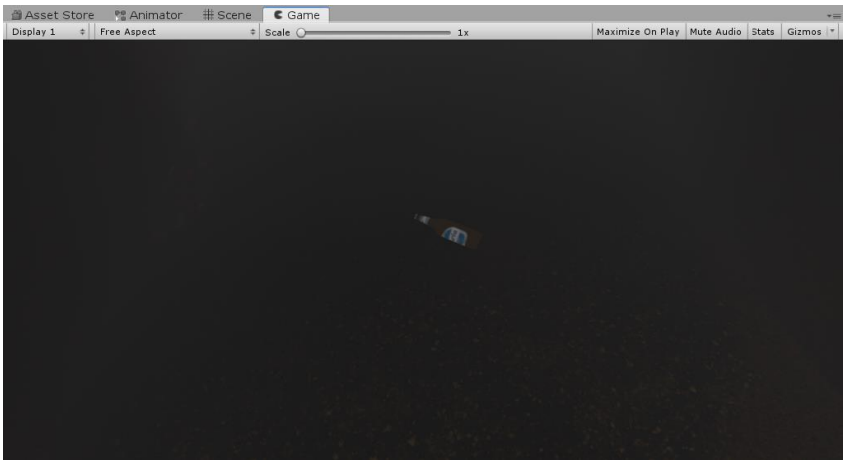


Figure 84: A beer bottle (simple interactable object).

Ο παίκτης έχει τη δυνατότητα βηματισμού απλού ή πιο γρήγορου, καθώς και διάδρασης με αλληλεπιδράσιμα αντικείμενα και έγγραφα που βρίσκονται διάσπαρτα στο περιβάλλον και από τα οποία αντλεί στοιχεία για επίλυση των γρίφων ή απλά μπορεί να τα παρατηρήσει πιο ενδελεχώς, λειτουργικότητα που συμβάλλει στην εμβύθιση του παίκτη στην εμπειρία. Η διάδραση με τα αλληλεπιδράσιμα αντικείμενα (interactable objects) περιλαμβάνει εξέταση των αντικειμένων με μορφή περιστροφής και zoom in/zoom out αυτών, καθώς και εύρεση ειδικών σημείων (inspect points) που δίνουν το έναυσμα σε animations και άλλες δυνατότητες. Ο παίκτης γνωρίζει αν ένα αντικείμενο είναι αλληλεπιδράσιμο, όταν κεντράρει το "βλέμμα" της κάμερας πρώτου προσώπου προς αυτό από κοντινή απόσταση (απόστασης ακτίνας – raycast): αν το αντικείμενο φωτιστεί, αυτό σημαίνει πως με το αριστερό κλικ του ποντικιού μπορεί να εκκινήσει η λειτουργία εξέτασης (examine mode) γι' αυτό το αντικείμενο.

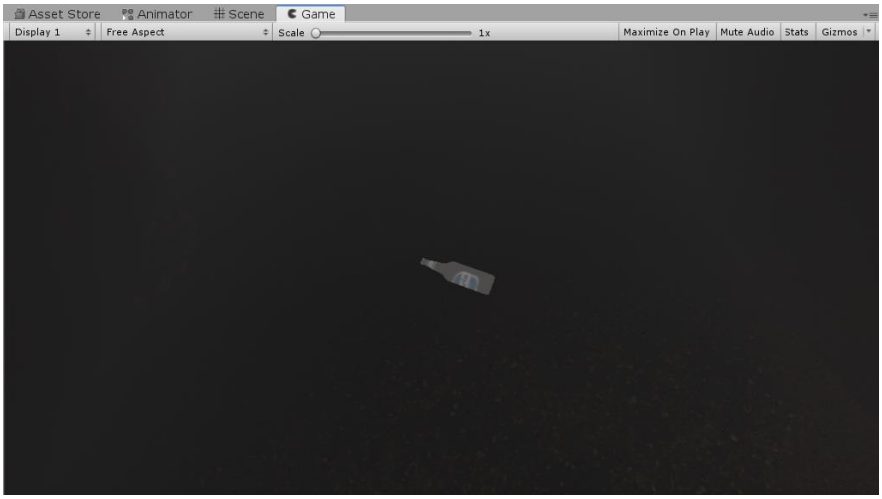


Figure 85: The beer bottle gets highlighted when player centers his camera view from a short distance using the invisible ray cast from player's camera that hits bottle's collider.

Υπάρχει επίσης η δυνατότητα συλλογής συγκεκριμένων αντικειμένων/κλειδιών που ξεκλειδώνουν συγκεκριμένες πόρτες ή ανοίγουν κλειδωμένα κουτιά που περιέχουν επί μέρους στοιχεία στο εσωτερικό τους, χρήσιμα για την πρόοδο στην ιστορία του παιχνιδιού και τα επόμενα επίπεδα. Διάσπαρτες στην ιστορία υπάρχουν επίσης συσκευές ανάγνωσης κωδικών/πληκτρολόγια που δέχονται συγκεκριμένους κωδικούς, κλειδαριές που δέχονται συγκεκριμένους συνδυασμούς αριθμών/κωδικούς για να ξεκλειδώσουν, συστήματα μοχλών που τίθενται σε λειτουργία μόνο με συγκεκριμένες ακολουθίες κινήσεων, καθώς και ηλεκτρικοί πίνακες των οποίων κάποιες ασφάλειες λείπουν και πρέπει ο παίκτης να τις περισυλλέξει από το περιβάλλον για να τεθούν σε λειτουργία μηχανισμοί χρήσιμοι για την μετάβαση στη συνέχεια της ιστορίας. Ο παίκτης μπορεί ανά πάσα στιγμή να προκαλέσει την παύση του παιχνιδιού, έτσι ώστε να έχει πρόσβαση στο μενού των επιλογών, στο αρχικό μενού ή στην έξοδο από την εφαρμογή πάλι στο περιβάλλον των Windows. Στην αρχή κάθε νέου επιπέδου, η πρόοδος του παίκτη αποθηκεύεται αυτόματα και ο παίκτης έχει τη δυνατότητα να συνεχίσει από το τελευταίο επίπεδο στο οποίο είχε πρόσβαση πριν την έξοδο από την εφαρμογή.

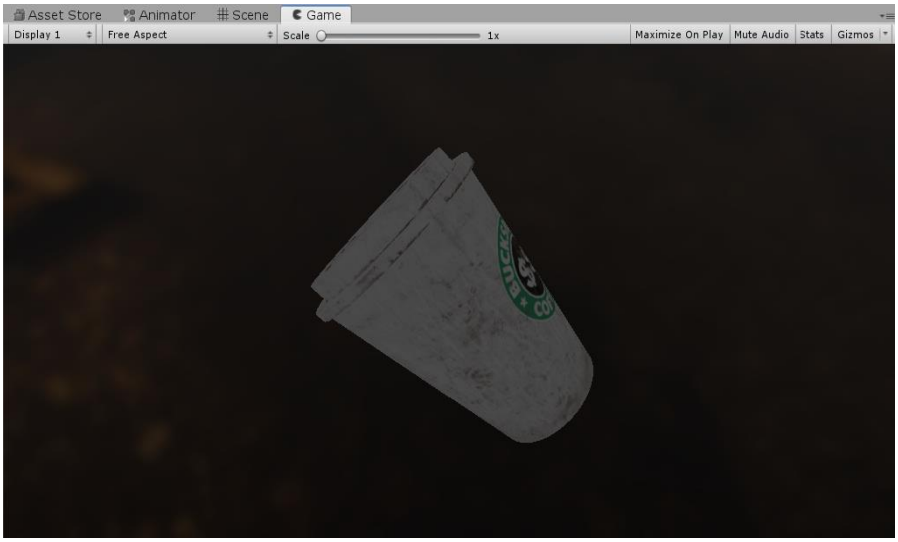


Figure 86: A plastic cup during Examine Mode.

5.2 Γρίφοι

Για να αντιμετωπίσει τις προκλήσεις που εμφανίζονται στην πορεία του παιχνιδιού ο παίκτης, αλλά και για να οδηγηθεί στην επίλυση των γρίφων [6], θα πρέπει είτε να αντλήσει στοιχεία από το περιβάλλον και να βρει το βαθύτερο νόημα που κρύβεται πίσω από τα ποιήματα που βρίσκονται διάσπαρτα σε έγγραφα, έτσι ώστε να ανακαλύψει τους μοναδικούς κωδικούς/αλληλουχίες αριθμών ή να βρεί τα απαραίτητα αντικείμενα/κλειδιά που μπορεί να περισυλλέξει και να προσθέσει στο inventory του, για να ξεκλειδώσει συγκεκριμένες πόρτες/κουτιά, να προσθέσει ασφάλειες που λείπουν από ηλεκτρικούς πίνακες ή να κινήσει μοχλούς διαδοχικά. Κάποια στοιχεία/έγγραφα/αντικείμενα δεν είναι διαθέσιμα στην πρώτη επαφή του παίκτη με την εκάστοτε πίστα [5], αλλά εμφανίζονται αφού πρώτα ο παίκτης εξερευνήσει αρκετά καλά το περιβάλλον και προβεί σε κάποιες απαραίτητες ενέργειες.

Πίστα 1

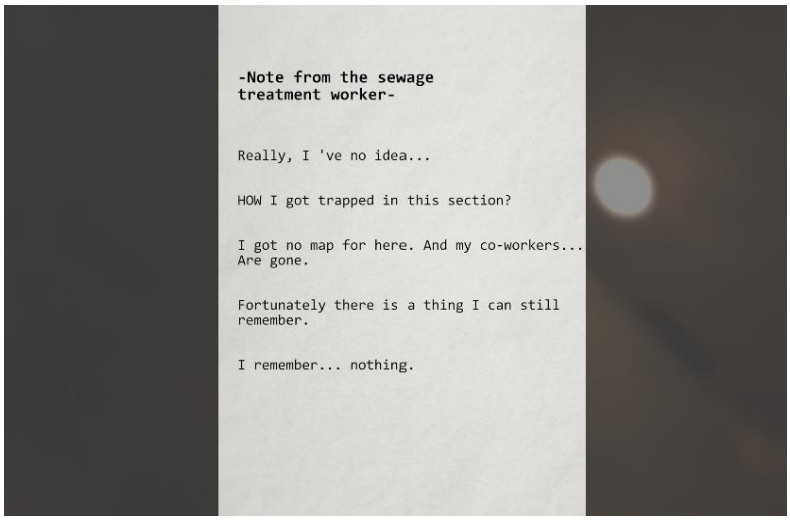


Figure 87: A note.

Figure 88: A keypad.

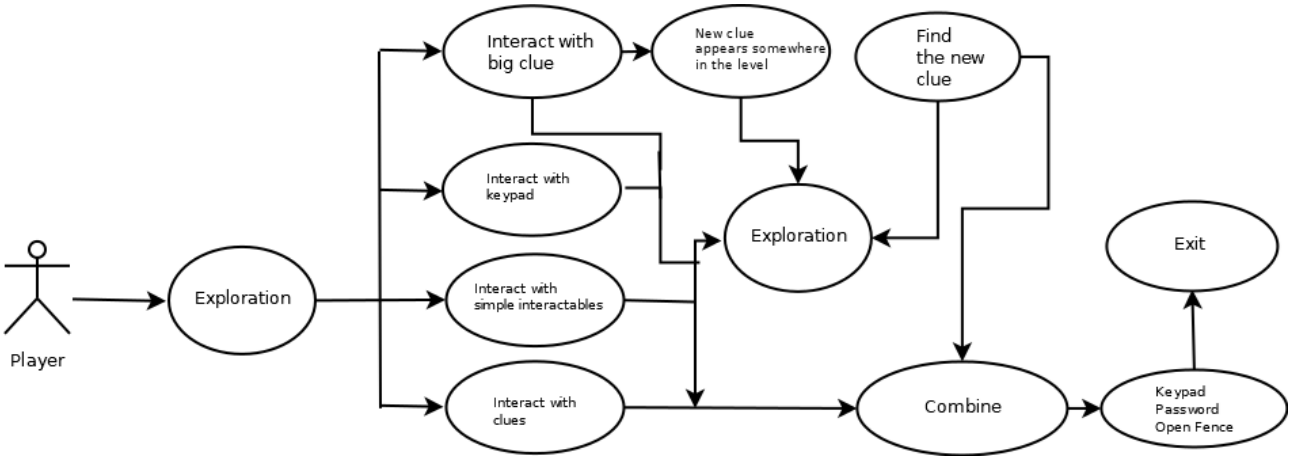


Figure 89: Level 1 Use Diagram

Το Level 1 αποτελεί ουσιαστικά την πρώτη επαφή του παίκτη με το περιβάλλον και τους γρίφους του παιχνιδιού. Ο χώρος είναι σχετικά μικρός σε μέγεθος, αν συγκριθεί με τις υπόλοιπες πίστες, ενώ ο γρίφος του στηρίζεται στην εξερεύνηση κι εξεύρεση και τον συνδυασμό στοιχείων, έτσι ώστε ο βρεθεί ο μοναδικός κωδικός που ξεκλειδώνει την επόμενη πίστα του παιχνιδιού.

Figure 90: A locked green door

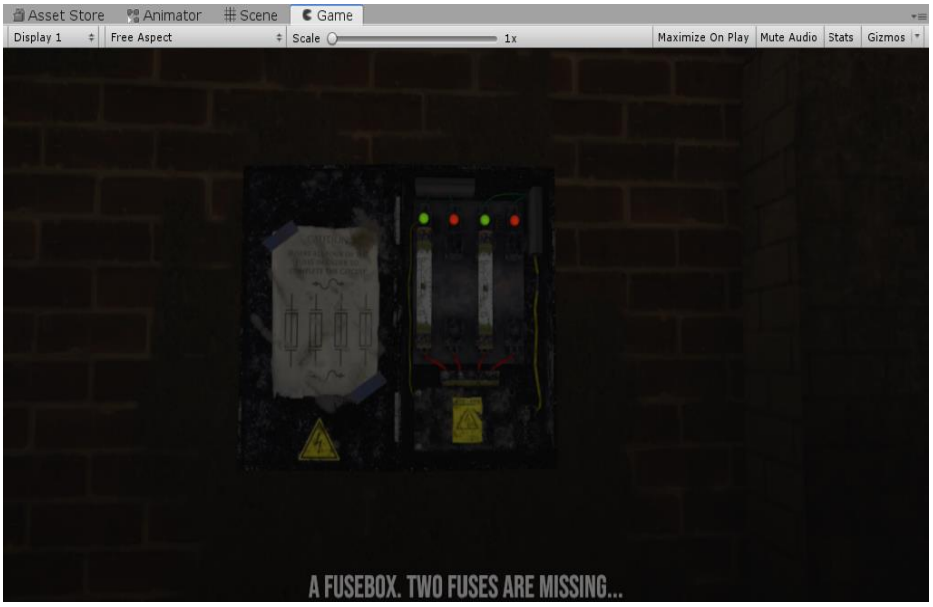


Figure 91: A fusebox with two fuses missing.

Figure 92: A fuse (collectable)

Figure 93: Fusebox with one fuse missing

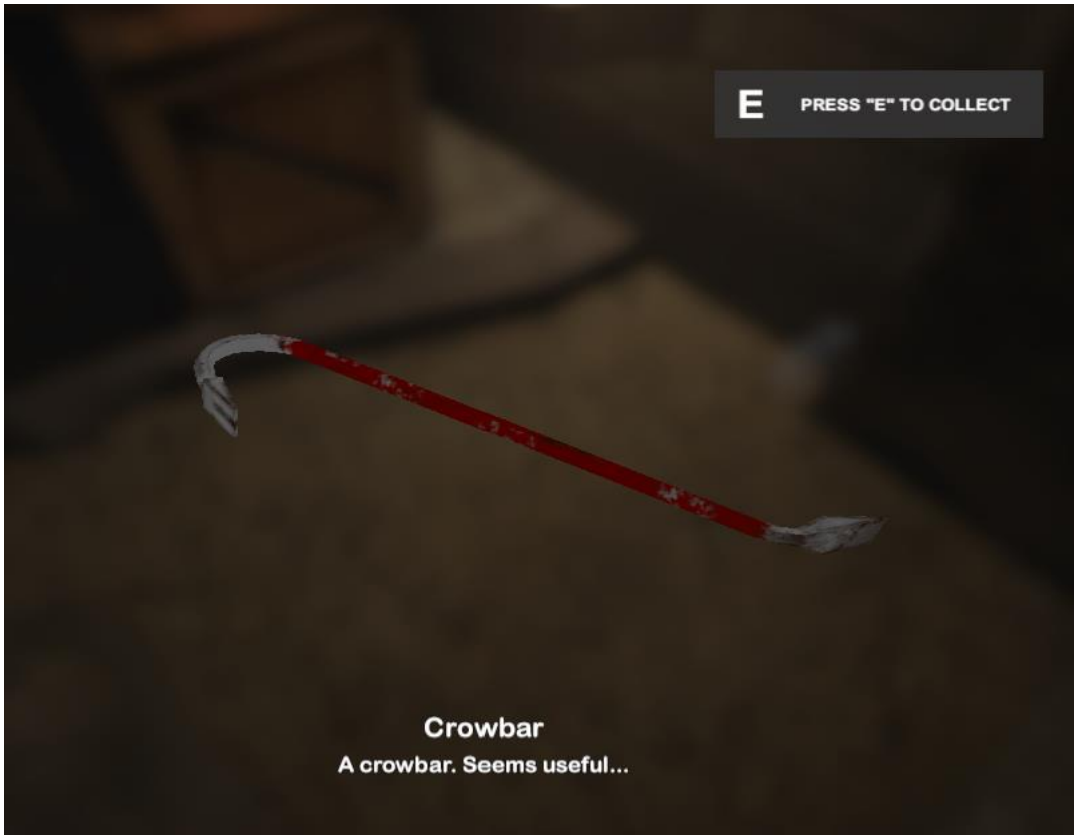


Figure 94: A collectable crowbar (key).

Figure 95: Crowbar collected (inventory).

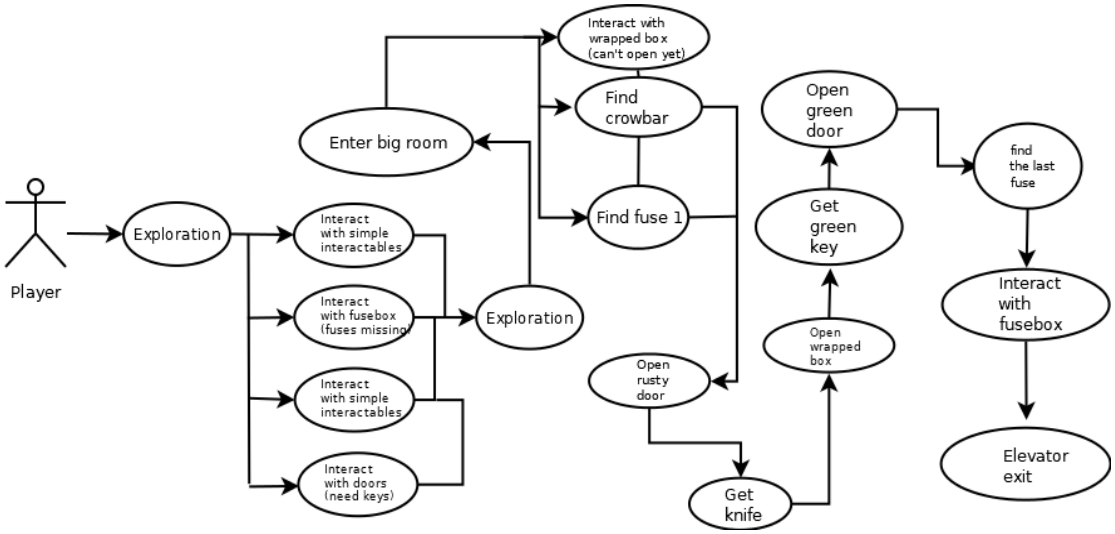


Figure 96: Level 2 Use case diagram

Το Level 2 αποτελείται από πολύ μεγαλύτερα τμήματα και χώρους που ο παίκτης θα πρέπει να εξερευνήσει για αρκετά μεγαλύτερο χρονικό διάστημα σε σχέση με την πρώτη πίστα, έτσι ώστε να βρει το σωστό μονοπάτι. Το σημείο εκκίνησης στην πίστα, είναι ένα σταυροδρόμι με οδούς και χώρους, στους οποίους η πρόσβαση δεν είναι αρχικά δυνατή. Οπτικά μηνύματα κατά την επαφή του παίκτη με κλειδωμένες πόρτες με την χρήση raycast και UI Canvas animations, δίνουν στοιχεία για τα πιθανά μοναδικά "κλειδιά", interactable αντικείμενα δηλαδή που πρέπει να περισυλλεχθούν, έτσι ώστε να ξεκλειδωθούν οι πόρτες και να βρεθεί ουσιαστικά η λύση στον κεντρικό γρίφο για να συνεχιστεί η ιστορία του παιχνιδιού προς την επόμενη πίστα. Ο παίκτης θα πρέπει αναγκαστικά να επισκεφθεί τους ίδιους χώρους για παραπάνω από μία φορές και επιπλέον να συλλέξει τα fuses, έτσι ώστε μέσω του ηλεκτρικού πίνακα, να θέσει πάλι σε λειτουργία τον ανελκυστήρα στο τελικό δωμάτιο που οδηγεί στην επόμενη πίστα.

Πίστα 3

Figure 97: Level 3 - Trapped

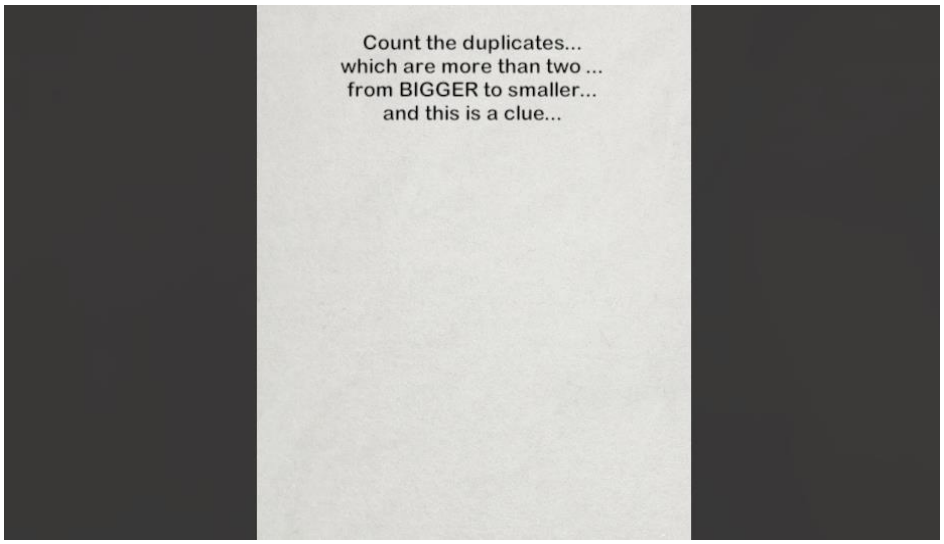


Figure 98: Level 3 poem (clue)

Figure 99: Levers in Level 3

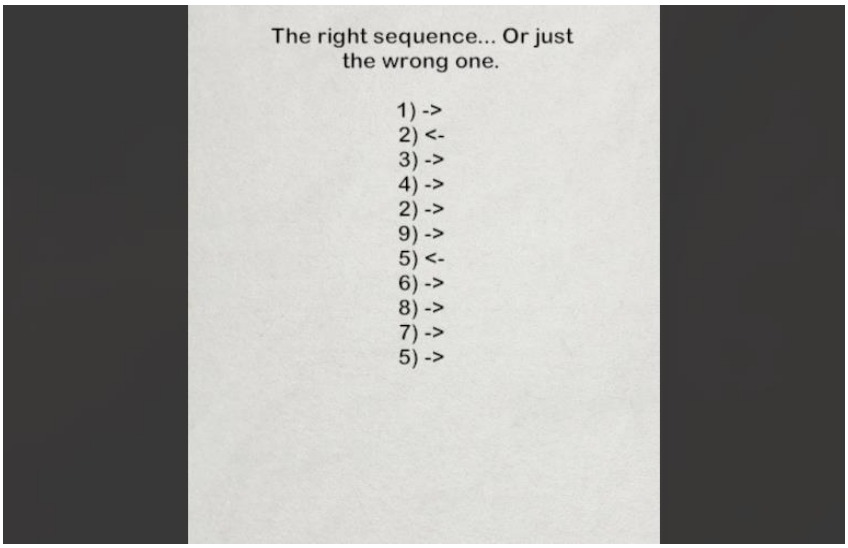


Figure 100: Another clue note (puzzle) in Level 3

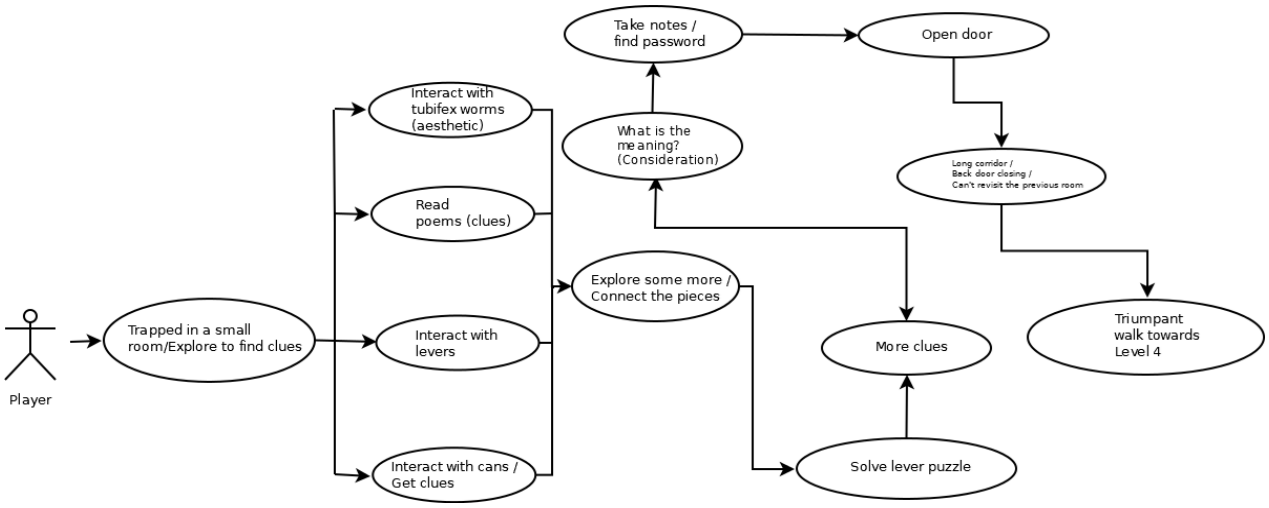


Figure 101: Level 3 use case diagram

Μετά το animation του ανελκυστήρα ο παίκτης βρίσκεται ξαφνικά παγιδευμένος και περιορισμένος σε ένα μικρό δωμάτιο. Με συνεχή εξερεύνηση και τα ελάχιστα στοιχεία που του δίνονται, πρέπει να βρει το νόημα σε έναν από τους δυσκολότερους γρίφους [2] του παιχνιδιού (ίσως τον δυσκολότερο), ο οποίος χωρίζεται σε δύο στάδια. Η επαναλαμβανόμενη μουσική με timer δίνει τον απαραίτητο χώρο στον παίκτη για περισυλλογή και αναζήτηση στοιχείων στο περιβάλλον. Αρχικά, δεν προσφέρονται όλα τα απαραίτητα clues [2] για την επίλυση του τελικού γρίφου του δωματίου, οπότε ο παίκτης θα πρέπει να ξεπεράσει το εμπόδιο του πρώτου γρίφου [2] για να αποκτήσει πρόσβαση σε νέα στοιχεία που ίσως μπορούν να δώσουν λύση στο μυστήριο. Μετά το άνοιγμα της πόρτας του αρχικού δωματίου, ένας νέος μακρόστενος μεταβατικός διάδρομος ξετυλίγεται μπροστά και η πόρτα προς τον αρχικό χώρο κλείνει, μην επιτρέποντας έτσι στον παίκτη να τον ξαναεπισκεφθεί. Ο παίκτης έχει επιπλέον ορατότητα προς έναν “δίδυμο” μυστικό διάδρομο, με τη δυνατότητα να κοιτάξει μέσα από τα κενά που σχηματίζονται ανάμεσα στις περσίδες στην άκρη του διαδρόμου και οι κινήσεις του περιορίζονται μόνο προς τα εμπρός και την πόρτα στο βάθος που οδηγεί προς το Level 4.

Πίστα 4

Figure 102: Padlock interface

Figure 103: Locked middle door

Figure 104: Mysterious book (left), a name is missing from the last page (right)

Figure 105:"R" letter

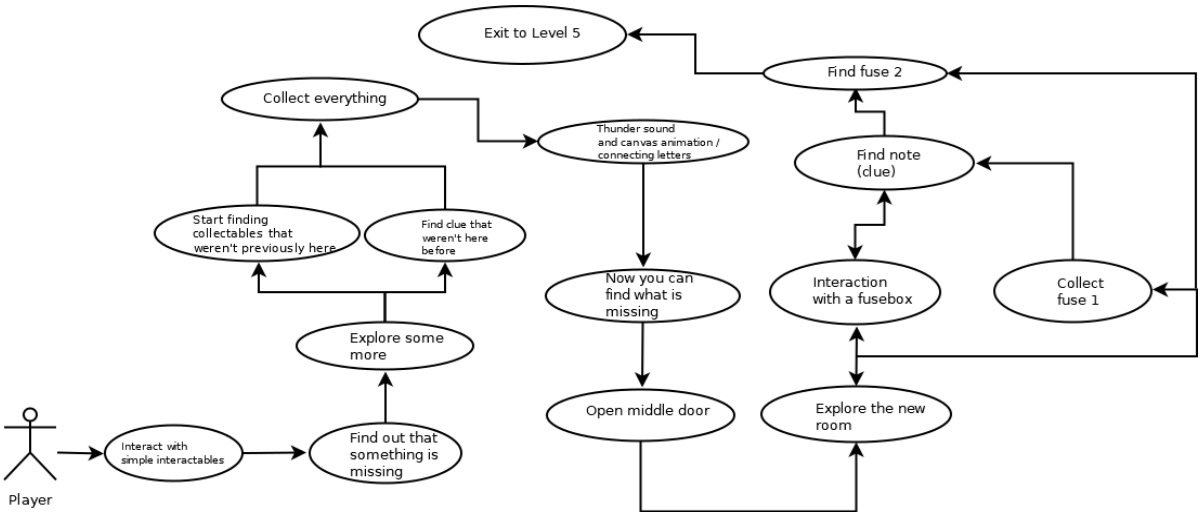


Figure 106: Level 4 use case diagram

Κατά την έναρξη της περιήγησης στο Level 4, ο παίκτης έχει πρόσβαση κυρίως σε απλά interactable αντικείμενα [5], τα οποία διαδραματίζουν ως επί το πλείστον αισθητικό ρόλο στο παιχνίδι. Απ' τη στιγμή που θα ανακαλύψει το κυρίαρχο σε αυτό το στάδιο interactable αντικείμενο και της συνειδητοποίησης ότι κάποιο στοιχείο λείπει, μέχρι πρότινος ανενεργά αντικείμενα της ιεραρχίας, καθίστανται πλέον ενεργά και ορατά στον παίκτη, διάσπαρτα στο περιβάλλον της πίστας και στόχος πλέον είναι να συλλεχθούν όλα, με μόνο "όπλο" του διαρκή αναζήτηση και με απώτερο σκοπο να καταστεί δυνατός ο σχηματισμός της λέξης/απάντησης στο γρίφο και να δοθεί η δυνατότητα για πορεία του παίκτη προς το μέχρι πρότινος μη-προσβάσιμο [5] δωμάτιο. Μέχρι τότε θα πρέπει να έχει βρεθεί ο τρόπος να ξεκλειδωθεί το υπάρχον padlock με την βοήθεια κρυμμένων clues. Στη συνέχεια, ο παίκτης θα πρέπει να ανακαλύψει νέα στοιχεία που θα τον οδηγήσουν στη συλλογή fuses για την ενεργοποίηση μηχανισμού που δίνει πρόσβαση στο μεταβατικό μονοπάτι προς το Level 5.

Figure 107: The city on top

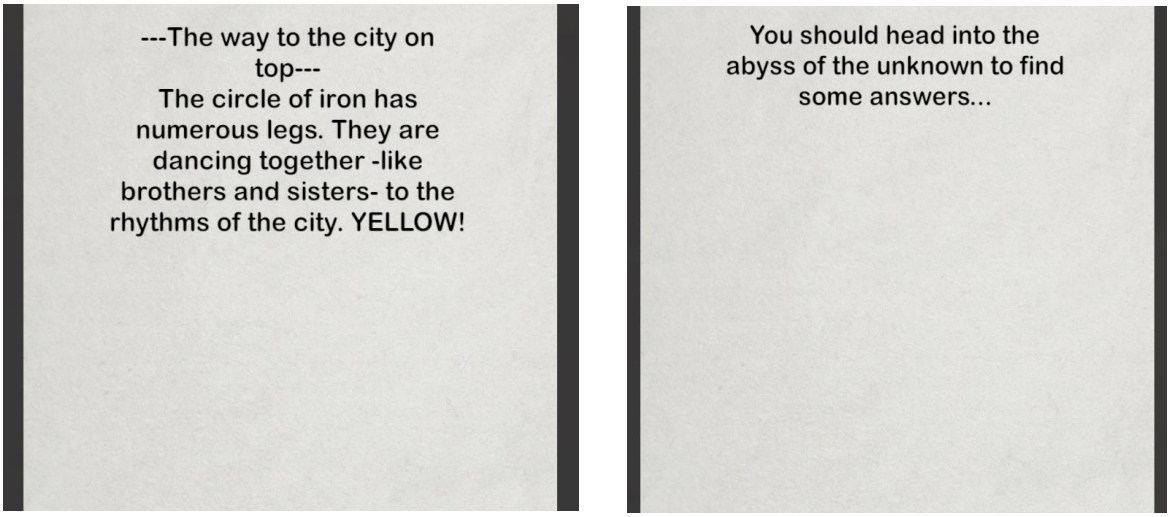


Figure 108: Allegorical poems/messages

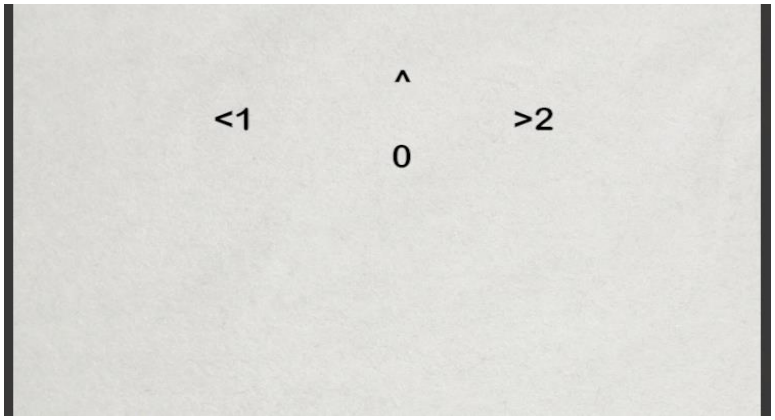


Figure 109: Weird numbers puzzle [6]

Figure 110: An old photo (simple interactable object)

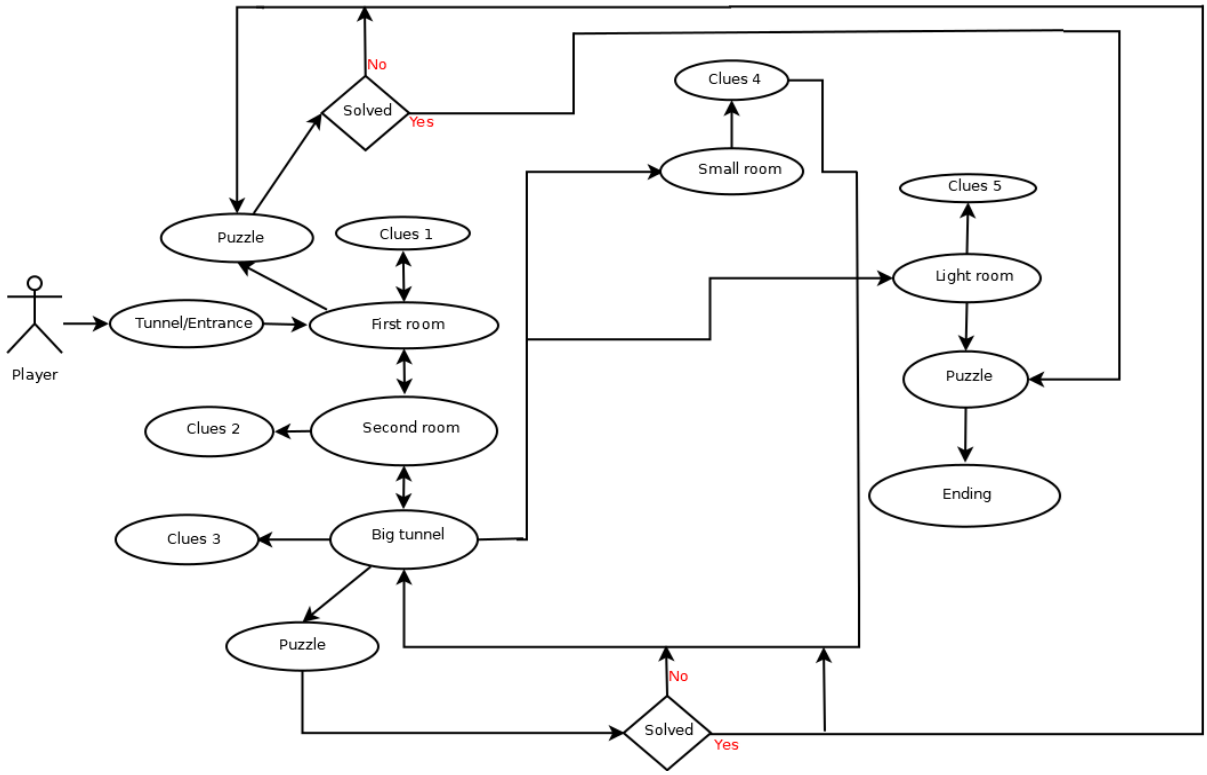


Figure 111: Level 5 use case diagram

Στη τελευταία πίστα του παιχνιδιού, για να οδηγηθεί ο παίκτης προς τον τερματισμό, θα πρέπει να αντιμετωπίσει την πρόκληση τριών διαφορετικών puzzles, στων οποίων την λύση θα πρέπει να προχωρήσει διαδοχικά, για να αποκτήσει αξιώσεις, ώστε να λύσει το κάθε επόμενο. Διαδοχική όμως δεν μπορεί να είναι η διαδρομή του εκάστοτε παίκτη στους χώρους της συγκεκριμένης πίστας. Τα διαφορετικά δωμάτια είναι αρκετά, όπως και τα στοιχεία που υπάρχουν διάσπαρτα στο περιβάλλον σε μορφή αλληγορικών ποιημάτων [6] (που μπορούν να περιπλέξουν την κατάσταση), τα οποία πρέπει να αποκρυπτογραφηθούν από τον παίκτη σε συνδυασμό με εξερεύνηση και ενδελεχή παρατήρηση στοιχείων του περιβάλλοντος, αλλά και με τη γνώση του τρόπου επίλυσης των γρίφων από τις προηγούμενες πίστες, οι οποίοι αναγκαστικά θα πρέπει να έχουν επιλυθεί, για να καταφέρει ο παίκτης να φτάσει μέχρι αυτή την πίστα. Ισχύουν φυσικά κι εδώ τα mechanics γρίφων από τις προηγούμενες πίστες του παιχνιδιού και μπορούν να γίνουν ορατά νέα στοιχεία μετά από την επίλυση, οπότε ο παίκτης θα πρέπει να εξερευνήσει αρκετές φορές τους χώρους, έτσι ώστε να συλλέξει και να ανακαλύψει ό,τι επιπλέον χρειάζεται για την εύρεση της λύσης και τον τερματισμό.

Figure 112: Pause menu.

Figure 113: Main menu.

Figure 114: Controls – Hints screen.

Figure 115: Οθόνη τερματισμού - credits.

Για τα mechanics των γρίφων και τη διάδραση με τα αντικείμενα, χρησιμοποιήσαμε το σύστημα Adventure Puzzle Kit [7] από το asset store της Unity και επεκτείναμε την λειτουργικότητά του με επεμβάσεις στον κώδικα, για να το φέρουμε στα μέτρα των αναγκών της υλοποίησης της εφαρμογής μας.

5.3 Κάμερα παιχνιδιού

Η κάμερα που χρησιμοποιήσαμε στο παιχνίδι είναι πρώτου προσώπου. Αυτό σημαίνει πως ο παίκτης βλέπει τον κόσμο μέσα από τα "μάτια" του ήρωα του παιχνιδιού. Η κάμερα είναι προσαρμοσμένη στο ύψος του κεφαλιού της αόρατης κάψουλας, η οποία συμβολίζει το σώμα του χαρακτήρα (ο capsule collider απαιτείται για την φυσική επαφή του παίκτη με το έδαφος και τους τοίχους), ενώ με το ποντίκι ο παίκτης έχει τη δυνατότητα να κοιτάξει προς όλες τις κατευθύνσεις, προσομοιώνοντας έτσι την συντονισμένη (ή και όχι) κίνηση του λαιμού και των ματιών, τον τρόπο δηλαδή με τον οποίο οι άνθρωποι

χρησιμοποιούν την όρασή τους για να αποκτήσουν οπτική αντίληψη του περιβάλλοντος.

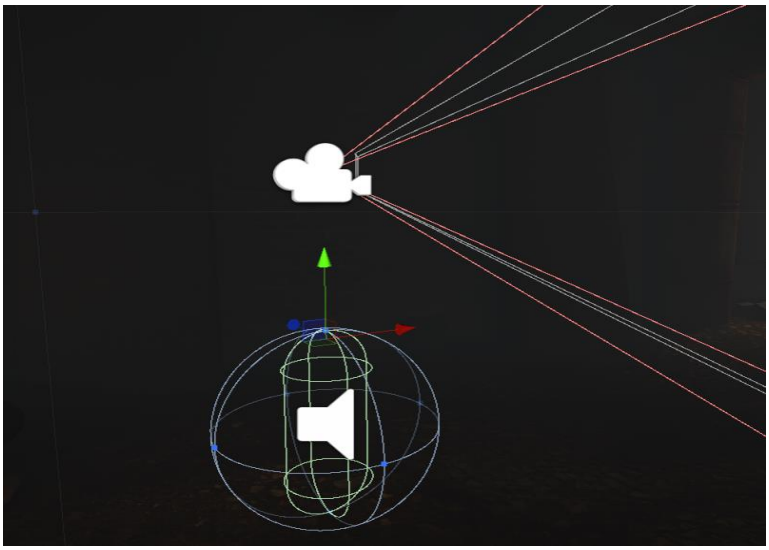


Figure 116: First person camera.

5.4 Το περιβάλλον υπονόμου

Μέχρι να καταλήξει στην τελική του μορφή το περιβάλλον του παιχνιδιού, πέρασε από αρκετά διαφορετικά στάδια και δέχθηκε αρκετές νέες προσθήκες και προεκτάσεις, αρκετοί από τους χώρους υπέστησαν επανεπεξεργασία, ενώ κάποιοι άλλοι σχεδιάστηκαν ξανά από την αρχή με περισσότερη λεπτομέρεια

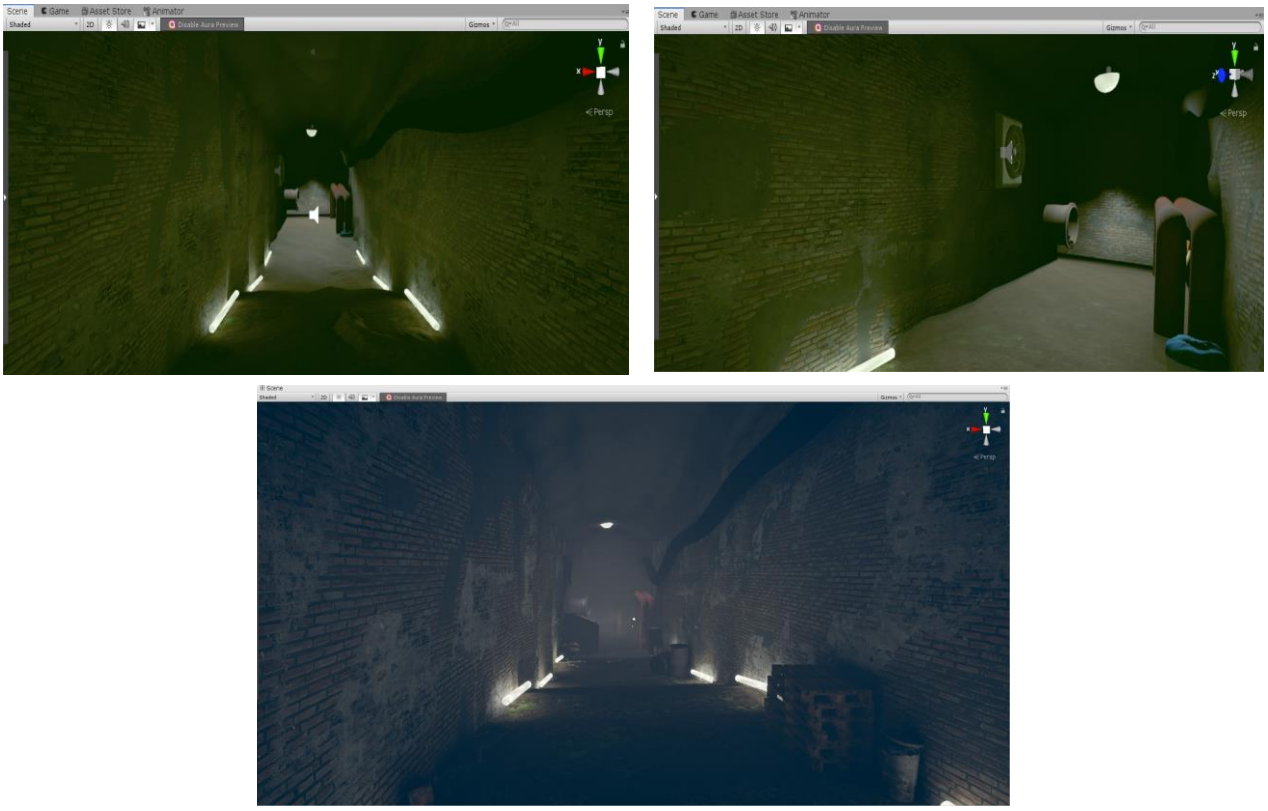


Figure 117: Old Level 1

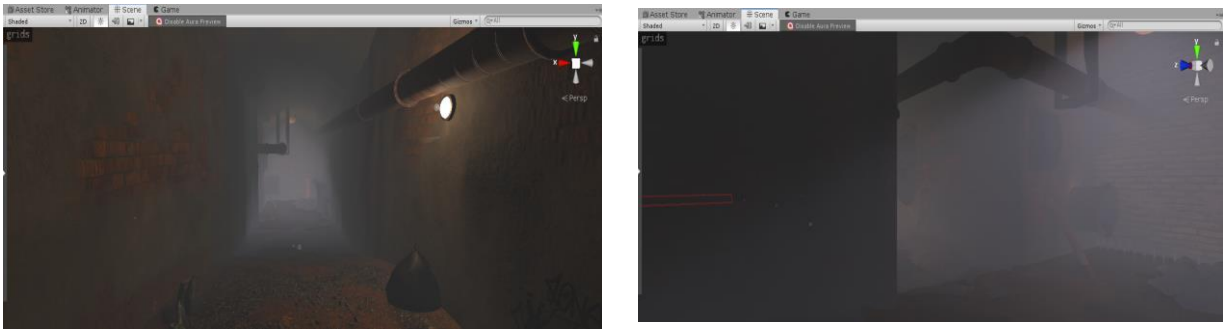


Figure 118: Final Level 1 with a new extension.

Figure 119: Level 1 top view

Κατά την ανάπτυξη του παιχνιδιού το workflow και οι εφαρμογές που χρησιμοποιούσαμε, άλλαξαν αρκετές φορές και ασφαλώς με την εκμάθηση νέων εργαλείων και τεχνικών καθ' όλο αυτό το διάστημα του development, προβήκαμε σε νέες δοκιμές, ανακατασκευές και βελτιστοποιήσεις όσον αφορά στη γεωμετρία των τρισδιάστατων μοντέλων ή ανέκυπταν προβλήματα των οποίων δεν υπήρχε η γνώση στο παρελθόν και έπρεπε να προχωρήσουμε στην επίλυσή τους. Επίσης, αναγκαστήκαμε να λάβουμε σχεδιαστικές αποφάσεις, οι οποίες έπρεπε να αλλάξουν σε μελλοντικό χρόνο, λόγω του ότι η όλη διαδικασία υλοποίησης ήταν τελικά μία καθαρά δυναμική διαδικασία [14]. Κάποιες πίστες, μπορεί για παράδειγμα, να ήταν μικρότερες σε μέγεθος απ' ό,τι επιθυμούσαμε και γι' αυτό προχωρούσαμε σε κάποιες προσθήκες ή μπορεί να ήταν αναγκαία η προσθήκη επιπλέον επιπέδων λεπτομέρειας κατά τη σχεδίαση, είτε μπορεί κάτι να μη μας ικανοποιούσε και γι' αυτό έπρεπε να κάνουμε export το μοντέλο με τον Fbx Exporter της Unity κι επανεπεξεργασία στο Blender και ξανά import στο Unity αρκετές φορές, για να φτάσουμε σε ένα επιθυμητό σχεδιαστικά κι αισθητικά αποτέλεσμα.

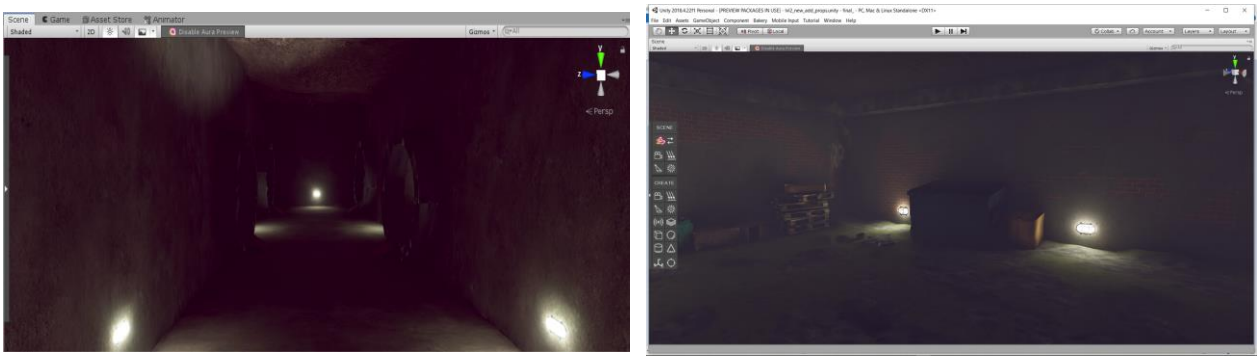


Figure 120: Old Level 2 crossroad (up-left) and small room (up-right)

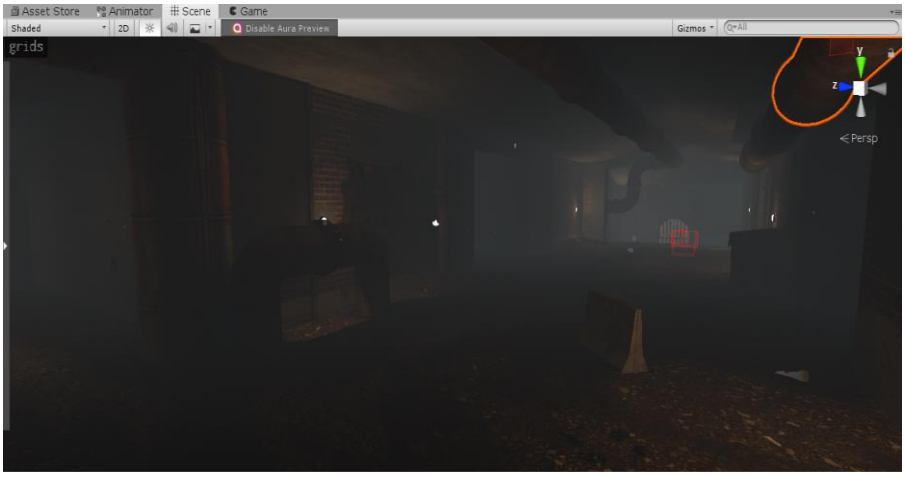


Figure 121: Final Level 2 crossroad remake (down).

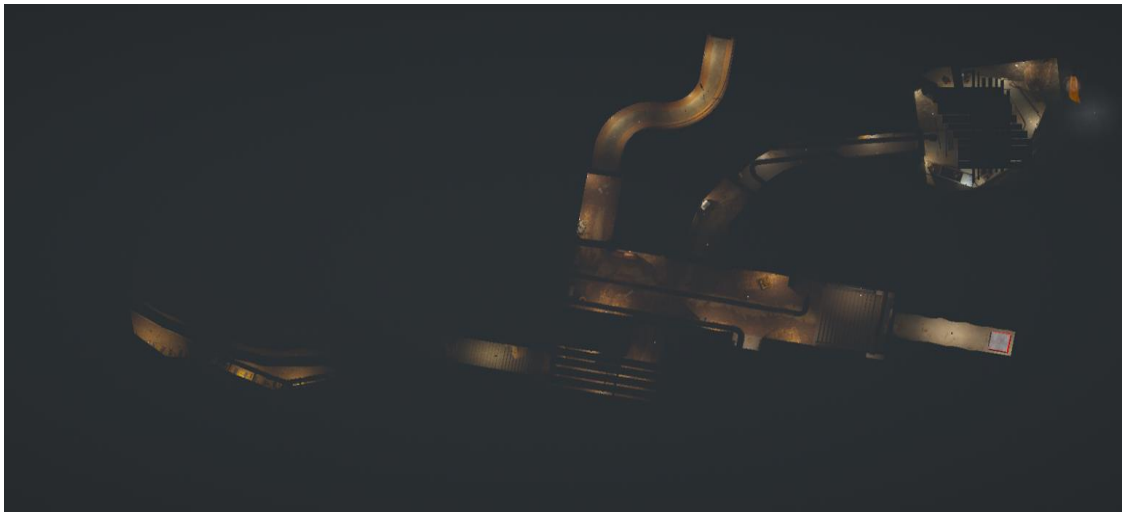


Figure 122: Level 2 top view

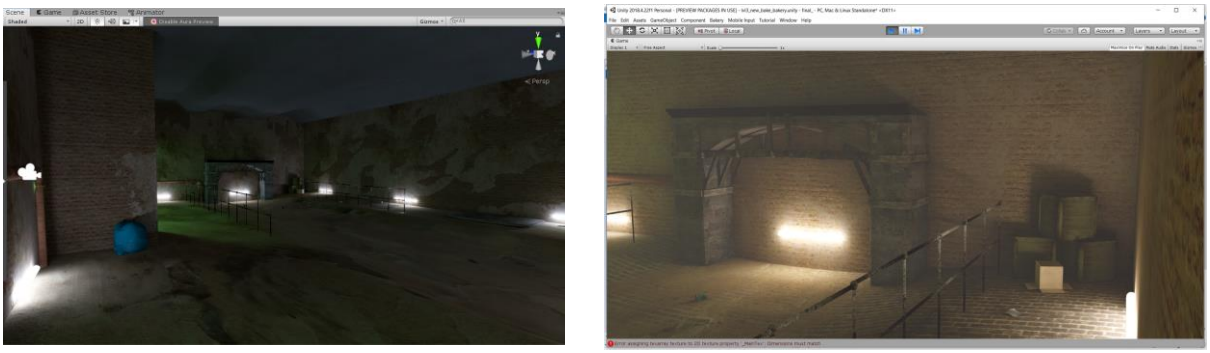


Figure 123: Old Level 3 part (up-left-right)

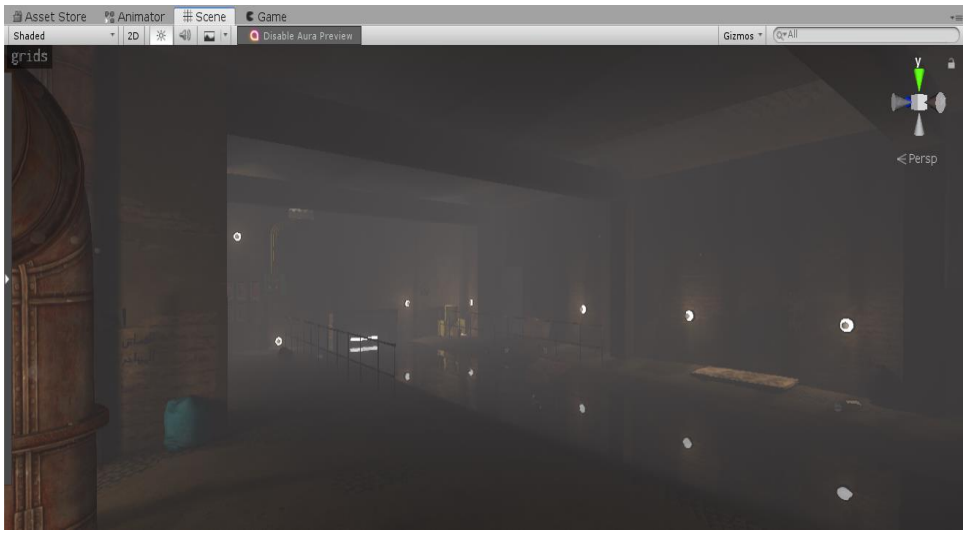


Figure 124: Final Level 3 with new details, rooms, extension and ceiling

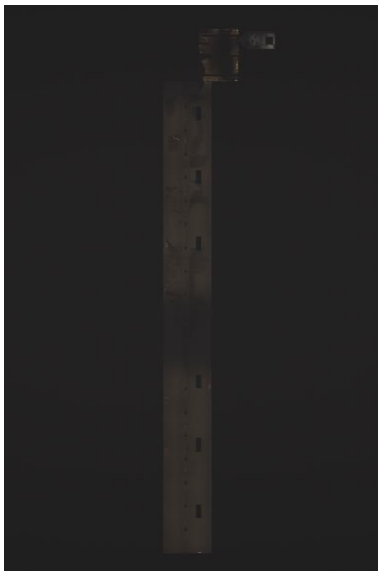


Figure 125: Level 3 top view with long extension

Στο Level 3 δεν υπήρχε αρχικά ο σχεδιασμός για τον μεγάλο διάδρομο που οδηγεί προς το Level 4, οπότε έπρεπε σχεδιαστικά να βρούμε μία λύση για τη μετάβαση από την μία πίστα στην άλλη. Χρησιμοποιήσαμε λοιπόν υλικό από το διαδίκτυο για concept art. Ούτως ή άλλως οι περισσότεροι χώροι του παιχνιδιού αποτελούν αντίγραφα ή η αρχιτεκτονική των χώρων έχει επηρεαστεί από

πραγματικές τοποθεσίες και προϋπάρχοντες χώρους πραγματικών υπονόμων που αναζητήσαμε για να προσδώσουμε ποικιλία στο περιβάλλον μας, αλλά και για να πάρουμε ιδέες για τη δημιουργία και το σχεδιασμό των χώρων του παιχνιδιού, έτσι ώστε να ξεφεύγουν από την στενή έννοια του υπονόμου που είχαμε πριν την κατάλληλη έρευνα.

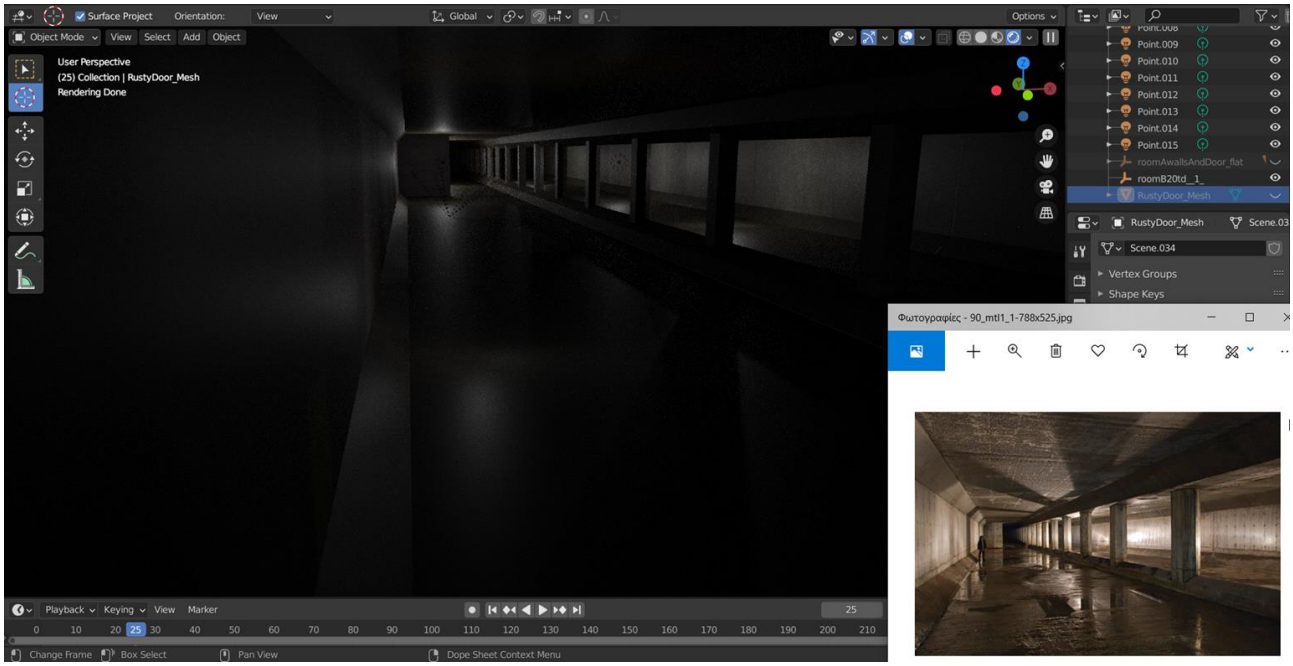


Figure 126: Level 3 long extension made with reference to a real location (via Sewer History org)

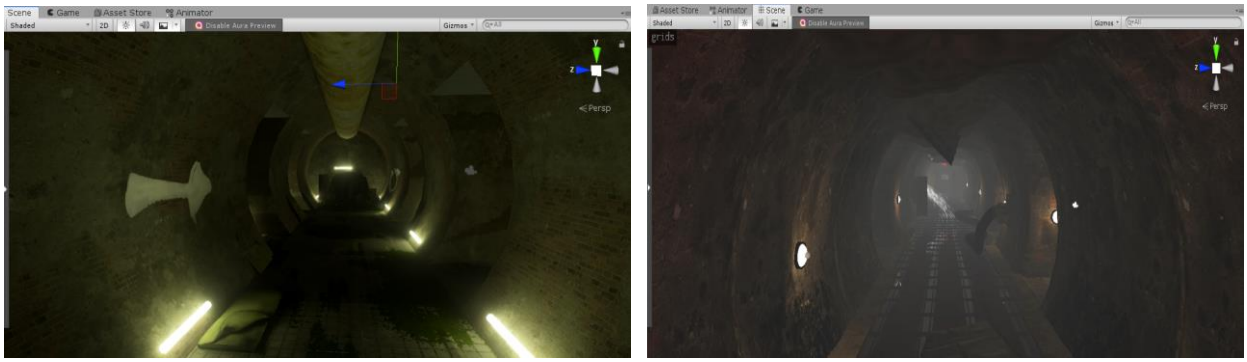


Figure 127: Part of Old Level 4 (up-left) vs. Part of the much longer Final Level 4 with a reworked floor, more details, props, fatberg ceiling and a new extension room (beyond the central door) (up-right).

Figure 128: Level 4 top view

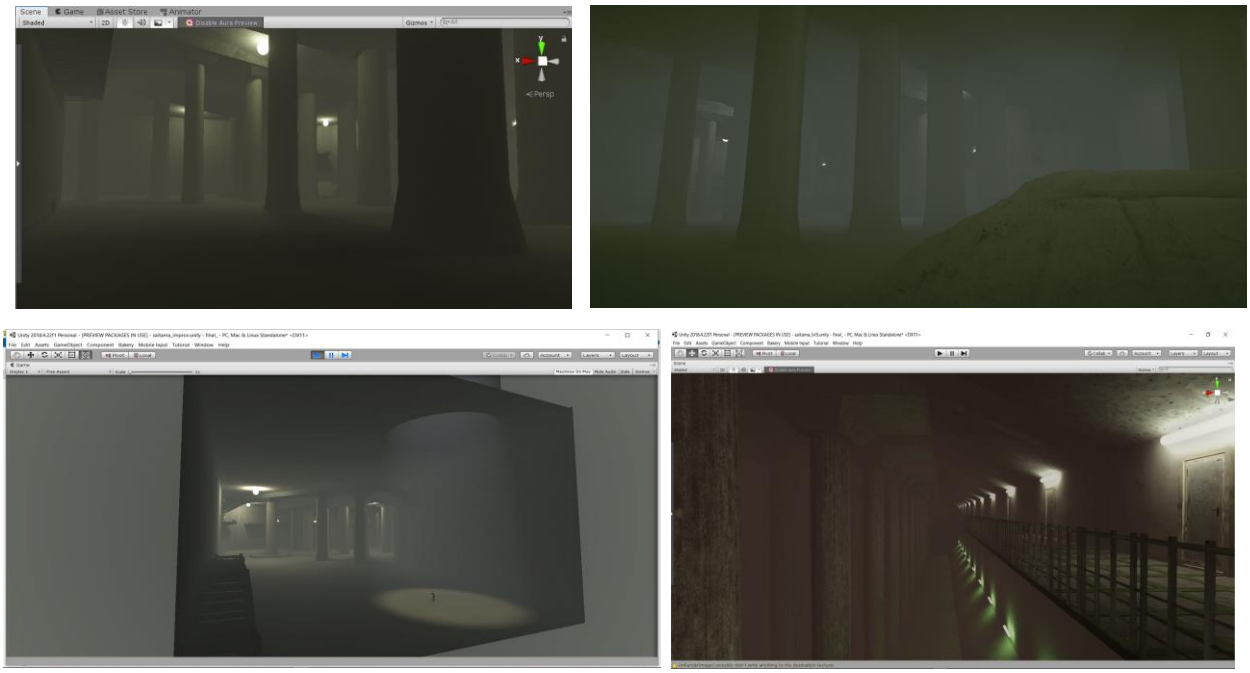


Figure 129: Old "Saitama Sewer[21] inspired" Level 5

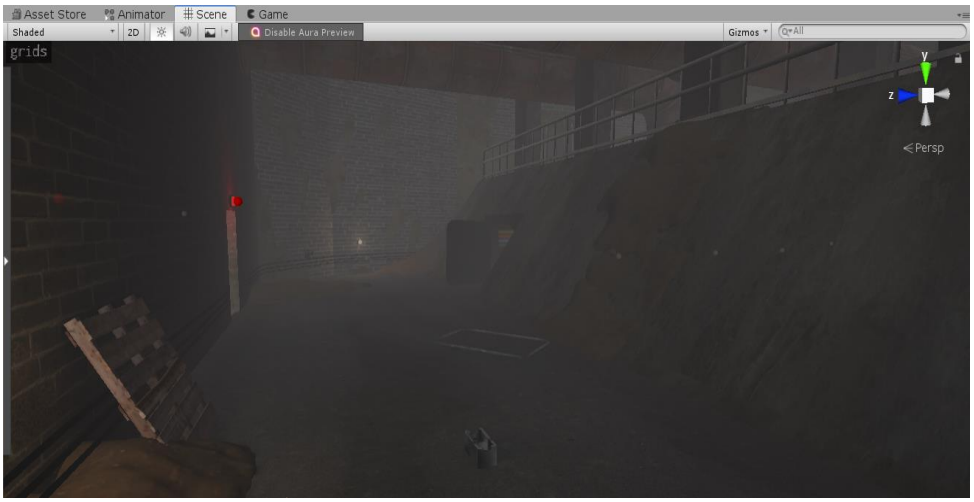


Figure 130: just a Room from the really bigger remade Final version of Level 5

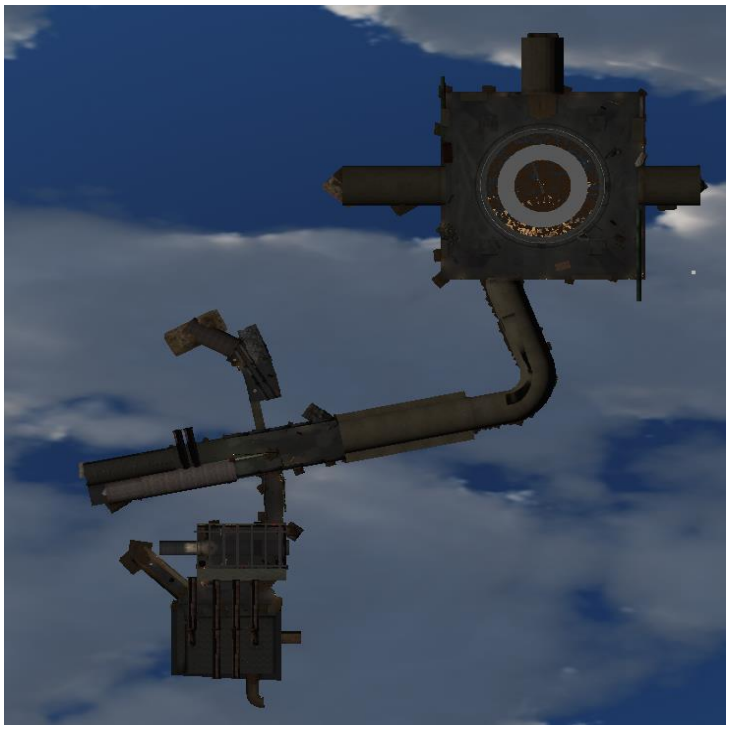


Figure 131: Level 5 top view

Εδώ πρέπει να σημειώσουμε ότι, για να έχουμε καλύτερη επίβλεψη του μεγάλου πλήθους από gameObjects και Prefabs που χρειάστηκε να χρησιμοποιήσουμε και για να έχουμε εύκολη πρόσβαση σε όλα τα ξεχωριστά αντικείμενα παιχνιδιού ανά πάσα στιγμή σε κάθε πίστα, ομαδοποιήσαμε τα αντικείμενα του κάθε scene στο Hierarchy Window.

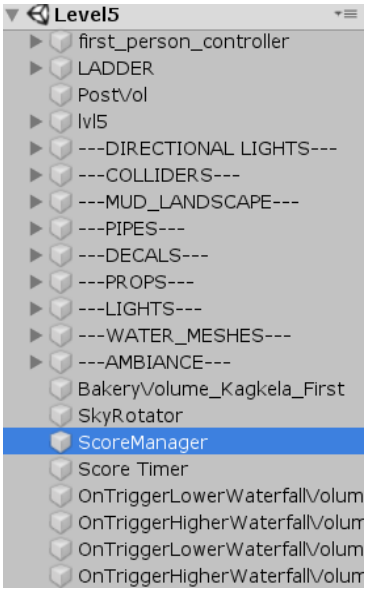


Figure 132: A typical "Drainage" level overview in the editor's hierarchy window

Κεφάλαιο 6: Υλοποίηση παιχνιδιού

Στο συγκεκριμένο κεφάλαιο θα αναφέρουμε κάποια στάδια της υλοποίησης του παιχνιδιού, προβλήματα που αντιμετωπίσαμε, καθώς επίσης θα παραθέσουμε κομμάτια κώδικα από scripts που υλοποιήσαμε σε C# για την ανάπτυξη της εφαρμογής και τις λειτουργικότητες που χρειαστήκαμε για τις ανάγκες του παιχνιδιού.

6.1 Flashing Light in Level 1

Καθώς στην έκδοση της Unity που χρησιμοποιήσαμε η υποστήριξη για realtime Emissive light από materials που χρησιμοποιούν τον Standard shader της Unity ήταν κάπως περιορισμένη, για να υλοποιήσουμε το ειδικό εφέ της λάμπας – πηγής φωτός που τρεμοπαίζει με συνοδεία από σπίθες την κατάλληλη στιγμή, υλοποιήσαμε ένα script και χρησιμοποιήσαμε ένα coroutine που το ξεκινήσαμε στην Start().

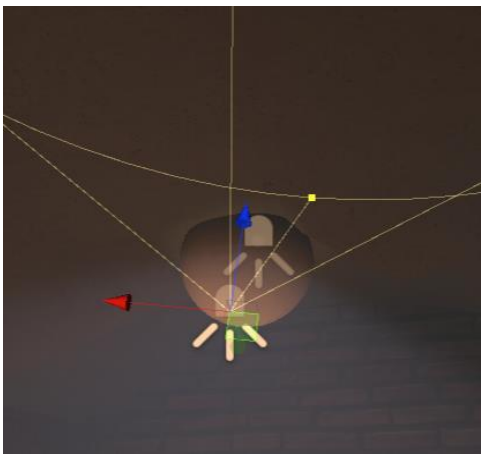


Figure 133: Spot Up Light

Επιθυμούσαμε όσο το Emissive material της λάμπας ήταν ενεργοποιημένο, καθώς δεν μπορούσαμε να έχουμε illumination από realtime emission και αυτό να έχει οπτικό αντίκρυσμα στα σκηνικά, να φωτίζεται και η περιοχή στην οροφή πάνω από τη λάμπα. Δημιουργήσαμε λοιπόν ένα spot Up Realtime Light με κατεύθυνση προς την οροφή. Δημιουργήσαμε επίσης ένα volumetric main Light (_auraLight) με κατεύθυνση προς το πάτωμα για τον κύριο φωτισμό προς τα σκηνικά κάτω από τη λάμπα.

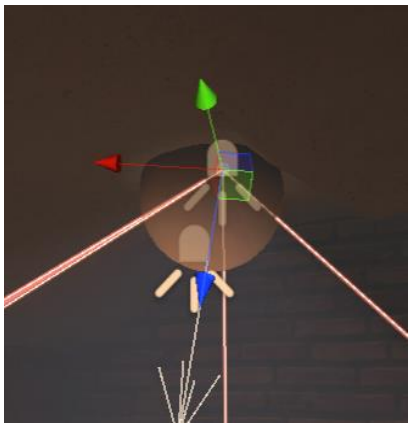


Figure 134: Main Volumetric Spot Light

Δημιουργήσαμε επίσης ένα Particle Effect με σπίθες που πέφτουν από την λάμπα προς το πάτωμα και πιο συγκεκριμένα μόνο όταν το emissive φως της λάμπας και τα φώτα ήταν disabled, έτσι ώστε λίγο μετά την αναπαραγωγή του εφέ σπίθας να ανάψει πάλι το φως. Επίσης χρησιμοποιήσαμε την έτοιμη συνάρτηση Random της Unity για να προσδώσουμε τυχαιότητα στις χρονικές στιγμές που θα τρεμοπαίξει το φως (άρα και το εφέ σπίθας). Ακολουθεί ο κώδικας:

```
void Start()
{
    testLight = GetComponent<Light>();
    _auraLight = GetComponent<AuraLight>();
    StartCoroutine(Flashing());
}
```

```
IEnumerator Flashing()
{
    while (true)
    {
        yield return new WaitForSeconds(0.1f);
        testLight.enabled = false;
        _auraLight.enabled = false;
        spotUpLight.enabled = false;
        material.DisableKeyword("_EMISSION");
        sparks.GetComponent<ParticleSystem>().Stop();
        yield return new WaitForSeconds(UnityEngine.Random.Range(minWaitTime, maxWaitTime));
        sparks.GetComponent<ParticleSystem>().Play();
        yield return new WaitForSeconds(0.36f);
        testLight.enabled = true;
        _auraLight.enabled = true;
        spotUpLight.enabled = true;
        material.EnableKeyword("_EMISSION");
    }
```

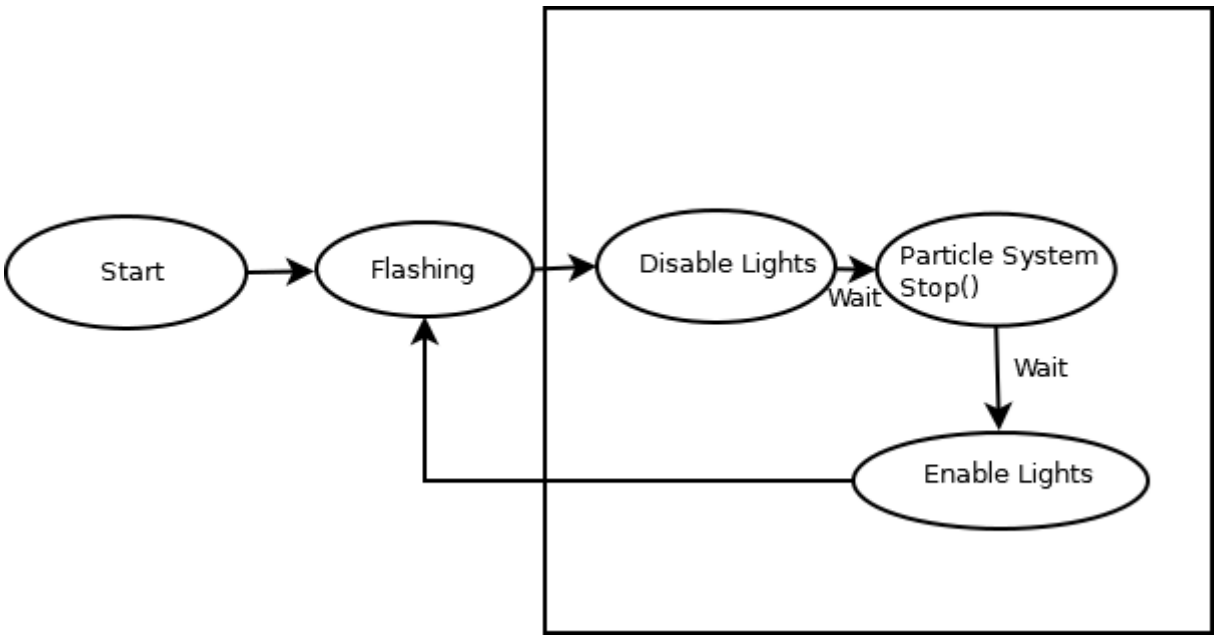


Figure 135: Visual representation of the loop we created with the Flashing() coroutine

6.2 Auto Save

Υλοποιήσαμε ένα απλό σύστημα για Auto-save της προόδου του παίκτη, η οποία αποθηκεύεται στην αρχή του κάθε level, έτσι ώστε ο παίκτης αφού επιλέξει Quit Game από το menu και πραγματοποιήσει έξοδο από την εφαρμογή, σε περίπτωση επανεκτέλεσής της να μπορεί να συνεχίσει από την πίστα στην οποίο βρισκόταν. Δημιουργήσαμε ένα trigger στην αρχή της κάθε πίστας που αποθηκεύει μέσω των **PlayerPrefs** το index της σκηνής, ενώ στη συνέχεια με χρήση ενός coroutine εμφανίζουμε ένα flashing Animated text "Saving" για να ενημερώσουμε τον παίκτη ότι αποθηκεύεται η πρόοδός του και μετά από λίγο να το απενεργοποιήσουμε μαζί με τον trigger, αφού αυτό που επιθυμούσαμε, δηλαδή η αποθήκευση προόδου και η οπτική ενημέρωση παίκτη για το γεγονός, έχει ήδη πραγματοποιηθεί.

```
public class TriggerAutoSave : MonoBehaviour
{
    private int currentSceneIndex;
    public GameObject savingTextAnimation;

    void OnTriggerEnter()
    {
        KeepCurrentSceneIndex();
    }
    public void KeepCurrentSceneIndex()
    {
        currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        PlayerPrefs.SetInt("SaveScene", currentSceneIndex);
        //SceneManager.LoadScene(0);
        StartCoroutine(AutoSaveTriggerDisappear());
    }
    IEnumerator AutoSaveTriggerDisappear()
    {
        yield return new WaitForSeconds(0.5f);
        Debug.Log("Saved scene's index is" + currentSceneIndex);
        savingTextAnimation.SetActive(true);
        yield return new WaitForSeconds(2f);
        savingTextAnimation.SetActive(false);
        yield return new WaitForSeconds(0.5f);
        gameObject.GetComponent<BoxCollider>().enabled = false;
        yield return null;

    }
}
```

Και για το Continue:

```
void Awake()
{
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
    savedSceneIndex = PlayerPrefs.GetInt("SaveScene");
}
```

και

```
AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(savedSceneIndex);
```

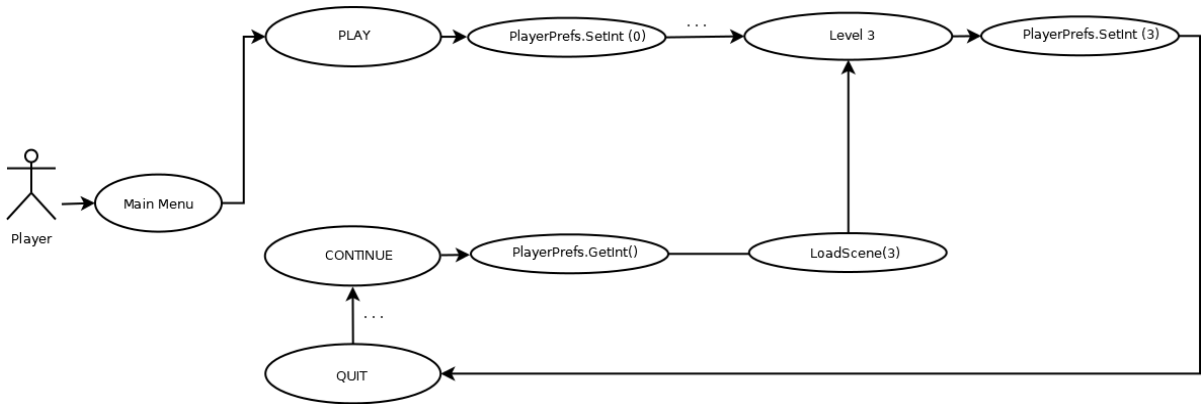


Figure 136: Visual representation of the auto-save procedure

6.3 Level Loader

```
public IEnumerator LoadNextLevel()
{
    loadingScreen.SetActive(true);

    AsyncOperation asyncLoad = SceneManager.LoadSceneAsync(nextLevel);

    asyncLoad.allowSceneActivation = false;

    while (!asyncLoad.isDone)
    {
        float progress = Mathf.Clamp01(asyncLoad.progress / 0.9f);
        slider.value = progress;

        if (asyncLoad.progress >= .9f)
        {
            loadingText.text = "Press Enter to continue";
            loadingIcon.SetActive(false);

            if (Input.GetKey(KeyCode.Return))
            {
                asyncLoad.allowSceneActivation = true;

                Time.timeScale = 1f;
            }
        }
        yield return null;
    }
}
```

Για την μετάβαση σε διαφορετικά Scenes-πίστες δημιουργήσαμε έναν Level Loader για asynchronous loading και χρησιμοποιήσαμε την συνάρτηση LoadSceneAsync του SceneManager της Unity, καθώς επιθυμούσαμε να έχουμε loading screen με progress bar που δείχνει την πρόοδο του Loading ενός nextLevel scene τύπου string που εισάγαμε στον inspector, να κάνουμε loading δηλαδή χωρίς την διακοπή του κύριου thread. Όταν το Scene έχει πια φορτώσει, ο παίκτης με το πλήκτρο Enter μπορεί να μεταβεί σε αυτό. Για κάθε πίστα δημιουργήσαμε ένα empty GameObject Level Loader, προσθέσαμε το LevelLoader script και ορίσαμε στον Inspector ποιο θα είναι το επόμενο Level. Μέσω ενός τελικού trigger στο τέλος της κάθε πίστας, ο παίκτης μεταβαίνει στην επόμενη πίστα κάθε φορά.

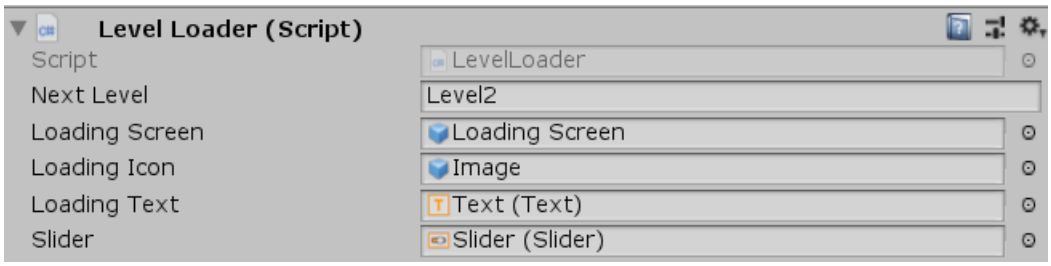


Figure 137: Level Loader Script component in the inspector

6.4 Random Drip Sound On Collision

Για το παιχνίδι μας θέλαμε να συνδυάσουμε το Particle Effect της σταγόνας από διαρροή σωλήνα, με ηχητικό εφέ και αναπαραγωγή ήχου κατά την σύγκρουση της σταγόνας με το έδαφος. Δοκιμάσαμε με έναν μοναδικό ήχο σταγόνας, αλλά δεν ήταν τόσο ρεαλιστικό κάθε επόμενη σταγόνα on collision με το πάτωμα, να παράγει τον ίδιο ήχο σταγόνας. Οπότε, προχωρήσαμε στην κατάτμηση ενός μεγάλου ενιαίου ηχητικού κλιπ πολλών λεπτών, με διαδοχικούς ήχους σταγόνας σε μία επιφάνεια, σε διαφορετικά audio samples σταγόνας για να υπάρχει ποικιλία και διαφορετικότητα μεταξύ των ήχων και γράψαμε ένα script που επιτρέπει την αναπαραγωγή random clips σταγόνας on particle collision χρησιμοποιώντας την έτοιμη συνάρτηση OnParticleCollision, ενώ ορίσαμε το κατάλληλο audio source και τα περιεχόμενα του array των drip audio clips στον inspector.

```
{
    public AudioSource randomDrip;
    public AudioClip[] audioSources;

    void RandomSoundness()
    {
        randomDrip.clip = audioSources[UnityEngine.Random.Range(0, audioSources.Length)];
        randomDrip.Play();
        //CallAudio();
    }
    void OnParticleCollision (GameObject other)
    {
        RandomSoundness();
        //Debug.Log("Particle Hit!");
    }
}
```

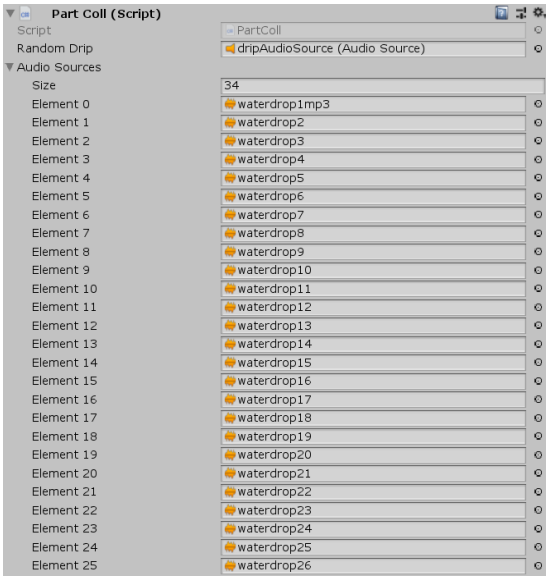


Figure 138: Different Drip Samples in the ParticleColl script component

6.5 Animation Camera Follow

Για το παιχνίδι μας χρειάστηκε να δημιουργήσουμε κάποια animations κατά την επίλυση γρίφων, ξεκλείδωμα πόρτας, κουτιών κ.α όποτε επιθυμούσαμε τα γεγονότα αυτά να γίνονται πιο εύκολα και στοχευμένα αντιληπτά από τον παίκτη. Για να το επιτύχουμε αυτό, αναπτύξαμε scripts έτσι ώστε η κάμερα του παίκτη να ακολουθεί αυτόματα το animated key/item κατά τη διάρκεια του animation, ενεργοποιώντας το κατάλληλο script component και το camera follow να σταματάει μετά το τέλος του animation με την απενεργοποίηση του CameraLookAt script. Ακολουθεί ο κώδικας:

```
public class CameraLookAtCrowBar : MonoBehaviour
{
    public Transform target;

    void Update()
    {
        transform.LookAt(target);
    }
}
```

```
public class PlayCrowbarDoorAnimation : MonoBehaviour
{

    private Animator crowbarDoorAnimator;
    public Camera lookingCamera;

    [Header("Animation")]
    [SerializeField] public AnimationClip crowbarAnimation;

    private void Awake()
    {
        crowbarDoorAnimator = gameObject.GetComponent<Animator>();
    }

    public void PlayCrowbarAnimation()
    {
        crowbarDoorAnimator.SetBool("beginCrowbarAnimation", true);
        StartCoroutine(LookAtCrowbar());
    }

    IEnumerator LookAtCrowbar()
    {
        lookingCamera.GetComponent<CameraLookAtCrowBar>().enabled = true;
        yield return new WaitForSeconds(4f);
        lookingCamera.GetComponent<CameraLookAtCrowBar>().enabled = false;
        yield return null;
    }
}
```

6.6 Elevator Scene and Problems

Για την μετάβαση από την δεύτερη στην τρίτη πίστα του παιχνιδιού, θέλαμε να δώσουμε έναυσμα σε animated ανελκυστήρα, η κίνηση του οποίου θα πυροδοτούνταν με την είσοδο του παίκτη σε αυτόν. Δημιουργήθηκε όμως πρόβλημα, καθώς η ύπαρξη rigidbody στον παίκτη και το υπάρχον parenting στη σκηνή μας (το οποίο δεν ήταν δυνατόν να αλλάξει κατά το προχωρημένο στάδιο της ανάπτυξης) δημιουργούσαν προβληματικά physics κατά την διάρκεια της πορείας καθόδου του ανελκυστήρα, κάτι που είχε σαν αποτέλεσμα ο παίκτης να αναπηδά και η κάμερα να τρέμει λόγω της αναπήδησης με αποτέλεσμα ανεπιθύμητο αισθητικά αποτέλεσμα και προβληματικό gameplay κατά την κίνηση του παίκτη μέσα στον κινούμενο προς-τα-κάτω ανελκυστήρα. Δεν μπορούσαμε επίσης στη συγκεκριμένη περίπτωση να δώσουμε κάποια λύση με Parenting και το IsKinematic του RigidBody component. Αποφασίσαμε λοιπόν να προχωρήσουμε σε μία “Smoke and Mirrors” [15] επίλυση του προβλήματος, καθώς είχε περάσει ήδη αρκετός χρόνος και κάθε άλλη προσπάθεια συνέχιζε να παρουσιάζει προβλήματα. Αντικαταστήσαμε λοιπόν OnTrigger τον παίκτη με μία free camera-child object του ανελκυστήρα, στο ύψος που βρισκόταν η ενσωματωμένη κάμερα στον capsule collider του παίκτη, μαζί με ένα γρήγορο blackout και γρήγορο fade in πάλι, έτσι ώστε αυτή η αλλαγή στις controllable cameras να μην γίνει ιδιαίτερα αντιληπτή από τον παίκτη, αλλά και να σταματήσουμε να έχουμε αυτό το πρόβλημα αναπήδησης λόγω βαρύτητας, δίνοντας επίσης την δυνατότητα στον παίκτη να χειρίζεται την κάμερά του κατά την κάθοδο του ανελκυστήρα και με τον collider της νέας κάμερας να πλησιάζει προς τον τελικό trigger που πυροδοτούσε την μετάβαση προς την επόμενη πίστα.

```
public class ElevatorEngineStart : MonoBehaviour
{
    public GameObject elevator;
    public GameObject playerMass;
    public GameObject oldPlayerCamera;
    public GameObject newPlayerCamera;

    void OnTriggerEnter(Collider collider)
    {
        Vector3 newRotation = new Vector3(oldPlayerCamera.transform.rotation.x, playerMass.transform.rotation.y, playerMass.transform.rotation.z);
        Vector3 newPosition = new Vector3(oldPlayerCamera.transform.position.x, oldPlayerCamera.transform.position.y, oldPlayerCamera.transform.position.z);
        playerMass.SetActive(false);
        newPlayerCamera.SetActive(true);
        newPlayerCamera.transform.eulerAngles = newRotation;
        newPlayerCamera.transform.position = newPosition;
        newPlayerCamera.GetComponent<FadeToBlackBlitter>().FlashBlackToWhite();
        elevator.GetComponent<ElevatorAnimation>().PlayElevatorAnimation();
    }
}
```

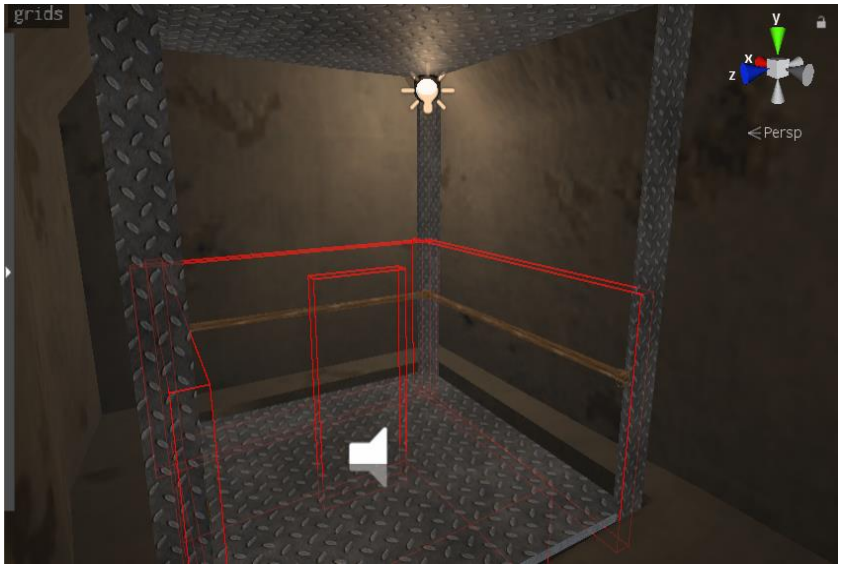


Figure 139: Elevator Engine Start trigger

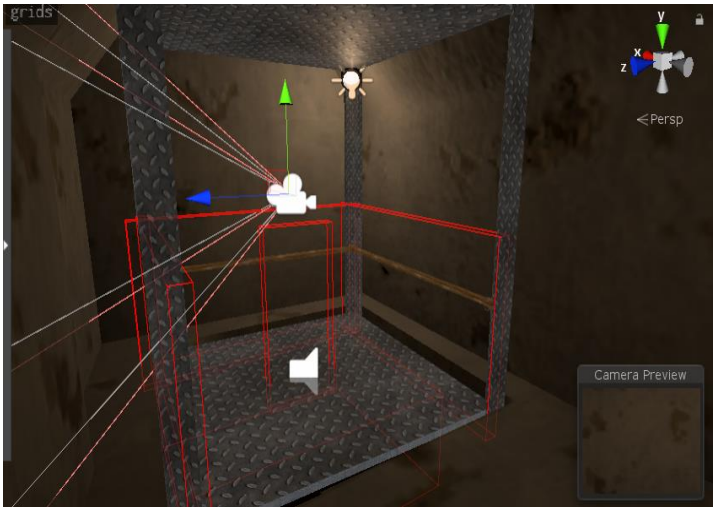


Figure 140: The previously disabled camera – child of the elevator is enabled after elevator engine start and at the same time the main player controller is disabled.

6.7 Αναπαραγωγή soundtrack του παιχνιδιού στο Level 2

Επειδή επιθυμούσαμε με την είσοδο του παίκτη σε διαφορετικούς χώρους της πίστας να αναπαράγονται και διαφορετικά κομμάτια μουσικής, τοποθετήσαμε triggers σε συγκεκριμένα σημεία του περιβάλλοντος, ενώ χρησιμοποιήσαμε έναν έτοιμο Audio Manager για να διαχειριστούμε τα διαφορετικά clips μουσικής και να επωφεληθούμε κάποιων συναρτήσεων που δεν μας παρέχονται by default από την Unity. Γράψαμε λοιπόν το εξής script, δηλώνοντας διαφορετικές boolean μεταβλητές για τα διαφορετικά triggers, το οποίο περάσαμε σαν component σε κάθε διαφορετικό trigger GameObject για αναπαραγωγή διαφορετικών κομματιών μουσικής, αλλά και για να απενεργοποιήσουμε τα triggers μετά την πρώτη αναπαραγωγή ή για να ενεργοποιήσουμε νέα triggers, σε περίπτωση που ο παίκτης δεν έφτανε στην λύση και έπρεπε να συνέχισει να περιφέρεται στην πίστα.

```
public class Lvl2ScoreTriggers : MonoBehaviour
{
    public bool isCrossroad;
    public bool isChoosingAPath;
    public bool isRoaming;
    public bool isRoomWithSthImportant;
    public bool isRoomFromTheFuture;
    public bool isProgressToLvl3;
    public bool isFatberg;

    public GameObject triggerDisappearer;
    public GameObject triggerAppearer;
    public GameObject audioManagerObject;
    private AudioManager audioManager;

    public void Start()
    {
        audioManager = audioManagerObject.GetComponent<AudioManager>();
    }
    void OnTriggerEnter()
    {
        if(isCrossroad)
        {
            audioManager.Play("lvl2FindingAnewRoomCrossroadfadedfadein", 1, 1f, 1f);
            StartCoroutine(TriggerDisappear());
        }
        else if(isChoosingAPath)
        {
            audioManager.Play("lvl2ChoosingApathfaded", 1, 1f, 1f);
            StartCoroutine(TriggerDisappear());
        }
```

```
    else if(isFatberg)
    {
        audioManager.Play("lvl2Fatbergfaded", 1, 1f, 1f);
        StartCoroutine(TriggerDisappear());
    }
    else if(isRoomWithSthImportant)
    {
        audioManager.Play("lvl2FindingAroomWithsthImportantfaded", 1, 1f, 1f);
        StartCoroutine(TriggerDisappear());
    }
    else if (isRoaming)
    {
        audioManager.Play("lvl2roamer1explorationfaded", 1, 1f, 1f);
        StartCoroutine(TriggerDisappear());
    }
    else if (isRoomFromTheFuture)
    {
        audioManager.Play("lvl2RoomFromtheFuture2faded", 1, 1f, 1f);
        StartCoroutine(TriggerDisappear());
    }
    else if (isProgressToLvl3)
    {
        audioManager.Play("lvl2ProgressToLevel3faded", 1, 1f, 1f);
        StartCoroutine(TriggerDisappear());
    }
}
```

```
IEnumerator TriggerDisappear()
{
    yield return new WaitForSeconds(1f);
    if (isRoomFromTheFuture || isProgressToLvl3 || isRoomWithSthImportant || isFatberg || isChoosingAPath || isCrossroad)
    {
        triggerDisappearer.GetComponent<BoxCollider>().enabled = false;
    }
    else if (isRoaming)
    {
        triggerDisappearer.GetComponent<BoxCollider>().enabled = false;
        triggerAppearer.GetComponent<BoxCollider>().enabled = true;
    }
    yield return null;
}
```

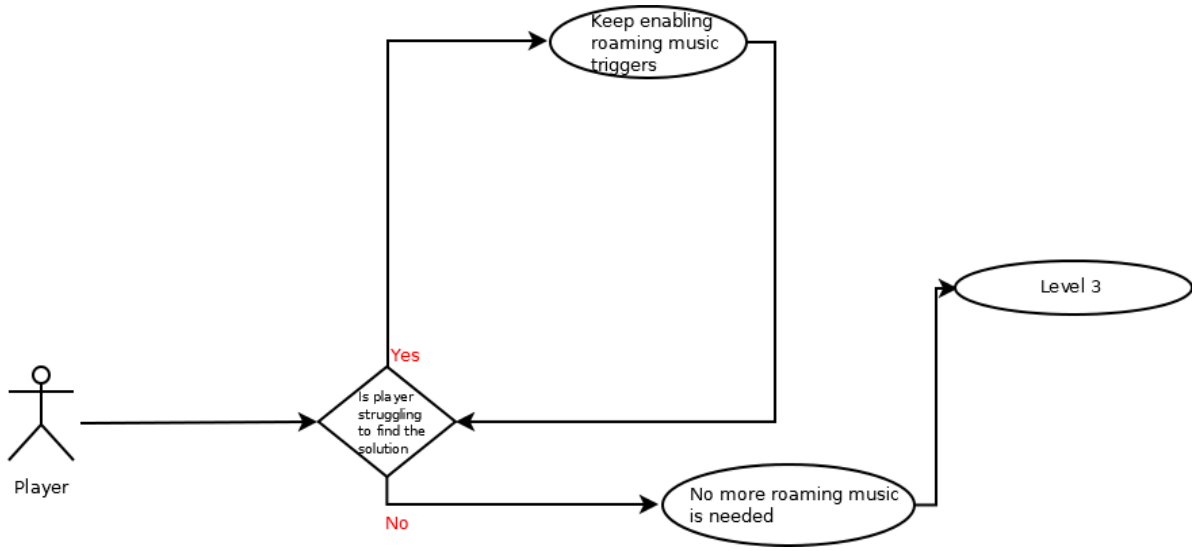


Figure 141: Visual representation of the music playing script in Level 2

6.8 Memory Recorder

Κατά την διάρκεια της υλοποίησης του παιχνιδιού, υπήρξε η ιδέα για την δημιουργία ενός Memory Recorder, έτσι ώστε ο παίκτης να έχει την δυνατότητα να καταγράφει τις μνήμες του από την περιπλάνηση στο παιχνίδι και να μπορεί ουσιαστικά να τις ξαναβιώσει. Αν και το συγκεκριμένο σύστημα δεν χρησιμοποιήθηκε εν τέλει στην τελική έκδοση του παιχνιδιού, εμείς το υλοποιήσαμε, πήραμε αρκετές πολύτιμες γνώσεις και παρουσιάζει ενδιαφέρον, γι΄ αυτό θα αναφερθούμε και σ' αυτό. Η αρχική ιδέα ήταν να χρησιμοποιήσουμε ένα έτοιμο video capture εργαλείο από το asset store της Unity, είδαμε όμως ότι κατά την καταγραφή του βίντεο είχαμε πτώση στα frames per second του παιχνιδιού, αλλά και ότι το αρχείο καταγραφής ήταν αρκετά μεγάλο όσον αφορά στο μέγεθος που καταλάμβανε στο σκληρό δίσκο, ενώ κατά την αναπαραγωγή δεν μπορούσαμε να έχουμε την ίδια υψηλή ανάλυση με το πρωτότυπο gameplay. Δημιουργήσαμε από την αρχή λοιπόν ένα σύστημα καταγραφής και αναπαραγωγής "βιώματος" στα πρότυπα Replay System παιχνιδιών ταχύτητας.
Το συγκεκριμένο σύστημα αποτελείται από δύο scripts το MemoryInventory και το Replay. Το πρώτο αφορούσε καθαρά το κομμάτι διεπαφής χρήστη και πως ο παίκτης θα είχε πρόσβαση στο Menu του Memory Inventory, έτσι ώστε να μπορεί να αναπαράγει παλιές του μνήμες. Η κύρια λειτουργικότητα υπάρχει στο Replay.cs. Δημιουργήσαμε μία κλάση PositionRotation για να αποθηκεύουμε διαδοχικά positions της κάψουλας του παίκτη και διαδοχικά Rotations της κάμερας.

```
public class PositionRotation
{

    public Vector3 position { get; set; }
    public Quaternion rotation { get; set; }

    public PositionRotation(Vector3 _position, Quaternion _rotation)
    {
        position = _position;
        rotation = _rotation;
    }

}
```

Και μία κλάση MemorySaved για να αποθηκεύσουμε την κάθε μνήμη του παίκτη στον δίσκο μετά απ΄ το τέλος της καταγραφής της, αλλά και να τη συνδέσουμε με το δικό της Button που γινόταν Initialize στο UI. Ίσως εδώ θα ήταν καλύτερο να αναπτύξουμε ένα σύστημα Object Pooling για να μην χρειάζεται να δημιουργούμε τα επιπλέον buttons κατά το runtime και να χρησιμοποιήσουμε structs, αλλά κάτι τέτοιο είναι εκτός του πεδίου εφαρμογής της συγκεκριμένης διπλωματικής.

```
[System.Serializable]
public class MemorySaved
{
    public GameObject memoryButton;
    public string memoryName;
    //public Image memoryThumbnail;
    //public int memoryLvl;

    public MemorySaved(GameObject _memoryButton, string _memoryName)
    {
        memoryName = _memoryName;
        memoryButton = _memoryButton;
        //memoryThumbnail = _memoryThumbnail;
        //memoryLvl = _memoryLvl;
    }
}
```

Στη συνέχεια στο κύριο μέρος του script, δημιουργήσαμε μία λίστα PositionsRotations για την καταγραφή διαδοχικών θέσεων του παίκτη και rotations της κάμερας.

```
void FixedUpdate()
{
    if (isRecording)
        Record();
}

void Record()
{

    //positionsRotations.Insert(0, new PositionRotation(transform.position, camera.transform.rotation));
    positionsRotations.Add(new PositionRotation(player.transform.position, camera1.transform.rotation));

}
```

Χρησιμοποιήσαμε την έτοιμη βιβλιοθήκη Newtonsoft.json για να έχουμε τη δυνατότητα να κάνουμε εύκολα Serialize – Deserialize, έτσι ώστε να αποθηκεύσουμε τις μνήμες στον δίσκο, αλλά και να έχουμε πρόσβαση σε αυτές και να μπορούμε να τις αναπαράγουμε ξανά.

Αρχικοποίηση:

```
void Start()
{
    if (Directory.Exists(Application.dataPath + "/" + "memories"))
    {

    }
    else
    {
        Directory.CreateDirectory(Application.dataPath + "/" + "memories");
    }

    string[] files = Directory.GetFiles(Application.dataPath + "/" + "memories" + "/", "*.json");

    foreach (string file in files)
    {
        replayFileName = Path.GetFileName(file);
        //Debug.Log(Path.GetFileName(file));
        buttonSpawn.CreateMemoryButtons();
    }
}
```

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.R))
        StartRecording();
    if (Input.GetKeyDown(KeyCode.T))
        StopRecording();
}
```

και καταγραφή μνήμης με αποθήκευση σε αρχείο στον δίσκο:

```
public string StopRecording()
{
    //stop the recording and save lists of positions
    //and rotations to a json file

    isRecording = false;

    var memoryList = Directory.GetFiles(Application.dataPath + "/" + "memories" + "/", "*.json");
    var splittedFileName = replayFileName.Split(delimiterChars);

    var fileCounter = memoryList.Length;
    splittedFileName[2] = fileCounter.ToString();
    var lvlName = SceneManager.GetActiveScene().buildIndex;
    Debug.Log("The lvlName is: " + lvlName);
    var splittedLvlName = lvlName.ToString();
    splittedFileName[1] = splittedLvlName;
    var connectedFileName = "memory_" + splittedFileName[1] + "_" + splittedFileName[2] + ".json";
    replayFileName = connectedFileName;
    File.WriteAllText(Application.dataPath + "/" + "memories" + "/" + replayFileName, JsonConvert.SerializeObject(positionsRotations));
    GetComponent<ButtonsSpawn>().CreateMemoryButtons();
    GameObject memButton = GetComponent<ButtonsSpawn>().button;
    memory.Insert(0, new MemorySaved(memButton, replayFileName));
    ClearList();

    recordingTxt = GameObject.FindWithTag("RecordingText").transform.GetChild(0).transform.GetComponent<Text>();
    recordingTxt.gameObject.SetActive(false);
    return replayFileName;
}
```

Ενώ για την αναπαραγωγή της μνήμης Deserialize:

```
public void GatherDataFromFile()
{
    memName = "memory_" + mName + ".json";
    string dataPath = Application.dataPath + "/" + "memories" + "/" + memName;
    Debug.Log("The full datapath is: " + dataPath);

    List<PositionRotation> deserializedPosRots = JsonConvert.DeserializeObject<List<PositionRotation>>(File.ReadAllText(dataPath));
    posRots = deserializedPosRots;

    replayCamera.GetComponent<Blitter>().enabled = true;
    replayCamera.GetComponent<DesaturateBlit>().enabled = true;
    lastPosition = replayPlayer.transform.position;
    lastRotation = replayCamera.transform.rotation;



    StartCoroutine(PlayFrom(0f));

}
```

και lerp μεταξύ των διαδοχικών positions και rotations [19]:

```
bool Playback(float time)
{

    // Convert the playback time to a position in our sample array.
    // (Here I'm assuming the current fixedDeltaTime is the same one used to record)
    float sample = time / Time.fixedDeltaTime;

    // Round down to the position & rotation sample just before this moment.
    // (Here I'm assuming time is non-negative)
    int previousIndex = (int)(sample);


    int last = posRots.Count - 1;
    if (previousIndex < last)
    {
        // We have another snapshot ahead, so we'll interpolate smoothly between them.
        int nextIndex = previousIndex + 1;
        float interpolation = sample - previousIndex;

        replayPlayer.transform.position = Vector3.Lerp(
                                posRots[previousIndex].position,
                                posRots[nextIndex].position,
                                interpolation);

        // Slerp is technically a more consistent option here, since it keeps
        // the rate of change even over the course of the interpolation.
        // But each step is usually too small for this difference to be visible.
        replayCamera.transform.rotation = Quaternion.Lerp(
                                posRots[previousIndex].rotation,
                                posRots[nextIndex].rotation,
                                interpolation);

        // Signal that there's still more replay past here.
        return true;
    }
```

6.9 Αναπαραγωγή soundtrack στα Level 3, Level 4 και Level 5

Καθαρά σχεδιαστικά κι επειδή ο απαιτούμενος χρόνος για πιθανή επίλυση γρίφων και περιήγησης για τον παίκτη αυξανόταν αρκετά, αποφασίσαμε να μην χρησιμοποιήσουμε triggers σε διαφορετικές

περιοχές πίστας για αναπαραγωγή της μουσικής περιήγησης, αλλά και για να μην έχουμε επικάλυψη των ηχητικών κλιπ μουσικής περιήγησης υλοποιήσαμε ένα script με timer και διαφορετικά audio clips για τα διαφορετικά levels.

Αρχικοποίηση:

```
public AudioManager scoreManager;
private float secondsCount;
[SerializeField]private bool isLvl3, isLvl4, isLvl5, isLvl1, isLvl2, isMenu;
// Start is called before the first frame update
void Start()
{   if (isLvl3)
    {
        scoreManager.PlayWithDelay("lvl3Roamerfaded", 5f, 2, 1f, Random.Range(0.8f, 1f));
    } else if (isLvl4)
    {
        scoreManager.PlayWithDelay("lvl4Exploration2faded", 5f, 3, 1f, Random.Range(0.8f, 1f));
    } else if (isLvl5)
    {
        scoreManager.PlayWithDelay("lvl5Exploration11faded", 5f, 4, 1f, Random.Range(0.8f, 1f));
    }

}
```

Και ο timer που τρέχει μέσα στην Update() και αναπαράγει το κατάλληλο audio track με σκοπό να αποφευχθούν επικαλύψεις μεταξύ των διαφορετικών audio clips. Χρησιμοποιήσαμε τον έτοιμο δωρεάν Audio Manager by Carter Games από το Asset Store της Unity για να επωφεληθούμε των έτοιμων συναρτήσεων για αναπαραγωγή μουσικής και να προσδώσουμε randomness αλλάζοντας το pitch σε κάθε νέα αναπαραγωγή:

```
using CarterGames.Assets.AudioManager;
```

```
void Update()
{
    UpdateTimerUI();
}

public void UpdateTimerUI()
{
    //set timer UI
    secondsCount += Time.deltaTime;
    if ((secondsCount >= 120) && (isLvl3))
    {
        if (!scoreManager.IsClipPlaying("lvl3findingAcluefaded") && !scoreManager.IsClipPlaying("lvl3fIndingAnotherCluefaded") &&!scoreManager.IsClipPlaying("lvl3Roamerfaded"))
        {
            secondsCount = 0;
            scoreManager.PlayWithDelay("lvl3Roamerfaded", 5f, 2, 1f, Random.Range(0.8f, 1f));
        }

    } else if ((secondsCount >= 120) && (isLvl4))
    {
        if (!scoreManager.IsClipPlaying("lvl4Exploration2faded") && !scoreManager.IsClipPlaying("lvl4FindAnotherCluefaded") && !scoreManager.IsClipPlaying("lvl4FindCluefaded") && !scoreManager.IsClipPlaying("lvl4rightdirectionfaded"))
        {
            secondsCount = 0;
            scoreManager.PlayWithDelay("lvl4Exploration2faded", 5f, 3, 1f, Random.Range(0.8f, 1f));
        }
    } else if ((secondsCount >= 120) && (isLvl5))
    {
        if (!scoreManager.IsClipPlaying("lvl5Exploration11faded") && !scoreManager.IsClipPlaying("lvl5Exploration21resonatfaded") && !scoreManager.IsClipPlaying("lvl5Explorationpart2faded"))
        {
            secondsCount = 0;
            int rand = Random.Range(0, 2);
            switch (rand)
            {
                case 0:
                    scoreManager.PlayWithDelay("lvl5Exploration11faded", 5f, 4, 1f, Random.Range(0.8f, 1f));
                    break;
                case 1:
                    scoreManager.PlayWithDelay("lvl5Exploration21resonatfaded", 5f, 4, 1f, Random.Range(0.8f, 1f));
                    break;
                case 2:
                    scoreManager.PlayWithDelay("lvl5Explorationpart2faded", 5f, 4, 1f, Random.Range(0.8f, 1f));
                    break;
                default: break;
            }
        }
    }

}
```

6.10 Εξειδικευμένη αναπαραγωγή μουσικής και ήχου από audio sources στο Level 5

Εδώ αντιμετωπίσαμε ένα πρόβλημα, καθώς το Audio System της Unity μας παρέχει μόνο σφαιρικό Audio Source.

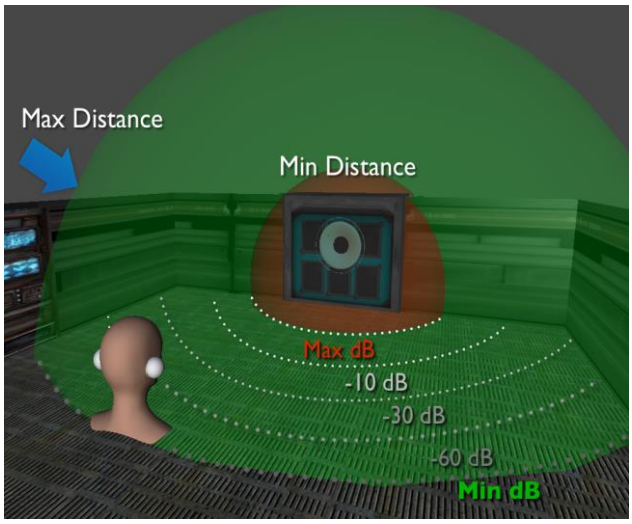


Figure 142: Spherical Audio Source

Ο ήχος ενός δυνατού σε ένταση Audio Source και με αρκετά αυξημένο max distance γινόταν αντιληπτός από τον listener σε σημεία της πίστας που δεν επιθυμούσαμε (πίσω από τοίχους με μεγάλο πλάτος στη συγκεκριμένη περίπτωση), οπότε χρησιμοποιήσαμε δύο triggers έτσι ώστε ο ήχος να ακούγεται δυνατά στο δωμάτιο 2 και ελάχιστα έως μηδαμινά στο δωμάτιο 1.

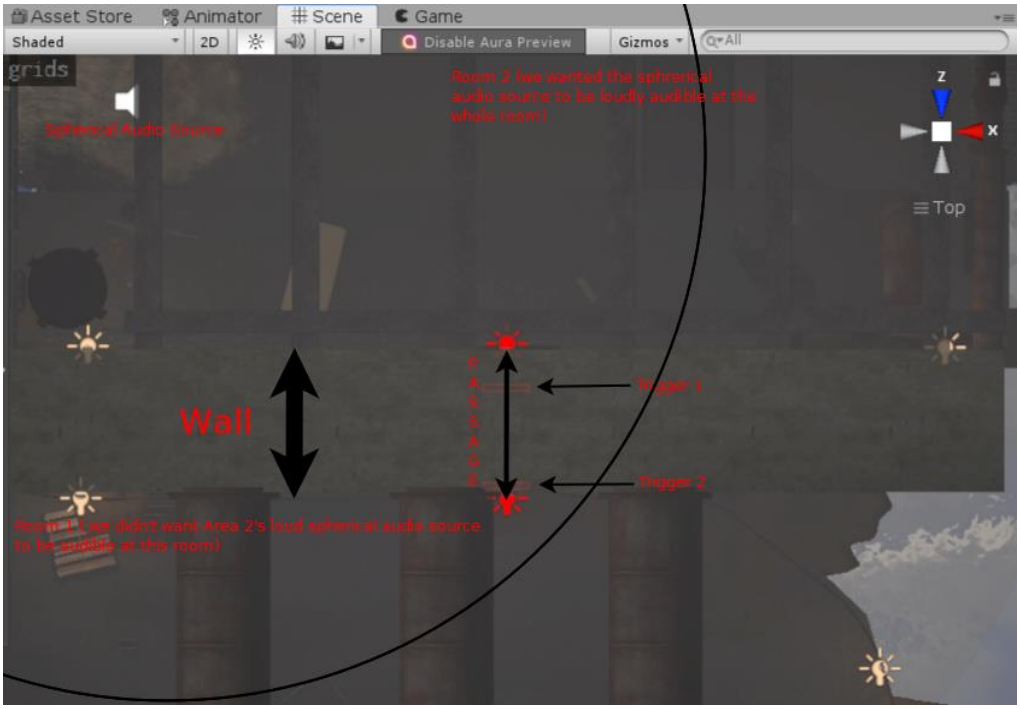


Figure 143: Κάτοψη του προβλήματος

Χρησιμοποιήσαμε διαφορετικές boolean μεταβλητές στο script για να εξακριβώσουμε σε ποιο σημείο βρίσκεται ο παίκτης και αν επιθυμούσαμε fade in ή fade out του continuously playing audio source.

Ο κώδικας για το παραπάνω:

```
void OnTriggerEnter()
{
    if (startWaterfall && !waterFallAlreadyPlaying)
    {
        waterFallAlreadyPlaying = true;
        stopWaterfallTrigger.GetComponent<OnTriggerAudioLvl5>().stopWaterfall = true;
        StartCoroutine(FadeInOutWaterFall(0.05f, 0.65f, 1.5f));
    }
    else if (startWaterfall && waterFallAlreadyPlaying)
    {
        stopWaterfallTrigger.GetComponent<OnTriggerAudioLvl5>().waterFallStop = false;
    }

    else if (stopWaterfall && waterFallStop)
    {
        startWatefallTrigger.GetComponent<OnTriggerAudioLvl5>().waterFallAlreadyPlaying = false;
        StartCoroutine(FadeInOutWaterFall(0.06f, 0.05f, 1.5f));
    }
    else if (stopWaterfall && !waterFallStop)
    {
        startWatefallTrigger.GetComponent<OnTriggerAudioLvl5>().waterFallAlreadyPlaying = false;
        waterFallStop = true;
        StartCoroutine(FadeInOutWaterFall(0.65f, 0.05f, 1.5f));
    }
```

```
IEnumerator FadeInOutWaterFall(float fromVal, float toVal, float duration)
{
    float counter = 0f;

    while (counter < duration)
    {
        if (Time.timeScale == 0)
            counter += Time.unscaledDeltaTime;
        else
            counter += Time.deltaTime;

        waterFallAudioSource.GetComponent<AudioSource>().volume = Mathf.Lerp(fromVal, toVal, counter / duration);
        yield return null;
    }
}
```

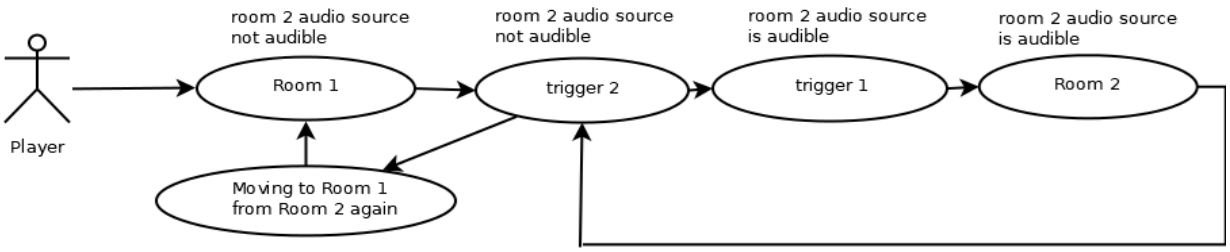


Figure 144: Visual representation of the the trigger system to mute audio source in different areas

Το soundtrack του παιχνιδιού αναπαραγόταν και σε αυτή την πίστα με timer, αλλά με κάποια customizations. Πιο συγκεκριμένα, επιθυμούσαμε ο timer να σταματάει κατά την είσοδο του παίκτη σε ένα μεγάλο tunnel έτσι ώστε όταν ο παίκτης φτάσει στο τέλος του, να αναπαραχθεί ένα κομμάτι score ειδικά για το μεγάλο τελικό δωμάτιο που έμπαινε ο παίκτης και κάθε φορά που ο παίκτης

ξαναεισερχόταν σε αυτό το μεγάλο δωμάτιο να αναπαράγεται και πάλι το ίδιο track. Επίσης, μετά την έξοδο του παίκτη πάλι από την είσοδο του tunnel, επιθυμούσαμε την επανενεργοποίηση του timer και αναπαραγωγή του roaming music track ανά τακτά χρονικά διαστήματα για τους υπόλοιπους χώρους της πίστας.
Ακολουθεί σχεδιάγραμμα του συστήματος με triggers που ενεργοποιούνται και απενεργοποιούνται για να έχουμε αναπαραγωγή μουσικής κατ' επιλογή.

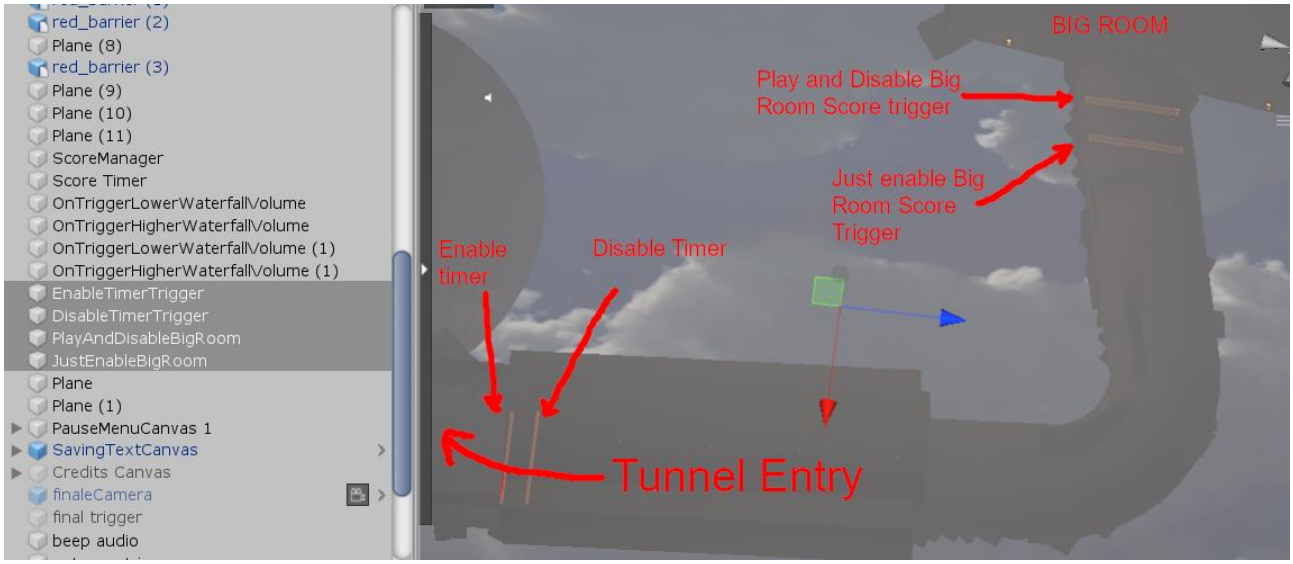


Figure 145: Level 5 music system map view diagram

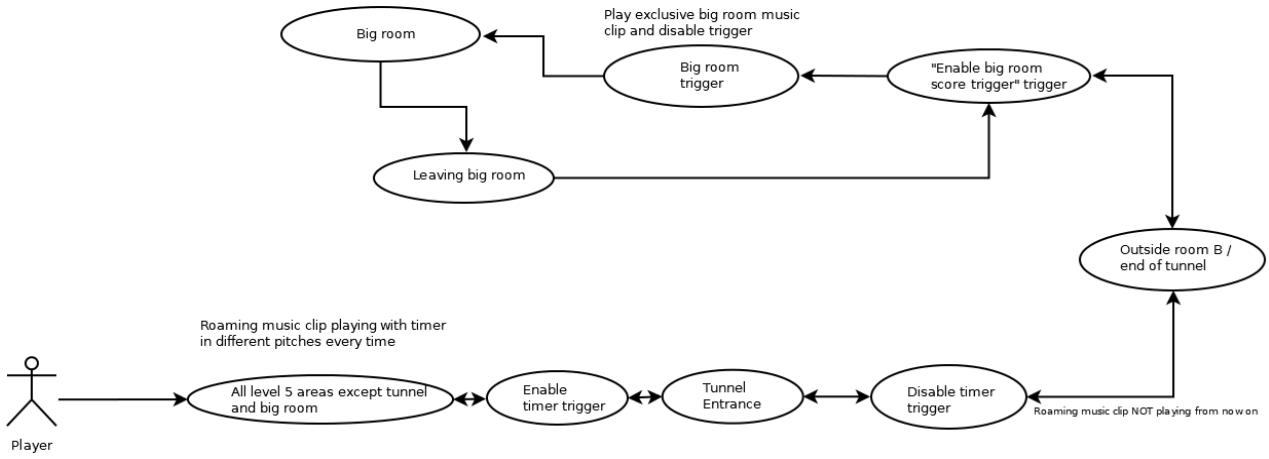


Figure 146: Level 5 main music system user case diagram

και οι συνθήκες για ενεργοποίηση-απενεργοποίηση triggers με την χρήση boolean μεταβλητών που καθόριζαν τον ρόλο του κάθε ξεχωριστού trigger:

```
else if (justEnableBigRoom)
{
    playAndDisableBigRoomTrigger.SetActive(true);
}
else if (playAndDisableBigRoom)
{
    if (!scoreManager.IsClipPlaying("lvl5AccomplishmentEndingorGoingtotherightdirectionfaded"))
    {
        scoreManager.PlayWithDelay("lvl5AccomplishmentEndingorGoingtotherightdirectionfaded", 1f, 4, 1f, Random.Range(0.9f, 1f));
    }
    playAndDisableBigRoomTrigger.SetActive(false);
}
else if (enableTimer)
{
    scoreTimer.SetActive(true);
}
else if (disableTimer)
{
    scoreTimer.SetActive(false);
}
```

6.11 Βελτιστοποίηση

Για βελτιστοποίηση της εφαρμογής μας χρησιμοποιήσαμε:

- το σύστημα Occlusion Culling της Unity,
- αφαιρέσαμε faces των μοντέλων που δεν ήταν ορατά από την κάμερα,
- χρησιμοποιήσαμε τον Physics debugger για να έχουμε χρωματική επισκόπηση των διαφορετικών τύπων colliders και να διορθώσουμε προβλήματα φυσικής και συγκρούσεων που είχαμε με αυτούς,
- μειώσαμε το poly count των τρισδιάστατων μοντέλων μας με χρήση του Decimate modifier στο Blender και
- χρησιμοποιήσαμε τα texture arrays και τους custom shaders του OneBatch [24], έτσι ώστε πολλά στατικά αντικείμενα της εκάστοτε σκηνής μας να μοιράζονται το ίδιο material και να επωφεληθούμε του Static Batching, με απώτερο στόχο να μειώσουμε τα batches και τα set pass calls, άρα να έχουμε πιο ταχείς χρόνους rendering, κέρδη στην απόδοση της εφαρμογής και τα frames ανά δευτερόλεπτο, άρα και καλύτερη εμπειρία για τον τελικό χρήστη.
- Τα lightmaps έγιναν bake με τον Bakery Gpu Lightmapper καθώς μπορούσαμε να επωφεληθούμε του denoiser του, που δεν ήταν διαθέσιμος by default στην έκδοση 2018.4 της Unity κι έτσι με λιγότερα texels per unit και samples για baking, να κάνουμε bake των lightmaps σε μικρότερους χρόνους, αλλά και να έχουμε αρκετά μικρότερα σε μέγεθος αρχεία lightmaps, άρα και μειωμένη χρήση μνήμης ram κατά το play mode και φυσικά στο runtime της τελικής εφαρμογής, με σχετικά καλή οπτική ποιότητα προϋπολογισμένου φωτισμού.
- Χρησιμοποιήσαμε επίσης τον Profiler και τον Frame Debugger της Unity για να ανιχνεύουμε τυχόν προβλήματα που προκαλούσαν πτώση στην απόδοση, frame drops και spikes στην χρήση πόρων του υλικού.
- Επίσης, με το εργαλείο αξιολόγησης Fraps μπορούσαμε να έχουμε πιο ακριβή εποπτεία των frames ανά δευτερόλεπτό, απ' ότι μας παρείχαν τα built-in stats της Unity, αλλά και για επιπλέον ελέγχους απόδοσης κατά τα δοκιμαστικά builds.
- Επιπλέον, για το τελικό build επιλέξαμε LZ4HC compression method και IL2CPP [26] scripting backend, καθαρά για λόγους performance, ενώ για το δοκιμαστικά builds χρησιμοποιήσαμε το Mono scripting backend, καθώς παρόλη την μειωμένη απόδοση, μας προσέφερε την ευκολία για γρήγορο testing, λόγω του μειωμένου χρόνου που απαιτούνταν για την ολοκλήρωση των builds.

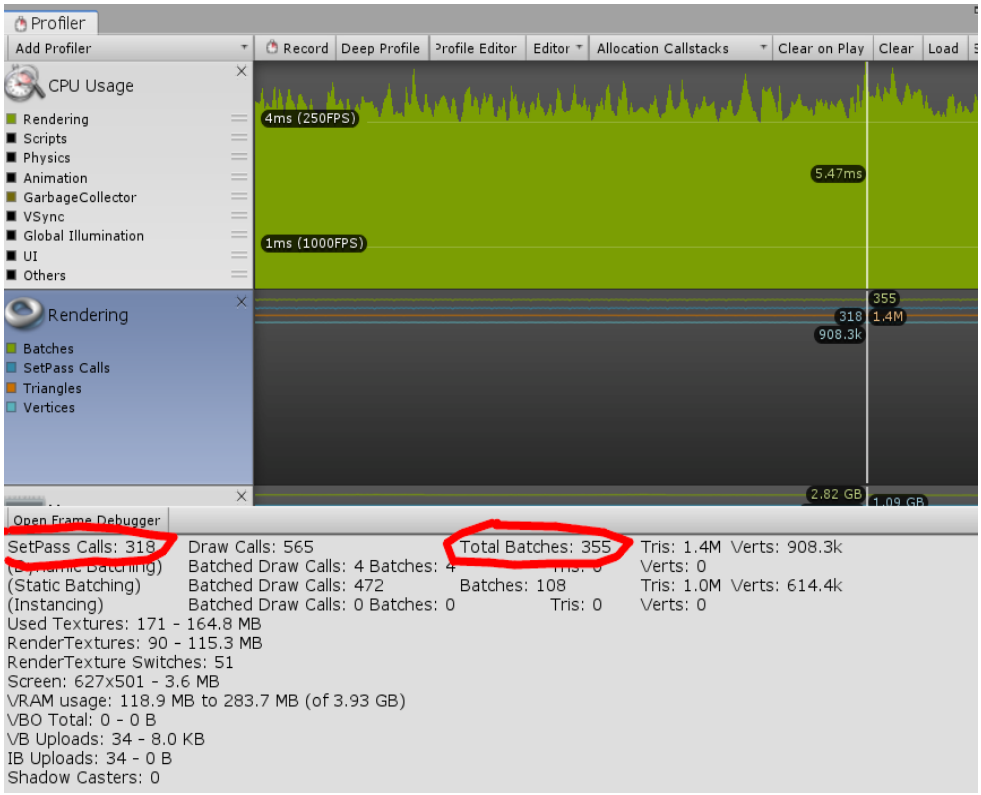


Figure 147: Without texture arrays

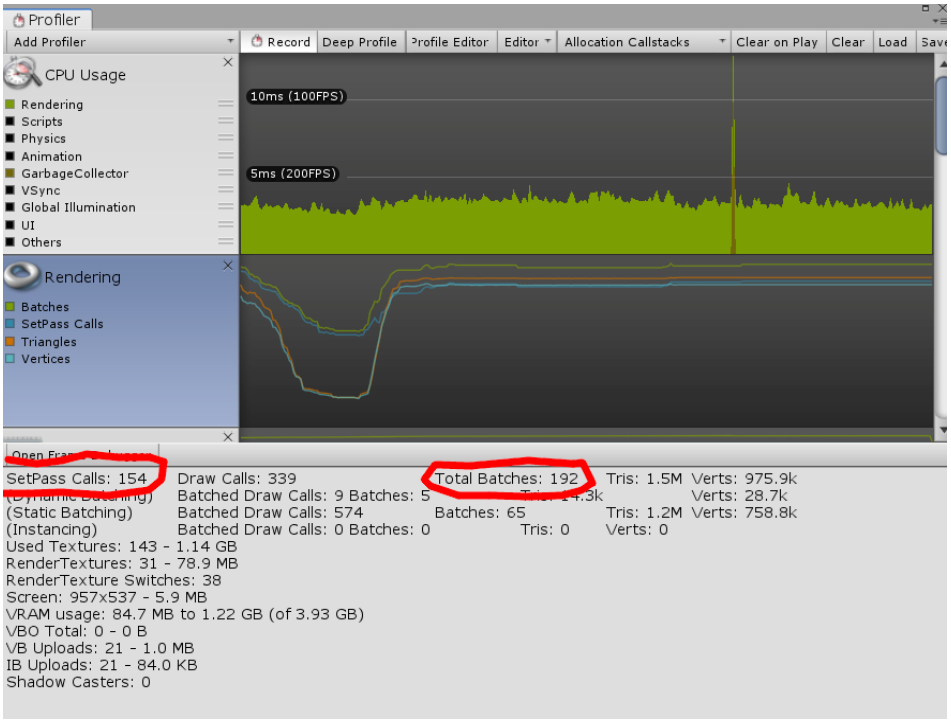


Figure 148: With texture arrays – Set pass calls and total batches decreasing dramatically

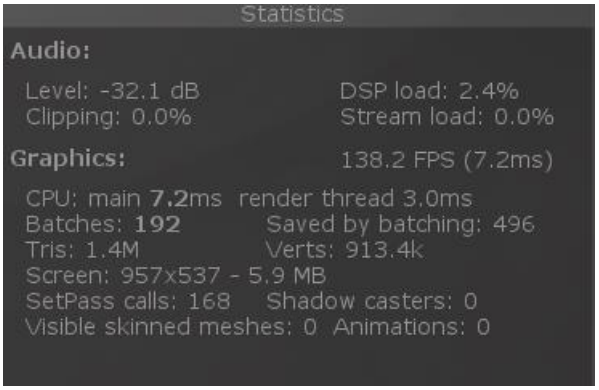


Figure 149: With occlusion culling 1.4M tris rendered in a heavy scene – 168 set pass calls

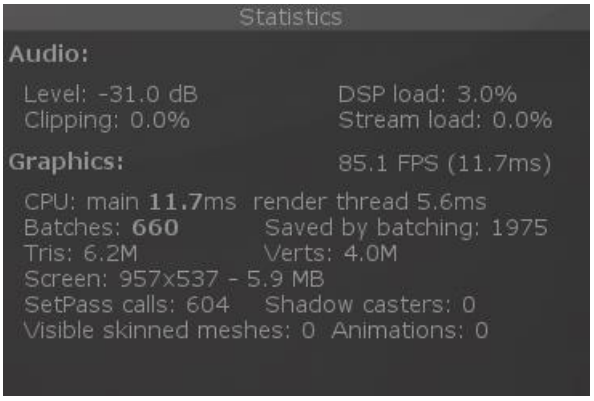


Figure 150: Without occlusion culling 6.2M tris rendered in the same heavy scene – 604 set pass calls

Κεφάλαιο 7: Το πείραμα

7.1 Σκοπός του πειράματος

Για το πείραμά μας και για να μπορέσουμε να εξάγουμε κάποια συμπεράσματα σχετικά με το πως διαφορετικές συνθήκες φωτισμού, contrast και ομίχλης μπορούν να επηρρεάσουν το συναίσθημα αβεβαιότητας του παίκτη, χωρίσαμε ουσιαστικά τους δεκαπέντε συμμετέχοντες του πειράματος σε τρεις διαφορετικές ομάδες των 5 ατόμων. Η Ομάδα 1 είχε αυξημένη ποσότητα ομίχλης, πολύ χαμηλό φωτισμό και χαμηλό contrast, η Ομάδα 3 καθόλου ομίχλη, δυνατό φωτισμό και αυξημένο contrast ενώ η Ομάδα 2 ενδιάμεσες ρυθμίσεις, ενώ υπήρχε ομοιογένεια μεταξύ των ρυθμίσεων των ίδιων ομάδων από την μία πίστα στην άλλη. Ο κάθε παίκτης είχε τον χρόνο να περιηγηθεί κατά την πρώτη του επαφή με το περιβάλλον του παιχνιδιού για 5 λεπτά σε κάθε πίστα και, μετά από το τέλος του συγκεκριμένου χρονικού ορίου, έπρεπε να απαντήσει στις ερωτήσεις ενός ερωτηματολογίου. Κάποιες από τις ερωτήσεις ήταν άμεσες και κάποιες άλλες έμμεσες, με σκοπό να εξάγουμε συμπεράσματα για την αβεβαιότητα που προκάλεσαν στον παίκτη τα διαφορετικά settings, αλλά και τον βαθμό engagement και ποια στοιχεία του παιχνιδιού κέντρισαν το ενδιαφέρον των συμμετεχόντων στις διαφορετικές ομάδες.

Οι ερωτήσεις ήταν οι ίδιες για την κάθε πίστα και για όλες τις ομάδες και τα ερωτηματολόγιά μας δημιουργήθηκαν ως Google Forms για να έχουμε εύκολη πρόσβαση στα αποτελέσματα και έτοιμα διαγράμματα.

7.2 Το ερωτηματολόγιο

Οι ερωτήσεις που τέθηκαν ήταν οι εξής:

2. Από 1 (λίγο) έως 5 (πολύ) πόσο πιστεύεις ότι ένιωσες αίσθηση μυστηρίου? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

3. Από 1 (λίγο) έως 5 (πολύ) πόσο ένιωσες την ανάγκη να περιεργαστείς αντικείμενα? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

4. Πόσα διαφορετικά αντικείμενα περιεργάστηκες πιο ενδελεχώς? *

_____

5. Περπατούσες συνέχεια (1) ή σταματούσες για να κοιτάξεις τριγύρω (5) ? Από 1 έως 5. *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

6. Κοιτούσες περισσότερο τους τοίχους, το πάτωμα ή το ταβάνι? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Τοίχους
- ◯ Πάτωμα
- ◯ Ταβάνι

7. Θα ήθελες να περιεργαστείς πιο ενδελεχώς κάποιο από τα αντικείμενα του περιβάλλοντος για το οποίο όμως δεν είχες τη δυνατότητα? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Ναι
- ◯ Όχι

8. Πόσο σε επηρέασε η μουσική σε συγκεκριμένα σημεία όσον αφορά στην ψυχική σου διάθεση? Από 1 (λίγο) έως 5 (πολύ). *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

9. Πόσο ένιωσες την ανάγκη να πλησιάσεις την πηγή των περιβαλλοντικών ήχων? Από 1 (λίγο) έως 5 (πολύ). *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

10. Από 1 (λίγο) έως 5 (πολύ) πόσο ένιωσες περιέργεια για το τι συμβαίνει και διάθεση να ανακαλύψεις? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

11. Έσκυψες καθόλου με το control? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Ναι
- ◯ Όχι

12. Ζούμαρες ή/και περιέστρεψες καθόλου τα αντικείμενα που περιεργάστηκες πιο ενδελεχώς? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Ναι
- ◯ Όχι

13. Υποψιάστηκες ποιος μπορεί να ήταν ο στόχος/τελικό δωμάτιο που θα σε οδηγήσει στην επόμενη πίστα? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Ναι
- ◯ Όχι

14. Αν υποψιάστηκες ποιος ήταν ο τελικός στόχος-δωμάτιο της πίστας, πόσο ξεκάθαρο ήταν κάτι τέτοιο από 1 (λίγο) έως 5 (πολύ)? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

15. Πόσο ένιωσες απορία? Από 1 (λίγο) έως 5 (πολύ). *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

16. Ένιωσες την ανάγκη να κάνεις παύση του παιχνιδιού? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Ναι
- ◯ Όχι

17. Πόσο θα ήθελες να έχεις τη δυνατότητα να τρέξεις γρηγορότερα? Από 1 (λίγο) έως 5 (πολύ). *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

18. Πόσο ρεαλιστικό σου φάνηκε το οπτικό αποτέλεσμα από 1 έως 5? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

19. Από 1 (λίγο) έως 5 (πολύ) πιστεύεις ότι το μέγεθος των αντικειμένων ήταν κοντά στο αντίστοιχο της πραγματικής ζωής? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

20. Από 1 (λίγο) έως 5 (πολύ) πόσο ένιωσες κλειστοφοβία? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

21. Από 1 έως 5 κατά πόσο ένιωσες μοναξιά (1) στο παιχνίδι ή ένιωσες ότι υπάρχει και κάποιος άλλος μαζί σου (5) μέσα στο παιχνίδι? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

22. Κατά πόσο ένιωσες την ανάγκη να αγγίξεις κάτι από τα χώρους ή τα αντικείμενα με τα ίδια σου τα χέρια? Από 1 (λίγο) έως 5 (πολύ). *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

23. Πιστεύεις ότι ο χώρος ήταν χαμηλοτάβανος (1) ή ψηλοτάβανος (5)? Από 1 έως 5. *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

24. Πόσο ατμοσφαιρική ήταν η εμπειρία από 1 έως 5? *

Να επισημαίνεται μόνο μία έλλειψη.

1	2	3	4	5
◯	◯	◯	◯	◯

25. Τι προσέδωσε στην ατμόσφαιρα περισσότερο κατά τη γνώμη σου? *

Να επισημαίνεται μόνο μία έλλειψη.

- ◯ Οι περιβαλλοντικοί ήχοι?
- ◯ Η μουσική?
- ◯ Ο φωτισμός?
- ◯ Οι χώροι?
- ◯ Οι περιβαλλοντικές συνθήκες?
- ◯ Οι υφές?
- ◯ Η γεωμετρία του χώρου?
- ◯ Το βάθος πεδίου?

26. Ποια στοιχεία από τα παραπάνω πιστεύεις ότι προσέδωσαν γενικά στην ατμόσφαιρα? Μπορείς να επιλέξεις περισσότερο από ένα. *

Επιλέξτε όλα όσα ισχύουν.

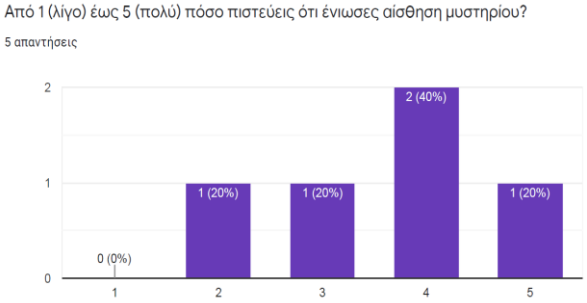
- ☐ Οι περιβαλλοντικοί ήχοι?
- ☐ Η μουσική?
- ☐ Ο φωτισμός?
- ☐ Οι χώροι?
- ☐ Οι περιβαλλοντικές συνθήκες?
- ☐ Οι υφές?
- ☐ Η γεωμετρία του χώρου?
- ☐ Το βάθος πεδίου?

7.3 Ανάλυση των αποτελεσμάτων

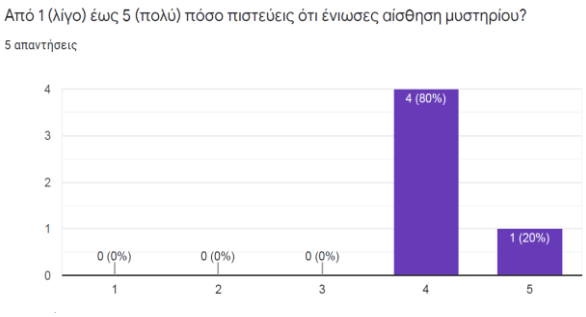
Αν και δεν το περιμέναμε εξαιτίας του μικρού αριθμού συμμετεχόντων του πειράματος, μπορέσαμε να εξάγουμε κάποια ιδιαίτερα ενδιαφέροντα συμπεράσματα.

<u>Στην πίστα 1:</u>

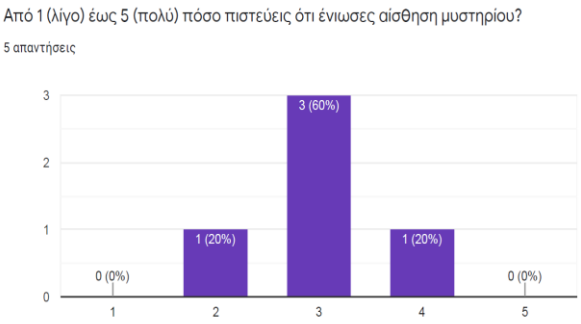
Φάνηκε ότι οι παίκτες ένιωσαν περισσότερο αίσθηση μυστηρίου στα ενδιάμεσα settings.



Team 1



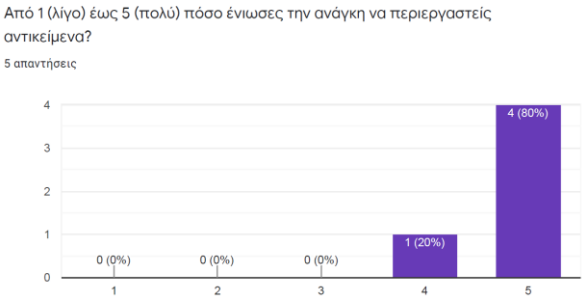
Team 2



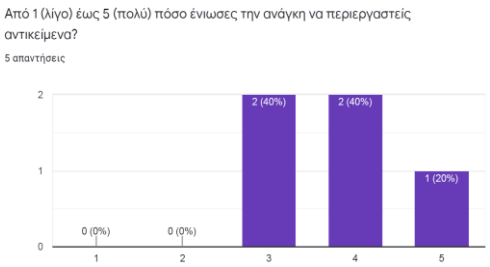
Team 3

Ένιωσαν περισσότερο την ανάγκη να περιεργαστούν αντικείμενα στις ενδιάμεσες ρυθμίσεις (Team 2).

Team 1



Team 2



Team 3

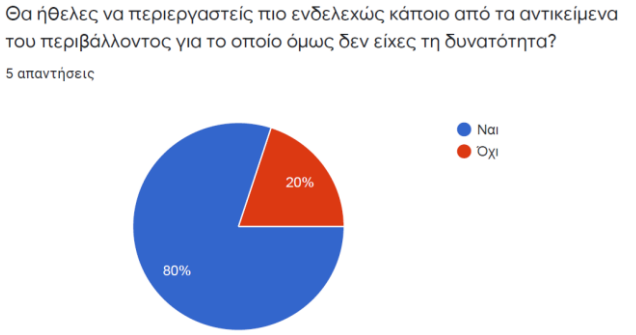
Κοιτούσαν περισσότερο το πάτωμα στις ενδιάμεσες ρυθμίσεις (Team 2).

Team 1

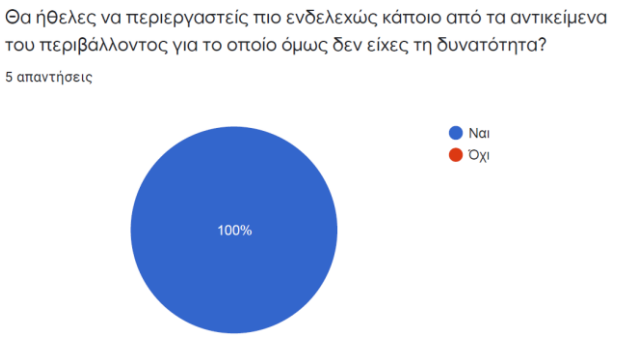
Team 2

Team 3

Ένιωσαν την ανάγκη να περιεργαστούν αντικείμενα για τα οποία όμως δεν υπήρχε η δυνατότητα περισσότερο στο Team 2 (ενδιάμεσες ρυθμίσεις).



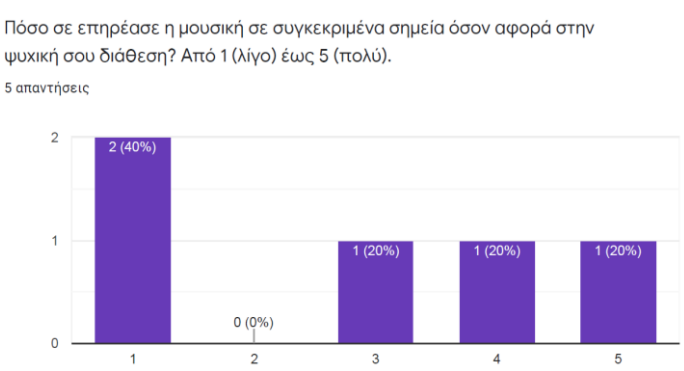
Team 1



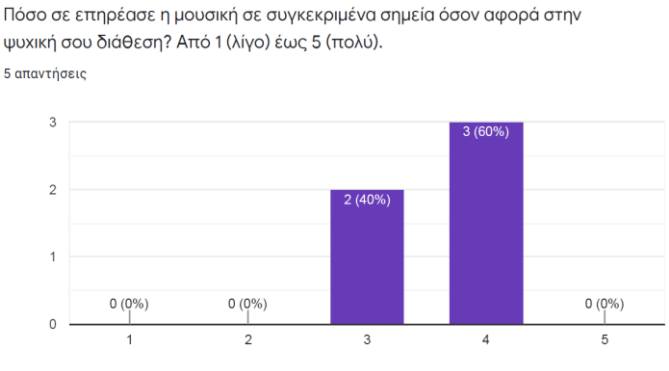
Team 2

Team 3

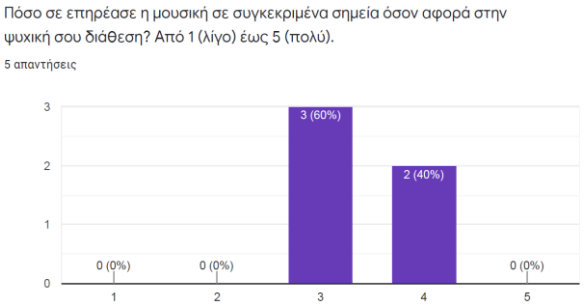
Απάντησαν ότι επηρεάστηκε η ψυχική τους διάθεση περισσότερο από τη μουσική στο Team 2.



Team 1

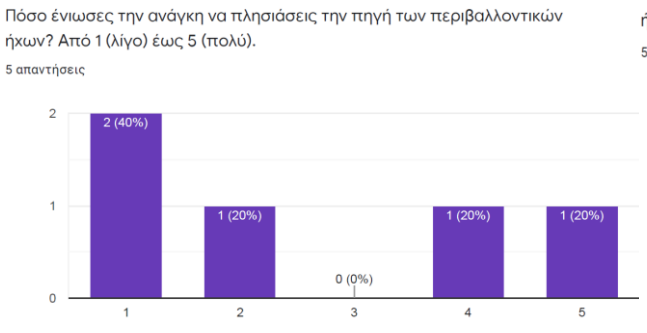


Team 2

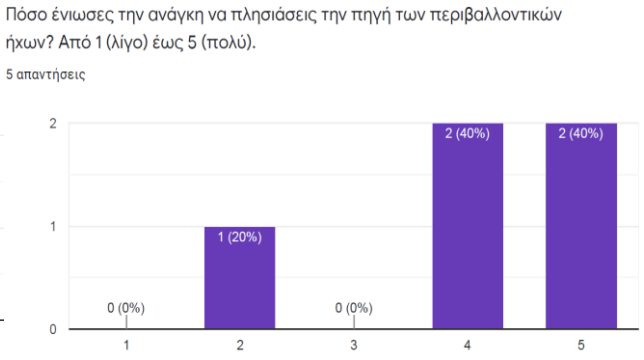


Team 3

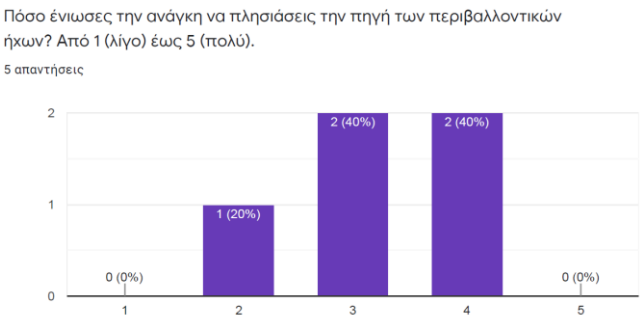
Απάντησαν ότι ένιωσαν περισσότερο την ανάγκη να πλησιάσουν την πηγή των περιβαλλοντικών ήχων στο Team 2.



Team 1

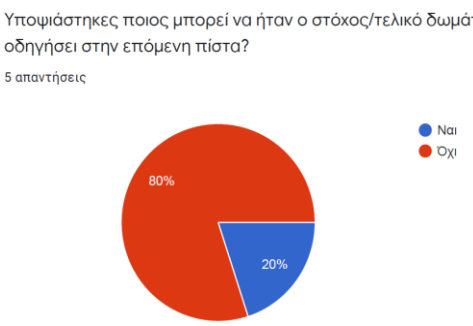


Team 2

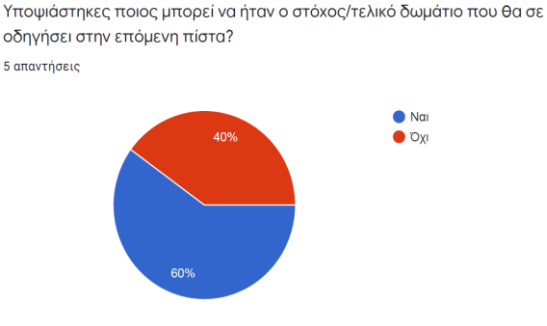


Team 3

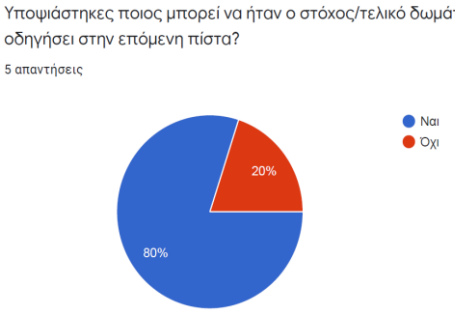
Ήταν πιο δύσκολο να βρουν το τελικό δωμάτιο που θα τους οδηγήσει στην επόμενη πίστα στο Team 1 (πολύ ομίχλη, χαμηλός φωτισμός, χαμηλό contrast).



Team 1



Team 2



Team 3

Στο Team 1 θεώρησαν ότι αυτό που καθιστούσε την εμπειρία περισσότερο ατμοσφαιρική ήταν ο φωτισμός, στο Team 2 οι περιβαλλοντικοί ήχοι και ο φωτισμός και στο Team 3 η μουσική.

Team 1

Team 2

Team 3

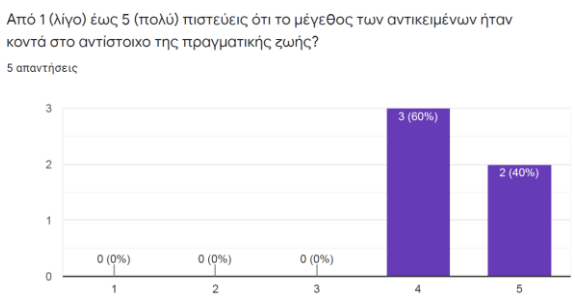
Στην ερώτηση για τι προσέδωσε γενικά στην ατμόσφαιρα με τη δυνατότητα πολλών διαφορετικών επιλογών απαντήσεων, στο Team 1 απάντησαν ο φωτισμός και οι περιβαλλοντικές συνθήκες, στο Team 2 οι περιβαλλοντικοί ήχοι και ο φωτισμός και στο Team 3 οι περιβαλλοντικοί ήχοι και η μουσική.

Team 1

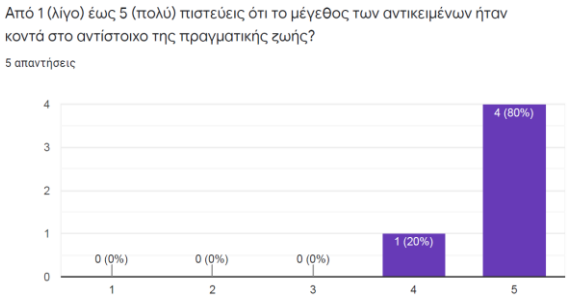
Team 2

Team 3

Στην Πίστα 2 οι απαντήσεις ήταν παρόμοιες με αυτές της Πίστας 1 κι επιπλέον οι συμμετέχοντες στο πείραμα θεώρησαν ότι το μέγεθος των αντικειμένων ήταν κοντά στο αντίστοιχο της πραγματικής ζωής περισσότερο στο Team 2.



Team 1



Team 2

Team 3

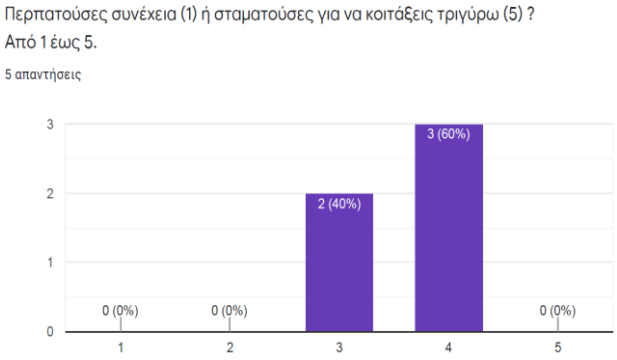
Θεώρησαν ότι προσέδωσαν περισσότερο στην ατμόσφαιρα οι περιβαλλοντικοί ήχοι και οι χώροι στο Team 1, ο φωτισμός στο Team 2 και οι περιβαλλοντικοί ήχοι στο Team 3.

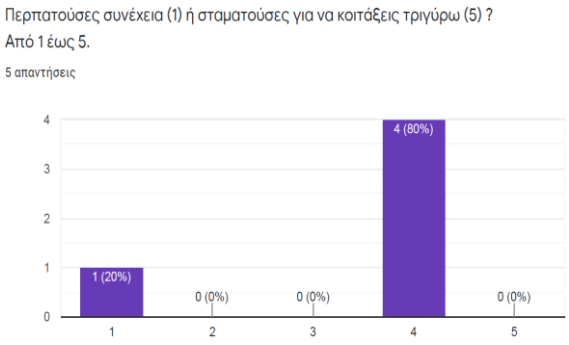
Team 1

Team 2

Team 3

Στην Πίστα 3 οι απαντήσεις ήταν παρόμοιες, ενώ οι συμμετέχοντες σταματούσαν περισσότερο για να κοιτάξουν τριγύρω στο Team 3.





Team 1 Team 2

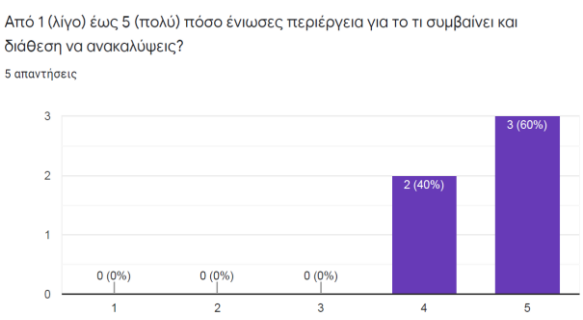
Team 3

Ενώ κοιτούσαν περισσότερο το πάτωμα στο Team 1.

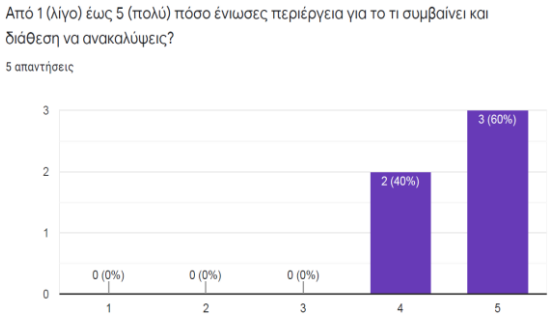
Team 1 Team 2

Team 3

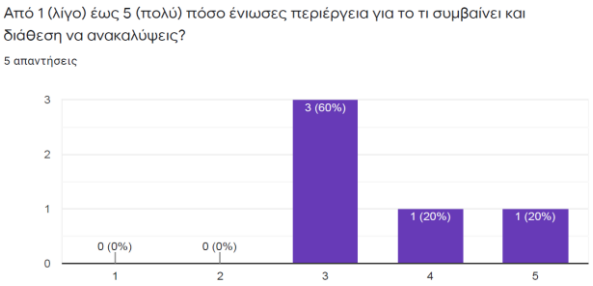
Ένιωσαν την ίδια ακριβώς μέγιστη περιέργεια στο Team 1 και στο Team 2.



Team 1



Team 2



Team 3

Στο Team 1 απάντησαν ότι πρόσθεσαν στην ατμόσφαιρα οι περιβαλλοντικοί ήχοι, ο φωτισμός και οι περιβαλλοντικές συνθήκες, στο Team 2 οι περιβαλλοντικοί ήχοι, ο φωτισμός και οι χώροι και στο Team 3 οι περιβαλλοντικοί ήχοι, η μουσική και οι χώροι.

Team 1

Team 2

Team 3

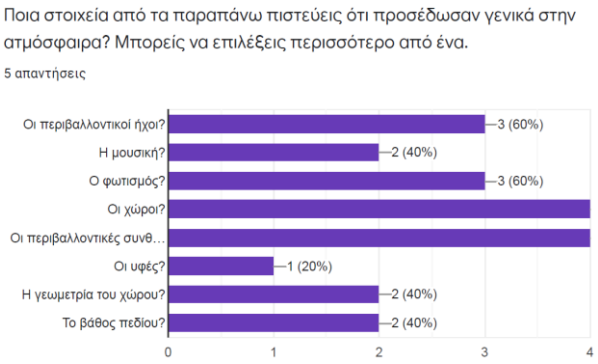
Στην Πίστα 4 οι απαντήσεις ήταν σχετικά παρόμοιες με αυτές από τις προηγούμενες πίστες, με τη διαφορά ότι οι συμμετέχοντες κοιτούσαν περισσότερο το ταβάνι, κάτι που είναι πολύ λογικό, καθώς τα interactable αντικείμενα ήταν προσκολλημένα στο Fatberg το οποίο βρισκόταν στο ταβάνι.

Team 1

Team 2

Team 3

Στην Πίστα 5 οι απαντήσεις ήταν παρόμοιες με εκείνες στις υπόλοιπες πίστες. Οι συμμετέχοντες στο Team 1 απάντησαν πως οι χώροι και οι περιβαλλοντικές συνθήκες προσέδωσαν περισσότερο στην ατμόσφαιρα, στο Team 2 οι περιβαλλοντικοί ήχοι, η μουσική, οι χώροι και ο φωτισμός, στο Team 3 οι χώροι και οι περιβαλλοντικές συνθήκες.



Team 1

Team 2

Team 3

Συμπεράσματα Πειράματος

Από την ανάλυση των αποτελεσμάτων του πειράματος και από την σύγκριση των απαντήσεων που δόθηκαν από τα διαφορετικά teams,

- Φάνηκε αρχικά ότι η μέγιστη ποσότητα ομίχλης, ο πιο χαμηλός φωτισμός και το χαμηλό contrast δυσκόλεψαν τους συμμετέχοντες και τους δημιούργησαν αβεβαιότητα [5] σχετικά με το ποιος ήταν ο τελικός στόχος-τελικό δωμάτιο της κάθε πίστας.
- Ένιωσαν περισσότερο αίσθηση μυστηρίου στις μεσαίες τιμές ομίχλης, φωτισμού και contrast.
- Ένιωσαν περισσότερο την ανάγκη να περιεργαστούν interactables πιο ενδελεχώς στις μεσαίες τιμές φωτισμού, ομίχλης και contrast.
- Κοιτούσαν περισσότερο το πάτωμα, άρα ένιωθαν περισσότερη αβεβαιότητα στις μεσαίες ρυθμίσεις (Team 2) και έψαχναν κάτι ή ήθελαν να καταλάβουν πάνω σε τι πατούσαν.
- Ήθελαν να περιεργαστούν αντικείμενα για τα οποία όμως δεν είχαν τη δυνατότητα, περισσότερο στις μεσαίες τιμές ομίχλης, φωτισμού και contrast.
- Επηρρεάστηκαν περισσότερο από τη μουσική όσον αφορά στην ψυχική τους διάθεση οι συμμετέχοντες του Team 2.
- Ένιωσαν περισσότερο την ανάγκη να πλησιάσουν την πηγή των περιβαλλοντικών ήχων στην Team 2 (ενδιάμεσες τιμές ομίχλης, φωτισμού, contrast).
- Οι συμμετέχοντες του Team 1 θεώρησαν σημαντικότερα ατμοσφαιρικά στοιχεία τις περιβαλλοντικές συνθήκες, τον φωτισμό και άλλα οπτικά στοιχεία. Απ' ότι φαίνεται λόγω της χαμηλής ορατότητας, χρησιμοποίησαν περισσότερο την αίσθηση της όρασης για να αποκτήσουν καλύτερη αντίληψη του περιβάλλοντος.
- Οι συμμετέχοντες του Team 2 θεώρησαν ότι προσέδιδαν στην ατμόσφαιρα και οπτικά, αλλά και ηχητικά στοιχεία του παιχνιδιού (φωτισμός, χώροι, περιβαλλοντικοί ήχοι, μουσική κτλ.), κάτι που μας ωθεί να πιστέψουμε πως στη συγκεκριμένη περίπτωση είχαμε αρκετά καλύτερο βαθμό engagement και μία περισσότερο εμβυθιστική εμπειρία για τον χρήστη, καθώς χρησιμοποιούσε δύο αισθήσεις αυτή τη φορά (όραση και ακοή), κάπως πιο ισορροπημένα.
- Οι συμμετέχοντες του Team 3 έδωσαν περισσότερη σημασία στον ήχο (μουσική και περιβαλλοντικοί ήχοι), καθώς οπτικά ήταν όλα ξεκάθαρα γι' αυτούς με μηδαμινή ποσότητα ομίχλης, δυνατά φώτα και αυξημένο contrast, οπότε το μόνο που τους διέγειρε την περιέργεια ήταν το ηχητικό κομμάτι.

Συμπεράναμε λοιπόν πως παρόλο που η πολύ πυκνή ομίχλη, χαμηλός φωτισμός και χαμηλό contrast του Team 1 δημιούργησαν αβεβαιότητα στους παίκτες, όσον αφορά στον εκάστοτε τελικό στόχο ανά πίστα, οι χρήστες ένιωσαν τη μέγιστη αβεβαιότητα, απορία, διάθεση να ανακαλύψουν, να εξερευνήσουν, να ψάξουν, να περιεργαστούν αντικείμενα-interactables ή μη και αίσθηση μυστηρίου στις μεσαίες και πιο ισορροπημένες τιμές ομίχλης, φωτισμού και contrast δηλαδή στο Team 2. Στα ίδια settings, επιπλέον, φάνηκε πως η εμπειρία ήταν πιο εμβυθιστική [5] για τους συμμετέχοντες, καθώς χρησιμοποίησαν δύο από τις αισθήσεις τους σε σχέση με τα άλλα settings που χρησιμοποίησαν κυρίως μόνο μία (όραση στο Team 1 και ακοή στο Team 3). Η μουσική έγινε περισσότερο αντιληπτή από τους χρήστες και τους επηρέασε περισσότερο ψυχικά, επίσης στις μεσαίες ρυθμίσεις ομίχλης, φωτισμού, contrast και οι συγκεκριμένες τιμές απ' ό,τι φαίνεται δημιούργησαν γι' αυτούς μία πιο ατμοσφαιρική εμπειρία.

Κεφάλαιο 8 Επίλογος

Επιλέγοντας το συγκεκριμένο θέμα διπλωματικής κληθήκαμε να δημιουργήσουμε ένα ολοκληρωμένο βιντεοπαιχνίδι εκ του μηδενός στην Unity Engine. Ήταν μία ιδιαίτερα χρονοβόρα διαδικασία, αλλά έχουμε την πεποίθηση ότι τα οφέλη ήταν πολλαπλάσια. Εμβαθύναμε στην ανάπτυξη βιντεοπαιχνιδιών μέσω της αυξημένης τριβής με την μηχανή και χρήσης της Unity και άλλων εφαρμογών, αναζητήσαμε αρκετά και μελετήσαμε παραδείγματα στο διαδίκτυο, χρησιμοποιήσαμε δημιουργική σκέψη για να ξεπεράσουμε τα εμπόδια για τα οποία δεν υπήρχε υπαρκτή λύση κάπου στο διαδίκτυο, μάθαμε αρκετά best practices και τεχνικές βελτιστοποίησης, είχαμε επαφή και μάθαμε πώς λειτουργούν πολλά από τα διαφορετικά συστήματα της μηχανής Unity και προσπαθήσαμε με το συνδυασμό της χρήσης αυτών, να δημιουργήσουμε μία ολοκληρωμένη αισθητικά εμπειρία για τον χρήστη, πήραμε πολύτιμες συμβουλές από τον επιβλέποντα καθηγητή που ήμαστε σε συνεννόηση, για το πώς θα μπορέσουμε να κάνουμε trigger το συναίσθημα αβεβαιότητας στον χρήστη με διαφορετικές επιλογές αρχιτεκτονικής χώρων και φωτισμού και άλλες προτάσεις κατά τις συναντήσεις μας, που ήταν ιδιαίτερα βοηθητικές και μας έδιναν κατεύθυνση και πολύτιμες γνώσεις κατά τη διάρκεια της ανάπτυξης, μάθαμε τεχνικές που χρησιμοποιούν επαγγελματίες του χώρου, αποκτήσαμε μία νέα αντίληψη για επίλυση προβλημάτων όσον αφορά στα βιντεοπαιχνίδια και αυξημένο επίπεδο κατανόησης του τι συμβαίνει στην πραγματικότητα και με ποιόν τρόπο έχουν αναπτυχθεί αγαπημένα εξ αυτών ή τμήματά τους και πήραμε γνώσεις που δεν είχαμε πριν την έναρξη της ανάπτυξης του βιντεοπαιχνιδιού μας. Χρησιμοποιήσαμε την φαντασία μας και μελέτη πηγών και υλικού από το διαδίκτυο για να δημιουργήσουμε το περιβάλλον ενός υπονόμου, επιλέξαμε ήχους και textures από τεράστιες δεξαμενές αρχείων, δημιουργήσαμε γρίφους για τον παίκτη σύμφωνα με τα διαθέσιμα mechanics, αλλά και τους χώρους που είχαμε δημιουργήσει, σε μία προσπάθεια να προκαλέσουμε immersion και να δημιουργήσουμε ένα συγκεκριμένο συναίσθημα στον χρήστη. Όλα τα παραπάνω αποτέλεσαν, μαζί με το πείραμα, μία ιδιαίτερα χρήσιμη και πολλές φορές διασκεδαστική εμπειρία που πιστεύουμε ότι μας προσέφερε εφόδια, feedback και πολύτιμες γνώσεις, ενώ τροφοδότησε την επιθυμία μας για μελλοντική ανάπτυξη και άλλων βιντεοπαιχνιδιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] The Feeling of Uncertainty Intensifies Affective Reactions --- Wilson and Gilbert, 2009
[2] The Puzzle Instinct: The Meaning of Puzzles in Human Life --- Marcel Danesi, 2004
[3] Belief, disbelief and uncertainty activate distinct brain regions --- Wiley, 2007
[4] The Relationship Between Uncertainty and Affect --- Eric C. Anderson, R. Nicholas Carleton, Michael Diefenbachand and Paul K. J. Han.
[5] Investigating Uncertainty in Digital Games and Its Impact On Player Immersion --- Shringi Kumari, University of York
[6] Uncertainty in Games --- Greg Costikyan, 2015
[7] SpeedTutor --- **https://www.youtube.com/c/SpeedTutor**
[8] Wikipedia, the free encyclopedia --- **https://en.wikipedia.org/wiki/**
[9] Unity 5 Shader Programming #1: An introduction to shaders --- digitalerr0r, 2015
[10] Unity Learn --- https://learn.unity.com/
[11] Rendering Pipeline In Unity --- https://medium.com/shader-coding-in-unity-from-a-to-z/rendering-pipe-line-f0471aa0904b
[12] Weather Effects Shader – Ground Water --- http://untitledgam.es/2017/02/weather-effects-shader-ground-water/
[13] Uncertainty in games – potential benefits and disadvantages --- Katarzyna Skok, 2020
[14] VideoGameSpaces: Η εμπειρία του χώρου --- Φλώρου Αθανασία, 2013
[15] Unity Forum --- https://forum.unity.com/
[16] Η Ιστορία των Βιντεοπαιχνιδιών --- Unboxholics
[17] Blender Guru --- **https://www.youtube.com/channel/UCOKHwx1VCdgnxwbjyb9Iu1g**
[18] Catlikecoding --- https://catlikecoding.com/unity/tutorials
[19]Code for memory recorder playback coroutine from --- https://gamedev.stackexchange.com/questions/141356/move-gameobject-with-defined-lists-of-position-and-rotation
[20] Η Ψυχολογία της Αβεβαιότητας: Τι είναι και πώς την αντιμετωπίζουμε --- https://www.psychografimata.com/η-ψυχολογία-της-αβεβαιότητας-τι-είναι/
[21] Impressive Storm Sewer System / Saitama, Japan --- https://www.archdaily.com/3591/giant-storm-sewer-system-sitama-japan --- Archdaily, 2008
[22] https://gamedev.stackexchange.com/questions/23/what-is-ambient-occlusion
[23] IL2CP - https://docs.unity3d.com/2019.3/Documentation/Manual/IL2CPP.html
[24]OneBatch --- https://docs.google.com/document/d/1SSgraFsomMqzaqoMBazGOgrpV9ijhuk1DFvxn7cOan8/edit?usp=sharing
[25] GIMP - https://el.wikipedia.org/wiki/GIMP
[26] Audacity - https://en.wikipedia.org/wiki/Audacity_(audio_editor)
[27] https://el.wikipedia.org/wiki/Blender