Πολυτεχνειο Κρητης

Σχολη Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

Σχεδιασμός και Υλοποίηση μιας Διαδραστικής Τρισδιάστατης Προσομοίωσης του Επιτραπέζιου Παιχνιδιού "Cluedo" με Χρήση Αυτόνομων Λογισμικών Αντιπάλων

Αλέξανδρος Μαυρομμάτης

Εξεταστική Επιτροπή

Αναπ. Καθ. Αικατερίνη Μανιά (ΗΜΜΥ)

Επικ. Καθ. Αντώνιος Δεληγιαννάκης (ΗΜΜΥ)

Αναπ. Καθ. Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)

Χανιά, Σεπτέμβριος 2013

Technical University of Crete, Greece
School of Electronic and Computer Engineering

Design and Implementation of an Interactive 3D Simulation of the Board Game "Cluedo" Using Autonomous Software Opponents

Alexandros Mavrommatis

Thesis Committee
Associate Professor Katerina Mania (ECE)
Assistant Professor Antonios Deligiannakis (ECE)
Associate Professor Michail G. Lagoudakis (ECE)

Chania, September 2013

Περίληψη

Στη παρούσα διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε ένα τρισδιάστατο παιχνίδι για προσωπικούς υπολογιστές, βασισμένο στο παγκόσμια διαδεδομένο αγγλικό επιτραπέζιο παιχνίδι *Cleudo*. Το *Cluedo* είναι ένα επιτραπέζιο παιχνίδι στρατηγικής στο οποίο κάθε παίχτης, μέσα από μια σειρά υποθέσεων, έχει ως στόχο να βρει τη λύση ενός μυστήριου φόνου. Αρχικά σχεδιάστηκε η γεωμετρία του παιχνιδιού με χρήση του εργαλείου Autodesk 3ds Max καθώς και οι χαρακτήρες των παιχτών. Στη συνέχεια με χρήση της μηχανής παιχνιδιών Unity3D υλοποιήθηκε η λειτουργία του παιχνιδιού. Τοποθετήθηκαν υφές, ήχοι, κινούμενα σχέδια (animations), εφφέ, τα οποία έδωσαν "ζωή" στο παιχνίδι μας. Έπειτα υλοποιήθηκαν οι υπομονάδες του παιχνιδιού όπως του διαχειριστή παιχνιδιού, διαχειριστή παίχτη και σχεδίασης των γραφικών διεπαφών. Τέλος υλοποιήθηκε και η τεχνητή νοημοσύνη των αντίπαλων παιχτών που τους έδωσε πλήρη αυτονομία κατά τη διάρκεια του παιχνιδιού. Η τεχνητή νοημοσύνη περιείχε την υλοποίηση ενός αλγορίθμου Path Planning, για την έξυπνη κίνηση των παιχτών, καθώς και μιας λογικής, μέσω συναρτήσεων χρησιμότητας, που θα ακολουθήσουν οι παίχτες κατά τη διάρκεια του παιχνιδιού. Το παιχνίδι μετά την ολοκλήρωσή του, δοκιμάστηκε από ένα πλήθος χρηστών διαφορετικών φύλων, ηλικιών και επαγγελμάτων και αξιολογήθηκε μέσω των ερωτηματολογίων QUIS (Questionnaire for User Interaction Satisfaction).

Abstract

In this diploma thesis we designed and implemented a 3D game for personal computers based on the worldwide popular English board game *Cleudo*. *Cluedo* is a strategy board game in which every player, in a series of suggestions, aims to find the solution of a mysterious murder. In the beginning, we designed the geometry of the game, using the graphics software Autodesk 3ds Max, as also the avatars of the players. Later we implemented the function of the game using the game engine Unity3D. We added textures, sounds, animations, effects, that gave "life" to our game. After that, we implemented the components of the game such as the game and player operators and the design of graphical interface. At the end, we implemented the artificial intelligence of the opponents that gave them full autonomy during the game. The artificial intelligence includes the implementation of a Path Planning algorithm, for the players to move smartly, and the implementation of a logic, via utility functions, that the players will follow during the game. The game, after its completion, it was tested by a number of users of different sexes, ages and occupations, and it was evaluated via the QUIS (Questionnaire for User Interaction Satisfaction).

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια Δρ. Κατερίνα Μανιά για την επίβλεψή της κατά την εκπόνηση της διπλωματικής μου εργασίας, καθώς και ότι μου έδωσε την ευκαιρία να κάνω πράξη μια ιδέα μου.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους καθηγητές της επιτροπής μου Δρ. Αντώνιο Δεληγιαννάκη και Δρ. Μιχαήλ Λαγουδάκη για το χρόνο που αφιέρωσαν στην ανάγνωση του κειμένου και ιδιαίτερα τον κ. Λαγουδάκη για την πολύτιμη βοήθειά του πάνω στα θέματα τεχνητής νοημοσύνης.

Θα ήθελα επίσης να ευχαριστήσω τον Γεώργιο-Αλέξανδρο Κουλιέρη για την βοήθειά του πάνω σε θέματα φωτισμού και σκιάσεων, και την φίλη μου Στέλλα Μαροπάκη για το χρόνο που αφιέρωσε στην δοκιμή του παιχνιδιού.

Ένα μεγάλο ευχαριστώ στον κ. Μανώλη Νταουντάκη, Group Brand Manager της εταιρείας Hasbro Hellas S.A. για τα όσα έκανε ώστε να πάρω την έγκριση από την εταιρεία Hasbro Inc. για την δημιουργία του παιχνιδιού.

Τέλος ευχαριστώ πολύ την οικογένειά μου και τους φίλους μου που με στήριξαν στα έξι χρόνια της φοίτησής μου.

Σεπτέμβριος 2013
Αλέξανδρος Μαυρομμάτης

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Κατάλογος Αλγορίθμων

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Στόχος της παρούσας διπλωματικής εργασίας ήταν η δημιουργία μιας τρισδιάστατης εφαρμογής που θα προσομοιώνει το παιχνίδι *Cleudo* της εταιρείας Hasbro. Στο τίτλος της εργασίας αναφέρονται οι όροι "διαδραστική" προσομοίωση και "αὐτόνομοι" αντίπαλοι. Με τον πρώτο όρο εννοούμε ότι ο χρήστης αλληλεπιδρά με το παιχνίδι και μπορεί ανάλογα με τις κινήσεις του να αλλάξει τον ρου των γεγονότων. Καταστρώνει την στρατηγική του με σκοπό να εμποδίσει τους αντιπάλους να φτάσουν προς τη λύση και να κερδίσει αυτός. Με τον δεύτερο όρο εννοούμε ότι οι αντίπαλοι έχουν πλήρη αυτονομία κατά τη διάρκεια του παιχνιδιού ως προς τις κινήσεις τους, καθώς και ως προς τις αποφάσεις που θα πάρουν. Στόχος κάθε αντιπάλου είναι να παραπλανήσει τους υπολοίπους και να φτάσει και αυτός με τη σειρά του πρώτος στη λύση. Το παιχνίδι ανήκει στη κατηγορία των ψηφιακών επιτραπέζιων παιχνιδιών και η μορφή που έχουν τέτοιου είδους παιχνίδια αναφέρεται αναλυτικότερα παρακάτω [ενότητα 2.3].

1.2 Σύντομη Περιγραφή

Το *Cleudo* είναι ένα επιτραπέζιο παιχνίδι μυστηρίου [1]. Ο χρήστης διαχειρίζεται έναν παίχτη ο οποίος βρίσκεται σε μια έπαυλη στα προάστια της πόλης στην οποία μόλις έχει λάβει χώρα μια δολοφονία. Πρέπει μέσα από έναν κύκλο υποθέσεων να λύσει το μυστήριο αυτής της δολοφονίας.

Σχήμα 1.1: Η εξέλιξη του *Cleudo* στο πέρασμα των χρόνων
Αριστερά: Το *Cleudo* στη δεκαετία του ′60 **Κέντρο:** Η έκδοση του 2003 **Δεξιά:** Η τελευταία έκδοση του 2013

Σχήμα 1.2: Περιεχόμενα επιτραπέζιου παιχνιδιού

Το *Cleudo* επινοήθηκε από έναν υπάλληλο δικηγορικού γραφείου το 1944 στο Leeds της Αγγλίας. Στόχος του ήταν το παιχνίδι να παίζεται κατά τη διάρκεια των πολύωρων παραμονών των πολιτών στις υπόγειες στοές κατά το βομβαρδισμό της πόλης από τους Γερμανούς στον δεύτερο παγκόσμιο πόλεμο. Πρωτοεκδόθηκε στην Αγγλία στα τέλη της δεκαετίας του ′40 και έκτοτε έχει μεταφερθεί και μεταφραστεί σε πολλές χώρες του κόσμου και έχει γίνει πηγή έμπνευσης βιβλίων, ταινιών, βιντεοπαιχνιδιών καθώς και διαδραστικής τηλεοπτικής σειράς. Κατά τη μεταφορά του και κατά το πέρασμα των δεκαετιών έχουν γίνει πολλές μετατροπές και εξελίξεις (σχήματα 1.1 και 1.2).

Παρακάτω παρατίθενται συνοπτικά οι οδηγίες του παιχνιδιού [2]:

1.2.1 Κανόνες Παιχνιδιού

Το παιχνίδι περιέχει 21 κάρτες. 6 κάρτες για τους υπόπτους, 6 για τα φονικά όπλα και 9 για τα δωμάτια. Στην αρχή επιλέγεται τυχαία μια κάρτα από κάθε κατηγορία και τοποθετείται στον φάκελο της λύσης. Οι υπόλοιπες κάρτες ανακατεύονται και μοιράζονται τυχαία και ισομερώς στους υπόλοιπους παίχτες.

Στόχος: Ο κύριος Black-το θύμα του παιχνιδιού-βρέθηκε νεκρός σε ένα από τα δωμάτια της έπαυλής του. Για να κερδίσει κάποιος πρέπει να απαντήσει στις εξής τρεις ερωτήσεις: Ποιός διέπραξε το φόνο; Που; Με τι όπλο;

Κινώντας τον παίχτη: Η Δεσποινίς Φλόγα παίζει πάντα πρώτη. Μετά, το παιχνίδι συνεχίζεται με τη φορά του ρολογιού. Σε κάθε γύρο, προσπαθείς να φτάσεις σε ένα διαφορετικό δωμάτιο της έπαυλης. Για να ξεκινήσει ο γύρος σου, κίνησε το παίχτη σου ανάλογα με τη ζαριά σου ή αν είσαι σε γωνιακό δωμάτιο, χρησιμοποίησε τα μυστικά περάσματα.

Ρίχνοντας το ζάρι: Ρίξε το ζάρι και προχώρησε τον παίχτη σου το νούμερο των τετραγώνων που έφερες. Μπορείς να κινηθείς οριζόντια ή κάθετα, εμπρός ή πίσω, αλλά όχι διαγώνια. Μπορείς να αλλάξεις κατεύθυνση όσες φορές θέλεις, όσο το επιτρέπει η ζαριά σου. Παρ΄ όλα αυτά, δεν μπορείς να εισέλθεις στο ίδιο τετράγωνο δύο φορές στον ίδιο γύρο. Δεν μπορείς να πατήσεις σε τετράγωνο που είναι ήδη κατειλημμένο από έναν άλλον ύποπτο.

Μυστικά περάσματα: Τα δωμάτια που βρίσκονται στις αντίθετες γωνίες της έπαυλης συνδέονται με μυστικά περάσματα. Αν είσαι σε ένα από αυτά τα δωμάτια στην αρχή του γύρου σου, μπορείς αν θέλεις, να χρησιμοποιήσεις ένα μυστικό πέρασμα αντί να ρίξεις το ζάρι. Για να κινηθείς σε ένα μυστικό πέρασμα, ανακοινώνεις την επιθυμία σου στους υπόλοιπους υπόπτους και μεταφέρεις τον παίχτη σου στο δωμάτιο της αντίθετης γωνίας.

Είσοδος και έξοδος από ένα δωμάτιο: Μπορείς να μπεις ή να βγεις από ένα δωμάτιο ρίχνοντας το ζάρι και περνώντας από μια πόρτα, ή περνώντας από ένα μυστικό πέρασμα. Μια πόρτα είναι το άνοιγμα στον τοίχο και όχι το τετράγωνο που βρίσκεται μπροστά στο κατώφλι της. Όταν φεύγεις από μια πόρτα, δεν υπολογίζεις την ίδια την πόρτα σαν τετράγωνο. Δεν μπορείς να περάσεις από πόρτα που εμποδίζεται από έναν

αντίπαλο. Δεν μπορείς να μπεις δύο φορές στο ίδιο δωμάτιο σε ένα μόνο γύρο. Είναι πιθανόν οι αντίπαλοι σου να εμποδίζουν μια ή και όλες τις πόρτες του δωματίου που βρίσκεσαι και να σε παγιδέψουν μέσα. Αν συμβεί αυτό, πρέπει να περιμένεις κάποιο παίχτη να κινηθεί και να ξεμπλοκάρει κάποια πόρτα ώστε να μπορείς να φύγεις.

1.2.2 Υπόθεση

Κάνοντας μια υπόθεση: Μόλις μπεις σε ένα δωμάτιο, κάνε μια υπόθεση. Κάνοντας υποθέσεις κατά τη διάρκεια του παιχνιδιού, προσπαθείς να καταλάβεις-με τη μέθοδο της απαλοιφής-ποιές τρεις κάρτες είναι στον φάκελο της λύσης. Για να κάνεις μια υπόθεση, μετακινείς έναν ύποπτο και ένα όπλο στο δωμάτιο που έχεις εισέλθει. Τότε υποθέτεις ότι το έγκλημα έγινε σε αυτό το δωμάτιο, με αυτό τον ύποπτο και με αυτό το όπλο.

Για παράδειγμα ας πούμε ότι είσαι η Δεσποινίς Φλόγα και έχεις εισέλθει στο Σαλόνι. Πρώτα μετακινείς έναν ύποπτο-Αιδεσιμότατος Πράσινος λόγου χάρη-στο Σαλόνι. Μετά χρησιμοποιείς ένα όπλο-το Γαλλικό κλειδί-στο Σαλόνι.

Δύο σημαντικά πράγματα: Πρέπει να είσαι στο δωμάτιο που αναφέρεις στην υπόθεση σου. Έχε στο νου σου όλους τους παίχτες ως υπόπτους, ακόμα και τον εαυτό σου.

Αποδεικνύοντας μια υπόθεση αληθή ή ψευδή: Μόλις κάνεις την υπόθεσή σου, οι αντίπαλοι σου με τη σειρά τους, προσπαθούν να αποδείξουν ότι είναι ψευδής. Ο πρώτος που προσπαθεί είναι ο παίχτης στα αριστερά σου. Αυτός ο παίχτης κοιτάει τις κάρτες του και κοιτάει αν έχει μια κάρτα από αυτές που ονομάτισες στην υπόθεσή σου. Αν έχει, τότε ο παίχτης αυτός πρέπει να τη δείξει μόνο σε εσένα. Αν έχει παραπάνω από μια, τότε επιλέγει μια από αυτές να σου δείξει. Αν δεν έχει καμία κάρτα, τότε την ευκαιρία να αποδείξει λάθος την υπόθεση σου, την έχει ο επόμενος από τα αριστερά παίχτης κ.λπ. Μόλις ένας αντίπαλος σου δείξει μια κάρτα από αυτές που ονομάτισες, τότε ξέρεις ότι δεν μπορεί να βρίσκεται στον φάκελο.

Τελείωσε τον γύρο σου βγάζοντας την συγκεκριμένη κάρτα από το σημειωματάριό σου (*notebook*). Αν κανένας παίχτης δεν μπόρεσε να αποδείξει ψευδή την υπόθεσή σου, μπορείς ή να τελειώσεις τον γύρο σου είτε να κάνεις μια κατηγορία [υποενότητα 1.2.3].

Ειδικές σημειώσεις για τις υποθέσεις:

1. Όταν κάνεις μια υπόθεση, μπορείς αν θέλεις, να ονοματίσεις μια ή περισσότερες κάρτες από αυτές που έχεις στο χέρι σου. Μπορεί έτσι να κερδίσεις πληροφορίες ή να παραπλανήσεις τους αντιπάλους.
2. Μπορείς να κάνεις μια υπόθεση και μια κατηγορία στον ίδιο γύρο.
3. Κάνεις **μόνο** μια υπόθεση όταν εισέρχεσαι σε ένα συγκεκριμένο δωμάτιο. Για να κάνεις την επόμενη υπόθεσή σου, πρέπει να εισέλθεις σε ένα διαφορετικό δωμάτιο ή κάποια στιγμή μετά τον επόμενο γύρο να επιστρέψεις στο συγκεκριμένο δωμάτιο που μόλις άφησες. **Δεν** μπορείς να περάσεις ένα γύρο ώστε να μείνεις σε ένα δωμάτιο. Αλλά αν είσαι παγιδευμένος σε ένα δωμάτιο επειδή οι αντίπαλοί σου, σου έχουν μπλοκάρει τις πόρτες, πρέπει να παραμείνεις εκεί μέχρι μια πόρτα να ξεμπλοκαριστεί και να βγεις έξω.
4. Μπορείς να κάνεις μια υπόθεση με έναν ύποπτο ή ένα αντικείμενο που βρίσκεται ήδη στο δωμάτιο. Σε αυτή την περίπτωση δεν θα μετακινηθούν. Αλλά αν η μετακίνηση είναι απαραίτητη, τα όπλα και οι ύποπτοι μένουν στο δωμάτιο που μεταφέρθηκαν μετά την υπόθεση.
5. Αν εσύ είσαι ο ύποπτος που μεταφέρθηκε, μπορείς στον επόμενο γύρο σου ή να βγεις από το δωμάτιο με τους συνήθεις τρόπος **είτε** να κάνεις μια υπόθεση σε αυτό το δωμάτιο. Αν αποφασίσεις να κάνεις εκεί μια υπόθεση **μην** ρίξεις το ζάρι σου.
6. Δεν υπάρχει όριο στο πόσοι παίχτες ή όπλα μπορούν να βρίσκονται στο ίδιο δωμάτιο, την ίδια στιγμή.

1.2.3 Κατηγορία

Κάνοντας μια κατηγορία: Όταν νομίζεις ότι έχεις καταλάβει ποιές τρεις κάρτες κρύβονται στον φάκελο, μπορείς στον γύρο σου, να κάνεις μια κατηγορία και να ονοματίσεις οποιαδήποτε τρία στοιχεία θέλεις. Λες "Κατηγορώ τον/την (ύποπτος) ότι διέπραξε το έγκλημα στο (δωμάτιο) με (όπλο)" .

Μετά, μόνο εσύ και κανείς άλλος, βλέπεις τις κάρτες του φακέλου. Σε μια υπόθεση το δωμάτιο που ονοματίζεις πρέπει να είναι το δωμάτιο που βρίσκεται ο παίχτης σου. Αλλά στην κατηγορία μπορείς να ονοματίσεις οποιοδήποτε δωμάτιο.

Προσοχή: Μπορείς να κάνεις μόνο μια κατηγορία κατά τη διάρκεια του παιχνιδιού.

Αν η κατηγορία σου είναι λανθασμένη: Αν οποιαδήποτε από τις κάρτες που ονομάτισες δεν βρίσκεται στην λύση, τότε έχεις χάσει το παιχνίδι.

Νίκη: Κερδίζεις το παιχνίδι αν η κατηγορία σου είναι εντελώς σωστή, δηλαδή αν βρήκες στο φάκελο όλες τις κάρτες που ονομάτισες.

1.3 Πλατφόρμα

Το παιχνίδι που υλοποιήθηκε υποστηρίζεται από υπολογιστές με λειτουργικό σύστημα *Microsoft Windows* και *Linux* στις εκδόσεις του που δέχονται το πρόγραμμα *Wine* ή αντίστοιχων. Οι υπολογιστές πρέπει να έχουν κάρτα γραφικών που υποστηρίζει τρισδιάστατα γραφικά με DirectX 9 [υποενότητα 2.4.1] ή μεγαλύτερο. Τέλος, θεωρήθηκε περιττή η υλοποίηση του παιχνιδιού για λειτουργικά συστήματα *Mac OS* αφού δημιουργήθηκε μόνο για εκπαιδευτικούς σκοπούς και η ανάπτυξη παιχνιδιών πάνω σε αυτά τα λειτουργικά συστήματα είναι πολύ μικρότερη. Η υλοποίηση του παιχνιδιού για πλατφόρμες κινητών θα ήταν ανέφικτη λόγω των χαμηλών υπολογιστικών πόρων.

1.4 Βασικά Τεχνικά Χαρακτηριστικά

Το παιχνίδι αποτελείται από τα παρακάτω συστήματα τα οποία είναι τα δομικά του συστατικά που συνδέθηκαν ώστε να δημιουργήσουν μια ενιαία εφαρμογή.

1.4.1 Σύστημα Γραφικών

Ήταν το σύστημα που χρησιμοποιήθηκε για να δημιουργήσει το τρισδιάστατο γραφικό περιβάλλον της εφαρμογής μας. Αποτελείται από λογισμικά και εργαλεία [κεφάλαιο 3] που δημιούργησαν αντικείμενα, τρισδιάστατα μοντέλα, χαρακτήρες· εισήγαγαν υφές, γραμματοσειρές και ότι χρειάζεται ένα παιχνίδι για να έχει μια ολοκληρωμένη εικόνα [ενότητα 4.3].

1.4.2 Σύστημα Φυσικής

Χρησιμοποιήθηκε το σύστημα φυσικής *PhysX* της εταιρείας γραφικών *NVIDIA* [3] που βρίσκεται ενσωματωμένο στη μηχανή παιχνιδιών που χρησιμοποιήσαμε [ενότητα 3.3]. Μας προσφέρει μοναδική και άμεσα φυσική και ρεαλιστική συμπεριφορά των αντικειμένων, με πολλές δυνατότητες όπως συγκρούσεις, επαφές, βαρύτητα, κινηματικά αντικείμενα κ.α. [ενότητες 2.10, 5.2 και υποενότητα 3.5.10].

1.4.3 Σύστημα Ήχου

Με αυτό το σύστημα μέσα από έναν ακροατή τοποθετημένο πάντα στην ενεργή κάμερα και μέσω αρκετών πηγών ήχου, μπορέσαμε να αναπαράγουμε τα ηχητικά μας αρχεία που προσέφεραν και αυτά με τη σειρά τους ρεαλιστικότητα και λειτουργικότητα στο παιχνίδι [υποενότητα 3.5.2 και ενότητα 5.8].

1.4.4 Σύστημα Φωτισμού

Με το σύστημα αυτό δώσαμε φως στα τρισδιάστατα γραφικά μας μέσω διαφόρων τύπου πηγών φωτός. Στο τέλος χρησιμοποιήσαμε το χάρτη χαρτογράφησης (*ligthmapping*), για να μειωθεί ο υπολογιστικός χρόνος επεξεργασίας του φωτός [ενότητα 2.6 και υποενότητες 3.3.3, 3.5.6].

1.4.5 Σύστημα Διαχείρισης Παιχνιδιού

Το σύστημα που διαχειρίζεται, ελέγχει τους κανόνες και εκτελεί όλες τις απαραίτητες λειτουργίες που χρειάζεται το παιχνίδι [ενότητα 5.9].

1.4.6 Σύστημα Διαχείρισης Παίχτη

Το σύστημα που καθορίζει το τρόπο εκτέλεσης και τη συμπεριφορά που ακολουθεί ο παίχτης, είτε είναι ο χρήστης είτε λογισμικός αντίπαλος, βάσει του διαχειριστή του παιχνιδιού και κατά τη διάρκειά του [ενότητα 5.10].

1.4.7 Σύστημα Γραφικής Διεπαφής Χρήστη

Το σύστημα που δημιουργεί όλο το απαραίτητο περιβάλλον διαχείρισης του παιχνιδιού από το χρήστη. Ξεκινάει από το μενού του παιχνιδιού και καταλήγει μέχρι τη διεπαφή κατά την εκτέλεσή του [ενότητες 4.4, 4.5, 5.6, 5.7].

1.4.8 Σύστημα Τεχνητής Νοημοσύνης

Τελευταίο αλλά ένα από τα κυριότερα συστήματα του παιχνιδιού. Δίνει την λειτουργικότητα και την αυτονομία στους λογισμικούς αντιπάλους να αντιδρούν στο παιχνίδι σαν ένας ορθολογικός πράκτορας· από το να κινούνται μέχρι να παίρνουν αποφάσεις [ενότητες 2.12, 5.11].

1.5 Δομή Εργασίας

Τελειώνοντας τον πρώτο αυτό κεφάλαιο-το οποίο αποτελεί εισαγωγικό στη εργασία μας -αναφέρουμε συνοπτικά τα επόμενα έξι που ακολουθούν:

- Στο κεφάλαιο 2 γίνεται μια συνοπτική αναφορά των γνώσεων που χρειάζεται να έχει κάποιος για να κατανοήσει την αρχιτεκτονική που σχεδιάσαμε και τον τρόπο που την υλοποιήσαμε. Κάνουμε μια αναδρομή στα βιντεοπαιχνίδια, αναφέρουμε τις βασικές έννοιες στα γραφικά και καταλήγουμε αναφερόμενοι σε μηχανές και λογισμικά γραφικών.

- Στο κεφάλαιο 3 περιγράφονται περιληπτικά τα εργαλεία που χρησιμοποιήσαμε στο σχεδιασμό και την υλοιποίηση του παιχνιδιού, καθώς και τις βασικές λειτουργιές τους που μας χρησίμευσαν.

- Στο κεφάλαιο 4 αναφέρεται αναλυτικά ο τρόπος που σχεδιάστηκε η γεωμετρία του παιχνιδιού και οι χαρακτήρες των παιχτών. Επίσης γίνεται μια πλήρης παρουσίαση του γραφικού περιβάλλοντος της εφαρμογής καθώς και της γραφικής διεπαφής του χρήστη.

- Στο κεφάλαιο 5 περιγράφεται η υλοποίηση του παιχνιδιού στα επιμέρους τμήματα του. Αναλύεται ο τρόπος με τον οποίο χρησιμοποιήθηκαν οι υπομονάδες των εργαλείων και ο τρόπος που υλοποιήθηκε η γραφική διεπαφή, οι διαχειριστές των παιχτών και του

παιχνιδιού. Τέλος αναλύεται η τεχνητή νοημοσύνη των αντιπάλων. Ποιοί αλγόριθμοι[1] υλοποιήθηκαν και για ποιό σκοπό.

- Στο κεφάλαιο 6 περιγράφεται η μέθοδος αξιολόγησης που χρησιμοποιήσαμε για το παιχνίδι· τι αποτελέσματα πήραμε και ποιά συμπεράσματα απορρέουν.

- Στο κεφάλαιο 7 και τελευταίο κεφάλαιο, κλείνουμε την εργασία με αναφορά στο κατά πόσο επιτεύχθηκαν οι στόχοι εν συγκρίσει με τα αποτελέσματα και αναφέρονται πιθανές μελλοντικές βελτιώσεις και επεκτάσεις στο παιχνίδι.

[1]Με τον όρο "Αλγόριθμος", από εδώ και στο εξής, θα αναφερόμαστε σε οποιαδήποτε αυτοματοποιημένη διαδικασία υπολογισμού και επεξεργασίας δεδομένων, που εκτελείται σε πεπερασμένο χρόνο, με χρήση πεπερασμένων εντολών. Η περιγραφή του θα γίνεται είτε σε μορφή ψευδοκώδικα, είτε σε κώδικα κάποιας γλώσσας προγραμματισμού είτε περιφραστικά με λόγια.

Κεφάλαιο 2

Επισκόπηση Σχετικής Έρευνας

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε εισαγωγικές έννοιες πάνω σε τρισδιάστατα παιχνίδια, διεπαφές γραφικών, γραφικά περιβάλλοντα και από τι αποτελούνται αυτά. Τέλος παρουσιάζουμε μηχανές και λογισμικά γραφικών τα οποία και συγκρίνουμε.

2.2 Ιστορική Αναδρομή Βιντεοπαιχνιδιών

Με τον όρο βιντεοπαιχνίδι (*video game*) καλείται οποιοδήποτε ηλεκτρονικό παιχνίδι το οποίο εμπεριέχει ανθρώπινη αλληλεπίδραση μέσω μια διεπαφής του χρήστη για να παραγάγει ανάδραση σε μια βιντεοσυσκευή [4]. Με τον όρο βίντεο (*video*) θεωρούνταν αρχικά οι συσκευές καθοδικού σωλήνα (*CRT*), αλλά πλέον στις μέρες μας θεωρείται οποιαδήποτε συσκευή μπορεί να προβάλει δισδιάστατες ή/και τρισδιάστατες εικόνες. Οι συσκευές αυτές, στις οποίες παίζουμε βιντεοπαιχνίδια ονομάζονται πλατφόρμες (σχήμα 2.1).
Οι κυριότερες και γνωστότερες στο ευρύ κοινό πλατφόρμες είναι:

- **Προσωπικοί Υπολογιστές (*PCs*):** Καθημερινά πλεόν στις ζωές όλων μας, για αυτό δεν θα είχαν λόγω να απουσιάζουν από τις πλατφόρμες. Εξελίσσονται συνεχώς στο επίπεδο hardware και ένα λόγος είναι για να μπορούν να υποστηρίζουν παιχνίδια, στα οποία τα γραφικά τους και σε αυτά, συνεχώς εξελίσσονται.

- **Κονσόλες βιντεοπαιχνιδιών (*Video game consoles*):** *Nintendo, Microsoft* και *Sony* οι τρεις μεγαλύτερες εταιρείες που ανταγωνίζονται στη δημιουργία της κα-

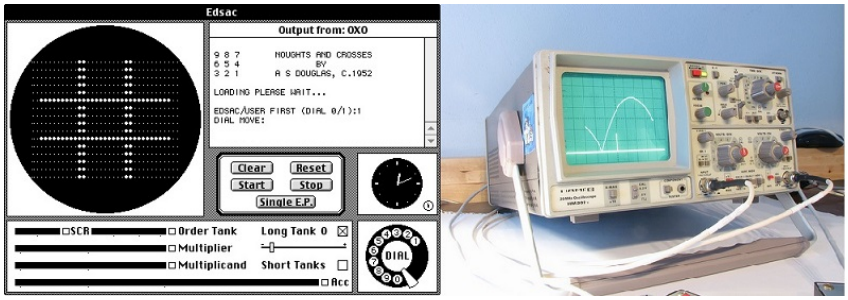
Σχήμα 2.1: Γνωστές κονσόλες βιντεοπαιχνιδιών
Πάνω αριστερά: Το *Sony PlayStation 4* **Κάτω Αριστερά:** Το *Atari 2600* **Πάνω κέντρο:** Το *Xbox 360* **Κάτω κέντρο:** Το *Nintendo Wii* **Δεξιά:** Το Arcade παιχνίδι *Mr. Pacman*

λύτερης βιντεοκονσόλας βγάζοντας συνεχώς και πιο εξελιγμένες. Γνωστές είναι τα *Wii, Xbox* και *PlayStation* αντίστοιχα για κάθε εταιρεία. Μην ξεχνάμε τις θρυλικές κονσόλες της ιαπωνικής εταιρείας *Atari* που έκανα θραύση την δεκαετία του ′80 [5].

- **Κονσόλες Arcade (*Arcade game consoles*):** Επανάσταση στις δεκαετίες ′80 και ′90 έφεραν τα παιχνίδια Arcade αφού άνοιγαν μαγαζιά μόνο και μόνο για να τα φιλοξενήσουν και πλοία, μπιλιαρδάδικα, λούνα παρκ διέθεταν ειδικούς χώρους για να τα τοποθετήσουν. Τα λεγόμενα στα ελληνικά ως "Φλιπεράκια" λειτουργούσαν με την εισαγωγή κέρματος αλλά πρόσφεραν προσωρινή χαρά στον χρήστη-μόλις ενός γύρου. Βρέθηκαν σε παρακμή στη δεκαετία του 2000 με την επανάσταση των βιντεοκονσολών αφού είχαν και μειονεκτήματα όπως ότι έπιαναν μεγάλο χώρο (ύψος≙1.5μ και βάθος≙0.5μ), φιλοξενούσαν μονάχα ένα παιχνίδι και είχαν δισδιάστατα γραφικά [6].

- **Κινητά τηλέφωνα και Υπολογιστές ταμπλέτα (*Mobile phones and Tablet PCs*):** Έντονη η διάδοσή τους την τελευταία δεκαετία δεν μπορούν να μην φιλοξενούν παιχνίδια μιας και γίνεται καθημερινή χρήση τους. Οι εταιρείες βιντεοπαιχνιδιών πλέον ειδικεύονται στον σχεδιασμό παιχνιδιών ειδικά σε αυτά καθώς οι απαιτήσεις τους σε γραφικά είναι μικρότερες και η διεπαφή με τον χρήστη διαφέρει εντελώς.

Τα πρώτα βιντεοπαιχνίδια εμφανίστηκαν στις αρχές της δεκαετίας του ′50 και είχαν απλή λογική και υποτυπώδη γραφική διεπαφή (σχήμα 2.2). Γνωστότερα είναι τα:



Σχήμα 2.2: Τα πρώτα βιντεοπαιχνίδια
Αριστερά: Το *OXO* (1952) **Δεξιά:** Το *Tennis For Two* (1958)

Σχήμα 2.3: Χειριστήρια βιντεοκονσολών

- **Nim:** Το οποίο είναι παιχνίδι μαθηματικής στρατηγικής και πρέπει ο παίχτης σε κάθε γύρο να αφαιρεί αντικείμενα από μια στοίβα [7].

- **OXO:** Η γνωστή σε όλους μας τρίλιζα.

- **Tennis For Two:** Ένα παιχνίδι που παιζόταν μέσα από την οθόνη ενός παλμογράφου και έχει την μορφή του ping-pong. Από τα πιο διαδεδομένα αρχικά βιντεοπαιχνίδια αφού παίζεται μέχρι σήμερα σε διάφορες μορφές του [8].

Τα πρώτο εμπορικό βιντεοπαιχνίδι ήταν το *Computer Space* και βγήκε στις αγορές το 1971. Τα βιντεοπαιχνίδια, στην αρχή, προσάρμοζαν τον χειρισμό τους ανάλογα με την πλατφόρμα για την οποία σχεδιάζονταν. Αργότερα εμφανίστηκα τα πρώτα *joysticks* τα οποία εμπνεύστηκαν την μορφή τους από το σχήμα που είχαν τα χειριστήρια στο πιλοτήριο του αεροπλάνου (σχήμα 2.3). Σήμερα ο χειρισμός γίνεται μέσω του πληκτρολογίου του υπολογιστή, ειδικών χειριστηρίων γνωστά ως *gamepads* ή *joypads* που παίρνουν μορφή ανάλογα με την κονσόλα, και μέσω οθονών επαφής.

2.3 Κατηγορίες Βιντεοπαιχνιδιών

Τα βιντεοπαιχνίδια χωρίζονται σε διάφορες κατηγορίες ανάλογα με την αλληλεπίδραση του χρήστη με το παιχνίδι (gameplay) και όχι με τις διαφορές στα γραφικά ή στο σενάριο του παιχνιδιού. Είναι επίσης ανεξάρτητες από τις ρυθμίσεις του καθώς και από το περιβάλλον του κόσμου που παρουσιάζει. Ένα παιχνίδι μπορεί να ανήκει σε μια κατηγορία ή υποκατηγορία, μπορεί να ανήκει σε μια κατηγορία που είναι υβρίδιο δυο άλλων καθώς μπορεί να κατασκευαστεί ένα καινούργιο παιχνίδι με ένα ιδιαίτερο gameplay που να δημιουργήσει μια καινούργια υποκατηγορία. Οι κατηγορίες που παρουσιάζονται παρακάτω είναι οι βασικές και οι ισχύουσες. Λόγω τις ραγδαίας εξέλιξης των βιντεοπαιχνιδιών, οι κατηγορίες αλλάζουν συνεχώς και είναι δύσκολο να ταξινομηθούν τα παιχνίδια [9].

2.3.1 Δράσης (Action)

Στα παιχνίδια δράσης απαιτείται ο παίχτης να έχει γρήγορα αντανακλαστικά, ακρίβεια στις κινήσεις του και ταχύτητα στο να αποφεύγει τα εμπόδια. Θεωρείται ίσως η κυριότερη κατηγορία παιχνιδιών και σίγουρα η πιο διαδεδομένη. Τα παιχνίδια δράσης τείνουν να κατασκευάζουν το gameplay τους, με έμφαση στη "μάχη" (σχήμα 2.4). Υπάρχουν πολλές υποκατηγορίες τους, οι κυριότερες εκ των οποίων είναι:

- **Πρώτου Προσώπου (First-Person Shooter):** Στα οποία διαχειρίζεσαι ένα παίχτη, συνήθως με όπλο, με σκοπό να αντιμετωπίσεις κάποιον ή κάποιους αντιπάλους. Είναι σε πρώτο πρόσωπο γιατί η κάμερα σου βλέπει όπως θα έβλεπε μέσα από τα μάτια του ο παίχτης.

- **Πρώτου Προσώπου Πολλαπλών Παιχτών (MMO First-Person Shooter):** Ακριβώς όπως τα πρώτου προσώπου μόνο που τώρα τους αντιπάλους σου τους διαχειρίζονται χρήστες μέσω του διαδικτύου.

- **Τρίτου Προσώπου (Third-Person Shooter):** Σαν τα πρώτου προσώπου μόνο που πλέον η κάμερα σου εστιάζει πάνω από τον παίχτη και τον βλέπεις να κινείται.

- **Παιχνίδια πάλης (Fighting games):** Στα οποία χρησιμοποιείς έναν χαρακτήρα και έρχεσαι αντιμέτωπος με έναν ή με περισσότερους αντιπάλους με τους οποίους παλεύεις.

Σχήμα 2.4: Παιχνίδια δράσης
Αριστερά: *Call of Duty 4-Modern Warfare* (2007-First-person shooter) **Κέντρο:** *Max Payne 3* (2012-Third-person shooter) **Δεξιά:** *UFC Undisputed 3* (2012-Fighting game)

- **Πλατφόρμας (Platform):** Στα οποία, ο παίχτης μεταφέρεται στο χώρο εξομοιώνοντας φυσικές αντιδράσεις ενός ανθρώπου όπως τρέξιμο, σκύψιμο, σύρσιμο, πήδημα ή το ανέβασμα μιας σκάλας.

2.3.2 Περιπέτειας (Adventure)

Τα παιχνίδια περιπέτειας πήραν τον όνομά τους από το πρώτο παιχνίδι που δημιουργήθηκε στην κατηγορία, στις αρχές της δεκαετίας του ′70, και λεγόταν *Adventure*. Θεωρούνται τα παιχνίδια αυτά που τα αντανακλαστικά και η δράση των παιχτών απουσιάζει. Ο παίχτης καλείται να λύσει μια σειρά από γρίφους αντιδρώντας συνήθως με ανθρώπους (συζητήσεις) και με το περιβάλλον (λήψη κάποιου αντικειμένου, άνοιγμα κάποιας πόρτας). Συνήθως επειδή τέτοιου είδους παιχνίδια δεν πιέζουν τον χρήστη στο να αντιδράσει σε κάτι ή να βιαστεί λόγο κάποιου χρονικού ορίου, προσελκύουν και κόσμο που συνήθως δεν ασχολείται με βιντεοπαιχνίδια (σχήμα 2.5). Οι υποκατηγορίες που έχουν είναι τα τρισδιάστατα πραγματικού χρόνου, τα γραφικά παιχνίδια περιπέτειας (με παλαιότερα γραφικά) και τα κειμένου στα οποία ο χρήστης δίνει ζωή στον παίχτη του μέσα από μια σειρά εντολών.

2.3.3 Ρόλων (RPGs)

Τα παιχνίδια ρόλων ξεκίνησαν επηρεασμένα από το επιτραπέζιο παιχνίδι Dungeons & Dragons. Σε αυτά τα παιχνίδια διαχειρίζεσαι ένα χαρακτήρα, ο οποίος έχει συγκεκριμένες δυνατότητες (όπλο ή/και μαγικά ξόρκια) και τον οποίο κινείς μέσα σε έναν ορισμένο κόσμο με κάποιο απώτερο σκοπό. Περνάς συνήθως μέσα από μεσαιωνικές πόλεις, κάστρα και κατακόμβες και αντιμετωπίζεις τέρατα, ζωτικά, μάγους και δράκους (σχήμα 2.6). Οι κυριότερες

Σχήμα 2.5: Παιχνίδια περιπέτειας
Αριστερά: *Realmyst* (2000) **Δεξιά:** *Runaway: A Road Adventure* (2001)

Σχήμα 2.6: Παιχνίδια ρόλων
Αριστερά: *World of Warcraft* (2004) **Κέντρο:** *Final Fantasy XIV* (2010) **Δεξιά:** *Diablo III* (2012)

κατηγορίες τους είναι τα τακτικής (tactical) όπου εισέρχεται και η έννοια της στρατηγικής, τα δράσης (action), τα οποία παίρνουν και στοιχεία από τα παιχνίδια δράσης και περιπέτειας, και τα πολλαπλών παιχτών (MMORPGs) τα οποία παίζονται στο διαδίκτυο.

2.3.4 Προσομοίωσης (Simulation)

Στα παιχνίδια αυτά προσομοιώνονται-εξού και το όνομά τους-καταστάσεις και στιγμές της πραγματικής ζωής ή της πλασματικής πραγματικότητας. Τα περισσότερα παιχνίδια στις μέρες μας, παρ΄ ότι μπορεί να μην ανήκουν σε αυτή τη κατηγορία, δανείζονται στοιχεία προσομοίωσης (σχήμα 2.7). Χωρίζονται σε τρεις βασικές κατηγορίες:

- **Προσομοίωση Κατασκευής και Επιχείρησης (Construction and Management Simulation):** Στα οποία ο χρήστης στόχο έχει να κατασκευάσει μια πόλη ή μια χώρα, να ικανοποιήσει τις απαραίτητες ανάγκες των κατοίκων όπως στέγη, πρό-

Σχήμα 2.7: Παιχνίδια προσομοίωσης
Αριστερά: *SimCity 4* (2013-Construction simulation) **Κέντρο:** *The Sims 3* (2009-Life simulation) **Δεξιά:** *Need for Speed: Underground 2* (2004-Vehicle simulation)

νοια, μετακίνηση, να διαχειριστεί μια επιχείριση ή μια πολιτική θέση.

- **Προσομοίωση Ζωής (Life Simulation):** Στα οποία ο χρήστης σκοπό έχει να διαχειριστεί κάποιον ή κάποιους χαρακτήρες όπως στην πραγματική ζωή, από δουλειά και στέγαση μέχρι ανάγκες, κοινωνική ζωή και διασκέδαση.

- **Προσομοίωση Οχημάτων (Vehicle Simulation):** Στα οποία ο χρήστης προσομοιώνει την λειτουργία οχημάτων όπως αυτοκίνητο, μοτοσυκλέτα μέχρι τραίνο ή αεροπλάνο. Οι προσομοιωτές αεροπλάνων (Flight Simulators) χρησιμοποιούνται κατά κόρον στην εκπαίδευση των υποψήφιων πιλότων.

2.3.5 Στρατηγικής (Strategy)

Στην συγκεκριμένη κατηγορία παιχνιδιών ο χρήστης πρέπει να έχει προσεγμένη και επιδέξια σκέψη στο πως θα εκτελέσει τις κινήσεις του. Τα παιχνίδια αυτά είναι εμπνευσμένα από τα αντίστοιχα επιτραπέζια στρατηγικής (σχήμα 2.8). Χωρίζονται σε πραγματικού χρόνου (real-time) και ανά γύρου (turn-based) ανάλογα με το αν ο παίχτης περιμένει τους υπόλοιπους αντιπάλους να τελειώσουν τις κινήσεις τους ή παίζουν όλοι ταυτόχρονα. Επίσης χωρίζονται βασισμένα στην στρατηγική ή στρατιωτική τακτική.

Σχήμα 2.8: Παιχνίδια στρατηγικής
Αριστερά: *Age of Empires II: The Age of Kings* (1999) **Δεξιά:** *Rise of Nations* (2003)

Σχήμα 2.9: Παιχνίδια αθλητισμού
Αριστερά: *Pro Evolution Soccer 2013* (2012) **Κέντρο:** *Football Manager 2013* (2012) **Δεξιά:** *NBA 2K13* (2012)

2.3.6 Αθλητισμού (Sport)

Από τις πρώτες κατηγορίες παιχνιδιών τα οποία προσομοιώνουν τους αγώνες γύρω από κάποιο άθλημα, με δημοφιλέστερα το ποδόσφαιρο και την καλαθοσφαίριση. Άλλα δίνουν έμφαση στο πραγματικό παίξιμο του αθλήματος και άλλα στην στρατηγική που θα ακολουθηθεί (σχήμα 2.9).

2.3.7 Επιτραπέζια και Παιχνίδια Καρτών (Board and Card games)

Η τελευταία αλλά η πιο σημαντική για εμάς κατηγορία-αφού σε αυτήν ανήκει το παιχνίδι μας-είναι τα επιτραπέζια παιχνίδια και παιχνίδια καρτών. Πολλά γνωστά επιτραπέζια παιχνίδια έχουν μετατραπεί σε ψηφιακά όπως επίσης και πάρα πολλά παιχνίδια με κάρτες ή τράπουλα.

Σχήμα 2.10: Επιτραπέζια και παιχνίδια καρτών
Αριστερά: *Monopoly* (2008) **Κέντρο:** *Trivial Pursuit* (2009) **Δεξιά:** *Tichu* (2000)

Σκάκι, ντάμα και Mah-jongg από τα αρχαιότερα επιτραπέζια παιχνίδια που έχουν μεταφερθεί στον υπολογιστή. Πάντα η τεχνητή νοημοσύνη βασικό στοιχείο, ώστε να δίνει ικανότητα στους αντιπάλους. Σήμερα δίνεται περισσότερο βάση στο σχεδιασμό τέτοιον παιχνιδιών για εφαρμογές στο διαδίκτυο (σχήμα 2.10). Χωρίζονται σε επιδεξιότητας, στρατηγικής, γνώσεων, μυστήριου ανάλογα με το είδος και το σενάριο.

Το *Cluedo* ανήκει στα επιτραπέζια παιχνίδια στρατηγικής. Αυτό που κάνει αυτά τα επιτραπέζια ενδιαφέροντα και να ξεχωρίζουν από τις υπόλοιπες κατηγορίες, είναι ότι ο παίχτης πρέπει να καταστρώσει τον τρόπο με τον οποίο θα κινηθεί μέσα στο παιχνίδι, ώστε να φτάσει πιο εύκολα στο στόχο του, και ταυτόχρονα να εμποδίσει τους αντιπάλους να το καταφέρουν.

2.4 Τρισδιάστατες Βιβλιοθήκες Γραφικών (3D APIs)

Οι γραφικές βιβλιοθήκες ή αλλιώς όπως είναι γνωστές διεπαφές προγραμματισμού εφαρμογών (3D APIs) δημιουργήθηκαν για να διευκολύνουν τις διαδικασίες σε όλα τα στάδια των γραφικών [10]. Επίσης έχει αποδειχθεί ότι έχουν ζωτική σημασία στους κατασκευαστές υλικού (hardware) γραφικών, αφού δίνουν τη δυνατότητα στους προγραμματιστές γραφικών να έχουν πρόσβαση στο hardware-σε ένα αφηρημένο επίπεδο βέβαια-οποιασδήποτε κάρτας γραφικών. Χωρίζονται σε χαμηλού και υψηλού επιπέδου. Αναφέρουμε περιληπτικά τις 3 κυριότερες βιβλιοθήκες χαμηλού επιπέδου που μας ενδιαφέρουν στην εργασία μας. Για τις υψηλού επιπέδου-επειδή συνήθως χρησιμοποιούνται παράλληλα με κάποια μηχανή γραφικών-γίνεται εκτενής περιγραφή παρακάτω [υποενότητα 2.11.2].

Η χρήση παρ΄ όλα αυτά των βιβλιοθηκών γραφικών είναι δύσκολη και πολύπλοκη. Η διαχείρισή τους, γίνεται συνήθως από έμπειρους προγραμματιστές γραφικών που επιθυμούν πρόσβαση στο υλικό (hardware), πάντα σε αφηρημένο επίπεδο. Οι σύγχρονες μηχανές γραφικών, όπως αυτή που χρησιμοποιήσαμε, διαχειρίζονται αυτόματα τις βιβλιοθήκες αυτές. Οπότε μια διαχείριση από μέρους μας ήταν περιττή και επικίνδυνη ως προς το αποτέλεσμα.

2.4.1 Direct3D

Η *Direct3D* είναι κομμάτι μιας ευρύτερης βιβλιοθήκης γραφικών (API) της εταιρείας *Microsoft*, της *DirectX* [11]. Είναι διαθέσιμη για όλα τα λειτουργικά *Windows* από την έκδοση 95 και έπειτα και για άλλες πλατφόρμες με χρήση του εργαλείου *Wine*. Είναι η βασική βιβλιοθήκη που χρησιμοποιείται στις βιντεοκονσόλες *Xbox* της ίδια εταιρείας και αποτελεί τον μεγάλο ανταγωνιστή της βιβλιοθήκης *OpenGL*. Χρησιμοποιείται για αναπαράσταση τρισδιάστατων γραφικών σε εφαρμογές που χρειάζονται υψηλή απόδοση, όπως τα παιχνίδια. Επιτρέπει την εκτέλεση των εφαρμογών σε κατάσταση πλήρους οθόνης αλλά και σε ενσωματωμένο παράθυρο μέσα στο λειτουργικό σύστημα. Χρησιμοποιεί επιτάχυνση του υλικού της κάρτας γραφικών σε όλη τη διαδικασία επεξεργασίας των γραφικών και έχει δυνατότητες όπως z-buffering, χωρική αντιταύτιση (spatial anti-aliasing), alpha blending, mipmapping, ατμοσφαιρικά εφφέ, σωστή απεικόνιση υφής, εφαρμογή εξομοίωσης κορυφής αλλά και εξομοίωσης εικονοστοιχείου μόνο αν το υποστηρίζει η κάρτα γραφικών.

2.4.2 OpenGL

Η *OpenGL* είναι μια διαγλωσσική, πολλαπλών πλατφόρμων βιβλιοθήκη γραφικών (API) για αναπαράσταση δισδιάστατων και τρισδιάστατων γραφικών [12]. Η βιβλιοθήκη αυτή χρησιμοποιείται κυρίως αφηρημένα για να αλληλεπιδρά με την κάρτα γραφικών ώστε να επιτυγχάνει αναπαράσταση με επιτάχυνση υλικού (harware)· παρ΄ όλα αυτά μπορεί να εφαρμοστεί και σε λογισμικό. Η *OpenGL* δημιουργήθηκε από την εταιρεία *Silicon Graphics Inc.* το 1992 και χρησιμοποιείται ευρέως σε σχεδιαστικά εργαλεία CAD, σε εικονική πραγματικότητα, επιστημονικές απεικονίσεις και απεικονίσεις πληροφοριών, προσομοιωτές πτήσεων και βιντεοπαιχνίδια.

2.4.3 X3D

Το *X3D* είναι ένας πρότυπου ISO τύπος αρχείου, βασισμένος στο *XML*, για αναπαράσταση τρισδιάστατων γραφικών [13]. Είναι διάδοχος της γλώσσας *VRML* και την προεκτείνει (π.χ. CAD, Geospatial, ανθρωποειδή κινούμενα σχέδια (humanoid animations), NURBS κ.α.), με δυνατότητα να κωδικοποιήσει μια σκηνή χρησιμοποιώντας συντακτικό της *XML* καθώς με την σύνταξη της *VRML97,* δυαδικής διαμόρφωσης και φυσικά ενισχυμένης βιβλιοθήκης γραφικών (API).

2.4.4 Direct3D ενάντια OpenGL

Κατά τις πρώτες του εκδόσεις το *Direct3D* καταπολεμήθηκε πολύ από μεγάλους σχεδιαστές γραφικών· σε επίπεδο να ζητήσουν από την *Microsoft* να σταματήσει να βγάζει καινούργιες εκδόσεις του και να υιοθετήσει το *OpenGL*. Οι λόγοι που οδήγησαν εκεί τους σχεδιαστές ήταν η μεγάλη πολυπλοκότητά του *Direct3D* και ότι ήταν ένα κακό αντίγραφο του *OpenGL*. Σήμερα μετά από πολλές εξελίξεις θεωρούνται εξίσου εύκολες και εξίσου δυνατές βιβλιοθήκες (σχήμα 2.11) [14].

Οι διαφορές τους έγκεινται κυρίως ότι έχουν σχεδιαστεί με διαφορετικούς τρόπους. Το μεν *Direct3D* ελευθερώνει τον προγραμματιστή από την σύνδεσή του με το hardware, ενώ το *OpenGL* όχι, γιατί σχεδιάστηκε ως σύστημα αναπαράστασης με επιτάχυνση στο hardware που θα εξομοιώνεται από το ίδιο το λογισμικό. Έτσι το *Direct3D*, περιμένει η ίδια η εφαρμογή να διαχειριστεί τους πόρους του υλικού όπως θέλει, ενώ το *OpenGL* κάνει από μόνο του αυτή τη δουλειά, καθιστώντας το πιο εύκολο στην υλοποίησή του αλλά πιο πολύπλοκο στο να το διαχειριστεί ένας σχεδιαστής γραφικών. Στο *Direct3D* ο σχεδιαστής έχει την ελαστικότητα να ορίσει τους πόρους για την εφαρμογή του, κάτι που το κάνει πιο αποδοτικό.

Μέχρι το 2005 μια άλλη διαφορά που είχαν ήταν ο τρόπος αναπαράστασης των υφών (textures). Στο *Direct3D* γινόταν με την κλήση μια συνάρτησης, που ήταν και βολικό, ενώ στο *OpenGL* χρειαζόντουσαν buffers για τα εικονοστοιχεία που το καθιστούσε επικίνδυνο. Μέτα από μια επέκταση που δημιούργησαν στο *OpenGL* το πρόβλημα λύθηκε.

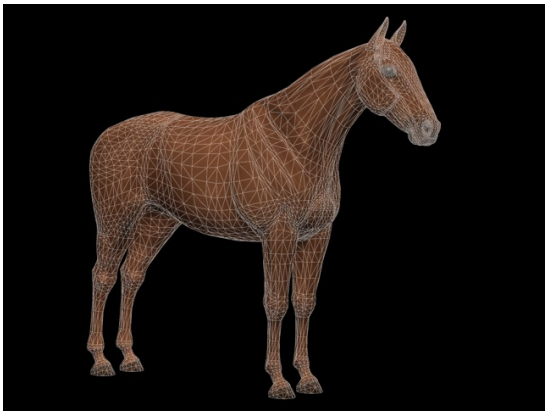
Σχήμα 2.11: *OpenGL* ενάντια *DirectX11*

2.5 Δίκτυα Γεωμετρίας (Meshes)

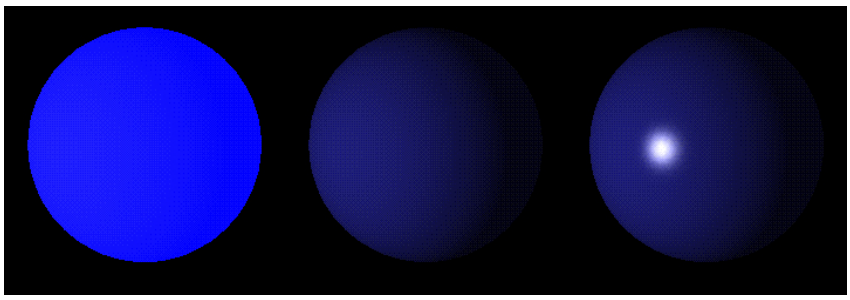
Όταν πρωτοδημιουργήθηκαν τα βιντεοπαιχνίδια είχαν δισδιάστατα γραφικά. Η αναπαράσταση των χώρων και των αντικειμένων γινόταν μέσα από γραμμές, τόξα, κύκλους και καμπύλες τα οποία ενώνονταν με τέτοιο τρόπο ώστε να μας δώσουν το τελικό αποτέλεσμα. Σε περίπτωση που θέλαμε να δώσουμε την υποτυπώδη αίσθηση των τριών διαστάσεων, αυτό γινόταν μέσα από υφές (textures) [15]. Όταν δημιουργήθηκαν τα τρισδιάστατα γραφικά έπρεπε πλέον τα αντικείμενα να κατασκευάζονται με κάποιο τρόπο. Ο παλαιότερος και πιο διαδεδομένος είναι τα πολύγωνα. Συνδυαζόμενα τα πολύγωνα δίνουν τρισδιάστατα αντικείμενα τα οποία αποτελούνται από ένα σύνολο κορυφών που ενώνονται με γραμμές έτσι ώστε να μας δίνουν το επιθυμητό αποτέλεσμα. Αυτός ο συνδυασμός των πολυγώνων ονομάζεται πλέγμα ή δίκτυο γεωμετρίας (mesh). Όσο πιο πολύπλοκη η καμπύλη στο αντικείμενο τόσο περισσότερα δίκτυα περιέχει (σχήμα 2.12) [16].

2.6 Φωτισμός (Lighting)

Με τον όρο φωτισμός στα γραφικά υπολογιστών εννοούμε την διαδικασία προσομοίωσης του φωτός πάνω στα γραφικά [17]. Η προσομοίωση αυτή μπορεί να είναι εξαιρετικά ακριβής, όπως η παρακολούθηση της ροής της ενέργειας του φωτός που αλληλεπιδρά με τα υλικά, μέσα από διάφορες υπολογιστικές τεχνικές, είτε πιο απλή εμπνευσμένη από τη φυσική του φωτός που μας δίνει όμως μια λιγότερο φωτορεαλιστική απόδοση. Τα είδη φωτός στα υπολογιστικά γραφικά είναι (σχήμα 2.13) [18]:



Σχήμα 2.12: Τρισδιάστατος σχεδιασμός ενός αλόγου με δίκτυα γεωμετρίας (meshes)



Σχήμα 2.13: Η αναπαράσταση του φωτός σε μια σφαίρα
Αριστερά: Καθολικό (Ambient) **Κέντρο:** Διάχυτο (Diffuse) **Δεξιά:** Κατοπτρικό (Specular)

- **Καθολικό Φως (Ambient Light):** Στο οποίο το φως φαίνεται να έχει ανακλαστεί τόσες φορές και σε τόσες πολλές επιφάνειες που κάνει τα αντικείμενα να φωτίζονται όλα το ίδιο.
- **Διάχυτο Φως (Diffuse Light):** Το οποίο είναι το φως που διαχέεται σαν να πέφτει σε θαμπές επιφάνειες.
- **Κατοπτρικό Φως (Specular Light):** Το οποίο είναι το φως που φεύγει όταν πέφτει σε ανακλαστικές, λαμπερές επιφάνειες (π.χ μέταλλο, πλαστικό)
- **Εκπέμπον Φως (Emissive Light):** Το οποία εκπέμπουν αυτόφωτα αντικείμενα.

Υπάρχουν πολλοί αλγόριθμοι στα γραφικά υπολογιστών για τον υπολογισμό του φωτός σε μια σκηνή. Γνωστότεροι είναι οι *Ray Tracing*, *Ambient Occlusion*, *Global Illumination*

και *Radiosity*. Παρ' ότι είναι αρκετοί, χρειάζονται πολλούς πόρους του hardware και είναι αρκετά χρονοβόροι για να εκτελεστούν κατά την διάρκεια του παιχνιδιού. Για αυτό το λόγο χρησιμοποιείται ο χάρτης φωτός (lightmapping) [19].

Ο χάρτης φωτός είναι μια δομή δεδομένων που περιέχει πληροφορίες για την φωτεινότητα και τις σκιάσεις των αντικειμένων μιας σκηνής. Ο χάρτης αυτός υλοποιείται μονάχα για τα στατικά αντικείμενα. Αλλά επειδή σε μια σκηνή συνήθως τα περισσότερα αντικείμενα είναι στατικά (εκτός από τους παίχτες ή τα αντικείμενα που μετακινούνται) καταφέρνουμε να γλυτώσουμε πολύ χρόνο από τον υπολογισμό τους φωτός κατά τη διάρκεια του παιχνιδιού και έτσι να το κάνουμε πιο γρήγορο.

2.7 Υφές (Textures)

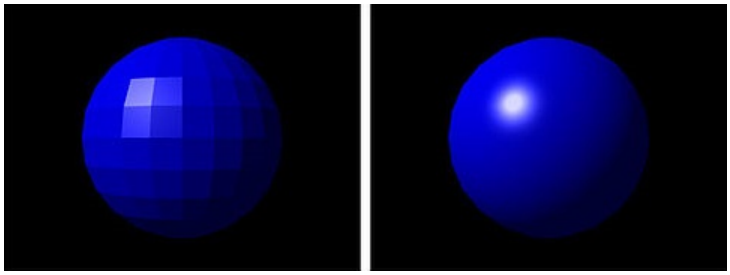
Υφή ονομάζεται οτιδήποτε τοποθετείται στην επιφάνεια ενός αντικειμένου και χαρακτηρίζεται από το μοτίβο, την τραχύτητα και τον κυματισμό που το συνθέτει. Στα υπολογιστικά γραφικά όταν μιλάμε για υφές εννοούμε την χαρτογράφηση υφών (texture mapping), κατά την οποία προσθέτονται υφές επιφανειών, λεπτομέρειες και χρώμα στα τρισδιάστατα αντικείμενα [20]. Ο χάρτης υφής εφαρμόζεται στην επιφάνεια ενός πολυγώνου ακριβώς σαν να εφαρμόζεται ένα χαρτί σε ένα λευκό κουτί. Σε κάθε κορυφή του πολυγώνου αποδίδονται συντεταγμένες υφής που ονομάζονται UV .

Υπάρχουν πολλά είδη χαρτογράφησης υφών. Ένα είναι η πολυχαρτογράφηση (multitexturing) όπου σε ένα πολύγωνο εφαρμόζεται παραπάνω από μια υφή σε μια χρονική στιγμή. Ένα άλλο είναι η χαρτογράφηση βάθους (bump mapping), με το οποίο δίνεται η αίσθηση ότι η υφή έχει διαφορετικό βάθος σε διάφορα σημεία είτε αλλάζει κατεύθυνση ώστε να επηρεάσει στον υπολογισμό του φωτός (σχήμα 2.14) [21]. Χωρίζεται σε normal mapping, το οποίο επηρεάζει το βάθος μόνο στην ίδια την υφή, και σε parallax mapping που επηρεάζει το βάθος και στο αντικείμενο στο οποίο τοποθετείται η υφή.

2.8 Σκιαστές (Shaders)

Στα γραφικά υπολογιστών, σκιαστής είναι ένα πρόγραμμα το οποίο χρησιμοποιείται για να σκιάσει τα αντικείμενα· να παράγει δηλαδή τα κατάλληλα επίπεδα του φωτός και του χρώματος μέσα σε μια εικόνα [22]. Πλέον στις μέρες μας σκίαση θεωρείται ακόμα, η δημιουργία

Σχήμα 2.14: Bump mapping
Αριστερά: Κανονική υφή (bitmap) **Δεξιά:** Υφή με χαρτογράφηση βάθους (bump mapping)



Σχήμα 2.15: Σκιαστές
Αριστερά: Σκιαστής Flat **Δεξιά:** Σκιαστής Phong

ειδικών εφφέ και η εκτέλεση μετεπεξεργασίας.

Υπάρχουν 3 είδη σκιαστών, οι σκιαστές κορυφής, εικονοστοιχείου και γεωμετρίας που αποτελούν και τους πιο σύγχρονους. Υπάρχουν πολλοί αλγόριθμοι για σκιάσεις, πιο γνωστοί από τους οποίους είναι το Flat shading, Phong shading, Lambert κ.α. (σχήμα 2.15) [23].

2.9 Κινούμενα Σχέδια (Animations)

Στα ελληνικά ο όρος Animation αποδίδεται ως Εμψύχωση, Κινούμενη Εικόνα ή Κινούμενα Σχέδια [24]. Το Animation είναι η ταχεία προβολή μιας σειράς από εικόνες (δισδιάστατες ή τρισδιάστατες) ή των θέσεων μιας φιγούρας, έτσι ώστε να δημιουργείται η ψευδαίσθηση της κίνησης. Είναι στην ουσία μια οπτική οφθαλμαπάτη της κίνησης και αυτό συμβαίνει εξαιτίας

του φαινομένου διατήρησης της εικόνας στο μάτι επί 1/12 του δευτερολέπτου, το λεγόμενο ως μεταίσθημα ή μετείκασμα. Κίνηση μπορεί να δημιουργηθεί και να παρουσιαστεί με πολλούς τρόπους. Η πιο διαδεδομένη μέθοδος απεικόνισης της κινούμενης εικόνας αποτελείται από ένα πρόγραμμα βίντεο, έναν προβολέα ή την οθόνη του υπολογιστή.

2.10 Φυσική (Physics)

Η φυσική στους υπολογιστές συνεπάγεται τη θέσπιση των νόμων της φυσικής σε μια μηχανής προσομοίωσης ή στα παιχνίδια, ειδικά σε τρισδιάστατα γραφικά, με σκοπό να κάνουν τα εφφέ να φαίνονται πιο ρεαλιστικά στον παρατηρητή [25]. Συνήθως η προσομοίωση της φυσικής είναι μόνο μια στενή προσέγγιση της πραγματικής φυσικής και ο υπολογισμός της γίνεται με διακριτές τιμές. Υπάρχουν 2 βασικά συστατικά:

- **Μηχανή Φυσικής:** Είναι ο κώδικας του προγράμματος που χρησιμοποιείται για την προσομοίωση της νευτώνειας φυσικής.
- **Ανίχνευση Συγκρούσεων:** Χρησιμοποιείται για να λύσει το πρόβλημα του προσδιορισμού, όταν οποιαδήποτε δύο ή περισσότερα αντικείμενα στο περιβάλλον έρχονται σε επαφή.

Υπάρχουν και άλλα στοιχεία που ενσωματώνονται στις μηχανές φυσικής όπως τα συστήματα σωματιδίων (particle systems), που προσομοιώνουν την φυσική που ακολουθούν σωματίδια όπως σε περιπτώσεις φωτιάς, έκρηξης, νερού, καπνού, ή όπως της φυσικής των νεκρών σωμάτων (ragdoll physics) που προσπαθεί να προσομοιώσει την ακαμψία που έχουν τα σώματα όταν πέφτουν νεκρά.

2.11 Περιβάλλοντα Ανάπτυξης Γραφικών

Η δημιουργία ενός παιχνιδιού-και ιδιαίτερα ενός τρισδιάστατου-προϋποθέτει μια πολύπλοκη διαδικασία στις μέρες μας. Η διαδικασία αυτή περιλαμβάνει πολλά στάδια από την σχεδίαση των γραφικών του παιχνιδιού, μέχρι την ανάπτυξη του λογισμικού για την λειτουργία του και την ενσωμάτωση της φυσικής σε αυτό. Για αυτό χρησιμοποιούνται πολλά εργαλεία για την υλοποίησή του, τα κυριότερα εκ των οποίων περιγράφονται σε αυτή την ενότητα.

2.11.1 Λογισμικά Σχεδιασμού Γραφικών

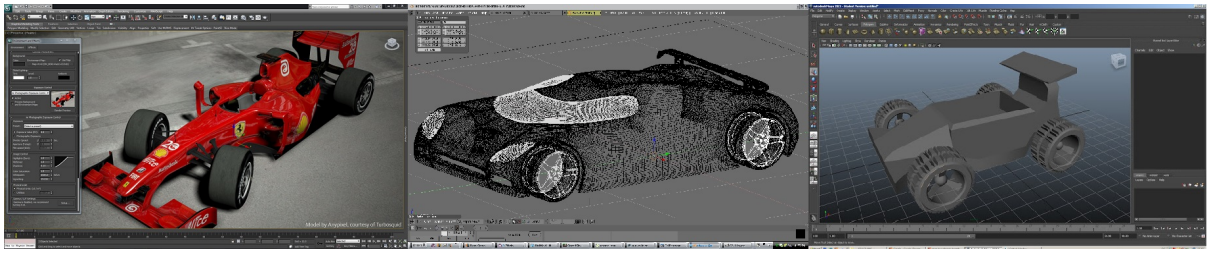
Τα λογισμικά σχεδιασμού γραφικών παραπέμπουν στα προγράμματα αυτά που χρησιμοποιούνται για να δημιουργούν υπολογιστικά-παραγόμενες τρισδιάστατες εικόνες [26]. Οι σχεδιαστές μπορούν να δημιουργήσουν και να τροποποιήσουν αντικείμενα μέσα από τρισδιάστατα δίκτυα γεωμετρίας (meshes). Οι σχεδιαστές μπορούν να προσθέσουν, να αφαιρέσουν, να εκτείνουν και γενικά να αλλάξουν τα δίκτυα γεωμετρίας όπως επιθυμούν. Τα αντικείμενα μπορούν να προβληθούν από διάφορες γωνίες, συνήθως ταυτόχρονα. Οι σχεδιαστές μπορούν να μετακινήσουν, να περιστρέψουν και να κλιμακώσουν τα αντικείμενα ή να τα μεγεθύνουν.

Τα αντικείμενα μπορούν να εξαχθούν σε αρχεία και να εισαχθούν σε άλλα προγράμματα ή μηχανές γραφικών όταν η συμβατότητα το επιτρέπει. Επίσης περιέχουν έναν αριθμό από δυνατότητες όπως φωτισμό, χαρτογράφηση υφών (texture mapping) και κινούμενα σχέδια (animations) (σχήμα 2.16). Κάποια διαδεδομένα λογισμικά σχεδιασμού γραφικών είναι:

- **Autodesk 3ds Max:** Το *Autodesk 3ds Max*, πρώην *3D Studio Max*, είναι ένα λογισμικό τρισδιάστατων γραφικών, για την δημιουργία τρισδιάστατων animations, μοντέλων και εικόνων [27]. Αναπτύχθηκε και παράχθηκε από την εταιρεία *Autodesk Media and Entertainment*. Έχεις πολλές δυνατότητες, ευέλικτα αρχιτεκτονικά εργαλεία και μπορεί να χρησιμοποιηθεί σε πλατφόρμες *Windows* και *Mac OS*. Χρησιμοποιείται συχνά από σχεδιαστές βιντεοπαιχνιδιών, εταιρείες τηλεοπτικών διαφημίσεων και αρχιτεκτονικών απεικονίσεων. Χρησιμοποιείται επίσης και για κινηματογραφικά εφφέ και προοπτικοποίησης ταινιών.

- **Blender:** Το *Blender* είναι ένα ελεύθερο και ανοιχτού κώδικα λογισμικό σχεδιασμού τρισδιάστατων γραφικών [28]. Χρησιμοποιείται για την δημιουργία ταινιών κινουμένων σχεδίων, οπτικών εφφέ, προϊόντων τέχνης, τρισδιάστατων εκτυπώσιμων μοντέλων, τρισδιάστατων διαδραστικών εφαρμογών και βιντεοπαιχνιδιών. Οι δυνατότητες του *Blender* περιλαμβάνουν σχεδιασμό τρισδιάστατων αντικειμένων, UV unwrapping, υφές, rigging και skinning, προσομοίωση (καπνού, υγρών, σωματιδίων, μαλακού σώματος), animation, παρακολούθηση κάμερας, απεικόνιση αντικειμένων, σύνθεση και επεξεργασία ταινιών. Επίσης περιέχει εγκαταστημένη μια δική του μηχανή παιχνιδιών.

- **Autodesk Maya:** Το *Autodesk Maya* ή πιο απλά *Maya*, είναι ένα λογισμικό τρισδιάστατων γραφικών για πλατφόρμες *Windows*, *Mac OS*, και *Linux* [29]. Αρχικά



Σχήμα 2.16: Η σχεδίαση ενός αυτοκινήτου με λογισμικά σχεδιασμού γραφικών **Αριστερά:** *Autodesk 3ds Max* **Κέντρο:** *Blender* **Δεξιά:** *Autodesk Maya*

σχεδιάστηκε από την εταιρεία *Alias Systems Corporation*, πρώην *Alias/Wavefront* αλλά πλέον ανήκει και αναπτύσσεται από την εταιρεία *Autodesk, Inc.* Χρησιμοποιείται για τρισδιάστατες διαδραστικές εφαρμογές συμπεριλαμβανομένων των βιντεοπαιχνιδιών, ταινιών κινουμένων σχεδίων, τηλεοπτικών σειρών και οπτικών εφφέ. Πήρε το όνομα του από το σανσκριτικό όνομα Maya, της ινδουιστικής θεάς της ψευδαίσθησης.

2.11.2 Μηχανές Γραφικών

Όπως αναφέραμε και σε προηγούμενη ενότητα [ενότητα 2.4] οι βιβλιοθήκες γραφικών υψηλού επιπέδου, συνήθως χρησιμοποιούνται μαζί με την κατάλληλη μηχανή γραφικών. Η χρήση τους, πέρα από την ανάπτυξη βιντεοπαιχνιδιών, είναι και η αρχιτεκτονική απεικόνιση, η δημιουργία εξομοιωτών για εκπαιδευτικούς σκοπούς και γενικά η σχεδίαση γραφικών σε πραγματικό χρόνο [15]. Παρ' όλα αυτά η χρήση τους για ανάπτυξη παιχνιδιών δεν είναι τόσο διαδεδομένη, λόγω της έλλειψης δυνατοτήτων που είναι βασικές για τα βιντεοπαιχνίδια, όπως εισαγωγή ήχου, διαχείριση δικτύου κ.α., που προσφέρουν οι μηχανές παιχνιδιών. Εντούτοις αν χρησιμοποιηθεί μια μηχανή γραφικών για τη δημιουργία ενός βιντεοπαιχνιδιού, το βιντεοπαιχνίδι που θα παραχθεί θα είναι ολοκληρωμένο (σχήμα 2.17). Διαδεδομένες μηχανές τρισδιάστατων γραφικών είναι:

- **Irrlicht:** Η μηχανή *Irrlicht* είναι μια ανοιχτού κώδικα, υψηλών επιδόσεων, πραγματικού χρόνου τρισδιάστατη μηχανή γραμμένη σε *C++* [30]. Είναι πολλαπλών πλατφόρμων, χρησιμοποιεί *Direct3D*, *OpenGL* αλλά και δικιές της βιβλιοθήκες, και περιέχει όλες τις δυνατότητες που μπορεί κανείς να βρει σε όλες τις εμπορικές τρισδιάστατες μηχανές. Πολλά προγράμματα έχουν σχεδιαστεί με χρήση αυτής της μηχανής που υποστηρίζει σύνταξη σε πολλές γλώσσες όπως *Java*, *Perl*, *Ruby*, *Basic*, *Python*, *Lua* κ.α.

Σχήμα 2.17: Παιχνίδια σχεδιασμένα με μηχανές γραφικών
Αριστερά: *Aerosion* (*Irrlicht*-2009) **Κέντρο:** *Stained* (*Ogre3D*-2012) **Δεξιά:** *Retago* (*Horde3D*-2010)

- **Ogre3D:** Η *Ogre3D* είναι μια προσανατολισμένη στη σκηνή, πραγματικού χρόνου, ευέλικτη μηχανή τρισδιάστατης απεικόνισης [31]. Είναι γραμμένη σε *C++* και έχει αναπτυχθεί ώστε να κάνει πιο εύκολη στους σχεδιαστές την δημιουργία εφαρμογών, αξιοποιώντας την επιτάχυνση υλικού (hardware acceleration). Η κλάση της βιβλιοθήκης της, αφαιρεί τις λεπτομέρειες από την χρήση των συστημικών βιβλιοθηκών, όπως της *Direct3D* και *OpenGL*, και παρέχει μια διεπαφή βασισμένη σε αντικείμενα και κλάσεις υψηλού επιπέδου.

- **Horde3D:** Είναι μια μικρή μηχανή προβολής τρισδιάστατων γραφικών, πολλαπλών πλατφόρμων [32]. Σχεδιάστηκε για να υποστηρίζει μεγάλο αριθμό animated χαρακτήρων και πολλές γλώσσες προγραμματισμού.

2.11.3 Μηχανές Παιχνιδιών

Μια μηχανή παιχνιδιών είναι ένα σύστημα σχεδιασμένο για την δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών [33]. Οι κορυφαίες μηχανές παιχνιδιών παρέχουν ένα πλήρες λογισμικό ώστε οι σχεδιαστές να δημιουργήσουν ένα βιντεοπαιχνίδι για βιντεοκονσόλες, κινητά τηλέφωνα και προσωπικούς υπολογιστές. Ο πυρήνας των λειτουργιών, που τυπικά παρέχεται από μια τέτοια μηχανή, περιέχει έναν προβολέα δισδιάστατων και τρισδιάστατων γραφικών, μια μηχανή φυσικής ή ένα σύστημα εντοπισμού συγκρούσεων και αντίδρασης σε αυτές, ήχους, σύνταξη κώδικα μέσω scripting, τεχνητή νοημοσύνη, διεπαφή με το δίκτυο, ροή δεδομένων, διαχείριση μνήμης, πολυνηματική εκτέλεση, υποστήριξη εντοπισμού και έναν γράφο της σκηνής μας (σχήματα 2.18, 2.19). Γνωστές μηχανές παιχνιδιών είναι:

- **Unity3D:** Η *Unity3D* είναι μια πολλαπλών πλατφόρμων μηχανή παιχνιδιών με εγκατεστημένο IDE[1], που αναπτύχθηκε από την εταιρεία *Unity Technologies* [35]. Χρησιμοποιείται για την ανάπτυξη βιντεοπαιχνιδιών στο διαδίκτυο, για πλατφόρμες επιτραπέζιων υπολογιστών, για βιντεοκονσόλες και κινητά τηλέφωνα, και έχει χρησιμοποιηθεί από πάνω από ένα εκατομμύριο σχεδιαστές. Αρχικά ήταν ένα υποστηρικτικό εργαλείο ανάπτυξης βιντεοπαιχνιδιών για πλατφόρμες *Mac OS* αλλά αργότερα εξελίχτηκε σε πολλαπλών πλατφόρμων.

  Με την τελευταία έκδοση, του Ιουλίου 2013, υποστηρίζει ανάπτυξη παιχνιδιών για λειτουργικά *iOS*, *Android*, *Windows*, *Blackeberry 10*, *Mac OS X*, *Linux*, για περιηγητές διαδικτύου, *Flash*, *PlayStation 3*, *Xbox 360*, *Windows Phone 8*, και *Wii U*. Η μηχανή γραφικών της, χρησιμοποιεί βιβλιοθήκες *Direct3D*, *OpenGL*, *OpenGL ES*, και κάποιες άλλες κατάλληλες για *Wii*. Στην τελευταία του έκδοση υποστηρίζει και *DirectX 11*. Χρησιμοποιεί την μηχανή φυσικής *NVIDIA PhysX* που περιγράφεται παρακάτω [υποενότητα 2.11.4].

  Υποστηρίζει εισαγωγή γεωμετριών από διάφορα λογισμικά όπως *Autodesk 3ds Max*, *Autodesk Maya*, *Softimage*, *Blender*, *Modo*, *ZBrush*, *Cinema 4D*, *Cheetah3D*, *Adobe Photoshop*, *Adobe Fireworks* κ.α. Υποστηρίζει σύνταξη κώδικα scripting μέσα από τρεις γλώσσες *C#*, *Java* και *Boo*, η οποία τελευταία είναι μια γλώσσα που ανέπτυξε η εταιρεία βασισμένη στην *Python*.

  Υπάρχουν δύο εκδόσεις της, μια ελεύθερη και μια Pro έναντι πληρωμής. Η *Unity3D* έχει μια μεγάλη online κοινότητα για συζήτηση και απορίες καθώς και ένα ηλεκτρονικό κατάστημα με δωρεάν και επί πληρωμή projects και τρισδιάστατα αντικείμενα.

- **Unreal Engine:** Η *Unreal Engine* είναι μια μηχανή παιχνιδιών που σχεδιάστηκε από την εταιρεία *Epic Games*, και πρωτοχρησιμοποιήθηκε στην σχεδίαση του παιχνιδιού πρώτου-προσώπου *Unreal*, το 1998 [36]. Παρ΄ ότι αρχικά χρησιμοποιήθηκε για παιχνίδια πρώτου-προσώπου, έχει χρησιμοποιηθεί με επιτυχία σε ένα μεγάλο πλή-

[1]IDE: Integrated/Interactive Development Environment. Ένα ολοκληρωμένο/διαδραστικό περιβάλλον ανάπτυξης είναι μια ολοκληρωμένη υποδομή, για προγραμματιστές υπολογιστών, με σκοπό την ανάπτυξη λογισμικού. Αποτελείται συνήθως από ένα πρόγραμμα επεξεργασίας πηγαίου κώδικα, εργαλεία χτισίματος (build) και ένα πρόγραμμα εντοπισμού σφαλμάτων [34].

θος διαφορετικών ειδών όπως stealth, ρόλων (RPGs), και ρόλων πολλαπλών παιχτών (MMORPGs). Με τον κώδικά της γραμμένο σε *C++*, η *Unreal Engine* διαθέτει υψηλό βαθμό φορητότητας και είναι ένα εργαλείο που χρησιμοποιείται από πολλούς σχεδιαστές παιχνιδιών στις μέρες μας.

Η τελευταία έκδοση *Unreal Engine 3*, έχει σχεδιαστεί για *Microsoft DirectX 9* (σε *Windows* και *Xbox 360*), *DirectX 10* (για *Windows Vista*) και *DirectX 11* (για εκδόσεις *Windows 7* και μετά)· *OpenGL* για *Mac OS X*, *Linux*, *PlayStation 3*, *Wii U*, *iOS*, *Android*· *Stage 3D* για *Adobe Flash Player 11*· και *JavaScipt/WebGL* για *HTML5*. Έχει επίσης δικιά της γλώσσα για scripting κώδικα την *UnrealScript*.

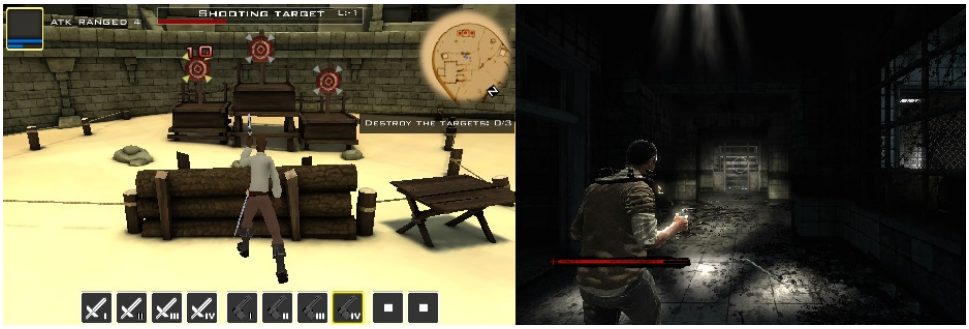
- **CryEngine:** Η *CryEngine* είναι μια μηχανή παιχνιδιών που σχεδιάστηκε από την γερμανική εταιρεία *Crytek* [37]. Το *CryEngine 3 Free SDK*[1] είναι η τωρινή έκδοση του συντάκτη εδάφους που χρησιμοποιεί η μηχανή. Στο λογισμικό αυτό παρέχονται εργαλεία όπως σύνταξη κώδικα scritping, animation και δημιουργία αντικειμένων. Ο συντάκτης αυτός δίνει έμφαση στα μεγάλα εδάφη και στο ελεύθερο στυλ του προγραμματισμού της αποστολής του παιχνιδιού.

  Ο συντάκτης εδάφους επίσης υποστηρίζει όλες τις δυνατότητες της μηχανής όπως οχήματα και φυσική, προηγμένο φωτισμό (φωτισμό πραγματικού χρόνου, κινούμενες σκιάσεις), σκιαστές, τρισδιάστατο ήχο, χαρακτήρες αντίστροφης κινηματικής, ανάμειξη animation, δυναμική μουσική, συστήματα μαλακών σωματιδίων πραγματικού χρόνου, αναβαλλόμενος φωτισμός και σύνθετη τεχνητή νοημοσύνη.

- **Panda3D:** Το *Panda3D* είναι μια μηχανή παιχνιδιών που περιέχει γραφικά, ήχο, συστήματα εισόδου/εξόδου, ανίχνευση συγκρούσεων και άλλες δυνατότητες που είναι σχετικές με την δημιουργία τρισδιάστατων παιχνιδιών [39]. Είναι ένα ελεύθερο λογισμικό ανοιχτού κώδικα, κάτω από την αναθεωρημένη άδεια BSD.

  Η γλώσσα που υποστηρίζει η μηχανή για την ανάπτυξη των παιχνιδιών είναι η *Python*. Η ίδια η μηχανή είναι γραμμένη σε *C++* και χρησιμοποιεί μια αυτόματη γεννήτρια για

[1] SDK: Software Development Kit. Βοήθημα ανάπτυξης λογισμικού δηλαδή ένα σύνολο εργαλείων που συμβάλλουν στην δημιουργία ενός λογισμικού. Μπορεί να περιλαμβάνει πράγματα όπως APIs, IDE, προγράμματα εντοπισμού σφαλμάτων κ.α. [38].



Σχήμα 2.18: Παιχνίδια σχεδιασμένα με μηχανές παιχνιδιών
Αριστερά: *Blackreef Pirates* (*Unity3D*-2011) **Δεξιά:** *Saw* (Unreal Engine 3-2009)

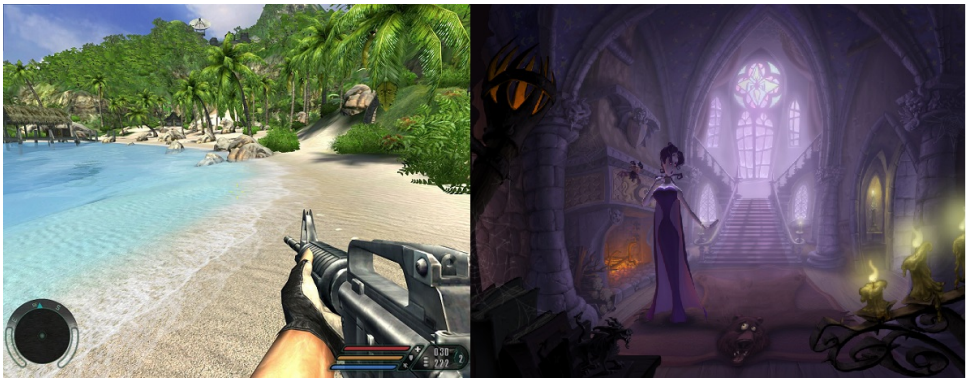
να παρέχει την πλήρη λειτουργικότητα της σε διεπαφή της *Python*. Αυτή η προσέγγιση δίνει στον σχεδιαστή τα πλεονεκτήματα της ανάπτυξης σε *Python*, όπως γρήγορη ανάπτυξη και προηγμένη διαχείριση μνήμης, αλλά διατηρεί ταυτόχρονα την δυνατότητα μιας μεταγλωττισμένης γλώσσας στον πυρήνα της μηχανής. Για παράδειγμα, η μηχανή έχει ενσωματωμένο τον διαχειριστή σκουπιδιών της *Python* και οι δομές της μηχανής διαχειρίζονται αυτόματα. Ο σχεδιαστής μπορεί επίσης εκτός από *Python*, να έχει άμεση πρόσβαση στη μηχανή χρησιμοποιώντας κώδικα *C++*.

Η *Panda3D* έχει μια μικρή κοινότητα που παρέχει στους χρήστες μεγάλη ποικιλία εμπορικών παιχνιδιών, κάποια projects ανοιχτού κώδικα και ένα χώρο συζήτησης που απαντώνται απορίες πάνω στη μηχανή.

Για την δική μας υλοποίηση, επιλέξαμε την μηχανή παιχνιδιών της *Unity3D*. Οι λόγοι, που μας οδήγησαν σε αυτή την επιλογή, προήλθαν μετά από εκτενή έρευνα γύρω από τις γνωστές μηχανές παιχνιδιών και περιγράφονται παρακάτω [ενότητα 3.4].

2.11.4 Μηχανές Φυσικής

Μια μηχανή φυσικής είναι ένα λογισμικό που παρέχει προσεγγιστική προσομοίωση ορισμένων φυσικών συστημάτων, όπως δυναμική άκαμπτου σώματος (συμπεριλαμβανομένης της ανίχνευσης συγκρούσεων), δυναμική μαλακού σώματος, και δυναμική υγρού, για χρήση σε διάφορους τομείς των γραφικών υπολογιστών, σε βιντεοπαιχνίδια και ταινίες. Η κύρια χρήση της είναι σε βιντεοπαιχνίδια, όπου εκεί οι προσομοιώσεις είναι σε πραγματικό χρόνο [40].



Σχήμα 2.19: Παιχνίδια σχεδιασμένα με μηχανές παιχνιδιών
Αριστερά: *Far Cry* (*CryEngine*-2004) **Δεξιά:** *A vampyre story* (*Panda3D-2008*)

Γενικά υπάρχουν δύο κλάσεις μηχανών φυσικής: οι πραγματικού χρόνου και οι υψηλής ακρίβειας. Οι μηχανές φυσικής υψηλής ακρίβειας απαιτούν περισσότερη ισχύ επεξεργασίας για να υπολογίσουν με ακρίβεια την φυσική και για αυτό συνήθως χρησιμοποιούνται από επιστήμονες και σε ταινίες κινουμένων σχεδίων. Οι μηχανές φυσικής πραγματικού χρόνου όπως χρησιμοποιούνται σε βιντεοπαιχνίδια και σε άλλες μορφές διαδραστικών εφαρμογών χρησιμοποιούν απλουστευμένους υπολογισμούς και μειωμένη ακρίβεια για να μπορέσει το παιχνίδι να αντιδράσει σε ένα εύλογο χρονικό διάστημα. Αρκετά διαδεδομένες μηχανές φυσικής είναι η *PhysX*, η *Havok* και η *ODE*.

Η *PhysX* είναι μια μηχανή φυσικής πραγματικού χρόνου που χρησιμοποιείται ως SDK πρόγραμμα ενσωματωμένο μέσα σε κάποιο λογισμικό [41]. Αναπτύχθηκε από την εταιρεία *Ageia* το 2004 και ανήκει πλέον στην *NVIDIA* από το Φεβρουάριο του 2008. Τα βιντεοπαιχνίδια που υποστηρίζουν επιτάχυνση υλικού από την μηχανή *PhysX*, μπορούν να επιταχυνθούν ή με χρήση της λειτουργίας *PhysX PPU* είτε με ενεργοποίηση της λειτουργίας *CUDA* σε κάρτες γραφικών *GeForce* της *NVIDIA* (αν έχουν τουλάχιστον 256MB μνήμης). Με αυτό τον τρόπο αφαιρούνται οι υπολογισμοί της φυσικής από τον επεξεργαστή, επιτρέποντάς του να εκτελεί και άλλες λειτουργίες ταυτόχρονα.

Η μηχανή *PhysX* είναι διαθέσιμη για πλατφόρμες *Microsoft Windows*, *Mac OS X*, *Linux*, *PlayStation 3*, *Xbox 360* και *Wii*. Είναι επίσης εγκατεστημένη σε πολλές μηχανές παιχνιδιών όπως τις *Unity3D*, *Unreal Engine 3*, *Panda3D*, *Torque*, σε μηχανές γραφικών όπως

την *Ogre3D* και σε λογισμικά τρισδιάστατων γραφικών όπως το *Autodesk 3ds Max* και το *Autodesk Maya*. Τέλος έχει χρησιμοποιηθεί σε πολλά δημοφιλή παιχνίδια όπως τα *Need for Speed: Shift*, *Mafia II* και *Bordelands 2*.

2.12 Τεχνητή Νοημοσύνη

Στα βιντεοπαιχνίδια, η τεχνητή νοημοσύνη χρησιμοποιείται για να δώσει την ψευδαίσθηση της νοημοσύνης στην συμπεριφορά των παιχτών που δεν τους διαχειρίζεται κάποιος χρήστης [42]. Οι τεχνικές που χρησιμοποιούνται, συνήθως αξιοποιούν μεθόδους από το πεδίο της τεχνητής νοημοσύνης. Παρ΄ όλα αυτά, ο όρος τεχνητή νοημοσύνη στα βιντεοπαιχνίδια, συνήθως χρησιμοποιείται για να αναφερθεί σε ένα σύνολο από αλγορίθμους που περιέχουν επίσης τεχνικές από τη θεωρία ελέγχου, ρομποτική, γραφικά υπολογιστών και γενικά την επιστήμη υπολογιστών.

Δεδομένου ότι η τεχνητή νοημοσύνη στα βιντεοπαιχνίδια επικεντρώνεται στην εμφάνιση της νοημοσύνης ενός καλού gameplay, η προσέγγιση του διαφέρει από αυτή της παραδοσιακής τεχνητής νοημοσύνης· διάφορες λύσεις και απατεωνιές είναι αποδεκτές. Παρ΄ όλα αυτά στις μέρες μας, σχεδόν σε όλα τα βιντεοπαιχνίδια, η τεχνητή νοημοσύνη που υλοποιείται προσεγγίζει τον τρόπο παιξίματος ενός ανθρώπινου παίχτη, έχοντας την αίσθηση του δίκαιου χωρίς απατεωνιές.

Οι σκοπιές από τις οποίες βλέπουμε την τεχνητή νοημοσύνη, και στη δική μας υλοποίηση αλλά και στα περισσότερα παιχνίδια, είναι πως θα κινηθούν οι παίχτες στο χώρο και έπειτα με ποιό τρόπο θα αποφασίσουν να αντιδράσουν. Η κίνηση του παίχτη υλοποιείται μέσω ενός αλγορίθμου εύρεσης βέλτιστου μονοπατιού (Path Planning). Ο αλγόριθμος που επιλέχθηκε είναι αυτός του A* [υποενότητα 2.12.2]. Ο A* είναι ο πιο διαδεδομένος και αποδοτικότερος αλγόριθμος σε αυτές τις περιπτώσεις. Για να έχουμε ταχύτερη εκτέλησή του, χρησιμοποιήθηκε με κάποιες βελτιώσεις [υποενότητα 2.12.3].

2.12.1 Λήψη Απόφασης (Decision Making)

Η λήψη απόφασης θεωρείται εν γένει η γνωστική διαδικασία με σκοπό την επιλογή της πορείας δράσης μεταξύ διαφόρων εναλλακτικών σεναρίων [43]. Κάθε διαδικασία λήψης απόφασης παράγει μια τελική επιλογή η οποία μπορεί να είναι μια ενέργεια ή μια γνωμοδότηση

της επιλογής. Η θεωρία παιγνίων είναι ο τομέας μελέτης της λήψης απόφασης. Ο τρόπος με τον οποίο θα ληφθεί μια απόφαση εξαρτάται άμεσα από τα στοιχεία του παιχνιδιού (αιτιοκρατικό/στοχαστικό, ατελής/τέλεια πληροφόρηση κ.α.). Υπάρχουν διάφοροι μέθοδοι και μοντέλα λήψης απόφασης εκ των οποίων γνωστά είναι οι συναρτήσεις χρησιμότητας, τα δίκτυα αποφάσεων *Bayes*, οι διαδικασίες απόφασης *Markov*, η ασαφής λογική κ.α.

Οι συναρτήσεις χρησιμότητας (utility functions) είναι από τις βασικότερες και πιο διαδεδομένες μεθόδους λήψης απόφασης στην θεωρία παιγνίων [44]. Με τις συναρτήσεις χρησιμότητας δίνεται μια αντικειμενική αριθμητική τιμή σε όλες τις πιθανές ενέργειες που μπορεί να εκτελέσει ο πράκτορας, και εκτιμούν πόσο γρήγορα θα τον οδηγήσουν στην τελική κατάσταση (νίκη σε ένα παιχνίδι). Αυτές οι συναρτήσεις παίρνουν διάφορες μορφές ανάλογα με το είδος του παιχνιδιού (π.χ. πιθανοτικό μοντέλο σε περίπτωση τυχαιότητας). **Μονοτονική** ονομάζεται η προτίμηση, όταν οι ενέργειες που προτιμούνται περισσότερο έχουν πάντα μεγαλύτερη αξία από αυτές που προτιμούνται λιγότερο. Σε αυτή την περίπτωση οι συναρτήσεις χρησιμότητας ονομάζονται **διατακτικές**, και είναι αυτές που μας ενδιαφέρουν στην παρούσα φάση.

2.12.2 Εύρεση Βέλτιστου Μονοπατιού (Path Planning)

Τα περισσότερα βιντεοπαιχνίδια στις μέρες μας δεν είναι στατικά αλλά υπάρχει κίνηση των παιχτών. Έτσι οι παίχτες τους οποίους διαχειρίζεται ο υπολογιστής, δεν πρέπει μόνο να σκέφτονται έξυπνα αλλά και να κινούνται έξυπνα. Αυτό γίνεται μέσα από την διαδικασία της εύρεσης του συντομότερου/βέλτιστου μονοπατιού μεταξύ δύο σημείων. Αυτή η διαδικασία είναι ένα από τα σημαντικότερα κομμάτια που απαρτίζουν την τεχνητή νοημοσύνη των βιντεοπαιχνιδιών. Η υλοποίησή της γίνεται μέσα από διάφορους αλγορίθμους όπως τον αλγόριθμο του *Dijkstra*, ο D* και ο επικρατέστερος A* [45].

Ο A* είναι ένας αλγόριθμος που χρησιμοποιείται ευρέως για την εύρεση μονοπατιού και την διάσχιση γράφων, την διαδικασία δηλαδή σχεδιασμού ενός αποδοτικού μονοπατιού μεταξύ δύο σημείων, που ονομάζονται κόμβοι [46]. Οφείλει την ευρεία χρήση του στον ότι είναι πλήρης, βέλτιστος και αρκετά αποδοτικός. Αποτελεί επέκταση του αλγορίθμου του *Dijkstra* χρησιμοποιώντας ευριστικές συναρτήσεις που τον κάνουν αποδοτικότερο.

Ο A* είναι ένας αλγόριθμος αναζήτησης πρώτα στο καλύτερο (best-first search), ο οποίος

βρίσκει το μικρότερο μονοπάτι μεταξύ ενός αρχικού και ενός τελικού κόμβου (μέσα σε ένα σύνολο τελικών κόμβων). Όπως ο A* διασχίζει τον γράφο, ακολουθεί ένα μονοπάτι προς την εκτιμώμενα φθηνότερη διαδρομή, κρατώντας μια ταξινομημένη ουράς προτεραιότητας με τα εναλλακτικά μονοπάτια.

Το εκτιμώμενο κόστος της φθηνότερης διαδρομής ($f(n)$) σε ένα κόμβο n, υπολογίζεται από το άθροισμα του κόστους διαδρομής από τον αρχικό κόμβο στον κόμβο n ($g(n)$), και το εκτιμώμενο κόστος της φθηνότερης διαδρομής από τον κόμβο n στον στόχο ($h(n)$):

$$f(n) = g(n) + h(n) \tag{2.1}$$

Η συνάρτηση $h(n)$ είναι η ευριστική συνάρτηση αφού είναι εκτίμηση του κόστους. Ο αλγόριθμος είναι βέλτιστος μόνο εάν η ευριστική συνάρτηση είναι συνεπής και παραδεκτή, κάτι το οποίο συμβαίνει όταν αυτή δεν υπερεκτιμά το κόστος επίτευξης του στόχου. Για να ισχύει αυτό πρέπει:

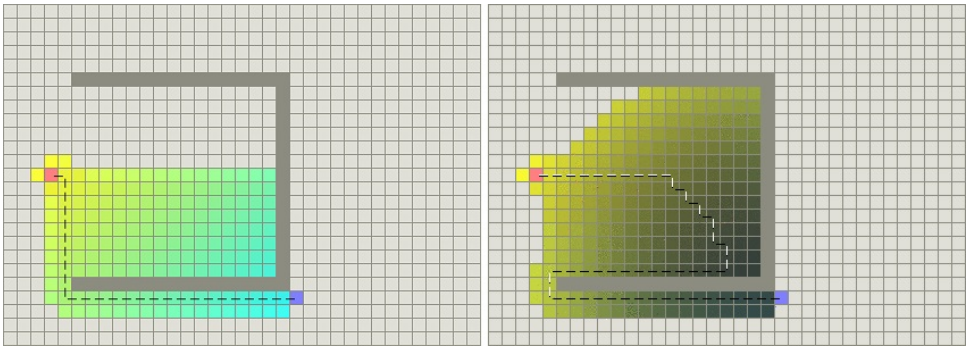
$$h(n) \leq c(n, a, n') + h(n') \tag{2.2}$$

Η εξίσωση 2.2 δείχνει ότι για κάθε κόμβο n και για κάθε διάδοχο n' του n που παράγεται από οποιαδήποτε ενέργεια a, το εκτιμώμενο κόστος της επίτευξης από τον n στο στόχο δεν είναι μεγαλύτερο από το κόστος βήματος της μετάβασης στον n' συν το εκτιμώμενο κόστος της επίτευξης από τον n' στο στόχο [44].

Υπάρχουν πολλές ευριστικές συναρτήσεις για την εύρεση βέλτιστου μονοπατιού όπως η διαγώνια συντόμευση, η ευκλείδεια απόσταση και πιο διαδεδομένη η απόσταση Manhattan, που είναι η απόσταση δύο σημείων που υπολογίζεται ως το άθροισμα των απόλυτων διαφορών των καρτεσιανών συντεταγμένων τους:

$$d(\mathbf{p}, \mathbf{q}) = |p_x - q_x| + |p_y - q_y| \tag{2.3}$$

Όπως όλοι οι αλγόριθμοι πληροφορημένης αναζήτησης, έτσι και ο A* ψάχνει το μονοπάτι που φαίνεται πιο πιθανό ότι θα φτάσει στο στόχο. Αυτό που διαχωρίζει τον A* από έναν άπληστο αλγόριθμο αναζήτησης πρώτα στο καλύτερο είναι ότι λαμβάνει υπ' όψιν την απόσταση που έχει διανυθεί μέχρι τότε. Η συνάρτηση $g(n)$ είναι το κόστος διαδρομής από τον αρχικό κόμβο και όχι μόνο από τον προηγούμενο (σχήμα 2.20).



Σχήμα 2.20: Εύρεση βέλτιστου μονοπατιού
Αριστερά: A* **Δεξιά:** Άπληστος αλγόριθμος αναζήτησης πρώτα στο καλύτερο (Greedy best-first search). Με ροζ σημειώνεται ο αρχικός κόμβος, μωβ ο τελικός και χρωματισμένοι οι κόμβοι που έχουν επεκταθεί.

Ξεκινώντας από τον αρχικό κόμβο, ο αλγόριθμος κρατάει μια ουρά προτεραιότητας των κόμβων που πρέπει να διασχίσει, γνωστός ως ανοιχτή λίστα. Όσο μικρότερη είναι η τιμή της $f(n)$ για ένα κόμβο n, τόσο μεγαλύτερη είναι η προτεραιότητά του. Σε κάθε βήμα του αλγορίθμου, ο κόμβος με την μικρότερη τιμή $f(n)$ αφαιρείται από την ουρά, οι τιμές των συναρτήσεων f και g των γειτόνων ενημερώνονται κατάλληλα, και αυτοί οι γείτονες προστίθενται στην ουρά. Ο αλγόριθμος συνεχίζει μέχρι ένας κόμβος στόχου να έχει μικρότερη τιμή f από οποιονδήποτε κόμβο στην ουρά (ή μέχρι να αδειάσει η ουρά). Η τιμή της f του στόχου είναι το μήκος του συντομότερου μονοπατιού, μιας και η συνάρτηση h στον στόχο έχει τιμή μηδέν αν η ευριστική συνάρτηση είναι παραδεκτή. Σε περίπτωση που η ευριστική συνάρτηση είναι συνεπής (όπως αυτές που αναφέραμε παραπάνω), χρησιμοποιείται μια κλειστή λίστα των κόμβων που ήδη διασχίστηκαν ώστε να κάνει την αναζήτηση πιο αποδοτική.

Ο αλγόριθμος παρουσιάζεται παρακάτω (αλγόριθμος 2.1) σε μορφή ψευδοκώδικα και μας επιστρέφει το μήκος της συντομότερης διαδρομής. Σε περίπτωση που θέλουμε να μας επιστραφεί τα βήματα του ίδιου του μονοπατιού τον παραλλάζουμε κατάλληλα.

Αλγόριθμος 2.1: Ψευδοκώδικας αλγορίθμου A*

```
function A* (start, goal)
    closedset = the empty set                ▷ The set of nodes already evaluated.
    openset = {start}                        ▷ The map of navigated nodes.
```

```
    g_score[start] = 0                          ▷ Cost from start along best known path.
                                                ▷ Estimated total cost from start to goal through y.
    while openset is not empty do
        current = the node in openset having the lowest f_score[ ] value
        if current == goal then
            return RECONSTRUCT_PATH (came_from, goal)
        end if
        remove current from openset
        add current to closeset
        for each neighbor in neighbor_nodes (current) do
            tentative_g_score = g_score[current] + dist_between (current, neighbor)
            if neighbor in closedset and tentative_g_score ≥ g_score[neighbor] then
                continue
            end if
            if neighbor not in openset or tentative_g_score < g_score[neighbor] then
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = g_score[neighbor] + heuristic_cost_estimate(neighbor,
goal)
                if neighbor not in openset then
                    add neighbor to openset
                end if
            end if
        end for
    end while
    return failure
end function

function RECONSTRUCT_PATH (came_from, current_node)
    if current_node in came_from then
        p = RECONSTRUCT_PATH (came_from, came_from[current_node])
        return (p + current_node)
    else
        return current_node
```

end if
end function

Η χρονική πολυπλοκότητα του A* εξαρτάται από την ευριστική συνάρτηση. Στην χειρότερη περίπτωση, ο αριθμός των κόμβων που θα επεκταθούν είναι εκθετικός ως προς το μέγεθος της λύσης (συντομότερο μονοπάτι), αλλά είναι πολυωνυμικός όταν ο χώρος αναζήτησης είναι δέντρο, υπάρχει μόνο ένας στόχος και η ευριστική συνάρτηση πληροί την παρακάτω συνθήκη:

$$|h(n) - h^*(n)| = O(\log h^*(n)) \tag{2.4}$$

όπου h^* είναι η βέλτιστη ευριστική συνάρτηση, δηλαδή το ακριβές κόστος να πάει από τον κόμβο n στο στόχο. Με άλλα λόγια, το σφάλμα της h δεν θα αυξηθεί ταχύτερα από τον λογάριθμο της "τέλειας" ευριστικής συνάρτησης που επιστρέφει την πραγματική απόσταση από τον κόμβο n στο στόχο.

2.12.3 Βελτίωση του Αλγόριθμου A*

Η πιο συνηθισμένη βελτίωση του αλγόριθμου A*, είναι η χρήση μια ουράς προτεραιότητας (priority queue) για την ανοιχτή λίστα αντί μιας απλής συνδεδεμένης λίστας [15]. Με αυτόν τον τρόπο αναζητείται ο κόμβος με το μικρότερο $f(n)$ και η χρονική πολυπλοκότητα του αλγορίθμου από $O(N)$ γίνεται $O(1)$. Η θεωρητική συνολική χρονική πολυπλοκότητα του αλγορίθμου δεν επηρεάζεται, όμως πρακτικά αποφεύγονται πολλές περιττές συγκρίσεις.

Η πιο συνηθισμένη δομή για την υλοποίηση της ουράς προτεραιότητας είναι αυτή του δυαδικού σωρού (binary heap) (αλγόριθμος 2.2) [47]. Ο δυαδικός σωρός είναι μια δομή πίνακα που μπορεί να αναπαρασταθεί σαν ένα ολοκληρωμένο δυαδικό δέντρο. Κάθε κόμβος του δέντρου αντιπροσωπεύει ένα στοιχείο του πίνακα. Ο πίνακας είναι γεμάτος σε όλα τα επίπεδα εκτός ίσως από τα τελευταία.

Αν $A[1]$ είναι το στοιχείο του πίνακα που αποτελεί ρίζα του δέντρου και i ο δείκτης ενός κόμβου τότε μπορούμε εύκολα να υπολογίσουμε τους δείκτες του πατέρα του, του αριστερού και του δεξιού του παιδιού.

Αλγόριθμος 2.2: Ψευδοκώδικας δυαδικού σωρού

```
function PARENT (i)
    return FLOOR (i/2)
end function
function LEFT (i)
    return 2i
end function
function RIGHT (i)
    return 2i + 1
end function
```

Επειδή θέλουμε όμως τα στοιχεία να είναι ταξινομημένα (ουρά προτεραιότητας), στην ουσία πρέπει να εκτελέσουμε ταξινόμηση σωρού (heapsort) στον πίνακα. Η ταξινόμηση σωρού είναι μια πάρα πολύ γρήγορη διαδικασία με μέση και χειρότερη περίπτωση $O(nlog(n))$. Παρ' όλα αυτά η ταξινόμηση quicksort, ενώ θεωρητικά έχει ίδια πολυπλοκότητα μέσης περίπτωσης, θεωρείται πιο γρήγορη από την ταξινόμηση σωρού. Έχει όμως πολυπλοκότητα χειρότερης περίπτωσης $O(n^2)$.

Ένας έξυπνος και αποδοτικός τρόπος ταξινόμησης, είναι η εσωστρεφής ταξινόμηση (introsort) [48]. Για τα πρώτα 16 στοιχεία του πίνακα εκτελείται ταξινόμηση με εισαγωγή (insertion sort). Για στοιχεία περισσότερα των 16 εκτελείται ταξινόμηση quicksort που είναι πολύ γρήγορη. Σε περίπτωση που τα διαμερίσματα που θα κοπεί ο πίνακας κατά τη διαδικασία της quicksort είναι περισσότερα από $2\lfloor log_2(N) \rfloor$ (όπου N το μέγεθος του πίνακα), τότε εκτελείται ταξινόμηση σωρού γιατί πλέον ξεπερνάει το κατώφλι εκείνο στο οποίο η quicksort είναι πιο γρήγορο από τη heapsort. Έτσι αυτή η υβριδική ταξινόμηση έχει πολυπλοκότητα μέσης και χειρότερης περίπτωσης $O(nlog(n))$ και ταυτόχρονα είναι πιο γρήγορη από την ταξινόμηση σωρού και δεν φτάνει στην χειρότερη περίπτωση της quicksort.

Κεφάλαιο 3

Τεχνολογική Βάση

3.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο θα κάνουμε μια αναφορά για το λογισμικό σχεδίασης γραφικών που επιλέξαμε· από ποιά μέρη απαρτίζεται, ποια βασικά εργαλεία έχει και πως τα χρησιμοποιήσαμε. Τέλος θα αναφερθούμε στη μηχανή παιχνιδιών που επιλέχθηκε. Πως διαχειρίζεται την γεωμετρία, ποιά είναι η δομή της και ποιά από τα στοιχεία της χρησιμοποιήσαμε για να υλοποιήσουμε το παιχνίδι.

3.2 Autodesk 3ds Max

Το *3ds Max* της *Autodesk* είναι ένα πλήρες λογισμικό σχεδίασης τρισδιάστατων γραφικών. Παρέχει ότι ακριβώς χρειάζεται ένας σχεδιαστής παιχνιδιών για να χτίσει την γεωμετρία του, από εργαλεία αντικειμένων μέχρι υφές, animations κ.α. Είναι από τα πιο διαδεδομένα λογισμικά σχεδίασης που χρησιμοποιούνται σήμερα σε παιχνίδια αλλά και σε ταινίες κινουμένων σχεδίων, εφφέ ταινιών και επεξεργασία διαφημιστικών σποτ.

Υπάρχουν και άλλα καλά λογισμικά που θα μπορούσαμε να έχουμε επιλέξει. Οι λόγοι που επιλέξαμε το συγκεκριμένο είναι:

- Πολλά άλλα λογισμικά του είδους του είναι ανοιχτού κώδικα ώστε να είναι σε ελεύθερη διάθεση στο κοινό, κάτι το οποίο τα κάνει να έχουν κάποια πλεονεκτήματα. Παρ' όλα αυτά τα λογισμικά ανοιχτού κώδικα έχουν τον κίνδυνο να εμφανίσουν σφάλματα πολύ

πιο εύκολα. Πολλές φορές τα σφάλματα αυτά διορθώνονται αλλά όχι πάντα με επιτυχία έτσι ώστε αυτό να έχει συνέπεια και σε επόμενες εκδόσεις.

- Η εταιρεία *Autodesk*, από την άλλη, έχει μια έμπειρη ομάδα προγραμματιστών που διορθώνουν, μετά από ελέγχους, τα σφάλματα στα λογισμικά της. Μιας και τα περισσότερα λογισμικά της είναι επί πληρωμή, αυτό δείχνει ευθύνη προς τον πελάτη-χρήστη. Επίσης διαθέτει μια φόρμα αναφοράς σφαλμάτων, στην οποία όταν αναφέρεις το σφάλμα σου, σου απαντούν σε σύντομο χρονικό διάστημα.

- Έχει μεγάλο αριθμό εργαλείων, αρκετό για να μπορέσει κάποιος να δημιουργήσει ο,τι επιθυμεί με πολλούς τρόπους.

- Υπάρχει ένα πλήρης οδηγός του στο διαδίκτυο, όπως και μια πληθώρα από εκπαιδευτικά βίντεο, που σε μαθαίνουν βήμα προς βήμα πως να το χρησιμοποιήσεις και ποιά είναι τα καλύτερα εργαλεία για κάθε ενέργεια που θες να κάνεις.

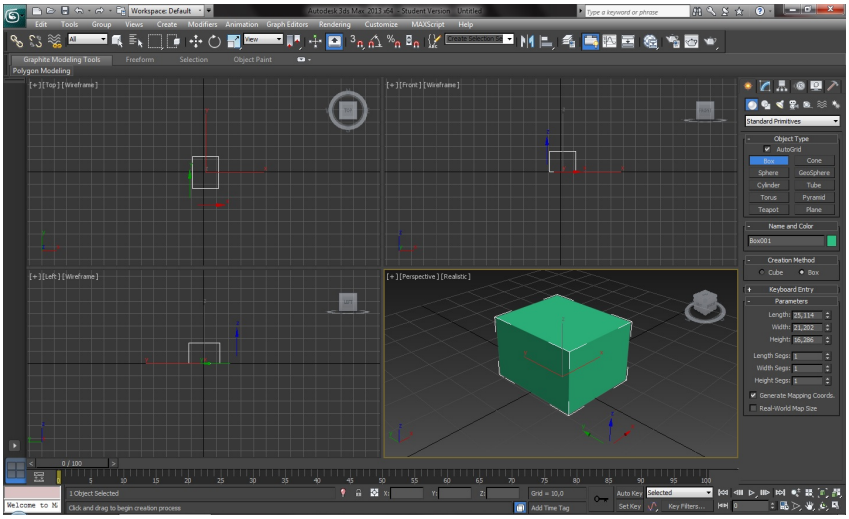
- Τέλος έχει μια διεπαφή πολύ φιλική προς τον χρήστη, που κάνει την εκμάθηση του πολύ εύκολη και σε πολύ μικρό χρονικό διάστημα σε σχέση με άλλα λογισμικά.

3.3 Δομή Εργαλείου Autodesk 3ds Max

Η βασική δομή του *Autodesk 3ds Max* αποτελείται από μια διεπαφή με την οποία επεξεργάζεσαι την εκάστοτε σκηνή σου (σχήμα 3.1) [49]. Η σκηνή μπορεί να περιέχει ένα ή και περισσότερα αντικείμενα. Τα αντικείμενα μπορείς να τα δεις μέσα σε διαφορετικά παράθυρα από μια πληθώρα διαφορετικών γωνιών (κάτοψη, πρόσοψη, προοπτική γωνία κ.α.) σε δισδιάστατη ή και τρισδιάστατη μορφή. Έχεις την δυνατότητα να περιστρέψεις την προβολή σου μέσα στα παράθυρα, να μετακινηθείς και να μεγεθύνεις. Μπορείς μέσα από την εργαλειοθήκη του να επιλέξεις ένα αντικείμενο ή ένα σύνολο αντικειμένων, να τα μετακινήσεις, να τα περιστρέψεις, να τα κλιμακώσεις και να τα ενώσεις σε μια ομάδα αντικειμένων. Τέλος μπορείς μέσα από τον πίνακα εντολών να σχεδιάσεις αντικείμενα, να δημιουργήσεις φωτισμούς, κάμερες και ότι άλλο χρειάζεται ώστε να έχεις μια ολοκληρωμένη σκηνή.

3.3.1 Αντικείμενα (Objects)

Μέσα από τον πίνακα εντολών μπορείς να δημιουργήσεις τα αντικείμενα της σκηνής σου (σχήμα 3.2(α΄)). Ο πίνακας προσφέρει κλασσικά τρισδιάστατα αντικείμενα όπως σφαίρα,

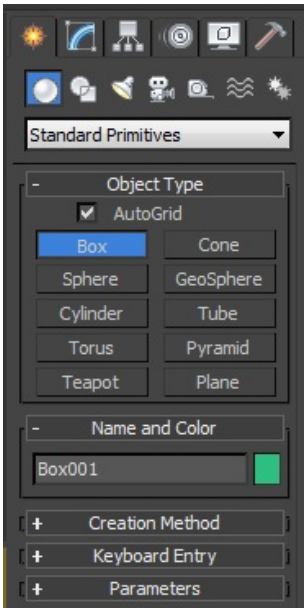


Σχήμα 3.1: Διεπαφή του *Autodesk 3ds Max*

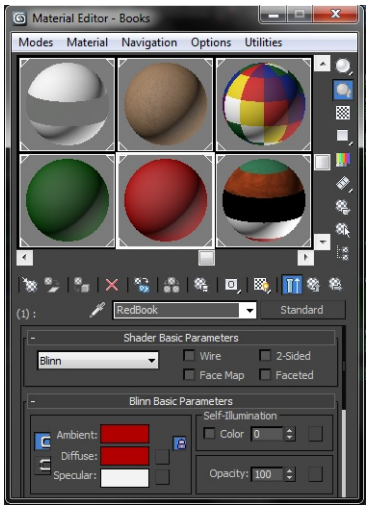
κύβο, πυραμίδα, κώνο κ.α. τα οποία μπορούν να συνδυαστούν και να επεξεργαστούν κατάλληλα ώστε να δώσουν το επιθυμητό αποτέλεσμα. Επίσης περιέχει δισδιάστατα σχέδια όπως γραμμή, τόξο, κύκλο, έλλειψη κ.α. στα οποία μπορείς συνδυάζοντάς τα να δημιουργήσεις δισδιάστατα αντικείμενα ή και τρισδιάστατα με χρήση συγκεκριμένων εργαλείων [υποενότητα 3.3.6]. Τέλος περιέχει μια πληθώρα αντικειμένων, που είναι πέρα των βασικών αλλά αποτελούν συχνή επιλογή των σχεδιαστών, όπως τοίχους, πόρτες, παράθυρα, φυλλώματα, συστήματα σωματιδίων κ.α. Μόλις δημιουργήσεις το εκάστοτε αντικείμενο, στον πίνακα εντολών δίνεις το όνομά του, τις διαστάσεις του και τιμές τους σε τυχόν άλλες παραμέτρους.

3.3.2 Υφές (Textures)

Το *Autodesk 3ds Max* έχει έναν συντάκτη δημιουργίας υφών που τοποθετούνται πάνω στα αντικείμενα (σχήμα 3.2(β′)). Στις υφές μπορεί κάποιος να τοποθετήσει μια εικόνα, ένα στέρεο χρώμα ή συνδυασμό αυτών. Μπορεί να φτιαχτεί οποιασδήποτε μορφής υφή αφού έχει ρυθμίσεις για το καθολικό, το διάχυτο και το κατοπτρικό χρώμα. Μπορούν να ρυθμιστούν τα επίπεδα κατοπτρικότητας, στιλπνότητας, αντανάκλασης, διάθλασης, διαφάνειας, αυτοφωτισμού και να τοποθετηθεί κάποιος χάρτης στο κάθε ένα. Τέλος υποστηρίζει τοποθέτηση πολλών υφών σε ένα αντικείμενο (multitexturing).



(α′) Αντικείμενα



(β′) Υφές

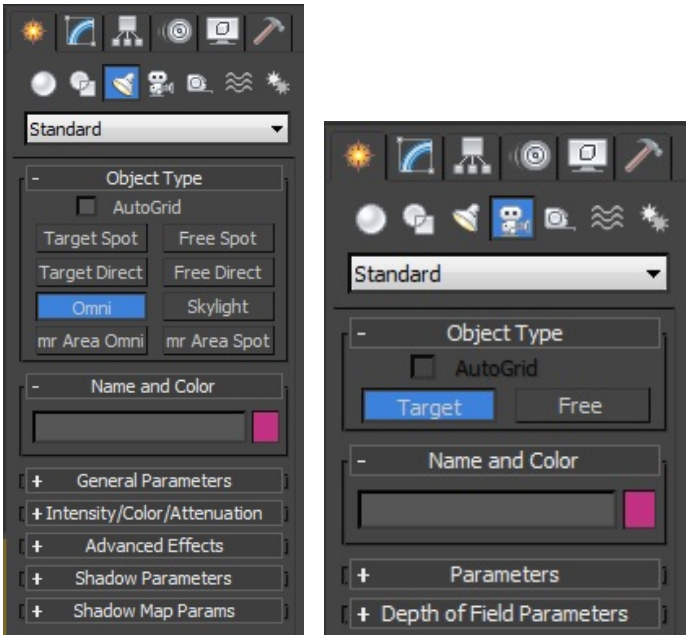
Σχήμα 3.2: Αντικείμενα και υφές στο *Autodesk 3ds Max*

3.3.3 Φωτισμός (Lighting)

Το *Autodesk 3ds Max* έχει πολλά είδη φωτισμού κατάλληλα για να φωτίσουν επαρκώς και με οποιοδήποτε τρόπο την σκηνή μας (σχήμα 3.3(α′)). Έχει κλασσικά φώτα όπως φώτα omni (φώτα προβολέα-τα οποία συμπεριφέρονται σαν το φως που βγαίνει από μια λάμπα), spot (φώτα σημείου-τα οποία αναπτύσσουν φως μέσα στο εύρος ενός κώνου), direct (απευθείας φώτα-τα οποία εκπέμπουν κάθετα από την πηγή προς μια κατεύθυνση) και μπορείς να επιλέξεις αν τα συγκεκριμένα φώτα θα βλέπουν σε κάποιο συγκεκριμένο στόχο ή όχι. Επίσης έχει φωτομετρικές πηγές και ουράνιο ηλιακό φως, για μεγαλύτερο ρεαλισμό στον φωτισμό. Τέλος σε κάθε πηγή μπορεί να ρυθμίσεις παραμέτρους όπως η ένταση, το χρώμα του φωτός, το μήκος απόσβεσής του, οι σκιάσεις κ.α.

3.3.4 Κάμερες (Cameras)

Στο λογισμικό αυτό παρέχονται δύο είδη καμερών: ελεύθερη και στόχου (σχήμα 3.3(β′)). Η στόχου κοιτάζει προς ένα σημείο και συνεπώς έχει μια συγκεκριμένη γωνία κατά την πλοήγηση ενώ με την ελεύθερη μπορείς να πλοηγηθείς με όποια γωνία θέλεις. Μπορείς να ρυθμίσεις παραμέτρους στις κάμερες όπως το εύρος και το μήκος του οπτικού πεδίου, το



(α′) Φωτισμοί (β′) Κάμερες

Σχήμα 3.3: Φωτισμοί και κάμερες στο *Autodesk 3ds Max*

μέγεθος των φακών, ορθογραφική ή μη προβολή κ.α.

3.3.5 Κινούμενα Σχέδια (Animations)

Στο κάτω μέρος του εργαλείου υπάρχει η λωρίδα χρόνου (timeline) για την προβολή των animations (σχήμα 3.4). Η λωρίδα μετράει το χρόνο σε δευτερόλεπτα ή χιλιοστά του δευτερολέπτου ή καρέ. Για κάθε αντικείμενο που θέλουμε να δημιουργήσουμε κινούμενη εικόνα δημιουργούμε συγκεκριμένα καρέ, όπου ανάμεσα τους αλλάζει μορφή, μέγεθος ή και θέση το αντικείμενο, και ονομάζονται καρέ ‘‘κλειδιά’’. Μπορούμε μεταξύ δυο καρέ κλειδιών να κάνουμε οποιεσδήποτε αλλαγές σε ένα αντικείμενο απ΄ το να το κλιμακώσουμε, να το περιστρέψουμε ή να το μετακινήσουμε μέχρι να του αλλάξουμε υφή και μορφή. Με αυτό τον τρόπο δημιουργείται αυτόματα από το λογισμικό η κινούμενη εικόνα μεταξύ των δύο καρέ που είναι και το επιθυμητό αποτέλεσμα.

Σχήμα 3.4: Λωρίδα χρόνου animations στο *Autodesk 3ds Max*

3.3.6 Τροποποιητές Αντικειμένων

Το Autodesk 3ds Max προσφέρει ένα μεγάλο αριθμό εργαλείων, που ονομάζονται τροποποιητές, και μας βοηθάνε να δώσουμε συγκεκριμένη μορφή στα αντικείμενά μας ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα [50]. Συνοπτικά κάποια από αυτά τα εργαλεία είναι:

- **Bend**: Λυγίζει ένα αντικείμενο επιλέγοντας το σημείο που θα λυγίσει, την κατεύθυνση και τις μοίρες.
- **Bind**: Ενώνει πολλαπλά αντικείμενα σε ένα ενιαίο.
- **Boolean**: Εκτελεί ένωση, τομή και αφαίρεση μεταξύ δύο αντικειμένων.
- **Editable Poly/Mesh**: Μετατρέπει ένα αντικείμενο σε επεξεργάσιμα πολύγωνα/δίκτυα γεωμετρίας. Μπορείς να τροποποιήσεις τα ίδια τα πολύγωνα, τις ακμές τους, τις κορυφές τους ή και ολόκληρα κομμάτια του αντικειμένου.
- **Lathe**: Μετατρέπει μια καμπύλη σε τρισδιάστατο αντικείμενο, περιστρέφοντάς την γύρω από έναν άξονα.
- **Slice**: Κόβει ένα αντικείμενο κάθετα προς έναν άξονα.
- **UVW mapping**: Δίνει τρισδιάστατες συντεταγμένες στις υφές ενός αντικειμένου, για σωστή προβολή τους πάνω σε ένα αντικείμενο.

3.3.7 Εξαγωγή σε μορφή VRML97

Πολλές φορές θέλουμε να δούμε τη γεωμετρία μας σε πραγματική κλίμακα και να πλοηγηθούμε μέσα σ' αυτήν, ακριβώς όπως θα κάνουμε κατά τη διάρκεια του παιχνιδιού. Ένας εύκολος και γρήγορος τρόπος για να υλοποιηθεί αυτό είναι μέσω της εξαγωγής της γεωμετρίας μας σε *VRML97*. Το *VRML* είναι μια τυποποιημένη μορφή αρχείου για την αναπαράσταση τρισδιάστατων διαδραστικών διανυσματικών γραφικών, σχεδιασμένο ειδικότερα για

Σχήμα 3.5: *VRML97* στο *Autodesk 3ds Max*

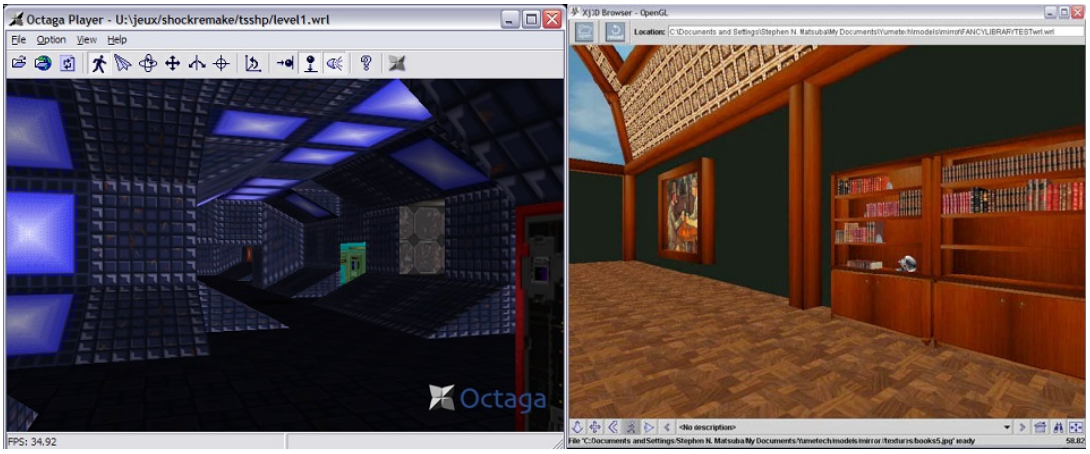
τον παγκόσμιο ιστό [51].

Κατά την εξαγωγή του ορίζεις έναν πλοηγό (navigator), ο οποίος θα κινηθεί μέσα στη γεωμετρία. Ρυθμίζεις παραμέτρους του, όπως ύψος, ταχύτητα βηματισμού, απόσταση στην οποία αντιλαμβάνεται την σύγκρουση κ.α. Ορίζεις ποιά ή ποιές θα είναι οι κάμερες προβολής και ορίζεις προαιρετικά φόντο, ήχους, αισθητήρες αφής/χρόνου/εγγύτητας κ.α. (σχήμα 3.5).

Επειδή πλέον το *VRML* έχει εκλείψει, έχει δώσει την θέση του στο *X3D* [υποενότητα 2.4.3]. Με έναν απλό μετατροπέα από *VRML* σε *X3D* μπορούμε να πάρουμε το αρχείο και να το ανοίξουμε με οποιονδήποτε περιηγητή ιστού που υποστηρίζει *X3D* όπως τους *Octaga Player*, *Xj3D* κ.α. (σχήμα 3.6).

3.4 Unity3D

Η *Unity3D* είναι μια ολοκληρωμένη μηχανή παιχνιδιών που παρέχει στο χρήστη πολλές δυνατότητες μέσω μια πλήρως διαμορφωμένης διεπαφής. Μέσα από τα εργαλεία της ο χρήστης έχει άμεση πρόσβαση στις λεπτομέρειες της γεωμετρίας, στην εισαγωγή στοιχείων όπως ήχους, animations, συστήματα σωματιδίων, φωτισμούς κ.α. καθώς και σε έναν συντάκτη κώδικα scripting, για την υλοποίηση της λειτουργίας του παιχνιδιού. Παρέχει εύκολη προσπέλαση των αντικειμένων του project, ιεραρχική δομή της γεωμετρίας και προεπισκόπηση



Σχήμα 3.6: Περιηγητές *X3D*
Αριστερά: *Octaga Player* **Δεξιά:** *Xj3D*

του παιχνιδιού [52].

Οι λόγοι τώρα, που μας οδήγησαν στην επιλογή της μηχανής *Unity3D* είναι οι παρακάτω:

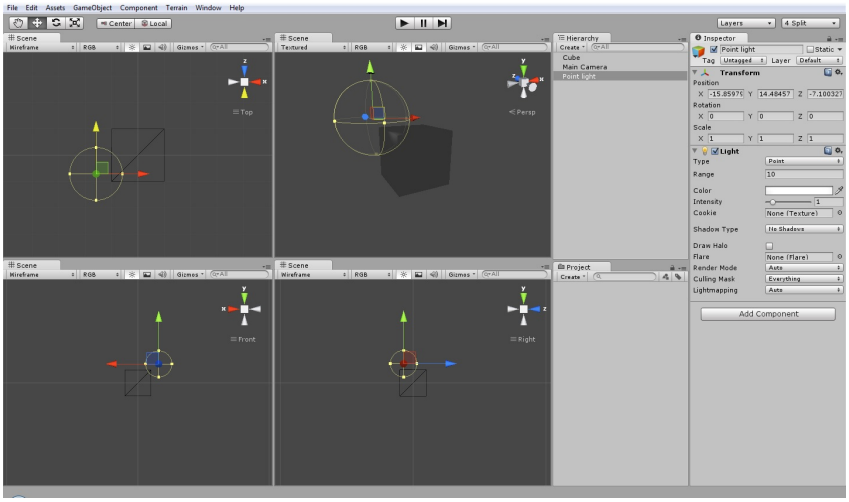
- Θεωρείται από τις πιο εύκολες μηχανές παιχνιδιών για αρχάριους, με μια αρκετά απλή και φιλική διεπαφή προς τον χρήστη.
- Είναι αρκετά γρήγορη μηχανή και χαμηλών απαιτήσεων συστήματος, σε σχέση με άλλες, ώστε να την κάνει προσιτή ακόμα και για υπολογιστές όχι τόσο μεγάλων δυνατοτήτων.
- Παρέχει δωρεάν έκδοση που οι περισσότερες μηχανές δεν έχουν.
- Στις περισσότερες μηχανές, η διεπαφή τους είναι βασισμένη ώστε να εξυπηρετούν περισσότερο ένα συγκεκριμένο είδος παιχνιδιού, ενώ η *Unity3D* από την άλλη είναι σχεδιασμένη για όλα τα είδη παιχνιδιών.
- Υποστηρίζει τις βασικές βιβλιοθήκες γραφικών *OpenGL* και *DirectX*.
- Υποστηρίζει σύνταξη κώδικα σε μορφή scripting σε γλώσσες *Javascript*, *C#* και *Boo*, με τις δύο πρώτες να είναι από τις πιο διαδεδομένες. Εν αντιθέσει, οι περισσότερες μηχανές γραφικών υποστηρίζουν scripting μόνο σε γλώσσες υλοποιημένες συγκεκριμένα για την μηχανή, με αποτέλεσμα να δαπανάται χρόνος για την εκμάθησή της.

- Κάνει εύκολη τη χρήση της μηχανής φυσικής της.
- Έχει άμεσο χτίσιμο του project σε οποιαδήποτε πλατφόρμα με το πάτημα ενός κουμπιού, εν συγκρίσει με άλλες μηχανές που χρειάζονται διαδικασίες για μετάβαση από μια πλατφόρμα σε μια άλλη.
- Υποστηρίζει άμεση και εύκολη εισαγωγή γεωμετρίας από πολλές μορφές αρχείων, ενώ σε άλλες μηχανές αυτό αποτελεί δύσκολη και χρονοβόρα διαδικασία.
- Τέλος η *Unity3D* έχει μια πλούσια κοινότητα στην οποία απατώνται άμεσα ερωτήσεις και απορίες χρηστών και επίσης έχει ένα ηλεκτρονικό κατάστημα με πληθώρα τρισδιάστατων αντικειμένων.

3.5 Δομή και Αρχιτεκτονική της Unity3D

Η δομή του *Unity3D* είναι ιεραρχική και είναι άμεσα αντιληπτή μέσω της διεπαφής του (σχήμα 3.7). Κάθε παιχνίδι αποθηκεύεται σε ένα project. Κάθε project αποτελείται από μια ή περισσότερες σκηνές (scenes), που μπορούμε να τις θεωρήσουμε σαν τα επίπεδα του παιχνιδιού (πίστες στις οποίες υπάρχει μια καινούργια γεωμετρία ή γραφικό περιβάλλον). Έχουμε τη δυνατότητα να επεξεργαζόμαστε μια σκηνή τη φορά. Κάθε σκηνή αποτελείται από τα αντικείμενα του παιχνιδιού (game objects), τα οποία μπορεί να ήδη σχεδιασμένα σε κάποιο λογισμικό και να εισαχθούν μέσω κάποιου αρχείου ή να δημιουργηθούν εντός του *Unity3D*. Από την άλλη κάθε αντικείμενο έχει τις υπομονάδες του (components) που χαρακτηρίζουν τη συμπεριφορά του. Ένα αντικείμενο περιέχει υποχρεωτικά την υπομονάδα του ονόματός του και αυτό του μετασχηματισμού του (transform). Από εκεί και πέρα μπορεί να περιέχει και άλλες υπομονάδες, όπως κάποιο αρχείο script, στοιχεία ήχου, κάμερα, ανάλογα με τις απαιτήσεις, οι βασικές εκ των οποίων περιγράφονται στις επόμενες υποενότητες. Τέλος κάθε υπομονάδα έχει διάφορες μεταβλητές (variables) που τους δίνουμε ανάλογα τις τιμές που θέλουμε (σχήμα 3.8) [53].

Η αρχιτεκτονική του *Unity3D* βασίζεται στις υπομονάδες του. Κάθε υπομονάδα ανήκει σε μια ευρύτερη κατηγορία ανάλογα με τη συμπεριφορά της. Για να μπορέσουμε να χρησιμοποιήσουμε με σωστό τρόπο τις υπομονάδες των αντικειμένων πρέπει να κατανοήσουμε αυτή την αρχιτεκτονική στην οποία είναι χτισμένη η συγκεκριμένη μηχανή παιχνιδιών (σχήμα 3.9).



Σχήμα 3.7: Διεπαφή του χρήστη στο *Unity3D*

Σχήμα 3.8: Ιεραρχική δομή του *Unity3D*

3.5.1 Μετασχηματισμός (Transform)

Η βασικότερη υπομονάδα των αντικείμενων είναι αυτή του μετασχηματισμού (Transform component) (σχήμα 3.10). Δίνει την δυνατότητα στον χρήστη να μετακινήσει το αντικείμενο στη σκηνή, να το περιστρέψει καθώς και να το κλιμακώσει έχοντας τρεις μεταβλητές για κάθε μια από τις παραπάνω ενέργειες.

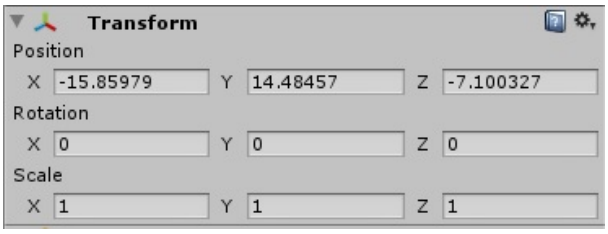
3.5.2 Ήχος (Audio)

Η ήχος στο *Unity3D* αποτελείται από δύο βασικές υπομονάδες (σχήμα 3.11):

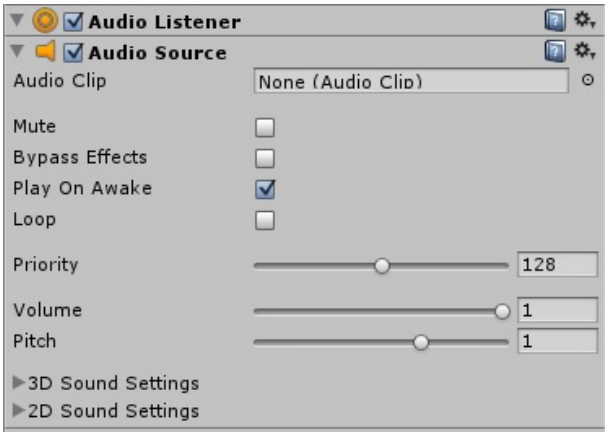
- **Audio Listener**: Ο ακροατής των ήχων, ο οποίος δέχεται τους ήχους όπως ακριβώς τους ακούει ο χρήστης κατά τη διάρκεια του παιχνιδιού. Πρέπει να υπάρχει πάντα μονάχα ένας ακροατής ενεργοποιημένος σε μια χρονική στιγμή κατά τη διάρκεια του παιχνιδιού. Συνήθως οι ακροατές τοποθετούνται στις κάμερες γιατί μέσα από εκεί βλέπει αλλά και ακούει ο χρήστης.

- **Audio Source**: Η πηγή του ήχου. Μπορεί να υπάρχουν μια αλλά και περισσότερες

Σχήμα 3.9: Αρχιτεκτονική και υπομονάδες του *Unity3D*



Σχήμα 3.10: Υπομονάδα μετασχηματισμού (Transform component)



Σχήμα 3.11: Υπομονάδα ήχου (Audio component)

πηγές ήχου μέσα στο παιχνίδι. Μπορείς μέσα από αυτή την υπομονάδα να ρυθμίσεις την ένταση του ήχου, την προτεραιότητά του σε σχέση με άλλες πηγές ήχου, αν είναι σε επανάληψη η αναπαραγωγή του ήχου, αν η αναπαραγωγή θα ξεκινήσει κατά την ενεργοποίηση της πηγής κ.α. Έχει επίσης ρυθμίσεις για δισδιάστατους, όσο και για τρισδιάστατους ήχους.

3.5.3 Γραφική Διεπαφή (GUI)

Η γραφική διεπαφή στο *Unity3D* περιέχει πολλά συστατικά και υπομονάδες που μας βοηθάνε στο να δημιουργήσουμε μια πλήρως λειτουργική και ευπαρουσίαστη γραφική διεπαφή προς τον χρήστη (σχήμα 3.12). Βασικά συστατικά της είναι:

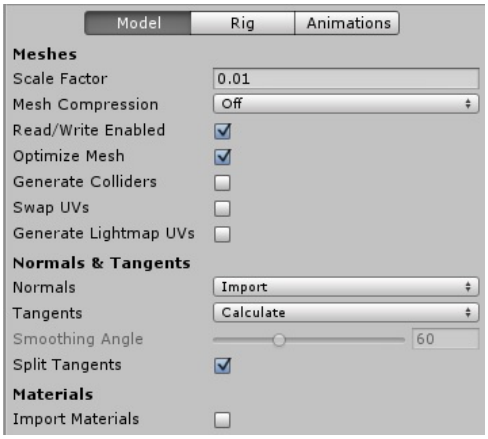
- **GUITexure**: Υφή η οποία χρησιμοποιείται εξ ολοκλήρου για γραφική διεπαφή.
- **GUIText**: Δισδιάστατο κείμενο για γραφικές διεπαφές.
- **GUIStyle**: Καθορίζει το ύφος που θα έχουν τα διάφορα στοιχεία της γραφικής διεπαφής μας. Γραμματοσειρές, χρώματα και υφές που τοποθετούνται σε παράθυρα, κουμπιά, ετικέτες, περιοχές κειμένου, κυλιόμενες μπάρες κ.α.
- **GUISkin**: Ένα ολοκληρωμένο σύνολο από GUIStyles.

Σχήμα 3.12: Ένα GUISkin με τα επιμέρους του GUIStyles

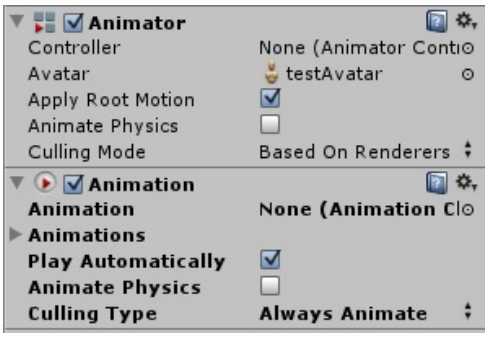
3.5.4 Εισαγωγή Γεωμετρίας

Η γεωμετρία ενός παιχνιδιού μπορεί να σχεδιαστεί σε οποιοδήποτε λογισμικό σχεδίασης γραφικών και να εισαχθεί στο *Unity3D* (σχήμα 3.13). Η εισαγωγή είναι εύκολη αφού το *Unity3D* υποστηρίζει τους βασικούς τύπους αρχείων που εξάγουν όλα τα γνωστά λογισμικά όπως τα *Autodeks 3ds Max*, *Autodesk Maya*, *Blender*, *Cinema4D*, *Modo*, *Lightwave*, *Cheetah3D* κ.α. Κατά την εισαγωγή της γεωμετρίας μπορείς να επιλέξεις τον συντελεστή κλιμάκωσης, τυχόν συμπίεση των δικτύων γεωμετρίας, παραγωγή ανιχνευτών σύγκρουσης στα αντικείμενα, εισαγωγή ή παραγωγή κάθετων και εφαπτόμενων διανυσμάτων κ.α.

Μετά την εισαγωγή προβάλλονται όλα τα αντικείμενα στο project μας, όπως τα έχουμε δημιουργήσει κατά τη σχεδίαση, καθώς και η ιεραρχία που τους έχουμε δώσει. Τα αντικείμενα που εισήχθησαν έχουν προεπιλεγμένες τις υπομονάδες του ονόματος, του μετασχηματισμού (Transform component) και αυτές του Mesh Filter και Mesh Renderer. Το Mesh Filter είναι υπεύθυνο για να σχηματίσει το δίκτυο της γεωμετρίας για το κάθε αντικείμενο και να το περάσει στο Mesh Renderer. Το Mesh Renderer, με τη σειρά του, είναι υπεύθυνο να προβάλλει το αντικείμενο στο χώρο ανάλογα με τις τιμές στην υπομονάδα του μετασχηματισμού και να δημιουργήσει τις αλληλεπιδράσεις του αντικειμένου με το φως, τις σκιάσεις που θα έχει και τις υφές του.



Σχήμα 3.13: Εισαγωγή γεωμετρίας στο *Unity3D*

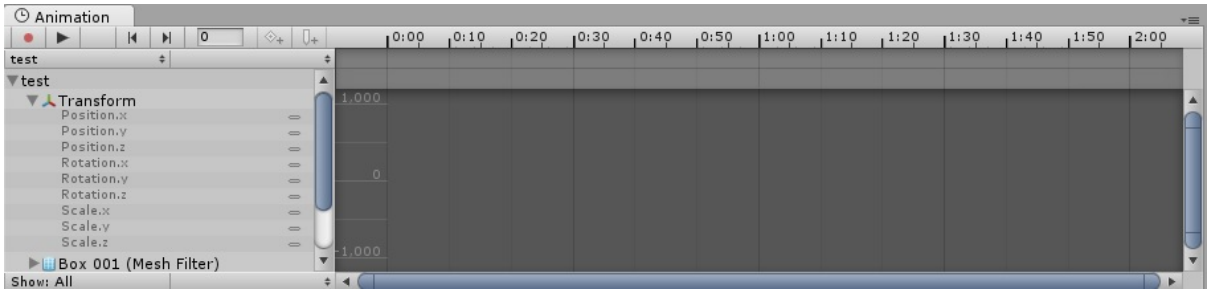


Σχήμα 3.14: Υπομονάδες για animations

3.5.5 Κινούμενα Σχέδια (Animations)

Τα κινούμενα σχέδια μπορούν ή να εισαχθούν στο *Unity3D* είτε να κατασκευαστούν εντός του. Κατά την εισαγωγή μιας γεωμετρίας σε περίπτωση που υπάρχουν animations, τότε αυτά αναγνωρίζονται αυτόματα και υπάρχει η δυνατότητα να εισαχθούν αν αυτό επιλεγεί. Όταν εισαχθούν τοποθετείται αυτόματα στο κατάλληλο αντικείμενο η υπομονάδα του Animation. Μια άλλη υπομονάδα που υπάρχει για animations, είναι αυτή του Animator. Αυτή η υπομονάδα διαχειρίζεται animations στα αντικείμενα-χαρακτήρες που εφαρμόζουν καμιά κίνηση (σχήμα 3.14).

Για την υλοποίηση ενός animation εντός του *Unity3D*, μέσα από το παράθυρο των animations, ορίζουμε σε ποιο αντικείμενο θα υλοποιηθεί το animation και για ποιές τιμές των υπομονάδων του (σχήμα 3.15). Η υλοποίηση γίνεται με τον γνωστό τρόπο, δηλαδή της



Σχήμα 3.15: Λωρίδα χρόνου animations στο *Unity3D*

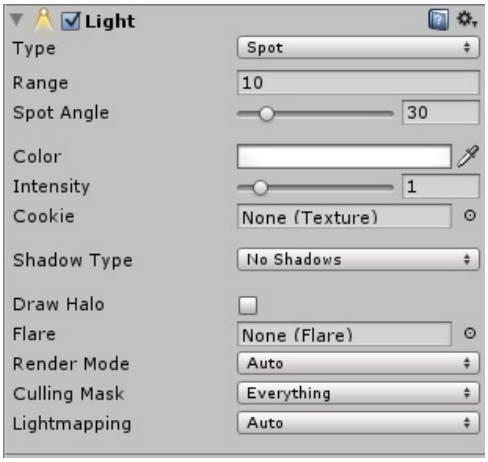
λωρίδας χρόνου και των κλειδιών καρέ.

3.5.6 Φωτισμός (Lighting)

Ο φωτισμός είναι αυτός που δίνει χρώμα και ύφος στο τρισδιάστατο περιβάλλον μας. Το *Unity3D* έχει πέντε τύπους φωτισμών:

- **Directional light**: Τοποθετείται απείρως μακριά απο την σκηνή και ο φωτισμός του επηρεάζει τα πάντα-σαν τον ήλιο.
- **Point light**: Εκπέμπει φως από ένα σημείο προς όλες τις κατευθύνσεις μέσα στα όρια μιας σφαίρας-σαν μια λάμπα.
- **Spot light**: Εκπέμπει φως από ένα σημείο προς μια κατεύθυνση και φωτίζει αντικείμενα που βρίσκονται στο εύρος εκπομπής ενός κώνου-σαν τα φώτα πορείας ενός αμαξιού.
- **Area light**: Χρησιμοποιείται μόνο για χαρτογράφηση φωτός και είναι φως που εκπέμπεται από ένα σημείο προς όλες τις κατευθύνσεις στα πλαίσια ενός παραλληλεπίπεδου.
- **Ambient light**: Το προεπιλεγμένο καθολικό φως, που φωτίζει όλα τα αντικείμενα με τον ίδιο τρόπο.

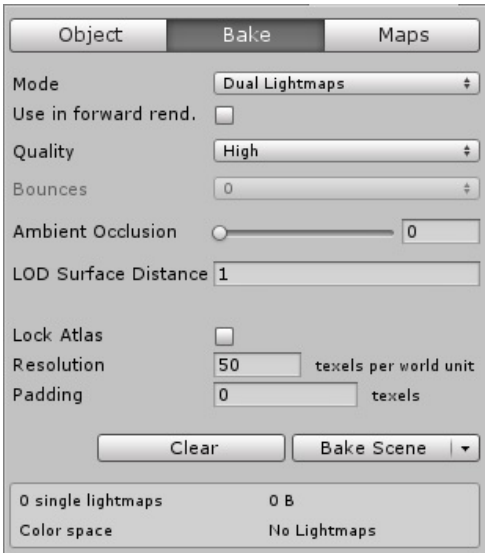
Κατά τη δημιουργία ενός φωτισμού δημιουργείται το αντικείμενο του φωτός με την κατάλληλη υπομονάδα (Light component) . Σε αυτή την υπομονάδα ρυθμίζονται παράμετροι όπως η ένταση του φωτός, η γωνία του κώνου στο spot light, η εμβέλεια, το χρώμα, ο τύπος των σκιάσεων κ.α. (σχήμα 3.16).



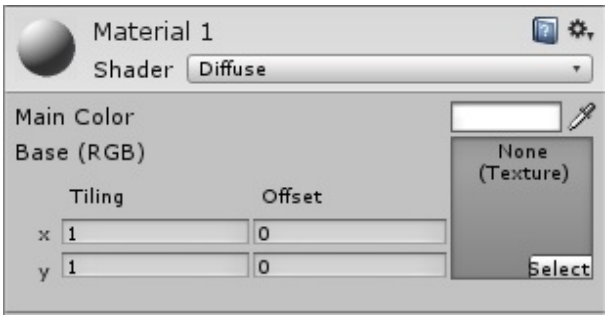
Σχήμα 3.16: Υπομονάδα φωτισμού (Light component)

Μια συχνή διαδικασία στο πεδίο του φωτισμού είναι αυτή της χαρτογράφησης φωτός (lightmapping). Οι φωτισμοί όταν τοποθετούνται αλληλεπιδρούν με τα αντικείμενα, τα φωτίζουν κατάλληλα και παράγουν τις σκιάσεις τους σε πραγματικό χρόνο. Με τον όρο πραγματικό χρόνο, εννοείται ότι ο υπολογισμός των φωτισμών και των σκιάσεων των αντικειμένων, γίνεται σε κάθε καρέ του που δημιουργείται κατά την εκτέλεση του παιχνιδιού. Σε περίπτωση που τα αντικείμενα και οι φωτισμοί είναι αρκετοί, αυτή είναι μια αρκετά χρονοβόρα διαδικασία που έχει ως αποτέλεσμα την αργή εκτέλεση του παιχνιδιού και την χρήση μεγάλου μέρους των υπολογιστικών πόρων. Τις περισσότερες φορές όμως ο φωτισμός δεν αλλάζει κατά τη διάρκεια του παιχνιδιού και τα περισσότερα αντικείμενα είναι στατικά. Έτσι μέσα από τη διαδικασία του lightmapping προϋπολογίζεται ο φωτισμός και οι σκιάσεις των αντικειμένων πριν την εκτέλεση του παιχνιδιού και αποθηκεύονται σε χάρτες οι οποίοι προβάλλονται πάνω στις υφές των αντικειμένων κατά την εκτέλεσή του. Αυτό έχει ως αποτέλεσμα να χρειάζεται να γίνουν υπολογισμοί μόνο για αντικείμενα τα οποία κινούνται-όπως οι χαρακτήρες-τα οποία είναι πολύ λιγότερα από τα στατικά και να είναι η εκτέλεση του παιχνιδιού πολύ πιο γρήγορη.

Ο αλγόριθμος που χρησιμοποιεί το *Unity3D* για χαρτογράφηση φωτός είναι ο *Beast*, της εταιρείας *Illuminate Labs*. Ο αλγόριθμος αυτός είναι πολύ γρήγορος και δίνει την δυνατότητα στον χρήστη να χρησιμοποιεί το *Unity3D* παράλληλα με τη διαδικασία της δημιουργίας των χαρτών, που ονομάζεται "ψήσιμο" (baking). Για τη διαδικασία του lightmapping ρυθμίζονται παράμετροι όπως ο τύπος της χαρτογράφησης, η ποιότητα και η ανάλυση των χαρτών, η απόφραξη περιβάλλοντος (ambient occlusion) κ.α. (σχήμα 3.17).



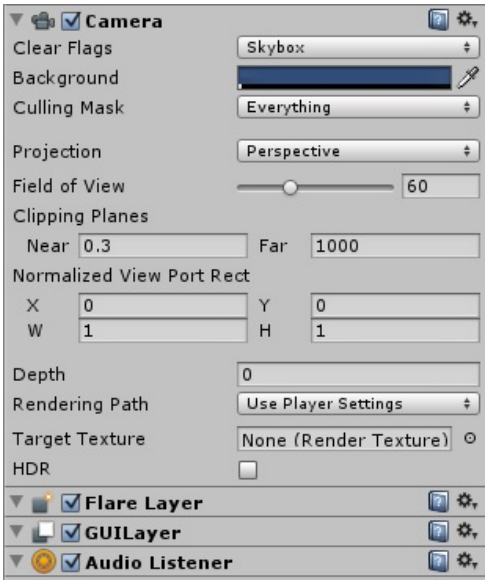
Σχήμα 3.17: Χαρτογράφηση φωτός (lightmapping) στο *Unity3D*



Σχήμα 3.18: Υπομονάδα υφής (Material component)

3.5.7 Υφές (Textures)

Οι υφές των αντικειμένων μπορούν είτε να δημιουργηθούν στο *Unity3D* είτε να εισαχθούν παράλληλα με την εισαγωγή της γεωμετρίας, σε περίπτωση που έχουν φτιαχτεί μαζί με τη γεωμετρία στο λογισμικός σχεδίασης. Το αντικείμενο έχει κάποια υφή μέσω της αντίστοιχης υπομονάδας (Material component). Στην υπομονάδα αυτή μπορούν να ρυθμιστούν παράμετροι όπως η εικόνα ή το χρώμα της υφής, ο σκιαστής που θα εφαρμοστεί πάνω της καθώς και η επανάληψη της υφής πάνω στην επιφάνεια του αντικειμένου σε μορφή πλακιδίων (σχήμα 3.18).



Σχήμα 3.19: Υπομονάδες κάμερας

3.5.8 Κάμερες (Cameras)

Όταν δημιουργούμε μια νέα σκηνή, εισάγεται αυτόματα σαν αντικείμενο μια κάμερα με το όνομα Main Camera. Αυτή είναι η κύρια κάμερα του παιχνιδιού. Μπορούμε να την διαγράψουμε ή να δημιουργήσουμε και άλλες κάμερες. Κατά τη διάρκεια του παιχνιδιού πρέπει μονάχα μια κάμερα να είναι ενεργοποιημένη. Έτσι όταν ενεργοποιούμε μια κάμερα πρέπει να απενεργοποιούμε αυτή που ήταν προηγουμένως ενεργή. Σε περίπτωση που αυτό δεν γίνεται η μηχανή εμφανίζει μήνυμα σφάλματος. Σε περίπτωση που δεν υπάρχει καμία κάμερα ενεργοποιημένη στο παιχνίδι, δεν μπορεί να προβληθεί τίποτα στην οθόνη μας καθώς και δεν μπορεί να αναπαραχθεί κανένας ήχος.

Κατά τη δημιουργία ενός αντικειμένου κάμερας εισάγονται αυτόματα οι υπομονάδες της κάμερας (Camera component), Flare Layer, GUILayer και Audio Listener (σχήμα 3.19). Η υπομονάδα της κάμερας είναι αυτή που περιέχει όλες τις ρυθμίσεις για την κάμερα όπως τον τύπο του φόντου, τον τύπο προβολής, το οπτικό πεδίο, το βάθος κ.α. Το Flayer Layer χρησιμοποιείται για να προβάλει την αναλαμπή του φυσικού φωτός στο φακό της κάμερα, όπως συμβαίνει στις πραγματικές κάμερες. Το GUILayer χρησιμοποιείται για να μπορεί η κάμερα να προβάλλει τα δισδιάστατα αντικείμενα της γραφικής διεπαφής και τέλος το Audio Listener είναι ο ακροατής των ήχων όπως περιγράφηκε παραπάνω [υποενότητα 3.5.2].

3.5.9 Σκιαστές (Shaders)

Κατά την δημιουργία μιας υφής, επιλέγεται και ο σκιαστής που θα εφαρμοστεί πάνω της. Ο σκιαστής είναι αυτός που θα ρυθμίσει την επίδραση του φωτός πάνω στην υφή. Μπορεί να χρησιμοποιηθεί κάποιος υπάρχων σκιαστής από αυτούς που έχει το *Unity3D*, ή να δημιουργηθεί κάποιος με σύνταξη μέσω ειδικού scripting. Τα βασικά είδη σκιαστών που έχει το *Unity3D* είναι:

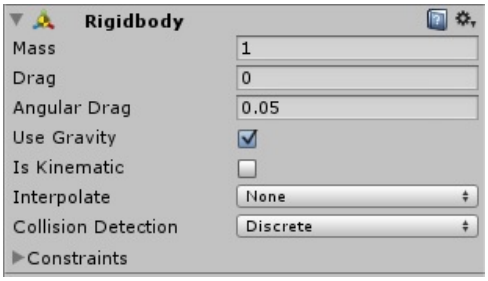
- **Diffuse**: Για διάχυση του φωτός πάνω στην υφή.
- **Specular**: Για κατοπτρικές υφές.
- **Bumped**: Για να δημιουργήσει ανάγλυφη υφή.
- **Parallax**: Για να δημιουργήσει ανάγλυφη υφή καθώς και να δώσει βάθος στην γεωμετρία στην οποία εφαρμόζεται η υφή.
- **Reflective**: Για να δημιουργήσει αντανάκλαση στην υφή όπως σε μέταλλα κ.α.
- **Gloss**: Για στιλπνότητα στην υφή.
- **Self-Illumin**: Για αυτόφωτες υφές.
- **Transparent**: Για διαφανείς υφές.

3.5.10 Φυσική (Physics)

Η μηχανής φυσικής που χρησιμοποιεί το *Unity3D* είναι η *PhysX* που περιγράφηκε παραπάνω [υποενότητα 2.11.4]. Οι βασικές κατηγορίες υπομονάδων που χρησιμοποιεί είναι του άκαμπτου σώματος, τον ανιχνευτών σύγκρουσης, των αρθρώσεων και των υφασμάτων.

Η υπομονάδα της άκαμπτου σώματος (Rigidbody component), είναι η υπομονάδα που δίνει τα βασικά συστατικά της νευτώνειας φυσικής σε ένα αντικείμενο (σχήμα 3.20). Ρυθμίζονται παράμετροι όπως η μάζα του σώματος, αν θα επιδρά η βαρύτητα πάνω του, αν είναι κινηματικό, αν θα ανιχνεύει συγκρούσεις σε διακριτή ή συνεχή μορφή κ.α.

Οι ανιχνευτές σύγκρουσης είναι πλέγματα που τοποθετούνται γύρω από τα αντικείμενα και χρησιμοποιούνται για να εντοπίζουν τυχόν συγκρούσεις μεταξύ των αντικειμένων. Χωρίζονται ανάλογα με το σχήμα τους σε κύβου (Box Collider), σφαίρας (Sphere Collider),



Σχήμα 3.20: Υπομονάδα άκαμπτου σώματος (Rigidbody component)

κάψουλας (Capsule Collider), δικτύου (Mesh Collider), τροχού (Wheel Collider) και εδάφους (Terrain Collider).

Οι υπομονάδες των αρθρώσεων (Joint component) χρησιμοποιούνται για να συνδέσει την φυσική ενός αντικειμένου με ένος άλλου, δηλαδή όταν υπακούει το ένα αντικείμενο στους νόμους φυσικής να υπακούει και το άλλο ανάλογα. Τέλος οι υπομονάδες των υφασμάτων (Cloth components) χρησιμοποιούνται για να προσομοιώσουν την φυσική που έχουν τα υφάσματα και τα ρούχα.

3.5.11 Προκατασκευές (Prefabs)

Ένα προκατασκευασμένο αντικείμενο (prefab), είναι ένα επαναχρησιμοποιήσιμο αντικείμενο που αποθηκεύεται στο project μας. Μπορεί να χρησιμοποιηθεί σε πολλαπλές σκηνές, και κάθε φορά που το εισάγουμε σε μια σκηνή, στην ουσία εισάγουμε μια αναφορά σε αυτό (instance). Σε περίπτωση που αλλάξει κάτι στο prefab αυτομάτως αλλάζουν και όλες οι αναφορές του. Τα αντικείμενα που είναι αναφορές σε κάποιο προκατασκευασμένο αντικείμενο, χρωματίζονται με μπλε χρώμα στην ιεραρχία των αντικειμένων της σκηνής, ενώ οι αναφορές των προκατασκευασμένων αντικειμένων που έχουν διαγραφεί, χρωματίζονται με κόκκινο. Αυτό υπάρχει ώστε ο σχεδιαστής να ξεχωρίζει πότε ένα αντικείμενο είναι αναφορά ενός προκατασκευασμένου αντικειμένου και πότε όχι.

3.5.12 Ρυθμίσεις

Το *Unity3D* παρέχει κάποιες ρυθμίσεις που σχετίζονται με λεπτομέρειες πάνω στο παιχνίδι και εφαρμόζονται σε ολόκληρο το project. Βασικές είναι οι παρακάτω:

- **Audio**: Αφορά ρυθμίσεις ήχου με παραμέτρους όπως η έντασή του, η κλίμακα απόσβεσής του, η ταχύτητά του, ο συντελεστής του Doppler και ο αριθμός των καναλιών του (μονοφωνικός/στερεοφωνικός κ.α.).

- **Input**: Είναι οι ρυθμίσεις που αντιστοιχίζουν κάθε είσοδο του παιχνιδιού στο αντίστοιχο πλήκτρο του πληκτρολογίου ή του χειριστηρίου.

- **Physics**: Ρυθμίζονται παράμετροι της φυσικής όπως η επιτάχυνση της βαρύτητας, ο άξονας που εφαρμόζεται η βαρύτητα και παράμετροι που αφορούν στις συγκρούσεις.

- **Player**: Αφορά τις ρυθμίσεις για το χτίσιμο του παιχνιδιού. Εικονίδια παιχνιδιού, όνομα, κέρσορας, ανάλυση οθόνης, προβολή σε κατάσταση πλήρους οθόνης και διάφορες άλλες ρυθμίσεις ανάλογα με την πλατφόρμα για την οποία υλοποιείται το παιχνίδι.

- **Render**: Ρυθμίσεις προβολής όπως ομίχλη με παραμέτρους εύρος, πυκνότητα και τύπο αυτής, χρώμα καθολικού φωτός, υφή ουρανού, ένταση αναλαμπής του φωτός στην κάμερα κ.α.

- **Quality**: Δημιουργείς επίπεδα ποιότητας των γραφικών του παιχνιδιού (π.χ. κακή, καλή, εξαιρετική κ.α.) και τα αντιστοιχείς σε όποια πλατφόρμα θέλεις. Ρυθμίζεις παραμέτρους όπως ανάλυση υφών, ανισότροπες υφές, αντιταύτιση (anti-aliasing), ανάλυση σκιάσεων, απόσταση σκιάσεων κ.α.

- **Time**: Ρυθμίζει τον συντελεστή του χρόνου και κάποιες ετικέτες του που χρησιμοποιούνται από συναρτήσεις του κώδικα scripting.

3.6 Scripting στο Unity3D

Μια γλώσσα δέσμης ενεργειών (scripting language), είναι μια γλώσσα προγραμματισμού που υποστηρίζει την σύνταξη προγραμμάτων για ειδικά περιβάλλοντα και αυτοματοποιεί την εκτέλεση διαδικασιών [54]. Το scripting είναι βασικό συστατικό σε όλα τα παιχνίδια· ακόμα και τα πιο απλά χρειάζονται scripts για να ανταποκριθούν στις εισόδους των χρηστών και να μεριμνήσουν για την εκτέλεση γεγονότων στο χρόνο που πρέπει. Πέρα από αυτό χρησιμοποιούνται για την δημιουργία γραφικών εφφέ, για τον έλεγχο της φυσικής συμπεριφοράς των αντικειμένων ή ακόμα την υλοποίηση της τεχνητής νοημοσύνης των χαρακτήρων.

Το *Unity3D* προσφέρει έναν πλήρη συντάκτη κώδικα scripting, τον *MonoDevelop*. Ο συντάκτη αυτός περιέχει έναν ελεγκτή λεκτικών, συντακτικών και σημασιολογικών σφαλμάτων και έναν μεταφραστή για κάθε γλώσσα που υποστηρίζει. Τα scripts που δημιουργούνται προσθέτονται σαν υπομονάδες στα αντικείμενα που θέλουμε να εφαρμοστούν. Μέσα από τα scripts υλοποιούμε την λειτουργικότητα του παιχνιδιού και μπορούμε να επιδράσουμε σε οποιαδήποτε μεταβλητή, οποιασδήποτε υπομονάδας, οποιουδήποτε αντικειμένου που βρίσκεται στην σκηνή μας. Μέσα από αυτή τη διαδικασία δημιουργούμε γεγονότα και αλλαγές στο περιβάλλον μας [55].

3.6.1 Γλώσσες Προγραμματισμού Scripting του Unity3D

Οι γλώσσες προγραμματισμού που υποστηρίζει το scripting της *Unity3D* είναι τρεις: *Javascript*, *C#* και *Boo*, η οποία τελευταία είναι μια παραλλαγή της *Python* που δημιούργησαν οι σχεδιαστές του *Unity3D*. Και οι τρεις γλώσσες είναι αντικειμενοστραφείς. Υλοποιούν τις έννοιες της κληρονομικότητας, του πολυμορφισμού και της ιεραρχίας των κλάσεων. Παρέχουν την ευκολία στον προγραμματιστή της ανυπαρξίας μη ασφαλών δεικτών και του αυτόματου συλλέκτη σκουπιδιών (garbage collector).

Αυτό που κάνει επίσης τις γλώσσες συγγενείς μεταξύ τους-με αποτέλεσμα να μην υπάρχουν προβλήματα συμβατότητας μεταξύ των scripts-είναι η ίδια σύνταξη τους. Υποστηρίζουν όλους τους γνωστούς τύπους μεταβλητών (integer, float, double, short, byte, string, character, boolean, enumarator κ.α.) και όλες τις γνωστές δομές εντολών: ελέγχου (if, else, switch, case), επανάληψης (do, for, foreach, in, while), διακλάδωσης (break, continue, default, goto, return, yield) και εξαιρέσεων (throw, try-catch, try-finally, try-catch-finally). Υποστηρίζει την εμβέλεια τον μεταβλητών (public, private, protected), την υλοποίηση κλάσεων, διεπαφών καθώς και συναρτήσεων.

3.6.2 Βασικές Συναρτήσεις του Unity3D

Κατά την σύνταξη του κώδικα μπορούν να υλοποιηθούν διάφορες κλάσεις που θα έχουν διάφορες λειτουργίες. Σε περίπτωση που θα δημιουργηθεί κάποια κλάση η οποία θα καλεί ή θα υλοποιεί συναρτήσεις του *Unity3D*, ή θα έχει άμεση επίδραση και θα διαχειρίζεται αντικείμενα της σκηνής, πρέπει αυτή η κλάση να οριστεί ως υποκλάση της *MonoBehavior*, ώστε να κληρονομήσει όλες τις απαραίτητες συναρτήσεις και μεταβλητές.

Η υλοποίηση των συναρτήσεων της *Monobehavior* βασίζεται στο καρέ. Καρέ (frame) είναι η εικόνα που σχεδιάζεται και προβάλλεται στην οθόνη μας κατά τη διάρκεια του παιχνιδιού. Συχνότητα καρέ (frame rate) είναι το πλήθος των καρέ που σχεδιάζονται στην μονάδα του χρόνου για να μας δώσουν την αίσθηση της ρεαλιστικής κινούμενης εικόνας. Παρ' όλα αυτά στην υλοποίηση των παιχνιδιών είναι κάτι πέρα από αυτό. Συχνότητα καρέ είναι η ταχύτητα με την οποία διεκπεραιώνονται όλες οι διεργασίες που χρειάζονται για την σχεδίαση ενός καρέ.

Οι βασικότερες συναρτήσεις του *Unity3D* είναι:

- **Awake()**: Καλείται κατά την φόρτωση της υπομονάδας του script όταν φορτώνονται στο παιχνίδι τα αντικείμενα και οι υπομονάδες του. Χρησιμοποιείται για αρχικοποίηση μεταβλητών και εκτελείται μια και μοναδική φορά κατά τη διάρκεια του παιχνιδιού.

- **Start()**: Καλείται μόλις έχει ενεργοποιηθεί η υπομονάδα του script. Παραπλήσια της *awake()* μόνο που εκτελείται μετά από αυτήν και εκτελείται κάθε φορά που ενεργοποιείται το script.

- **Update()**: Η βασικότερη συνάρτηση της κλάσης *Monobehavior* που εκτελείται για κάθε καρέ του παιχνιδιού.

- **FixedUpdate()**: Η λειτουργία της είναι ίδια με της *Update()*, με την μόνη διαφορά ότι εκτελείται για κάθε καρέ στην συχνότητα καρέ που έχουμε ορίσει εμείς.

- **OnGUI()**: Έχει ίδια λειτουργία με την *Update()*, μόνο που χρησιμοποιείται για διαχείριση γεγονότων που έχουν να κάνουν με γραφικές διεπαφές.

- **OnEnable(), OnDisable(), OnDestroy()**: Εκτελούνται όταν ενεργοποιείται, απενεργοποιείται και καταστρέφεται ένα αντικείμενο αντίστοιχα.

- **OnMouseDown()**: Εκτελείται όταν ο κέρσορας του ποντικιού πατήσει ένας ανιχνευτή σύγκρουσης ή ένα αντικείμενο γραφικής διεπαφής. Αντίστοιχης φιλοσοφίας είναι και οι *OnMouseEnter()*, *OnMouseOver()*, *OnMouseExit()*, *OnMouseDown()*, *OnMouseUp()*, *OnMouseUpAsButton()*, *OnMouseDrag()*.

- **OnTriggerEnter()**: Εκτελείται όταν ένας ανιχνευτής σύγκρουσης (Collider component) έρθει σε επαφή με έναν άλλον και ο δεύτερος έχει ενεργοποιημένη τη λει-

τουργία του εναύσματος (isTrigger) ώστε να διαχειρίζεται τα γεγονότα. Αντίστοιχης φιλοσοφίας και οι *OnTriggerExit()*, *OnTriggerStay()*.

- **OnCollisionEnter()**: Εκτελείται όταν δύο αντικείμενα που έχουν ως υπομονάδα τον ανιχνευτή σύγκρουσης (Collider component) ή του άκαμπτου σώματος (Rigidbody component) έρθουν σε επαφή. Αντίστοιχης φιλοσοφίας και οι *OnCollisionExit()*, *OnCollisionStay()*.

3.6.3 Συναρτήσεις Γραφικής Διεπαφής

Όπως αναφέρθηκε παραπάνω [υποενότητα 3.6.2], η σχεδίαση των γραφικών διεπαφής, εκτελείται για κάθε καρέ που σχεδιάζεται μέσα από την συνάρτηση *OnGUI()*. Οι δυο βασικές κλάσεις σχεδίασης γραφικών είναι οι *GUI* και *GUILayout*. Σε κάθε αντικείμενο γραφικής διεπαφής που σχεδιάζεται, περνιέται σαν όρισμα το μέγεθος του αντικειμένου (μετριέται σε πλήθος εικονοστοιχείων), η θέση του στην οθόνη και διάφοροι τυχόν παράμετροι που χρειάζεται το κάθε αντικείμενο. Σε περίπτωση που το αντικείμενο που δημιουργείται ανήκει στην κλάση *GUILayout*, τότε τοποθετείται αυτόματα στην οθόνη βάσει κάποιου σχεδίου (layout), και δεν χρειάζεται να οριστεί η θέση του. Οι συναρτήσεις που χρησιμοποιούνται συχνότερα είναι:

- **Box()**: Για την δημιουργία ενός γραφικού κουτιού.
- **Button()**: Για την δημιουργία ενός γραφικού κουμπιού.
- **DrawTexture()**: Για την σχεδίαση δισδιάστατης υφής.
- **HorizontalScrollbar()**: Για την δημιουργία οριζόντιας γραμμής κύλισης. Αντίστοιχα και η κάθετη με την *VerticalScrollBar()*.
- **HorizontalSlider()**: Για την δημιουργία οριζόντιου ολισθητή. Αντίστοιχα και ο κάθετος με την *VerticalSlider()*.
- **Label()**: Για την σχεδίαση δισδιάστατης ετικέτας.
- **TextArea()**: Για την δημιουργία πεδίου κειμένου πολλών γραμμών.
- **TextField()**: Για την δημιουργία πεδίου κειμένου μιας γραμμής.
- **Toggle()**: Για την δημιουργία κουμπιών εναλλαγής (toggle buttons).

- **Window()**: Για την σχεδίαση παραθύρου, με σκοπό να τοποθετηθούν αντικείμενα γραφικής διεπαφής.

Οι παραπάνω συναρτήσεις, όπως και πολλές άλλες που υπάρχουν ανάλογα με την εκάστοτε λειτουργία που θέλουμε να πραγματοποιήσουμε, χρησιμεύον στην σχεδίαση γραφικών μενού (στην έναρξη ή κατά την παύση του παιχνιδιού) και γραφικών διεπαφών του χρήστη κατά την εκτέλεση του παιχνιδιού.

3.6.4 Συναρτήσεις Προγραμματιστή

Πολλές φορές οι συναρτήσεις που παρέχει το *Unity3D*, τόσο σε επίπεδο λειτουργιών όσο και σε επίπεδο γραφικών, δεν επαρκούν για την λειτουργικότητα όλων των παιχνιδιών. Τότε ο σχεδιαστής είναι αναγκασμένος να υλοποιήσει δικές τους συναρτήσεις. Η κλήση τους παρ΄ όλα αυτά πρέπει να γίνεται με προσοχή διότι, αν και δεν είναι άμεσα αντιληπτό από τον προγραμματιστή, οι συναρτήσεις που εκτελούνται σε κάθε καρέ (*Update()*, *FixedUpdate()*, *OnGUI()*) είναι πολυνηματικές. Έτσι μπορεί να εμφανιστούν σφάλματα που θα προκύπτουν από την ανεπιθύμητη συνεχόμενη εκτέλεση μιας συνάρτησης, κάτι για το οποίο η αποσφαλμάτωση είναι αρκετά δύσκολη.

Η σύνταξη των συναρτήσεων γίνεται με τον κλασσικό τρόπο και η κλήση τους γίνεται κανονικά. Πολύ συχνό φαινόμενο είναι να θέλουμε να περιμένουμε κάποια λειτουργία (συνήθως ανάδραση από τον χρήστη), πριν συνεχίσουμε την εκτέλεση του κώδικα. Αυτό γίνεται μέσω της εντολής *yield*. Επειδή όπως είπαμε η εκτέλεση των συναρτήσεων είναι πολυνηματική για κάθε καρέ, η αναμονή του κώδικα θα πρέπει να εκτελεστεί εντός μια συνάρτησης, η οποία υποχρεωτικά πρέπει να έχει τύπο επιστροφής *IEnumerator*. Σε αυτή την περίπτωση όμως, για να καλεστούν τέτοιες συναρτήσεις, πρέπει να γίνει χρήση της συνάρτησης *StartCoroutine(FuncName(FunAttrs))*.

Τέλος στην δική μας περίπτωση υλοποιήθηκε μια πληθώρα συναρτήσεων για την διαχείριση του παίχτη, την σχεδίαση της γραφικής διεπαφής, το χειρισμό των ήχων, της φυσικής, της κάμερας καθώς και για την υλοποίηση της τεχνητής νοημοσύνης. Εκετενής περιγραφή αυτών των συναρτήσεων γίνεται παρακάτω [κεφάλαιο 5].

Κεφάλαιο 4

Σχεδιασμός Παιχνιδιού και Γραφικής Διεπαφής Χρήστη

4.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο [κεφάλαιο 3], κάναμε μια εκτενή αναφορά των λογισμικών που χρησιμοποιήσαμε και των βασικών τους εργαλείων. Σε αυτό το κεφάλαιο θα περιγράψουμε αναλυτικά πως διαχειριστήκαμε τα εργαλεία αυτά για τη σχεδίαση, της γεωμετρίας μας και της γραφικής διεπαφής του χρήστη.

4.2 Σενάριο

Δυστυχώς, ήρθατε αργά για να συναντήσετε τον κύριο Black. Μόλις δολοφονήθηκε. Το 29χρονο θύμα ήταν ο ανιψιός του τελευταίου Κόμη Black, του προηγούμενου ιδιοκτήτη της έπαυλης Tudor. Ορφανός από μικρή ηλικία, ο νεαρός κύριος Black ήρθε στην έπαυλη Tudor, για να τον μεγαλώσει ο θείος του. Ο κύριος Black κάποια στιγμή έφυγε για να σπουδάσει και επέστρεψε στην έπαυλη μετά το θάνατο του θείου του. Ο Κόμης, ο οποίος δεν παντρεύτηκε ποτέ, εμπιστεύτηκε την περιουσία του σε διαχειριστές, για να την κληρονομήσει ο ανηψιός του όταν θα γινόταν 30 χρονών.

Σύντομα, ο κύριος Black ανακάλυψε τη σταθερή χορήγηση μέρους της περιουσίας του θείου του σε ένα άγνωστο πρόσωπο, για άγνωστους λόγους. Μήπως ο Κόμης ήταν θύμα εκβιασμού ή υποστήριζε κάποιον στα κρυφά; Ο κύριος Black αποφάσισε να ψάξει τις ύπο-

πτες συναλλαγές εις βάθος. Έτσι κάλεσε τους πρώην συνεργάτες του Κόμη στην έπαυλη για το σαββατοκύριακο. Εκείνο το σαββατοκύριακο ήταν που ο κύριος Black δολοφονήθηκε.

Ο χρήστης βάσει των κανόνων που αναφέρθηκαν σε προηγούμενο κεφάλαιο [ενότητα 1.2], διαχειρίζεται ένα από τους παίχτες-υπόπτους μέσα στην έπαυλη. Σκοπός του να διασχίσει τα δωμάτια, να κάνει υποθέσεις με στόχο να βρει ποιός πραγματικά δολοφόνησε τον κύριο Black, και τι κρυβόταν πίσω από αυτό το στυγερό έγκλημα. Ο παίχτης πρέπει να κινηθεί έξυπνα ώστε να εμποδίσει τους άλλους παίχτες και να βρεθεί αυτός πρώτος στη λύση.

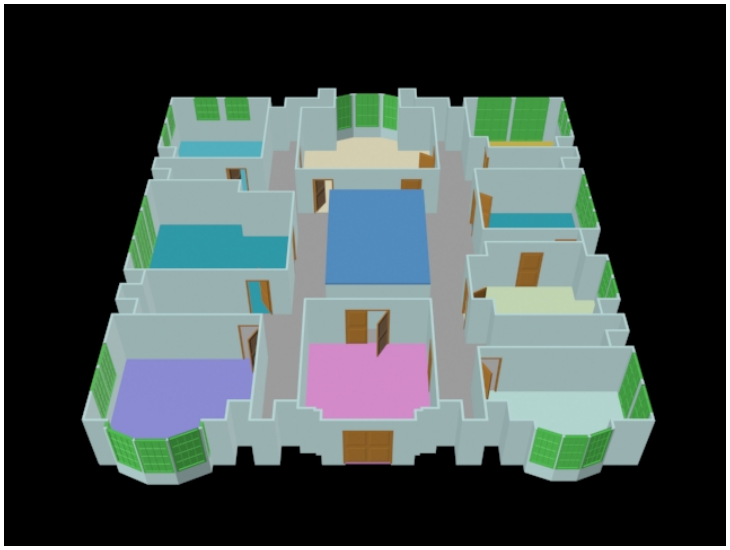
4.3 Σχεδιασμός Παιχνιδιού

Για το σωστό σχεδιασμού του παιχνιδιού, κρίθηκε αναγκαίο να μελετηθούν παραπλήσια επιτραπέζια παιχνίδια τα οποία έχουν γίνει ψηφιακά, καθώς και προηγούμενες εκδόσεις του *Cluedo* που υπάρχουν για προσωπικούς υπολογιστές.

Στην παρούσα ενότητα γίνεται αναλυτική περιγραφή του τρόπου σχεδιασμού των γραφικών σε όλα τα επίπεδα.

4.3.1 Σχεδιασμός Πίστας

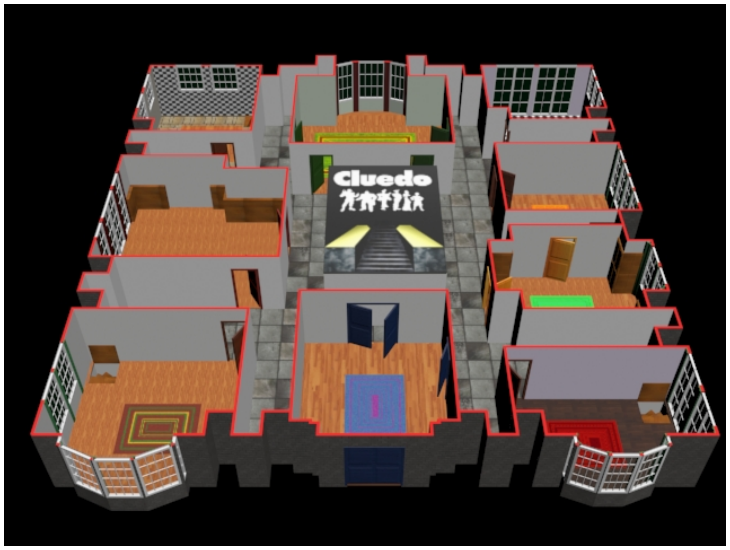
Η πίστα που χρειαζόταν να σχεδιάσουμε είναι το εσωτερικό της έπαυλης. Θα μπορούσαμε να χρησιμοποιήσουμε δικές μας τεχνικές και με τη χρήση της φαντασίας μας να υλοποιήσουμε το σπίτι όπως θα θέλαμε. Παρ' όλα αυτά για να είναι ο σχεδιασμός σωστός και ρεαλιστικός προτιμήσαμε να αντιγράψουμε το εσωτερικό του σπιτιού, όπως αυτό παρουσιάζετε στο ταμπλό του παιχνιδιού της έκδοσης του 2003 (σχήμα 1.2). Χρησιμοποιήθηκε ένα φωτοαντίγραφο του ταμπλό ώστε να μπορέσουμε να ορίσουμε τις διαστάσεις των δωματίων και των αντικειμένων, και να τις μεταφέρουμε σε πραγματική κλίμακα. Ο λόγος για τον οποίον χρησιμοποιήθηκε τέτοια ακρίβεια, είναι γιατί η πίστα θα έπρεπε να είναι σχεδιαστικά ωραία αλλά και ταυτόχρονα λειτουργική και εύκολα προσπελάσιμη από τον χρήστη, σαν να κινείται σε ένα πραγματικό σπίτι. Εντούτοις το ταμπλό παρουσιάζει μια δισδιάστατη κάτοψη της έπαυλης. Έτσι η πιστή αντιγραφή με το μάτι ήταν κάποιες φορές δύσκολη. Σε περίπτωση ατελειών κατά την αντιγραφή, προσπαθήσαμε μέσα από τεχνάσματα να δημιουργήσουμε τρισδιάστατη μορφή, με σκοπό να ξεγελάσουμε το μάτι του χρήστη.



Σχήμα 4.1: Η σχεδίαση της γεωμετρίας του σπιτιού στο πρώτο στάδιο

Ο σχεδιασμός ξεκίνησε με την υλοποίηση της αρχιτεκτονικής του σπιτιού στο *Autodesk 3ds Max*. Βάσει των συντεταγμένων και των διαστάσεων που είχαν προκύψει από το ταμπλό και έπειτα από την κλιμάκωσή τους, δημιουργήσαμε τους τοίχους. Δημιουργήσαμε τρύπες σε αυτούς, με χρήση του εργαλείου boolean, και τοποθετήθηκαν οι πόρτες και τα παράθυρα του σπιτιού. Δημιουργήσαμε τρύπες στο δάπεδο για τις καταπακτές. Τοποθετήθηκαν τα πόμολα στις πόρτες και οι κουρτίνες στις άκρες όλων των παραθύρων. Διαιρέθηκε το δάπεδο σε πλακάκια, με χρήση του τροποποιητή slice, στα οποία προχωράει ο παίχτης κατά την διάρκεια του παιχνιδιού, και ξεχωρίσαμε τα δάπεδα των δωματίων (σχήμα 4.1).

Στο επόμενο στάδιο τοποθετήθηκαν υφές για όλα τα παραπάνω. Για τα δάπεδα των δωματίων και για τις πόρτες χρησιμοποιήθηκαν υφές ξύλου και υφή πέτρας για το δάπεδο κουζίνας. Οι υφές αυτές, βρέθηκαν μετά από αναζήτηση στο διαδίκτυο, ώστε να ταιριάζουν όσο το δυνατόν περισσότερο με αυτές του ταμπλό του παιχνιδιού. Για τα πλακάκια των διαδρόμων, χρησιμοποιήθηκε ένα μέρος των υφών που υπάρχουν στο ταμπλό, έπειτα από σάρωση του. Σχεδιάστηκε από εμάς η υφή του δαπέδου του θερμοκηπίου, οι υφές των χαλιών και όλες οι υπόλοιπες υφές η οποίες είχαν ως βάση ένα στέρεο χρώμα. Ανάλογα με το υλικό του αντικειμένου, δόθηκε η ανάλογη στιλπνότητα και ανταυακλαστικότητα στην υφή. Για τις υφές των τζαμιών των παραθύρων δόθηκε επίσης διαφάνεια (σχήμα 4.2).

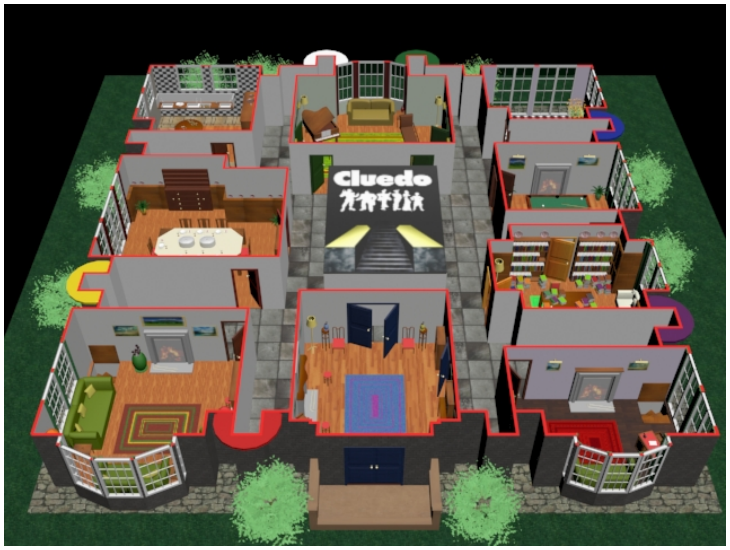


Σχήμα 4.2: Η σχεδίαση της γεωμετρίας του σπιτιού στο δεύτερο στάδιο

Στην συνέχεια σχεδιάστηκαν όλα τα αντικείμενα του σπιτιού ένα προς ένα, προχωρώντας από δωμάτιο σε δωμάτιο. Προσπαθήσαμε να τα αντιγράψουμε όσο το δυνατόν καλύτερα γινόταν και ταυτόχρονα να δώσουμε την αίσθηση του ρεαλισμού. Χρησιμοποιήθηκαν τα εργαλεία και οι τροποποιητές που αναφέρθηκαν σε προηγούμενο κεφάλαιο [υποενότητα 3.3.6], ώστε να δώσουν το κατάλληλο σχήμα στο κάθε αντικείμενο. Για κάποια αντικείμενα που ήταν δύσκολο να σχεδιαστούν, όπως τα φυτά ή το γραμμόφωνο, έγινε η απαραίτητη αναζήτηση στο διαδίκτυο για τον εντοπισμό τους, σε σελίδες που διαθέτουν ελεύθερα τρισδιάστατα αντικείμενα [56, 57, 58]. Τοποθετήθηκαν οι κατάλληλες υφές στα αντικείμενα. Οι υφές είτε αναζητήθηκαν από ελεύθερους διανομείς φωτογραφιών στο διαδίκτυο, είτε σχεδιάστηκαν από εμάς μέσω του *Adobe Photoshop* (σχήμα 4.3).

Για να μπορέσουμε να δούμε τη γεωμετρία μας και τις λεπτομέρειές της σε βάθος, έπρεπε να πλοηγηθούμε μέσα σε αυτή. Έτσι αφού τοποθετήσαμε κάποιους υποτυπώδεις φωτισμούς και μια κάμερα, κάναμε εξαγωγή της γεωμετρίας σε μορφή *VRML97*. Έπειτα την μετατρέψαμε, με της χρήση ενός απλού μετατροπέα, σε μορφή *X3D*. Μέσω του *Octaga Player* μπορέσαμε και κινηθήκαμε μέσα στο σπίτι, εντοπίσαμε ατέλειες που μέσα από το *Autodesk 3ds Max* δεν θα μπορούσαμε να δούμε, και τις διορθώσαμε (σχήμα 4.4).

Στη συνέχεια, ήταν η ώρα να τοποθετήσουμε την γεωμετρία μας στο *Unity3D*. Την εξά-



Σχήμα 4.3: Η σχεδίαση της γεωμετρίας του σπιτιού στο τελευταίο στάδιο

γαμε από το *Autodesk 3ds Max* σε μορφή αρχείου .fbx και την εισάγαμε στο *Unity3D* χρησιμοποιώντας τις κατάλληλες ρυθμίσεις. Μέσω κλιμάκωσης, κάναμε τη γεωμετρία 100 φορές μεγαλύτερη, ώστε να είναι ευδιάκριτη και από το περιβάλλον του *Unity3D*. Σε αυτό το σημείο παρατηρήθηκε ότι το πλήθος των πολυγώνων της γεωμετρίας ήταν υπερβολικά αυξημένο, σε τέτοιο επίπεδο, όπου το παιχνίδι να καθυστερεί πάρα πολύ κατά την εκτέλεση του και να έχουμε μειωμένη συχνότητα καρέ. Το μεγάλο πλήθος πολυγώνων οφείλεται στην απειρία μας κατά το σχεδιασμό, κάτι το οποίο είναι λογικό. Παρ΄ όλα αυτά καταφέραμε και μειώσαμε τα πολύγωνα, σε ένα ποσοστό της τάξης του 70%, στα αντικείμενα που είχαν υπερβολικά μεγάλο αριθμό. Αυτό έγινε μέσω του εργαλείου ProOptimizer του *Autodesk 3ds Max*.

Τελικό στάδιο στο σχεδιασμό της πίστας ήταν ο φωτισμός της. Προσπαθήσαμε, όσο μπορούσαμε, να προσομοιώσουμε τον φωτισμό που αναπαριστάται στο ταμπλό του παιχνιδιού. Αν και ο φωτισμός και οι σκιάσεις των αντικειμένων στο ταμπλό δεν είναι ρεαλιστικοί, εμείς καταφέραμε εν μέρει να προσεγγίσουμε αυτό το φωτισμό μέσα από ρεαλιστικές διαδικασίες. Χρησιμοποιήθηκαν spot lights, point lights και προβολείς φωτός. Οι προβολείς φωτός δημιουργούν την αίσθηση του φωτισμού που βγαίνει από μια λάμπα, με την μόνη διαφορά ότι συμπεριφέρονται σαν υφή και δεν επηρεάζουν στις σκιάσεις των αντικειμένων. Επειδή ολόκληρη η γεωμετρία μας είναι στατική, ο φωτισμός και οι σκιάσεις έμειναν σταθερά μέσω της

Σχήμα 4.4: Πλοήγηση εντός του σπιτιού μέσω *X3D*

Σχήμα 4.5: Η χαρτογράφηση φωτός της γεωμετρίας του σπιτιού
Αριστερά: Η γεωμετρία χωρίς φωτισμούς **Δεξιά:** Η γεωμετρία με χαρτογράφηση φωτός

διαδικασίας της χαρτογράφησης φωτός (lightmapping). Δημιουργήθηκαν οι χάρτες φωτός, βάσει των ρυθμίσεων που επιτρέπει η ελεύθερη έκδοση της *Unity3D*, αποθηκεύτηκαν στο project μας και προβλήθηκαν πάνω στη γεωμετρία (σχήμα 4.5).

4.3.2 Σχεδιασμός Χαρακτήρων

Η σχεδίαση της φιγούρας ενός χαρακτήρα αποτελεί το δυσκολότερο σημείο στο σχεδιασμό των γραφικών ενός παιχνιδιού. Για να μπορέσουν να αναπαρασταθούν με ακρίβεια το σώμα και τα χαρακτηριστικά του προσώπου ενός χαρακτήρα, χρειάζεται μεγάλη λεπτομέρεια κατά τη σχεδίαση. Έτσι, για τον σχεδιασμό τους, επιστρατεύονται έμπειροι σχεδιαστές που γνωρίζουν με μεγάλη λεπτομέρεια τα λογισμικά σχεδιασμού γραφικών. Στην δική μας

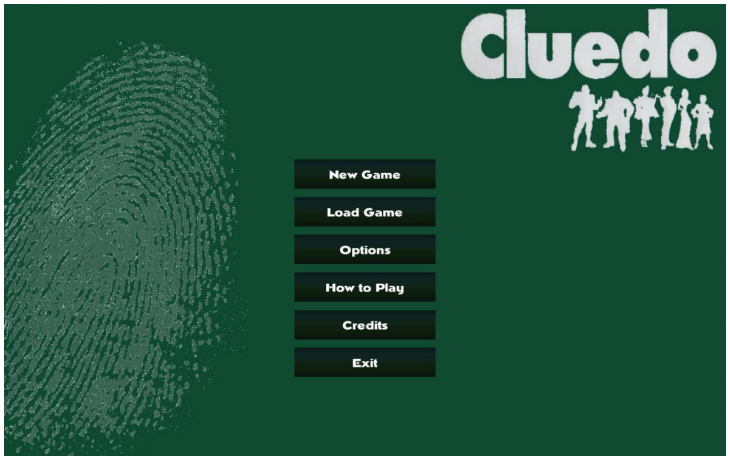
Σχήμα 4.6: Ένας από τους χαρακτήρες του παιχνιδιού
Αριστερά: Ολόκληρος ο χαρακτήρας **Δεξιά:** Ο σκελετός του χαρακτήρα

περίπτωση, θα ήταν ανέφικτο να σχεδιάσουμε τους χαρακτήρες, και αν το προσπαθούσαμε δεν θα φτάναμε σε ένα επιθυμητό αποτέλεσμα. Για την ευκολία μας χρησιμοποιήθηκε το εργαλείο *Project Pinocchio* της *Autodesk* [59].

Με το *Project Pinocchio* δίνεται η δυνατότητα στο χρήστη να δημιουργήσει έναν χαρακτήρα από το μηδέν. Ρυθμίζει το φύλο του χαρακτήρα, τη μορφή του προσώπου (στοιχεία όπως μάτια, αυτιά, μύτη, πιγούνι, στόμα κ.α.), το χρώμα του δέρματος και των ματιών, την κόμμωση και το χρώμα των μαλλιών, το σωματότυπο και το ρουχισμό του. Όλα τα παραπάνω προσφέρονται μέσα από μια μεγάλη ποικιλία ώστε να μπορεί ο χρήστης να δημιουργήσει τον χαρακτήρα που επιθυμεί.

Κατά την εξαγωγή του χαρακτήρα από το *Project Pinocchio* παράγεται ένα αρχείο του *Autodesk 3ds Max*. Το αρχείο αυτό περιέχει, όλο το σώμα του χαρακτήρα με τα χαρακτηριστικά του και από πάνω τον ρουχισμό του. Εσωτερικά του σώματος περιέχεται ένας υποτυπώδης σκελετός, για χρήση του στην υλοποίηση των animations του χαρακτήρα (σχήμα 4.6).

Τέλος έγινε εξαγωγή των χαρακτήρων σε μορφή .fbx από το *Autodesk 3ds Max*. Εισήχθη-



Σχήμα 4.7: Παράθυρο αρχικού μενού

σαν οι γεωμετρίες και τα animations τους στο *Unity3D* για την περαιτέρω χρήση τους. Τοποθετήκαν δύο φωτισμοί τύπου spot lights πάνω σε κάθε παίχτη, για να τον φωτίζουν κάθε φορά που παίζει.

4.4 Γραφικό Περιβάλλον Έναρξης

Σε κάθε παιχνίδι, πριν την εκτέλεσή του, εμφανίζεται ένα γραφικό περιβάλλον έναρξης, στο οποίο ο χρήστης μπορεί να κάνει τις κατάλληλες επιλογές για το παιχνίδι του και να ελέγξει κάποιες ρυθμίσεις. Πριν από αυτό το γραφικό περιβάλλον, συνήθως εμφανίζονται μια σειρά εικόνων, που περιέχουν τα λογότυπα των εταιρειών που υλοποίησαν το παιχνίδι και κατέχουν τα πνευματικά του δικαιώματα. Στην περίπτωσή μας, εμφανίζεται η εικόνα της μηχανής του *Unity3D* (υποχρεωτικά αφού χρησιμοποιούμε την ελεύθερη έκδοση του εργαλείου), το λογότυπο του Πολυτεχνείου Κρήτης και αυτό της εταιρείας *Hasbro, Inc.*. Έπειτα ο χρήστης βρίσκεται στο κύριο μενού του παιχνιδιού.

Το μενού αποτελείται από ένα σύνολο ευδιάκριτων κουμπιών, με τα οποία ο χρήστης μπορεί να πλοηγηθεί σε μια σειρά από επιμέρους παράθυρα (σχήμα 4.7). Τα κουμπιά αυτά, καθώς και η περαιτέρω πλοήγηση τους είναι:

- **New Game**: Πατώντας το συγκεκριμένο κουμπί ο χρήστης, επιλέγει να δημιουργήσει ένα νέο παιχνίδι.

Σχήμα 4.8: Παράθυρο επιλογής παίχτη

Αρχικά, ο χρήστης βρίσκεται στο παράθυρο επιλογής παίχτη (σχήμα 4.8). Εκεί μπορεί να διαλέξει τον παίχτη της αρεσκείας του επιλέγοντας την φωτογραφία του. Σε περίπτωση που ο χρήστης έχει μετανιώσει για την επιλογή του στη δημιουργία ενός νέου παιχνιδιού, μπορεί να επιλέξει το κουμπί "Back", και να επιστρέψει στο αρχικό μενού. Επιλέγοντας το κουμπί "Next", προχωράει στο επόμενο παράθυρο.

Το επόμενο παράθυρό είναι αυτό της επιλογής αντιπάλων (σχήμα 4.9). Σε αυτό εμφανίζονται οι φωτογραφίες όλων των παιχτών, εκτός αυτού που επιλέχθηκε ως παίχτης του χρήστη, στο προηγούμενο παράθυρο. Οι παίχτες έχουν κάτω από τις φωτογραφίες τους ένα κουμπί πολλαπλών επιλογών (combo button), στο οποίο ο χρήστης μπορεί να επιλέξει αν θέλει ένα παίχτη ανενεργό ως αντίπαλο, ή ενεργό σε τρία επίπεδα δυσκολίας (Easy, Moderate, Hard). Και εδώ υπάρχει ένα κουμπί "Back", για επιστροφή στο προηγούμενο παράθυρο. Εδώ, επιλέγοντας ο χρήστης το κουμπί "Next", ξεκινάει την διαδικασία της φόρτωσης του παιχνιδιού. Ο χρήστης είναι υποχρεωμένος, βάσει των κανόνων, να έχει τουλάχιστον δύο αντιπάλους. Σε περίπτωση που δεν πληρείται αυτός ο όρος, τότε κατά την επιλογή του κουμπιού "Next", εμφανίζεται ένα παράθυρο που τον ενημερώνει ότι πρέπει να έχει τουλάχιστον δύο αντιπάλους επιλεγμένους.

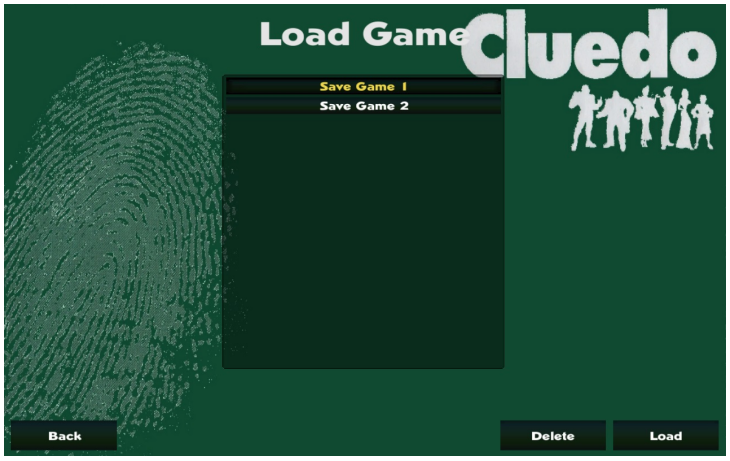
Κατά την φόρτωση του παιχνιδιού, εμφανίζεται ένα παράθυρο με την ένδειξη "Loading...", ώστε ο χρήστης να αναμείνει για την φόρτωση του παιχνιδιού.

Σχήμα 4.9: Παράθυρο επιλογής αντιπάλων

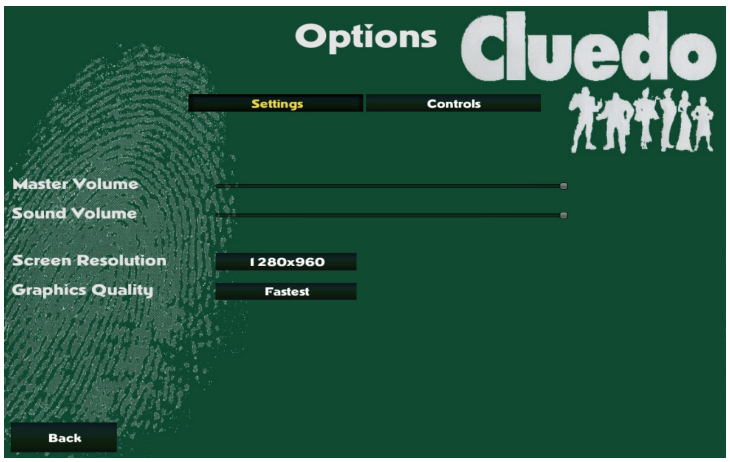
- **Load Game**: Κατά την επιλογή αυτού του κουμπιού ο χρήστης βρίσκεται στο παράθυρο φόρτωσης αποθηκευμένου παιχνιδιού (σχήμα 4.10). Εκεί εμφανίζεται μια λίστα των παιχνιδιών που έχουν αποθηκευτεί παλαιότερα. Ο παίκτης μπορεί να επιλέξει το επιθυμητό παιχνίδι, βάσει του ονόματός του, και να πατήσει το κουμπί "Load" ώστε να ξεκινήσει η φόρτωση του παιχνιδιού από το σημείο που είχε αποθηκευτεί. Με το κουμπί "Delete", ο χρήστης μπορεί να διαγράψει το επιλεγμένο αποθηκεύμενο παιχνίδι από τη λίστα. Σε αυτή την περίπτωση, του εμφανίζεται ένα προειδοποιητικό παράθυρο με το μήνυμα αν είναι σίγουρος ότι θέλει να διαγράψει το παιχνίδι. Ανάλογα με την απόφαση του ο παίχτης επιλέγει το κουμπί "Yes" ή το κουμπί "No". Επίσης, και σε αυτό το παράθυρο, υπάρχει και το κουμπί "Back" για επιστροφή στο αρχικό μενού.

- **Options**: Με την επιλογή αυτού του κουμπιού, ο χρήστης βρίσκεται στο παράθυρο επιλογών του παιχνιδιού. Στο πάνω μέρος του παραθύρου ο χρήστης μπορεί να επιλέξει μεταξύ των κουμπιών "Settings" και "Controls", ανάλογα με το αν θέλει να προβληθούν στο παράθυρο οι ρυθμίσεις ή ο χειρισμός του παιχνιδιού αντίστοιχα. Με το κουμπί "Back" επιστρέφει στο αρχικό μενού.

  Στις ρυθμίσεις του παιχνιδιού, ο χρήστης μπορεί, μέσω δύο οριζόντιων ολισθητών, να ρυθμίσει την ένταση του γενικού ήχου (Master Volume) και του ήχου των εφφέ (Sound Volume). Επίσης μέσα από δύο πλήκτρα πολλαπλών επιλογών, μπορεί να επιλέξει την ανάλυση της οθόνης του, καθώς και την ποιότητα των γραφικών (σχήμα 4.11).



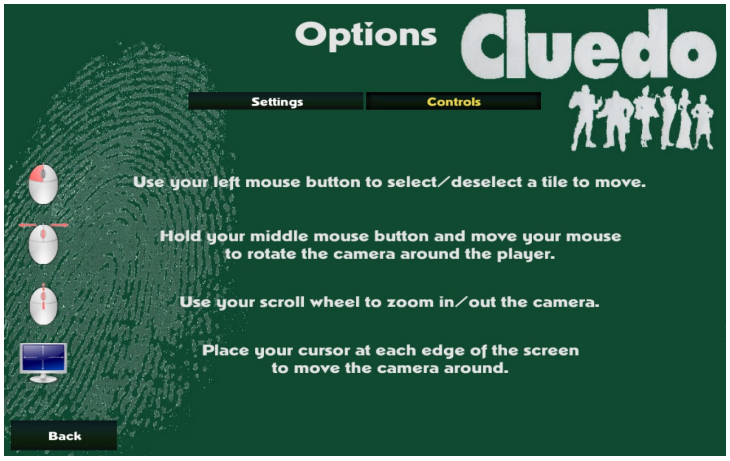
Σχήμα 4.10: Παράθυρο φόρτωσης αποθηκευμένου παιχνιδιού



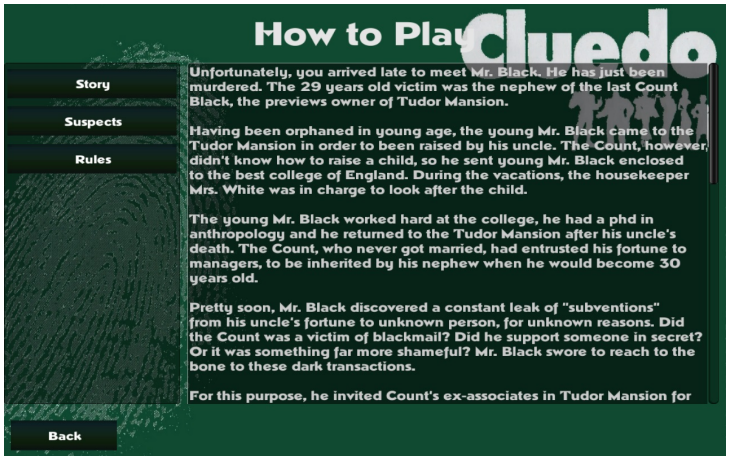
Σχήμα 4.11: Παράθυρο ρυθμίσεων

Στο χειρισμό, ο χρήστης μπορεί να ενημερωθεί για το πως θα χειρίζεται την κίνηση του παίκτη και την κάμερα του, για εύκολη πλοήγηση εντός του παιχνιδιού. Ο χειρισμός του παιχνιδιού μπορεί να γίνει μέσω ποντικιού τριών κουμπιών με ενσωματωμένο τροχό (mouse wheel) (σχήμα 4.12).

- **How to play**: Επιλέγοντας αυτό το κουμπί, ο χρήστης βρίσκεται σε ένα παράθυρο που ενημερώνεται για το πως θα παίξει το παιχνίδι (σχήμα 4.13). Στα αριστερά του παραθύρου υπάρχουν τρία κουμπιά: "Story", "Suspects" και "Rules". Με το πρώτο κουμπί εμφανίζεται ένα πεδίο κειμένου, που αναφέρει το σενάριο το οποίο διαδραματί-



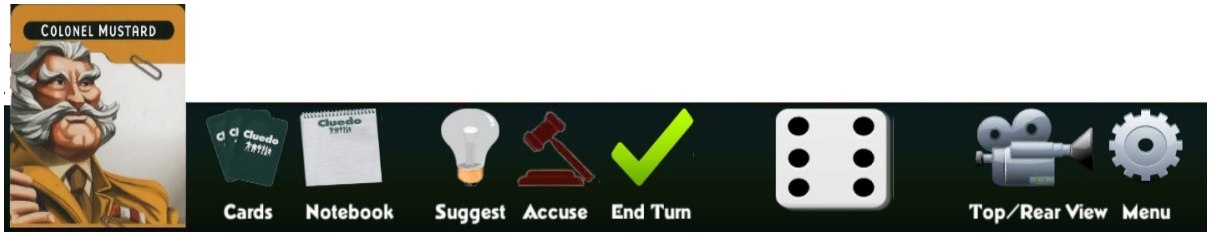
Σχήμα 4.12: Παράθυρο χειρισμού



Σχήμα 4.13: Παράθυρο πληροφοριών παιχνιδιού

ζεται κατά τη διάρκεια του παιχνιδιού. Με το δεύτερο κουμπί, εμφανίζεται ένα πλήθος κουμπιών, που το κάθε ένα έχει πάνω του το όνομα ενός παίχτη-υπόπτου. Πατώντας αυτά τα κουμπιά, εμφανίζεται ένα πεδίο κειμένου, με πληροφορίες σχετικά με τον κάθε ύποπτο ξεχωριστά. Με το τρίτο κουμπί, εμφανίζονται μια σειρά κουμπιών, που περιέχουν τις κατηγορίες των οδηγιών του παιχνιδιού. Επιλέγοντας την κάθε κατηγορία εμφανίζεται στα δεξιά το πεδίο κειμένου με τους αντίστοιχους κανόνες. Με το κουμπί "Back" επιστρέφει στο αρχικό μενού.

- **Credits**: Με την επιλογή αυτού του κουμπιού, προβάλλεται ένα παράθυρο με κυλιόμενες πληροφορίες σχετικά με την κατασκευή του παιχνιδιού, ποιοί συνετέλεσαν στην



Σχήμα 4.14: Head-up display

δημιουργία του και για ποιό σκοπό κατασκευάστηκε. Με το κουμπί "Back" επιστρέφει στο αρχικό μενού.

- **Exit**: Με το τελευταίο κουμπί εμφανίζεται ένα παράθυρο που προειδοποιεί τον χρήστη αν είναι σίγουρος ότι θέλει να εγκαταλείψει το παιχνίδι. Αν επιλέξει το κουμπί "Yes", το παιχνίδι τερματίζεται.

4.5 Γραφική Διεπαφή Χρήστη

Πέρα από το γραφικό περιβάλλον έναρξης, τα περισσότερα παιχνίδια διαθέτουν, κατά την διάρκεια του παιχνιδιού, μια γραφική διεπαφή για να δίνουν πληροφορίες στον χρήστη και να τον βοηθούν στη διαχείριση του παιχνιδιού. Αυτή η γραφική διεπαφή ονομάζεται head-up display (συντ. HUD). Στη δική μας περίπτωση, το HUD αποτελείται από ένα σύνολο κουμπιών και παραθύρων που δίνουν ακριβώς τις ίδιες δυνατότητες στον χρήστη, με αυτές που έχει όταν παίζει το επιτραπέζιο παιχνίδι (σχήμα 4.14). Το HUD είναι τοποθετημένο στο κάτω μέρος της οθόνης μας και αποτελείται από τα παρακάτω στοιχεία (από τα αριστερά προς τα δεξιά):

- **Cards**: Επιλέγοντας αυτό το κουμπί, εμφανίζεται το παράθυρο με τις κάρτες που έχει ο παίχτης στο χέρι του (σχήμα 4.15).

- **Notebook**: Επιλέγοντας αυτό το κουμπί, εμφανίζεται το παράθυρο του σημειωματάριου (σχήμα 4.16). Εκεί ο χρήστης μπορεί να επιλέξει μέσα από τέσσερα διαφορετικά κουμπιά εναλλαγής (toggle button), ποιόν ύποπτο, όπλο ή δωμάτιο έχει αποκλείσει ή υποψιάζεται ή δεν γνωρίζει για αυτό. Μπορεί να διαχειριστεί με όποιον τρόπο εκείνος θέλει τα κουμπιά επιλογών. Επίσης δίπλα σε κάθε αντικείμενο υπάρχει και ένα πεδίο

Σχήμα 4.15: Παράθυρο καρτών

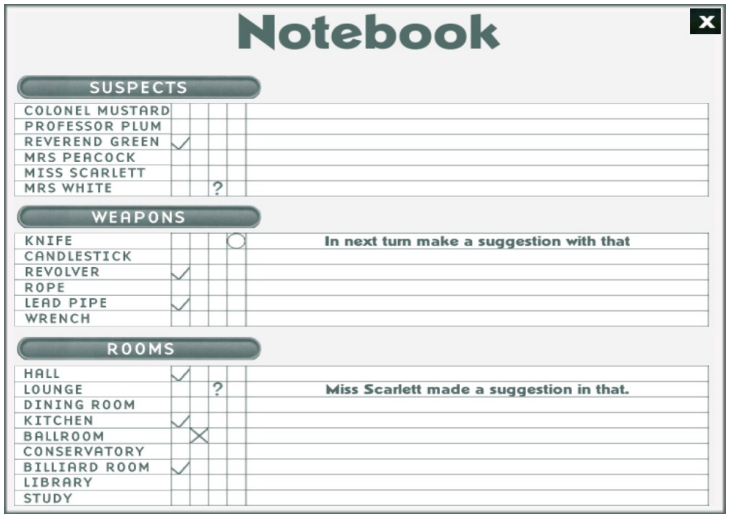
κειμένου, για να σημειώσει ο χρήστης, γράφοντας με το πληκτρολόγιο, οτιδήποτε για το εκάστοτε αντικείμενο.

- **Suggest**: Το κουμπί αυτό ενεργοποιείται και μπορεί να πατηθεί, μόνο όταν ο παίχτης είναι μέσα σε ένα δωμάτιο. Σε αυτή την περίπτωση εμφανίζεται το παράθυρο της υπόθεσης, όπου ο χρήστης μέσω δύο κουμπιών πολλαπλών επιλογών, μπορεί να επιλέξει τον ύποπτο και το όπλο για την υπόθεσή του (το δωμάτιο είναι προεπιλεγμένο με αυτό στο οποίο βρίσκεται ο παίχτης). Πατώντας το κουμπί "OK", επικυρώνει την υπόθεσή του (σχήμα 4.17).

- **Accusation**: Το κουμπί αυτό ενεργοποιεί το παράθυρο της κατηγορίας. Εδώ ο χρήστης μέσω τριών κουμπιών πολλαπλών επιλογών, επιλέγει τον ύποπτο, το όπλο και το δωμάτιο που επιθυμεί να κατηγορήσει. Πατώντας το κουμπί "OK", ο χρήστης επικυρώνει την κατηγορία του (σχήμα 4.18).

- **End Turn**: Χρησιμοποιείται από τον χρήστη για να ολοκληρώσει τον γύρο του.

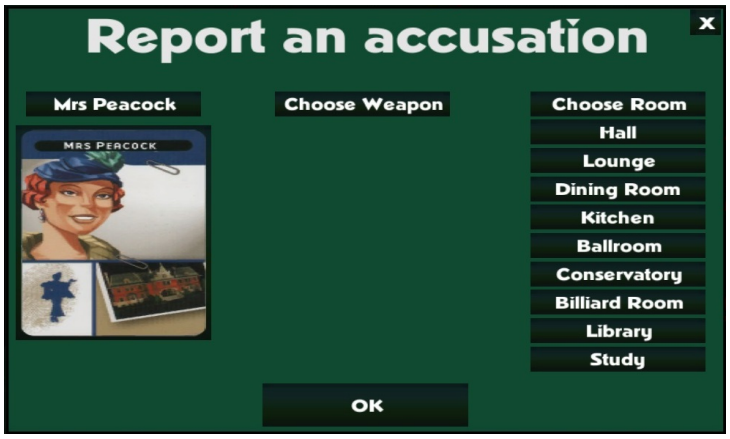
- **Ζάρι**: Το ζάρι το χρησιμοποιεί ο χρήστης στο γύρο του, όταν θέλει να κινηθεί. Σε περίπτωση που πατηθεί εκτός γύρου δεν έχει καμία λειτουργία. Παρ' όλα αυτά, η ζαριά που φέρνει ο κάθε παίχτης στο γύρο του εμφανίζεται στο συγκεκριμένο κουμπί.



Σχήμα 4.16: Παράθυρο σημειωματάριου

- **Top/Rear View**: Χρησιμοποιείται για εναλλαγή της κάμερας μεταξύ εσωτερικής προβολής και προβολής κάτοψης.
- **Menu**: Με το συγκεκριμένο κουμπί ή πατώνατς το πλήκτρο ESC εμφανίζεται το παράθυρο του μενού (σχήμα 4.19). Το μενού αυτό περιέχει τα κουμπιά:
  - **Resume**: Για επιστροφή στο παιχνίδι.
  - **Save Game**: Επιλέγοντας αυτό το κουμπί, εμφανίζεται ένα παράθυρο για αποθήκευση του παιχνιδιού. Το παράθυρο περιέχει ένα πεδίο κειμένου, για να γράψει ο χρήστης το όνομα του αρχείου αποθήκευσης, και δύο κουμπιά "OK" και "Cancel". Με το κουμπί "OK", ο χρήστης επικυρώνει την αποθήκευση του παιχνιδιού. Σε περίπτωση που το όνομα του αρχείου υπάρχει, εμφανίζεται παράθυρο με κουμπιά "Yes" και "No", που ρωτάει τον χρήστη αν επιθυμεί να αντικαταστήσει το αρχείο. Όταν το αρχείο αποθηκευτεί με επιτυχία, εμφανίζεται παράθυρο με αντίστοιχο μήνυμα.
  - **Settings**: Επιλέγοντας το συγκεκριμένο κουμπί, εμφανίζεται το παράθυρο των ρυθμίσεων, όπως αυτό περιγράφηκε σε προηγούμενη ενότητα [ενότητα 4.4].
  - **Back to Main Menu**: Επιλέγοντας το κουμπί αυτό, εμφανίζεται ένα παράθυρο με κουμπιά "Yes" και "No", που ρωτάει τον χρήστη αν επιθυμεί να επιστρέψει στο αρχικό μενού.

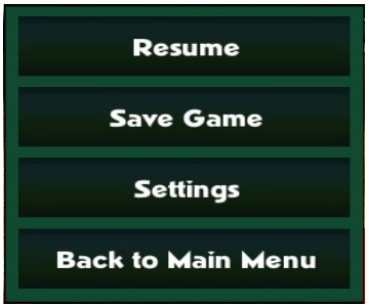
Σχήμα 4.17: Παράθυρο υπόθεσης



Σχήμα 4.18: Παράθυρο κατηγορίας

Τέλος, εκτός των παραθύρων που εμφανίζονται μέσα από τα κουμπιά του HUD, εμφανίζονται κατάλληλα παράθυρα όταν:

- Ένας παίχτης εισέρχεται σε ένα δωμάτιο. Το μήνυμα στο παράθυρο αναφέρει ότι ο εκάστοτε παίχτης μπαίνει στο X δωμάτιο.
- Ένας παίχτης εξέρχεται από ένα δωμάτιο. Το μήνυμα στο παράθυρο αναφέρει ότι ο εκάστοτε παίχτης έφυγε από το X δωμάτιο.
- Ένας παίχτης εξέρχεται από ένα δωμάτιο και εισέρχεται σε ένα άλλο. Το μήνυμα στο παράθυρο αναφέρει ότι ο εκάστοτε παίχτης έφυγε από το X δωμάτιο και μπήκε στο Υ δωμάτιο.



Σχήμα 4.19: Παράθυρο εσωτερικού μενού

- Ένας παίχτης, εκτός του χρήστη, κάνει μια υπόθεση. Εμφανίζεται ένα παράθυρο με την υπόθεση του χρήστη.
- Ένας παίχτης δείχνει μια κάρτα σε έναν άλλον παίχτη, για να αντικρούσει την υπόθεση του. Εμφανίζεται ένα παράθυρο με το μήνυμα ότι ο Χ έδειξε κάρτα στον Υ.
- Ένας παίχτης δείχνει μια κάρτα στον χρήστη, για να αντικρούσει την υπόθεση του. Εμφανίζεται ένα παράθυρο με το μήνυμα ότι ο Χ δείχνει την Υ κάρτα και εμφανίζεται η κάρτα.
- Ο χρήστης πρέπει να δείξει μια κάρτα, για να αντικρούσει την υπόθεση ενός άλλου παίχτη. Εμφανίζεται ένα παράθυρο που περιέχει τις κάρτες προς επιλογή και ένα κουμπί "OK" για την επικύρωσή της.
- Ένας παίχτης κάνει μια κατηγορία. Εμφανίζεται ένα παράθυρο με την κατηγορία. Αν είναι λανθασμένη υπάρχει το κουμπί "OK" ώστε να συνεχίσει το παιχνίδι. Αν είναι σωστή ή την κατηγορία την έχει κάνει ο χρήστης, τότε το παιχνίδι έχει τελειώσει και εμφανίζεται το κουμπί "Back to Main Menu" που επιστρέφει στο αρχικό μενού.

4. ΣΧΕΔΙΑΣΜΟΣ ΠΑΙΧΝΙΔΙΟΥ ΚΑΙ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ

Κεφάλαιο 5

Υλοποίηση Παιχνιδιού

5.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο θα γίνει αναλυτική περιγραφή των σταδίων υλοποίησης του παιχνιδιού. Θα γίνει αναφορά στις λειτουργίες της μηχανής παιχνιδιών και στο πως τις χρησιμοποιήσαμε για να δώσουμε ζωή στο παιχνίδι μας. Τέλος θα αναφερθούμε στο προγραμματιστικό μέρος που υλοποιήσαμε για την δημιουργία των γραφικών, τον χειρισμό του παιχνιδιού καθώς και την τεχνητή νοημοσύνη.

5.2 Φυσική (Physics)

Λόγω του ότι το παιχνίδι είναι επιτραπέζιο, η κίνηση των παιχτών στην ουσία, αντιπροσωπεύει μια κίνηση πιονιών πάνω σε ένα ταμπλό. Για αυτό ο ρεαλισμός στην φυσική ήταν ανύπαρκτος. Τοποθετήθηκαν colliders σε όλα τα αντικείμενα που εισήχθησαν ή δημιουργήθηκαν στο *Unity3D* αυτόματα. Παρ΄ όλα αυτά χρησιμοποιήθηκαν μονάχα οι παρακάτω:

- **Character Controller**: Ο Character Controller αποτελεί στην ουσία έναν Capsule Collider ο οποίος χρησιμοποιείται για να περιβάλλει έναν παίχτη (συνήθως σε παιχνίδια πρώτου και τρίτου προσώπου), ο οποίος δεν έχει ρεαλιστική φυσική, δεν εισάγει την υπομονάδα του άκαμπτου σώματος (Rigidbody), και απλώς θέλουμε να αντιδράει σε συγκρούσεις με τα άλλα αντικείμενα. Στην περίπτωση μας δεν χρησίμευσε ούτε καν αυτό, καθώς ο παίχτης κατευθύνεται μόνος του ανάλογα με τη ζαριά, και εντός των δωματίων μένει σε πάγια θέση· αυτό έχει σαν αποτέλεσμα να αποφεύγονται οι

συγκρούσεις. Χρησιμοποιήθηκε απλά για την κίνηση του παίχτη καθώς και για αντιδράει με το δάπεδο του σπιτιού ώστε να στέκεται πάνω του. Ρυθμίστηκαν παράμετροι όπως το ύψος του ανιχνευτή σύγκρουσης (collider), η ακτίνα του και η θέση του πάνω στο παίχτη.

- **Box Collider**: Οι ανιχνευτές σύγκρουσης σε σχήμα κύβου, χρησιμοποιήθηκαν στα πλακάκια των διαδρόμων καθώς και στα δάπεδα των δωματίων. Ο λόγος που τοποθετήθηκαν ήταν για να μπορεί ο χρήστης να πατάει τα διάφορα σημεία του δαπέδου ώστε να κινήσει τον παίκτη του (αλγόριθμος 5.1). Στους συγκεκριμένους ανιχνευτές σύγκρουσης ενεργοποιήθηκε η επιλογή του εναύσματος (isTrigger), για να εκτελείται αμέσως ο κώδικας όταν το δάπεδο πατηθεί με το ποντίκι.

Αλγόριθμος 5.1: Ενεργοποίηση ανιχνευτών σύγκρουσης δαπέδου

```
void OnMouseUpAsButton() {

 if(!occupied && enable){
        GameObject.Find("cluedo_house").GetComponent<GameController>().↩
            newTilePos = new Vector3(transform.position.x,transform.position.y,↩
            transform.position.z);
  GameObject.Find("cluedo_house").GetComponent<GameController>().newTileName = ↩
      this.name;
  GameObject.Find("cluedo_house").GetComponent<GameController>().newTile = true;
 }
   }
```

5.3 Κινούμενα Σχέδια (Animations)

Τα animations που υλοποιήθηκαν στο παιχνίδι μας είναι:

- **Animation χαρακτήρων**: Υλοποιήσαμε συνολικά 6 χαρακτήρες (όσοι και οι παίχτες του παιχνιδιού) και 2 animations σε κάθε χαρακτήρα, μέσω του *Autodesk 3ds Max*, όπως περιγράφηκε σε προηγούμενη ενότητα [ενότητα 3.3.5]. Τα animations που υλοποιήσαμε είναι:
  - **Idle**: Στην κατάσταση αυτή ο παίχτης βρίσκεται σε αδράνεια. Με μια ανεπαίσθητη κίνηση των χεριών και του κεφαλιού, φαίνεται σαν να στέκεται ο παίκτης και να αναπνέει.

- **Movement**: Στην κατάσταση αυτή, ο παίχτης κινείται. Δημιουργήθηκε ένα animation στο οποίο κινούνται χέρια και πόδια με τέτοιο τρόπο ώστε να προσομοιώνεται ρεαλιστικά η κίνηση.

- **Animation φωτιάς**: Για να αναπαραστήσουμε την φωτιά μέσα στα τζάκια του σπιτιού φτιάξαμε ένα animation. Στην ουσία αποτελούσε μια εναλλαγή τεσσάρων διαφορετικών υφών φωτιάς όπου διαρκούσε ένα δευτερόλεπτο (ένα τέταρτο του δευτερολέπτου για κάθε υφή) και υλοποιήθηκε μέσω του κώδικα script.

- **Animation έναρξης**: Στην αρχή του παιχνιδιού γίνεται η επιλογή των καρτών που αποτελούν την λύση του μυστηρίου. Δημιουργήθηκαν 2 αντικείμενα με την υπομονάδα του Animation, ένα για τις κάρτες και ένα για τον φάκελο που περιέχει τη λύση. Με αυτά τα αντικείμενα σχεδιάσαμε στο *Unity3D* το animation της τοποθέτησης των καρτών μέσα στο φάκελο.

Όλα τα παραπάνω animations εκτελούνται κατά τη διάρκεια του παιχνιδιού τη στιγμή που πρέπει. Η σωστή εκτέλεση τους γίνεται μέσω του κώδικα script με χρήση των συναρτήσεων animation.Play(AnimationClip) και animation.Stop().

5.4 Κίνηση Κάμερας

Η κάμερα είναι από τα πιο σημαντικά κομμάτια στα βιντεοπαιχνίδια. Αποτελούν τα μάτια, και τις περισσότερες φορές τα αυτιά του χρήστη, αφού μέσα από αυτή βλέπει, ακούει και αντιλαμβάνεται οποιοδήποτε γεγονός λαμβάνει χώρα κατά τη διάρκεια του παιχνιδιού. Βασικοί παράγοντες που πρέπει να ληφθούν υπ΄ όψιν, κατά τη δημιουργία της κάμερας και την υλοποίηση της κίνησής της, είναι να μπορεί ο χρήστης να παρατηρεί άμεσα και με ευκολία την εξέλιξη του παιχνιδιού καθώς και να μην του δημιουργεί πρόβλημα και να τον ζαλίζει. Το *Unity3D* περιέχει προεγκατεστημένα δύο αρχεία κώδικα script, τα οποία διαχειρίζονται την κάμερα, για παιχνίδια πρώτου και τρίτου προσώπου.

Η διαχείριση των καμερών, που είχε προεγκατεστημένη το *Unity3D*, δεν μας ικανοποιούσε στην δική μας περίπτωση. Έτσι, οι κάμερες που υλοποιήθηκαν από εμάς, για χρήση τους κατά τη διάρκεια του παιχνιδιού είναι:

- **Κάμερα κάτοψης**: Η κάμερα κάτοψης είναι μια σταθερή κάμερα που προβάλλει το σπίτι από πάνω.

- **Κάμερα κίνησης**: Είναι μια κάμερα τρισδιάστατης προβολής, η οποία βρίσκεται μέσα στο σπίτι και ακολουθεί τον εκάστοτε παίχτη σε κάθε κίνηση του. Η κάμερα αυτή τοποθετείται κάθε φορά που παίζει ένας παίχτης, ως αντικείμενο-παιδί στο αντικείμενο του παίχτη, για να μπορεί να κινείται παράλληλα μαζί του. Μέσω κώδικα script δόθηκε η δυνατότητα στο χρήστη να περιστρέφει την κάμερα, να την μεγεθύνει καθώς και να την μετακινεί στο χώρο (αλγόριθμος 5.2).

Αλγόριθμος 5.2: Διαχείριση κάμερας

```
if(Input.GetAxis("Mouse ScrollWheel")>0){ //zoom in camera
 cam.camera.fieldOfView = Mathf.Lerp(cam.camera.fieldOfView,20,Time.deltaTime*5f);
}
else if(Input.GetAxis("Mouse ScrollWheel")<0){ //zoom out camera
 cam.camera.fieldOfView = Mathf.Lerp(cam.camera.fieldOfView,120,Time.deltaTime*5f);
}

if(Input.GetKey(KeyCode.Mouse2)==true){ //rotate camera
 if(Input.GetAxis("Mouse X")>0)
  cam.transform.RotateAround(new Vector3(GameObject.Find(activePlayer).transform.position.x,GameObject.Find(activePlayer).transform.position.y,GameObject.Find(activePlayer).transform.position.z),Vector3.up,3);
 else if(Input.GetAxis("Mouse X")<0)
  cam.transform.RotateAround(new Vector3(GameObject.Find(activePlayer).transform.position.x,GameObject.Find(activePlayer).transform.position.y,GameObject.Find(activePlayer).transform.position.z),Vector3.up,-3);
}

//move camera
if(Input.mousePosition.y <= 5 && cameraBoundY != 1){
 cam.transform.Translate(new Vector3(0,-0.1f,-0.1f),Space.Self);
 if(cam.transform.position.x > maxCameraBound){
  cameraBoundY = 1;
  cam.transform.position = new Vector3(maxCameraBound,cam.transform.position.y,cam.transform.position.z);
 }
 else if(cam.transform.position.x < minCameraBound){
  cameraBoundY = 1;
  cam.transform.position = new Vector3(minCameraBound,cam.transform.position.y,cam.transform.position.z);
 }
 else if(cam.transform.position.z > maxCameraBound){
  cameraBoundY = 1;
  cam.transform.position = new Vector3(cam.transform.position.x,cam.transform.position.y,maxCameraBound);
```

```
     }
     else if(cam.transform.position.z < minCameraBound){
      cameraBoundY = 1;
      cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
          .position.y,-24);
     }
     else{
      cameraBoundY = 0;
     }
    }
    else if(Input.mousePosition.y >= Screen.height - 5 && cameraBoundY != 2){
     cam.transform.Translate(new Vector3(0,0.1f,0.1f),Space.Self);
     if(cam.transform.position.x > maxCameraBound){
      cameraBoundY = 2;
      cam.transform.position = new Vector3(maxCameraBound,cam.transform.position.↩
          y,cam.transform.position.z);
     }
     else if(cam.transform.position.x < minCameraBound){
      cameraBoundY = 2;
      cam.transform.position = new Vector3(minCameraBound,cam.transform.position.↩
          y,cam.transform.position.z);
     }
     else if(cam.transform.position.z > maxCameraBound){
      cameraBoundY = 2;
      cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
          .position.y,maxCameraBound);
     }
     else if(cam.transform.position.z < minCameraBound){
      cameraBoundY = 2;
      cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
          .position.y,minCameraBound);
     }
     else{
      cameraBoundY = 0;
     }
    }
    else if(Input.mousePosition.x <= 5 && cameraBoundX != 1){
     cam.transform.Translate(new Vector3(-0.2f,0,0),Space.Self);
     if(cam.transform.position.x > maxCameraBound){
      cameraBoundX = 1;
      cam.transform.position = new Vector3(4,cam.transform.position.y,cam.↩
          transform.position.z);
     }
     else if(cam.transform.position.x < minCameraBound){
      cameraBoundX = 1;
      cam.transform.position = new Vector3(-24,cam.transform.position.y,cam.↩
          transform.position.z);
     }
     else if(cam.transform.position.z > maxCameraBound){
      cameraBoundX = 1;
      cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
```

```
            .position.y,4);
        }
        else if(cam.transform.position.z < minCameraBound){
         cameraBoundX = 1;
         cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
            .position.y,-24);
        }
        else{
         cameraBoundX = 0;
        }
       }
       else if(Input.mousePosition.x >= Screen.width- 5 && cameraBoundX != 2){
        cam.transform.Translate(new Vector3(0.2f,0,0),Space.Self);
        if(cam.transform.position.x > maxCameraBound){
         cameraBoundX = 2;
         cam.transform.position = new Vector3(4,cam.transform.position.y,cam.↩
            transform.position.z);
        }
        else if(cam.transform.position.x < minCameraBound){
         cameraBoundX = 2;
         cam.transform.position = new Vector3(-24,cam.transform.position.y,cam.↩
            transform.position.z);
        }
        else if(cam.transform.position.z > maxCameraBound){
         cameraBoundX = 2;
         cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
            .position.y,4);
        }
        else if(cam.transform.position.z < minCameraBound){
         cameraBoundX = 2;
         cam.transform.position = new Vector3(cam.transform.position.x,cam.transform↩
            .position.y,-24);
        }
        else{
         cameraBoundX = 0;
        }
       }
      }
```

Για να βρούμε τον τρόπο που θα τοποθετηθούν οι κάμερες, έπρεπε να μελετηθούν παραπλήσια παιχνίδια και να δούμε πως τις διαχειρίζεται εκεί ο χρήστης. Για την κάμερα κάτοψης δεν υπήρξε ιδιαίτερο πρόβλημα. Για αυτή της κίνησης υπήρξε η σκέψη να τοποθετηθούν πολλές κάμερες, μια σε κάθε παίχτη. Αυτό όμως μας δημιουργούσε πρόβλημα στη διαχείριση τους μέσω του κώδικα. Έτσι επιλέχθηκε μια κάμερα που αρχικά αποφασίσαμε να την τοποθετήσουμε σε ισομετρική 3/4 προβολή.

Η ισομετρική 3/4 προβολή, είναι μια μέθοδος για την οπτική αντιπροσώπευση τρισδιάστατων

Σχήμα 5.1: Ισομετρική 3/4 προβολή
Diablo II (*Blizzard*-2000)

αντικειμένων σε δύο διαστάσεις [60]. Είναι μια αξονομετρική προβολή στην οποία εμφανίζονται οι τρεις άξονες των συντεταγμένων στην ίδια κλίμακα και οι γωνίες μεταξύ κάθε δύο από αυτών είναι πάντα 120 μοίρες. Είναι εύκολη στη διαχείριση και πολύ βολική στο χρήστη. Χρησιμοποιείται κατά κόρον σε βιντεοπαιχνίδια και ιδίως στα παιχνίδια ρόλου (RPG) (σχήμα 5.1). Παρ΄ όλα αυτά, επειδή οι χώροι του σπιτιού ήταν αρκετά στενοί, η προβολή αυτής της κάμερα δεν βοηθούσε στην κίνηση του παίχτη. Έτσι επιλέχθηκε η ισομετρική προβολή αλλά από την μπροστινή μεριά του παίχτη και όχι από το πλάι.

Τέλος στις υπομονάδες των καμερών ρυθμίστηκαν το μέγεθος και το οπτικό πεδίο, η κάμερα κάτοψης επιλέχθηκε σε ορθογραφική προβολή ενώ η κίνησης σε προοπτική προβολή, δόθηκαν οι αρχικές του θέσεις και τοποθετήθηκαν υφές νυχτερινού ουρανού (skybox).

5.5 Σκιαστές (Shaders)

Όπως αναφέραμε σε προηγούμενη ενότητα [υποενότητα 3.5.9], οι σκιαστές οι οποίοι υπάρχουν προεγκατεστημένοι στο *Unity3D* επαρκούσαν στο να δημιουργηθούν όλα τα είδη υφών που θέλαμε. Το μόνο πράγμα που αποτέλεσε εξαίρεση είναι οι ετικέτες των δωματίων οι οποίες εμφανίζονται όταν η προβολή γίνεται από την κάμερα κάτοψης. Σε αυτή την περίπτω-

ση θέλαμε να δημιουργήσουμε έναν σκιαστή με διαφάνεια (σύνθεση alpha) όπως επίσης να είναι και αυτόφωτος (self-illuminated), κάτι το οποίο δεν υπήρχε. Μετά από μια σύντομη αναζήτηση στο διαδίκτυο βρέθηκε η υλοποίηση του [61]. Αντιγράφηκε σε ένα αρχείο .shader και προστέθηκε σαν σκιαστής στην υφή μας.

5.6 Γραφικό Περιβάλλον Έναρξης

Όπως έχει ήδη αναφερθεί [ενότητα 4.4], στην αρχή του παιχνιδιού δημιουργήθηκε το γραφικό μενού. Η υλοποίησή του έγινε σε ένα ξεχωριστό αρχείο κώδικα script. Στην αρχή τοποθετήθηκαν οι εικόνες-λογότυπα, οι οποίες είχαν σταδιακή εμφάνιση και εξαφάνιση στην οθόνη (αλγόριθμος 5.3). Στην συνέχεια έγινε η σχεδίαση όλων των παραθύρων του μενού και των στοιχείων που τοποθετήθηκαν πάνω τους.

Αλγόριθμος 5.3: Σταδιακή εμφάνιση και εξαφάνιση εικόνων-λογότυπα

```
/**Fade in function*/
IEnumerator FadeIn (){
 float time= Time.time;
       while (Time.time - time < 1) {
           alpha = Time.time - time;
           yield return 0;
       }
}

/**Fade out function*/
IEnumerator FadeOut (){
 float time= Time.time;
 while (Time.time - time < 1) {
           alpha = 1- Time.time + time;
           yield return 0;
       }
}
```

Για την σχεδίαση των γραφικών των παραθύρων, χρησιμοποιήθηκαν οι συναρτήσεις γραφικής διεπαφής (GUI) του *Unity3D* που έχουν αναφερθεί [υποενότητα 3.6.3]. Η σχεδίαση του κάθε παραθύρου έγινε σε ξεχωριστή συνάρτηση, αφού η συνάρτηση GUI.Window(), η οποία εκτελεί την σχεδίαση του παραθύρου, έχει ως όρισμα μια συνάρτηση για την εκτέλεση

του κώδικα των γραφικών στοιχείων που θα τοποθετηθούν σε αυτό, με πρώτο και κύριο την εικόνα φόντου (παράδειγμα παραθύρου αλγόριθμος 5.4). Η κλήση των συναρτήσεων GUI.Window() γίνεται μέσα στην συνάρτηση OnGUI(). Ανάλογα με το ποιό κουμπί πατάει ο χρήστης και σε ποιό παράθυρο βρίσκεται, γίνεται η κατάλληλη εναλλαγή μεταξύ των παραθύρων. Η εναλλαγή γίνεται μέσω ενός animation, που δείχνει το ένα παράθυρο να κυλάει σταδιακά προς την μια μεριά και να έρχεται το επόμενο από την άλλη (παράδειγμα αλγόριθμος 5.5).

Αλγόριθμος 5.4: Σχεδίαση κύριου παραθύρου

```
/**Main Window*/
void Window1(int id){

 GUI.DrawTexture(new Rect(0,0,width,height),texture1);
 GUILayout.BeginArea(new Rect(width/2-100,height/2-120,200,400));

 //change window for each option
 if(GUILayout.Button("New Game",GUILayout.Height(50))){
  activeWindow = 2;
  passWindow = 1;
  anim = true;
 }
 GUILayout.Space(10);
 if(GUILayout.Button("Load Game",GUILayout.Height(50))){
  activeWindow = 3;
  passWindow = 1;
  anim = true;
 }
 GUILayout.Space(10);
 if(GUILayout.Button("Options",GUILayout.Height(50))){
  activeWindow = 4;
  passWindow = 1;
  anim = true;
 }
 GUILayout.Space(10);
 if(GUILayout.Button("How to Play",GUILayout.Height(50))){
  activeWindow = 5;
  passWindow = 1;
  anim = true;
 }
 GUILayout.Space(10);
 if(GUILayout.Button("Credits",GUILayout.Height(50))){
  creditsRoll = 0;
  activeWindow = 6;
  passWindow = 1;
```

```
   anim = true;
  }
  GUILayout.Space(10);
  if(GUILayout.Button("Exit",GUILayout.Height(50))){
   activeWindow = 7;
   passWindow = 1;
  }

  GUILayout.EndArea();
  return;
 }
```

Αλγόριθμος 5.5: Εναλλαγή μεταξύ των παραθύρων

```
    if(anim == true){
     if(oneShot == true){
      sound.PlayOneShot(pageTurn,soundVolume);
      oneShot = false;
     }

     if(animPos>-width){
      switch(passWindow){
      case 2:
       GUI.Window(2,new Rect(animPos-=5,0,width,height),Window2,new GUIContent());
       break;
      case 3:
       GUI.Window(3,new Rect(animPos-=5,0,width,height),Window3,new GUIContent());
       break;
      case 4:
       GUI.Window(4,new Rect(animPos-=5,0,width,height),Window4,new GUIContent());
       break;
      case 5:
       GUI.Window(5,new Rect(animPos-=5,0,width,height),Window5,new GUIContent());
       break;
      case 6:
       GUI.Window(6,new Rect(animPos-=5,0,width,height),Window6,new GUIContent());
       break;
      }
      GUI.Window(1,new Rect(width+animPos,0,width,height),Window1,new GUIContent());
     }
     else{
      anim = false;
      animPos = 0;
      oneShot = true;
     }
    }
    else{
```

```
    GUI.Window(1,new Rect(0,0,width,height),Window1,new GUIContent());
  }
```

Τις περισσότερες φορές χρησιμοποιήθηκαν τα γραφικά αντικείμενα της κλάσης GUILayout, γιατί αποδείχθηκε πιο εύκολη η αυτόματη τοποθέτηση των αντικειμένων από έναν προεγκατεστημένο διαχειριστή παραθύρου. Το GUILayout έχει αρκετούς διαχειριστές παραθύρων, που τοποθετούν είτε πάντα οριζόντια, είτε πάντα κάθετα, είτε σε ένα συγκεκριμένο μέρος του παραθύρου τα αντικείμενα. Με εμφωλευμένη χρήση αυτών των διαχειριστών, μπορέσαμε και τοποθετήσαμε τα αντικείμενα στα επιθυμητά σημεία.

Κάπως έτσι και με την χρήση όλων των συναρτήσεων σχεδιάστηκαν τα παράθυρα. Άξια αναφοράς είναι ακόμα η κλιμάκωση των γραφικών ώστε να ταιριάζουν πάντα στην ανάλυση της οθόνης (αλγόριθμος 5.6), η διαχείριση αρχείων για ανάκτηση και διαγραφή αποθηκευμένων παιχνιδιών, η αποθήκευση των επιλογών του χρήστη, για μεταφορά τους μεταξύ των σκηνών του παιχνιδιού με χρήση της κλάσης PlayerPrefs, η αλλαγή της ανάλυσης της οθόνης και της ποιότητας των γραφικών με χρήση των συναρτήσεων Screen.SetResolution() και QualitySettings.SetQualityLevel() αντίστοιχα και τέλος, η φόρτωση σκηνής και ο τερματισμός του παιχνιδιού με χρήση των συναρτήσεων Application.LoadLevel() και Application.Quit() αντίστοιχα.

Αλγόριθμος 5.6: Κλιμάκωση γραφικών στην ανάλυση της οθόνης

```
scale.y = Screen.height/height; // calculate vert scale
scale.x = Screen.width/width; // this will keep your ratio base on Vertical scale
scale.z = 1;
GUI.matrix = Matrix4x4.TRS(Vector3.zero, Quaternion.identity, scale); //scale the ↩
    screen
```

5.7 Γραφική Διεπαφή Χρήστη

Η υλοποίηση της γραφικής διεπαφής του χρήστη ακολουθεί την ίδια φιλοσοφία με αυτή της σχεδίασης του γραφικού περιβάλλοντος έναρξης, που περιγράφηκε στην προηγούμενη ενότητα [ενότητα 5.6]. Χρησιμοποιήθηκαν οι συναρτήσεις των κλάσεων GUI και GUILayout,

για να σχεδιαστούν τα παράθυρα που αναφέρθηκαν σε παραπάνω ενότητα [ενότητα 4.5] καθώς και τα αντικείμενα τους.

Αυτό που κάνει την γραφική διεπαφή χρήστη να διαφέρει στην υλοποίηση από το γραφικό περιβάλλον έναρξης είναι ότι πολλές φορές τα παράθυρα ενεργοποιούνται από εισόδους του χρήστη ή από γεγονότα που λαμβάνουν χώρα κατά τη διάρκεια του παιχνιδιού. Έτσι παράθυρα όπως της υπόθεσης ενός άλλου παίχτη, της κατηγορίας, της απόδειξης ότι μια υπόθεση ενός παίχτη είναι λανθασμένη, είναι αναδυόμενα παράθυρα τα οποία εμφανίζονται αυτόματα από τον διαχειριστή του παιχνιδιού μέσω του διαχειριστή του παίχτη που περιγράφονται παρακάτω (παράδειγμα αλγόριθμος 5.7) [ενότητες 5.9, 5.10].

Αλγόριθμος 5.7: Παράθυρο υπόθεσης ενός παίχτη

```
/**Other player's suggest a crime Window*/
void OtherSuggestWindow(int id){
 GUI.DrawTexture(new Rect(0,0,700,500), Resources.Load("smallBackground") as Texture)↩
     ;
 GUI.skin = skin4;
 GUI.Label(new Rect(0,0,700,140),GetComponent<PlayerController>().otherSuggest[0]+" ↩
     makes a suggestion");

 GUI.skin = skin3;
 GUI.Box(new Rect(10,170,190,250),"");
 GUI.DrawTexture(new Rect(15,175,180,240),Resources.Load(GetComponent<↩
     PlayerController>().otherSuggest[1]) as Texture);
 GUI.Box(new Rect(252,170,190,250),"");
 GUI.DrawTexture(new Rect(257,175,180,240),Resources.Load(GetComponent<↩
     PlayerController>().otherSuggest[2]) as Texture);
 GUI.Box(new Rect(500,170,190,250),"");
 GUI.DrawTexture(new Rect(505,175,180,240),Resources.Load(GetComponent<↩
     PlayerController>().otherSuggest[3]) as Texture);

 //play sound
 if(makesSuggestionSound){
  makesSuggestionSound = false;
  StartCoroutine("MakesSuggestionSound");
 }

 //press button end return to inital values
 if(GUI.Button(new Rect(250,440,200,50),"OK")){ //if the button pressed
  otherSuggestEnabled = false;
  otherSuggestClosed = true;
  makesSuggestionSound = true;
```

```
  }
}
```

Ένα πράγμα που είναι αρκετά σημαντικό στην γραφική διεπαφή είναι η προτεραιότητα των παραθύρων. Πολλές φορές δύο ή και περισσότερα παράθυρα μπορεί να είναι ενεργά σε μια γραφική διεπαφή οπότε ο σχεδιαστής πρέπει να ορίσει ποιά παράθυρα θα βρίσκονται μπροστά από ποιά. Με τις συναρτήσεις GUI.BringWindowToFront() και GUI.FocusWindow(), ο σχεδιαστής μπορεί να ορίσει αν ένα παράθυρο θα είναι μπροστά από όλα τα άλλα ενεργά παράθυρα και αν μόνο αυτό το παράθυρο θα μπορεί να διαχειρίζεται ο χρήστης αντίστοιχα.

Τέλος στη γραφική διεπαφή μεγάλο ρόλο παίζουνε οι υφές που θα χρησιμοποιηθούν. Χρησιμοποιήθηκαν οι υφές της έκδοσης του *Cluedo* που κατασκευάσαμε, για να είναι ακόμα πιο πιστή η αντιγραφή και να είναι εύκολο ο χρήστης, σε περίπτωση που έχει παίξει το επιτραπέζιο, να διαχειριστεί γρήγορα και το βιντεοπαιχνίδι. Σημαντικό ρόλο έχουν οι υφές που χρησιμοποιήθηκαν στα κουμπιά του HUD, αφού μέσω αυτών πρέπει ο χρήστης να καταλαβαίνει αμέσως την λειτουργία των κουμπιών.

5.8 Ήχος (Audio)

Ένα βασικό στοιχείο, που δίνει ρεαλισμό σε ένα παιχνίδι και προσελκύει τον χρήστη, είναι ο ήχος. Τοποθετήθηκαν διάφοροι ήχοι στο περιβάλλον έναρξης καθώς και κατά τη διάρκεια του παιχνιδιού. Κατάλληλοι ήχοι βρέθηκαν ύστερα από αναζήτηση στο διαδίκτυο, και χρησιμοποιήθηκε επίσης το εργαλείο *Text-to-Speech* της *Oddcast*, για την δημιουργία των ομιλιών [62].

Οι ήχοι που τοποθετήθηκαν είναι συνοπτικά οι παρακάτω:

- Μουσική του περιβάλλοντος έναρξης χρησιμοποιήθηκε το τραγούδι *The Den* του *Thomas Smith*.
- Ήχος βροχής που χρησιμοποιήθηκε κατά τη διάρκεια του παιχνιδιού.
- Ήχος κεραυνού, ο οποίος πέφτει μέσα στο παιχνίδι κάθε τριάντα δευτερόλεπτα (αλγόριθμος 5.8).

- Ήχοι για κάθε κουμπί της γραφικής διεπαφής, που πατάει ο παίκτης κατά τη διάρκεια του παιχνιδιού.
- Ήχοι νίκης και ήττας του παιχνιδιού.
- Ήχος πτώσης ζαριού.
- Ήχος φωτιάς που χρησιμοποιείται εντός των δωματίων για να δώσει ρεαλισμό στο τζάκι (αλγόριθμος 5.9).
- Ήχος αντρικού και γυναικείου βηματισμού καθώς και ξύλινης σκάλας για την χρήση των καταπακτών.
- Ομιλίες στα αγγλικά που χρησιμοποιούνται για να ειδοποιούν τον χρήστη για κάθε γεγονός του παιχνιδιού.

Αλγόριθμος 5.8: Αναπαραγωγή ήχου κεραυνού

```
/**Sound for fire*/
IEnumerator SoundFire(){
 sound[1].PlayOneShot(fileSound,soundVolume);
 yield return new WaitForSeconds(2.3f);
 soundFire = true;
}
```

Αλγόριθμος 5.9: Αναπαραγωγή ήχου φωτιάς

```
/**Light thunder function*/
IEnumerator ThunderFunc(){
 Thunder.SetActive(true);
 sound[0].PlayOneShot(thunderSound,soundVolume);
 yield return new WaitForSeconds(0.2f);
 Thunder.SetActive(false);
}
```

Τέλος οι ήχοι εκτελούνται την κατάλληλη στιγμή με χρήση της συνάρτησης sound.PlayOneShot(), που αναπαράγει έναν ήχο σε συγκεκριμένη ένταση (δίνονται σαν ορίσματα) για μια φορά. Με την sound.Stop(), σταματάει οποιαδήποτε αναπαραγωγή εκτελείται στην πηγή του ήχου τη δεδομένη στιγμή.

5.9 Διαχείριση Παιχνιδιού

Ο διαχειριστής του παιχνιδιού είναι το βασικότερο σύστημα που υλοποιήθηκε. Είναι αυτός, που βάσει των κανόνων του παιχνιδιού, διαχειρίζεται κατάλληλα τα γεγονότα. Κατά το ξεκίνημα της εκτέλεσής του, γίνεται αρχικοποίηση των μεταβλητών και ελέγχεται αν το παιχνίδι που θα εκτελεστεί, είναι καινούργιο ή είναι ανάκτηση κάποιου αποθηκευμένου. Μετά περνάει στην εκτέλεση του παιχνιδιού, που έχει τα παρακάτω στάδια:

1. Ενεργοποίηση των ανιχνευτών σύγκρουσης (colliders) του δαπέδου, για να μπορεί ο χρήστης επιλέγοντας τα πλακάκια του δαπέδου να μετακινήσει τον παίχτη του. Σε περίπτωση που ο ενεργός παίχτης δεν είναι του χρήστη οι ανιχνευτές απενεργοποιούνται.

2. Χειρισμός κάμερας. Αναλύθηκε εκτενώς σε προηγούμενη ενότητα [ενότητα 5.4].

3. Ο διαχειριστής αναμένει να κάνει ο παίχτης την κίνησή του. Ελέγχει εάν ο παίχτης βρίσκεται μέσα σε δωμάτιο ή όχι και αν ναι, περιμένει αν ο παίχτης θα κινηθεί με τη χρήση του ζαριού ή αν θα χρησιμοποιήσει την καταπακτή στα κατάλληλα δωμάτια.

4. Ο διαχειριστής προχωράει στο στάδιο της κίνησης. Το κομμάτι κώδικα αυτό εκτελείται κάθε φορά που ο χρήστης επιλέγει ένα πλακάκι του δαπέδου για να μετακινηθεί έως ότου ολοκληρώσει την κίνηση του. Ελέγχει κάθε φορά αν ο παίχτης εξέρχεται από ένα δωμάτιο ή εισέρχεται σε ένα (παράδειγμα αλγόριθμος 5.10), αν χρησιμοποιήθηκε καταπακτή ή αν το πλακάκι του δαπέδου που επέλεξε είναι γειτονικό με το προηγούμενο (παράδειγμα αλγόριθμος 5.11). Τέλος, ελέγχει αν έχει ολοκληρώσει την κίνηση του είτε με το να φτάσει σε ένα δωμάτιο είτε με το να τελειώσει η ζαριά του.

Αλγόριθμος 5.10: Είσοδος στην αίθουσα χορού

Για κάθε δωμάτιο, οι συντεταγμένες κάθε παίχτη είναι μοναδικές και πάγιες.

```
      if(newTileName.Equals("Floor_Ballroom")){
       if(tileNames[tileCount].Equals("Tile52") || tileNames[tileCount].Equals("↩
           Tile69") || tileNames[tileCount].Equals("Tile130") || tileNames[↩
           tileCount].Equals("Tile160")){
        endMovement = true;
        tileCount++;
        tileNames[tileCount] = "Floor_Ballroom";
        switch(activePlayer){ //specific position for each player
        case "colonel_mustard":
         tilePos[tileCount] = new Vector3(-9.895205f,0.1f,-19.26668f);
         break;
        case "miss_scarlett":
         tilePos[tileCount] = new Vector3(-10.00368f,0.1f,-17.37564f);
         break;
        case "mrs_peacock":
         tilePos[tileCount] = new Vector3(-11.13853f,0.1f,-17.39787f);
         break;
        case "mrs_white":
         tilePos[tileCount] = new Vector3(-11.01237f,0.1f,-19.24046f);
         break;
        case "professor_plum":
         tilePos[tileCount] = new Vector3(-11.08105f,0.1f,-18.2511f);
         break;
        case "reverend_green":
         tilePos[tileCount] = new Vector3(-10.03753f,0.1f,-18.34514f);
         break;
        }
       }
      }
```

Αλγόριθμος 5.11: Είσοδος σε νέο πλακάκι του δαπέδου

```
      //entering new tile
      else if((Mathf.Approximately(tilePos[tileCount].x - 0.9f,newTilePos.x) && ↩
          Mathf.Approximately(tilePos[tileCount].z,newTilePos.z)) || (Mathf.↩
          Approximately(tilePos[tileCount].x + 0.9f,newTilePos.x) && Mathf.↩
          Approximately(tilePos[tileCount].z,newTilePos.z)) || (Mathf.↩
          Approximately(tilePos[tileCount].z - 0.9f,newTilePos.z) && Mathf.↩
          Approximately(tilePos[tileCount].x,newTilePos.x)) || (Mathf.↩
          Approximately(tilePos[tileCount].z + 0.9f,newTilePos.z) && Mathf.↩
          Approximately(tilePos[tileCount].x,newTilePos.x))){  //if next tile is ↩
          legal
       bool existTile = false;
       for(int i=0; i<=tileCount; i++){ //check if the player pressed an existing ↩
           tile
        if(newTilePos == tilePos[i]){
         existTile = true;
```

```
        break;
       }
      }

      if(!existTile){ //if the tile doesn't exist
       tileCount++;
       tilePos[tileCount] = new Vector3(newTilePos.x,newTilePos.y,newTilePos.z);
       tileNames[tileCount] = newTileName;
       GameObject.Find("Tiles").transform.FindChild(newTileName).transform.↩
           FindChild("Text").GetComponent<TextMesh>().text = tileCount.ToString()↩
           ;
      }
     }
```

5. Στο επόμενο στάδιο ο διαχειριστής βρίσκεται στην κατάσταση της υπόθεσης. Ο ενεργός παίχτης στέλνει την υπόθεσή του στο διαχειριστή του παιχνιδιού. Ο διαχειριστής του παιχνιδιού την κοινοποιεί σε όλους τους παίχτες και ενεργοποιούνται τα κατάλληλα παράθυρα της γραφικής διεπαφής. Τέλος μετακινείται ο παίχτης, που χρησιμοποιήθηκε ως ύποπτος στην υπόθεση, στο δωμάτιο που έγινε η υπόθεση (όπως επιβάλλουν οι κανόνες). Ο παίχτης αυτός ενεργοποιείται ως εξαναγκασμένος σε περίπτωση που δεν ήταν ήδη σε εκείνο το δωμάτιο. Ο εξαναγκασμένος παίχτης, στο γύρο του, μπορεί άμα θέλει να μην μετακινηθεί και να κάνει υπόθεση στο δωμάτιο που τον μεταφέρανε (παράδειγμα αλγόριθμος 5.12).

Αλγόριθμος 5.12: Υπόθεση στην αίθουσα χορού με ύποπτο τον Συνταγματάρχη Μουστάρδα

```
    //move the suggested player
    if(suggestRoom.Equals("Ballroom")){

     switch(suggestPlayer){ //specific position for each player
     case "Colonel Mustard":
      GameObject.Find("colonel_mustard").transform.position = new Vector3↩
          (-9.895205f,0.1f,-19.26668f);

      //unoccupy current tile
      if(GameObject.Find("colonel_mustard").GetComponent<PlayerController>().↩
          posName.Contains("Tile")){
       GameObject.Find("Tiles").transform.FindChild(GameObject.Find("↩
           colonel_mustard").GetComponent<PlayerController>().posName).↩
           GetComponent<TilePress>().occupied = false;
      }

      //if you are still in the room
```

```
if(!GameObject.Find("colonel_mustard").GetComponent<PlayerController>().↩
    posName.Equals("Floor_Ballroom")){
 GameObject.Find("colonel_mustard").GetComponent<PlayerController>().↩
     posName = "Floor_Ballroom";
 GameObject.Find("colonel_mustard").GetComponent<PlayerController>().↩
     forcedRoom = true;
}
break;
```

6. Το στάδιο που έχει σειρά μετά είναι η εμφάνιση κάρτας από έναν παίχτη, για να αποδειχθεί λανθασμένη η υπόθεση που έκανε ο ενεργός παίχτης. Σε αυτή την περίπτωση ελέγχονται, ένας προς ένα, όλοι οι παίχτες του παιχνιδιού ξεκινώντας από τα αριστερά αυτού που έκανε την υπόθεση, μέχρις ότου βρεθεί ο πρώτος που έχει κάρτα για να αποδείξει ότι η υπόθεση είναι λανθασμένη. Υπάρχει βέβαια και η περίπτωση που δεν βρέθηκε κανένας και η υπόθεση δεν μπορεί να αποδειχθεί λανθασμένη.

7. Το επόμενο στάδιο είναι αυτό της κατηγορίας. Ελέγχεται η κατηγορία που έκανε ο ενεργός παίχτης (σε περίπτωση που έχει κάνει). Σε περίπτωση που είναι σωστή τερματίζει το παιχνίδι ενώ σε περίπτωση που είναι λάθος ο παίχτης αποχωρεί από το παιχνίδι.

8. Τελευταίο στάδιο είναι η μετάβαση στον επόμενο παίχτη. Αφού ο παίχτης έχει περάσει από τα προηγούμενα στάδια, δίνει εντολή στο διαχειριστή για την ολοκλήρωση του γύρου του. Βρίσκεται ο πρώτος παίχτης από τα αριστερά του ενεργού, που δεν είναι εκτός παιχνιδιού, και γίνεται πλέον αυτός ενεργός. Τοποθετείται η κάμερα ώστε να βλέπει πάνω σε αυτόν και ο διαχειριστής ξεκινάει πάλι την ίδια διαδικασία από την αρχή (αλγόριθμος 5.13).

Αλγόριθμος 5.13: Μετάβαση στον επόμενο παίχτη

```
int count = i+1;
while(count != i){ //find next active player
 if(count == enabledPlayers.Length)
  count = 0;
 if(state[count] != stateEnum.Inactive1 && state[count] != stateEnum.↩
     Inactive2){
  activePlayer = enabledPlayers[count];
  cam.transform.parent = GameObject.Find(activePlayer).transform;
  cam.transform.localPosition = new Vector3(0.30f,7.51f,5.87f);
```

```
        cam.transform.localEulerAngles = new Vector3(54.79762f,180f,0);

        //load human camera
        if(enabledPlayers[count] == humanPlayer){
         cam.transform.localPosition = humanCamera[0];
         cam.transform.localEulerAngles = humanCamera[1];
        }

        break;
       }
       count++;
      }
```

5.10 Διαχείριση Παίχτη

Ο διαχειριστής του παίχτη είναι εκείνο το σύστημα, που είναι αρμόδιο να εκτελέσει τις κινήσεις κάθε παίχτη κατά τη διάρκεια του παιχνιδιού. Χωρίζεται σε δύο καταστάσεις: ενεργή και ανενεργή. Στην ενεργή κατάσταση εκτελούνται όλες αυτές οι λειτουργίες, που χρειάζεται να γίνουν κατά τη διάρκεια του γύρου του παίχτη. Στην ανενεργή κατάσταση, αρχικοποιούνται οι μεταβλητές του παίχτη και ο παίχτης περιμένει να έρθει η σειρά του να παίξει. Κάθε μια από τις παραπάνω καταστάσεις χωρίζονται σε παίχτης-χρήστης και αυτόνομος παίχτης.

Στην ενεργή κατάσταση του παίχτη-χρήστη, το σύστημα αποτελείται από τα παρακάτω στάδια:

1. Το σύστημα περιμένει τον χρήστη να επιλέξει μεταξύ χρήσης του ζαριού μέσω της γραφικής διεπαφής για μετακίνηση, ή χρήσης κάποια καταπακτής στα κατάλληλα δωμάτια ή μετάβασης απευθείας στο στάδιο της υπόθεσης, σε περίπτωση που βρίσκεται σε ένα δωμάτιο μετά από εξαναγκασμένη μετακίνηση.

2. Γίνεται η μετακίνηση του παίχτη μέσω κατάλληλης συνάρτησης, σύμφωνα με τις κινήσεις που έχει επιλέξει, τις οποίες στέλνει ο διαχειριστής του παιχνιδιού (αλγόριθμος 5.14). Αυτό το στάδιο, όπως και τα επόμενα, εκτελούνται σε περίπτωση που ο χρήστης δεν είναι μπλοκαρισμένος σε κάποιο δωμάτιο και μπορεί να μετακινηθεί.

Αλγόριθμος 5.14: Κίνηση παίχτη

```
if(Mathf.Approximately(newPos[i].x,currentPos.x) && newPos[i].z > currentPos.z↩
    ){

 //do not rotate camera
 GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
     parent = null;
 transform.rotation = Quaternion.Euler(new Vector3(0,0,0)); //rotate player
 GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
     parent = GameObject.Find(GameObject.Find("cluedo_house").GetComponent<↩
     GameController>().activePlayer).transform;

 while(newPos[i].z>transform.position.z){ //while haven't reached the new ↩
     position
  animation.Play("Walk");
  transform.Translate(new Vector3(0,0,0.05f),Space.World);
  yield return null;
 }
 transform.position = new Vector3(transform.position.x,transform.position.y,↩
     newPos[i].z); //new position
 currentPos.z = transform.position.z;
}

//going up
else if(Mathf.Approximately(newPos[i].x,currentPos.x) && newPos[i].z < ↩
    currentPos.z){

 //do not rotate camera
 GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
     parent = null;
 transform.rotation = Quaternion.Euler(new Vector3(0,180,0)); //rotate player
 GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
     parent = GameObject.Find(GameObject.Find("cluedo_house").GetComponent<↩
     GameController>().activePlayer).transform;

 while(newPos[i].z<transform.position.z){ //while haven't reached the new ↩
     position
  animation.Play("Walk");
  transform.Translate(new Vector3(0,0,-0.05f),Space.World);
  yield return null;
 }
 transform.position = new Vector3(transform.position.x,transform.position.y,↩
     newPos[i].z); //new position
 currentPos.z = transform.position.z;
}

//goind left
else if(Mathf.Approximately(newPos[i].z,currentPos.z) && newPos[i].x > ↩
    currentPos.x){

 //do not rotate camera
 GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
```

```
        parent = null;
    transform.rotation = Quaternion.Euler(new Vector3(0,90,0)); //rotate player
    GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
        parent = GameObject.Find(GameObject.Find("cluedo_house").GetComponent<↩
        GameController>().activePlayer).transform;

    while(newPos[i].x>transform.position.x){ //while haven't reached the new ↩
        position
     animation.Play("Walk");
     transform.Translate(new Vector3(0.05f,0,0),Space.World);
     yield return null;
    }
    transform.position = new Vector3(newPos[i].x,transform.position.y,transform.↩
        position.z); //new position
    currentPos.x = transform.position.x;
   }

   //going right
   else if(Mathf.Approximately(newPos[i].z,currentPos.z) && newPos[i].x < ↩
       currentPos.x){

    //do not rotate camera
    GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
        parent = null;
    transform.rotation = Quaternion.Euler(new Vector3(0,270,0)); //rotate player
    GameObject.Find("cluedo_house").GetComponent<GameController>().cam.transform.↩
        parent = GameObject.Find(GameObject.Find("cluedo_house").GetComponent<↩
        GameController>().activePlayer).transform;

    while(newPos[i].x<transform.position.x){ //while haven't reached the new ↩
        position
     animation.Play("Walk");
     transform.Translate(new Vector3(-0.05f,0,0),Space.World);
     yield return null;
    }
    transform.position = new Vector3(newPos[i].x,transform.position.y,transform.↩
        position.z); //new position
    currentPos.x = transform.position.x;
   }
```

3. Αν δεν βρίσκεται σε δωμάτιο περνάει απευθείας στο στάδιο 5, αλλιώς περνάει στο στάδιο της υπόθεσης. Εκεί, ανάλογα με τις κάρτες που θα επιλέξει ο χρήστης μέσω της γραφικής διεπαφής, στέλνεται η υπόθεση στο διαχειριστή του παιχνιδιού (αλγόριθμος 5.15).

Αλγόριθμος 5.15: Εκτέλεση υπόθεσης του παίχτη-χρήστη

```
    if((endMove && makeSuggestion) || forcedRoom){
     makeSuggestion = true;
     if(suggestionMade){
      GameObject.Find("cluedo_house").GetComponent<GameController>().↩
          suggestPlayer = suggestPlayer;
      GameObject.Find("cluedo_house").GetComponent<GameController>().↩
          suggestWeapon = suggestWeapon;
      GameObject.Find("cluedo_house").GetComponent<GameController>().suggestRoom↩
           = suggestRoom;
      GameObject.Find("cluedo_house").GetComponent<GameController>().suggestion ↩
          = true;
      makeSuggestion = false;
      suggestionMade = false;
      endMove = true; //you cannot move anymore
      diceEnabled = true; //you cannot roll the dice
      forcedRoom = false; //you cannot make any suggestion in this room
     }
    }
```

4. Περνάει στο στάδιο όπου ενεργοποιείται το παράθυρο της γραφικής διεπαφής για εμφάνιση της κάρτας που απορρίπτει την υπόθεση.

5. Τέλος το σύστημα περιμένει τον χρήστη να επιλέξει το κουμπί "End Turn" της γραφικής διεπαφής, για να περάσει στην ανενεργή κατάσταση. Σε περίπτωση που ο χρήστης κάνει μια κατηγορία, αυτή η λειτουργία γίνεται απευθείας μεταξύ της γραφικής διεπαφής και του διαχειριστή του παιχνιδιού.

Στην ενεργή κατάσταση του αυτόνομου παίχτη, το σύστημα διαχειρίζεται τις κινήσεις που θα κάνει ο παίχτης, βάσει μιας υλοποιημένης τεχνητής νοημοσύνης που του δίνει την αυτονομία. Τα στάδια από τα οποία αποτελείται είναι:

1. Έλεγχος για το αν ο παίχτης έχει μετακινηθεί εξαναγκαστικά σε κάποιο δωμάτιο. Σε αυτή την περίπτωση επιλέγεται ο παίχτης να κάνει υπόθεση, μόνο εάν το δωμάτιο που βρίσκεται δεν το έχει αποκλείσει. Σε οποιαδήποτε άλλη περίπτωση, επιλέγεται να μετακινηθεί.

2. Εκτελεί την κίνηση του και την στέλνει στον διαχειριστή του παιχνιδιού. Αν βρίσκεται σε δωμάτιο που έχει καταπακτή, και το δωμάτιο στο οποίο θα μετακινηθεί δεν το έχει αποκλείσει, επιλέγει να χρησιμοποιήσει την καταπακτή. Σε οποιαδήποτε άλλη περίπτωση ρίχνει το ζάρι και ελέγχει αν με τη ζαριά που έχει μπορεί να μετακινηθεί ή είναι

μπλοκαρισμένος. Σε περίπτωση που είναι μπλοκαρισμένος ελέγχει αν υπάρχει καταπακτή ώστε να την χρησιμοποιήσει υποχρεωτικά, αλλιώς δεν προχωράει και παραμένει εκεί μπλοκαρισμένος. Σε περίπτωση που μπορεί να κινηθεί, εκτελεί αλγόριθμο εύρεση βέλτιστου μονοπατιού, ο οποίος αναλύεται εκτενώς παρακάτω [υποενότητα 5.11.1] (αλγόριθμος 5.16).

Αλγόριθμος 5.16: Επιλογή κίνησης αυτόνομου παίχτη

```
if(sendMove){ //search for new movement once
 sendMove = false;

 //in case you want to use the secret passage
 if(posName.Equals("Floor_Lounge") && rooms.Contains("Conservatory") && !↩
     diceEnabledButton){
  diceEnabledButton = true;
  GameObject.Find("Floors").transform.FindChild("Passage_Lounge").↩
      GetComponent<TilePress>().pressed();
 }
 else if(posName.Equals("Floor_Conservatory") && rooms.Contains("Lounge") ↩
     && !diceEnabledButton){
  diceEnabledButton = true;
  GameObject.Find("Floors").transform.FindChild("Passage_Conservatory").↩
      GetComponent<TilePress>().pressed();
 }
 else if(posName.Equals("Floor_Study") && rooms.Contains("Kitchen") && !↩
     diceEnabledButton){
  diceEnabledButton = true;
  GameObject.Find("Floors").transform.FindChild("Passage_Study").↩
      GetComponent<TilePress>().pressed();
 }
 else if(posName.Equals("Floor_Kitchen") && rooms.Contains("Study") && !↩
     diceEnabledButton){
  diceEnabledButton = true;
  GameObject.Find("Floors").transform.FindChild("Passage_Kitchen").↩
      GetComponent<TilePress>().pressed();
 }
 else{
  if(!diceEnabledButton){ //roll dice
   diceEnabledButton = true;
   diceNumber = Random.Range(1,7); //rolling dice
   GameObject.Find("cluedo_house").GetComponent<GameController>().activeHUD↩
       .rollDice = true;
   GameObject.Find("cluedo_house").GetComponent<GameController>().activeHUD↩
       .diceNumber = diceNumber;
   diceEnabled = true;
  }
```

```
        PathFinder path = new PathFinder();

        //if you are not blocked with the dice number
        if(!path.blocked(posName,diceNumber)){
         StartCoroutine("sendMovement");
        }
        else{
         //in case you are blocked use the secret passage anyway
         if(posName.Equals("Floor_Lounge"))
          GameObject.Find("Floors").transform.FindChild("Passage_Lounge").↩
              GetComponent<TilePress>().pressed();
         else if(posName.Equals("Floor_Conservatory"))
          GameObject.Find("Floors").transform.FindChild("Passage_Conservatory").↩
              GetComponent<TilePress>().pressed();
         else if(posName.Equals("Floor_Study"))
          GameObject.Find("Floors").transform.FindChild("Passage_Study").↩
              GetComponent<TilePress>().pressed();
         else if(posName.Equals("Floor_Kitchen"))
          GameObject.Find("Floors").transform.FindChild("Passage_Kitchen").↩
              GetComponent<TilePress>().pressed();
        }
       }
      }
```

3. Γίνεται η μετακίνηση του παίχτη μέσω της ίδιας συνάρτησης, όπως και στην περίπτωση του παίχτη-χρήστη (αλγόριθμος 5.14).

4. Αν δεν βρίσκεται σε δωμάτιο περνάει στο στάδιο 6, αλλιώς στο στάδιο της υπόθεσης. Εδώ ανάλογα με το επίπεδο του παίχτη, επιλέγεται ο κατάλληλος αλγόριθμος για τον σχηματισμό της υπόθεσης και στέλνεται στο διαχειριστή του παιχνιδιού (αλγόριθμος 5.17).

Αλγόριθμος 5.17: Επιλογή υπόθεσης αυτόνομου παίχτη

```
      if(posName.Contains("Floor")){

       string[] suggestion;

       //find the appropriate suggestion
       if(state == GameController.stateEnum.Easy)
        suggestion = makeSuggestionDum();
       else
        suggestion = makeSuggestionClev();
```

```
        //execute the suggestion
        GameObject.Find("cluedo_house").GetComponent<GameController>().↩
            suggestPlayer = suggestion[0];
        GameObject.Find("cluedo_house").GetComponent<GameController>().↩
            suggestWeapon = suggestion[1];
        GameObject.Find("cluedo_house").GetComponent<GameController>().suggestRoom↩
             = posRoom;
        GameObject.Find("cluedo_house").GetComponent<GameController>().suggestion ↩
            = true;
        mySuggestion[0] = suggestion[0];
        mySuggestion[1] = suggestion[1];
        mySuggestion[2] = posRoom;

        diceEnabledButton = true; //you cannot roll the dice
       }
```

5. Περνάει στο στάδιο της απάντησης του διαχειριστή για την απόρριψη της υπόθεσής του. Μόλις γίνει αυτό, υπολογίζει πρώτα το βάθος στο οποίο βρίσκεται ο παίχτης που του έδειξε την κάρτα. Έπειτα ανάλογα με το επίπεδο του, χρησιμοποιεί έναν κατάλληλο αλγόριθμο για να εκτιμήσει την υπόθεση που έκανε και τέλος κοινοποιεί την υπόθεση του σε όλους τους παίχτες (αλγόριθμος 5.18). Ο λόγος για τον οποίο κάνει εκτιμήσεις των υποθέσεων του περιγράφονται παρακάτω [υποενότητα 5.11.2].

Αλγόριθμος 5.18: Εκτίμηση υπόθεσης αυτόνομου παίχτη

```
        //find the player's index and the show card player's index and calculate ↩
            the depth
        int depth = 0, myIndex = 0, playerIndex = 0;
        for(int i=0; i<GameObject.Find("cluedo_house").GetComponent<↩
            GameController>().enabledPlayers.Length; i++){
         if(this.name == GameObject.Find("cluedo_house").GetComponent<↩
             GameController>().enabledPlayers[i])
          myIndex = i;
         if(suggestResponsePlayer == GameObject.Find("cluedo_house").GetComponent↩
             <GameController>().enabledPlayers[i])
          playerIndex = i;
        }

        //calculate depth
        if(playerIndex >= myIndex)
         depth = playerIndex-myIndex;
        else
         depth = GameObject.Find("cluedo_house").GetComponent<GameController>().↩
             enabledPlayers.Length+playerIndex-myIndex;
```

```
        //choose the appropriate response suggestion function
        if(state == GameController.stateEnum.Easy)
         responseSuggestionDum(suggestResponseCard,depth,mySuggestion);
        else
         responseSuggestionClev(suggestResponseCard,depth,mySuggestion);
```

6. Στο τελευταίο στάδιο, ο αυτόνομος παίχτης ελέγχει αν έχει βρει τη λύση του μυστηρίου. Αν ναι, σχηματίζει την κατηγορία του και στην στέλνει στον διαχειριστή του παιχνιδιού. Αν όχι, τελειώνει τον γύρο του.

Στην ανενεργή κατάσταση ο παίχτης περιμένει να έρθει η σειρά του να παίξει. Στο μόνο το οποίο αντιδράει, είναι όταν πρέπει να δείξει κάποια κάρτα για να απορρίψει την υπόθεση ενός άλλου παίχτη. Αν είναι παίχτης-χρήστης, ο χρήστης μέσω της γραφικής διεπαφής επιλέγει την κάρτα που θα δείξει και ειδοποιείται ο διαχειριστής του παιχνιδιού. Αν είναι αυτόνομος παίχτης, ανάλογα με το επίπεδό του, χρησιμοποιείται ο κατάλληλος αλγόριθμος για την επιλογή κάρτας.

5.11 Τεχνητή Νοημοσύνη

Όπως αναφέραμε και στην προηγούμενη ενότητα [ενότητα 5.10], οι αυτόνομοι παίχτες οφείλουν την αυτονομία τους σε ένα σύστημα τεχνητής νοημοσύνης που υλοποιήθηκε. Το σύστημα αυτό υλοποιήθηκε εξ ολοκλήρου από εμάς, αφού το *Unity3D* δεν περιέχει κάποια αντίστοιχη υπομονάδα. Αυτά που συγκεκριμένα μας ενδιέφεραν να υλοποιηθούν είναι το σύστημα εύρεσης βέλτιστου μονοπατιού (path planning), που χρησιμοποιεί ο παίχτης για να κινηθεί κατάλληλα στο παιχνίδι, και το σύστημα λήψης απόφασης, που χρησιμοποιείται από τον παίχτη για να καθοδηγηθεί προς τη λύση του μυστηρίου.

5.11.1 Εύρεση Βέλτιστου Μονοπατιού (Path Planning)

Πρώτος και βασικός στόχος του αυτόνομου παίχτη είναι να μπορεί να κινηθεί έξυπνα μέσα στο παιχνίδι. Η κίνηση του βασίζεται σε μια αλληλουχία γειτονικών πλακιδίων πάνω στο δάπεδο του σπιτιού, ακριβώς όπως και στο επιτραπέζιο παιχνίδι. Η έννοια του να κινηθεί έξυπνα, σημαίνει ότι ο αυτόνομος παίχτης θα πρέπει να επιλέξει, ανάλογα με το που πρέπει να πάει, την συντομότερη διαδρομή βάσει της ζαριάς του, όπως ακριβώς θα την επέλεγε ένας

άνθρωπος. Εκεί επιστρατεύεται ο αλγόριθμος της εύρεσης του συντομότερου μονοπατιού.

Για την υλοποίηση του χρησιμοποιήθηκε ο αλγόριθμος A*, που όπως έχουμε αναφέρει είναι ο αποδοτικότερος και επικρατέστερος σε αυτές τις περιπτώσεις, γιατί βρίσκει βέλτιστο μονοπάτι σε πολύ μικρό χρόνο. Μετά από αναζήτηση στο διαδίκτυο βρέθηκε η υλοποίησή του σε γλώσσα *Java* [63]. Δημιουργήσαμε ένα project και εισάγαμε τον παραπάνω κώδικα. Μετά από αλλαγές που έγιναν και αφαιρώντας περιττά σημεία που δεν χρειάζονταν για την δική μας υλοποίηση, ελέγχθηκε αν και κατά πόσο ο αλγόριθμος είναι λειτουργικός. Μετά τον έλεγχο του, ο αλγόριθμος μεταφέρθηκε σε γλώσσα *C#* και ενσωματώθηκε στο παιχνίδι μας (αλγόριθμος 5.19).

Αλγόριθμος 5.19: Εύρεση βέλτιστου μονοπατιού (Path Planning)

```
public List<Point> calcShortestPath(int startX, int startY, int goalX, int ↩
    goalY) {

        //mark start and goal node
        map.setStartLocation(startX, startY);
        map.setGoalLocation(goalX,goalY);

        //Check if the goal node is also an obstacle (if it is, it is ↩
            impossible to find a path there)
        if (map.getNode(goalX, goalY).isObstacle) {
                return null;
        }

        map.getStartNode().setDistanceFromStart(0);
        closedList.Clear();
        openList.clear();
        openList.add(map.getStartNode());


        //while we haven't reached the goal yet
        while(openList.size() != 0) {

                //get the first Node from non-searched Node list, sorted by ↩
                    lowest distance from our goal as guessed by our heuristic
                Node current = openList.getFirst();

                // check if our current Node location is the goal Node. If it ↩
                    is, we are done.
                if(current.getX() == map.getGoalLocationX() && current.getY() ↩
                    == map.getGoalLocationY()) {
```

```
                return reconstructPath(current);
        }

        //move current Node to the closed (already searched) list
        openList.remove(current);
        closedList.Add(current);

        //go through all the current Nodes neighbors and calculate if ↩
            one should be our next step
        foreach(Node neighbor in current.getNeighborList()) {
                bool neighborIsBetter;

                //if we have already searched this Node, don't bother ↩
                    and continue to the next one
                if (closedList.Contains(neighbor))
                        continue;

                //also just continue if the neighbor is an obstacle
                if (!neighbor.isObstacle) {

                        // calculate how long the path is if we choose↩
                             this neighbor as the next step in the ↩
                            path
                        float neighborDistanceFromStart = (current.↩
                            getDistanceFromStart() + map.↩
                            getDistanceBetween(current, neighbor));

                        //add neighbor to the open list if it is not ↩
                            there
                        if(!openList.contains(neighbor)) {
                                openList.add(neighbor);
                                neighborIsBetter = true;
                                //if neighbor is closer to start it ↩
                                    could also be better
                        } else if(neighborDistanceFromStart < current.↩
                            getDistanceFromStart()) {
                                neighborIsBetter = true;
                                //neighbor.setDepth(neighbor.getDepth↩
                                    ()+1);
                        } else {
                                neighborIsBetter = false;
                                //neighbor.setDepth(neighbor.getDepth↩
                                    () + 1);
                        }
                        // set neighbors parameters if it is better
                        if (neighborIsBetter) {
                                neighbor.setPreviousNode(current);
                                neighbor.setDistanceFromStart(↩
                                    neighborDistanceFromStart);
                                neighbor.setHeuristicDistanceFromGoal(↩
                                    heuristic.↩
```

```
                                                    getEstimatedDistanceToGoal(↩
                                                    neighbor.getPoint(), map.↩
                                                    getGoalPoint()));
                                        }
                                    }

                                }
                            }
                            return null;
                        }
```

Οι βασικές αλλαγές οι οποίες έγιναν στον κώδικα είναι:

- Χρησιμοποιήσαμε για ευριστική συνάρτηση την απόσταση Manhattan όπως περιγράφηκε στην εξίσωση 2.3 (αλγόριθμος 5.20).

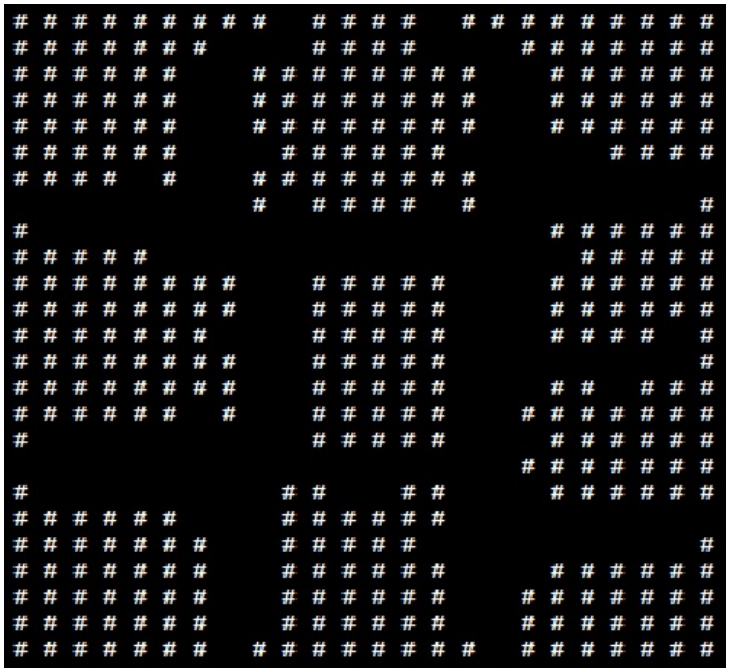
Αλγόριθμος 5.20: Απόσταση Manhattan

```
//calculates the manhattan distance between two points
public float getEstimatedDistanceToGoal(Point start, Point goal) {
 return Mathf.Abs(goal.x-start.x) + Mathf.Abs(goal.y-start.y);
}
```

- Χρησιμοποιήθηκε μια ταξινομημένη λίστα για την υλοποίηση της ανοιχτής λίστας που κρατάει τους κόμβους που δεν έχουν αναζητηθεί ακόμα. Το κριτήριο για την ταξινόμησή τους είναι το εκτιμώμενο κόστος φθηνότερης διαδρομής για κάθε κόμβο, όπως περιγράφηκε στην εξίσωση 2.1.

- Υλοποιήθηκε επίσης μια συνάρτηση για να ελέγχει εάν ο παίχτης είναι μπλοκαρισμένος και δεν μπορεί να κινηθεί βάσει της ζαριάς του. Η συνάρτηση αυτή, ακολουθεί παραπλήσια λογική με αυτή του A*, με τη μόνη διαφορά ότι όταν διασχίσει ένα κόμβο δεν τον τοποθετεί σε μια κλειστή λίστα, γιατί έχει σκοπό να τον αναζητήσει εκ νέου. Αυτό γίνεται γιατί δεν μας ενδιαφέρει πλέον η συντομότερη διαδρομή, αλλά απλώς να υπάρχει μια τουλάχιστον διαδρομή.

Ο αλγόριθμος, όπως περιγράφηκε παραπάνω, πέρα των αντικειμένων της κλάσης κόμβου (Node), χρησιμοποιεί και αντικείμενα των κλάσεων Point και AreaMap. Η πρώτη κλάση περιγράφει ένα σημείο, το οποίο αντιστοιχίζεται σε ένα ζευγάρι ακέραιων συντεταγμένων.



Σχήμα 5.2: Χάρτης πλέγματος που χρησιμοποιεί ο αλγόριθμος A*
Τα εμπόδια αναπαριστώνται με δίεση (#)

Η δεύτερη είναι αυτή που περιγράφει τον χώρο στον οποίο αναζητά ο αλγόριθμος τη συντομότερη διαδρομή, στην ουσία δηλαδή τον χάρτη της πίστας. Ο αλγόριθμος A* λειτουργεί πάντα με γνώμονα ότι το περιβάλλον στο οποίο αναζητείται η διαδρομή είναι διακριτό. Στα περισσότερα παιχνίδια ο χώρος είναι συνεχής, οπότε μεταφράζεται σε ένα πλέγμα πάρα πολύ μικρών διακριτών σημείων, στα οποία μπορεί να κινηθεί ο παίχτης. Στην περίπτωση μας, αυτό δεν μας δυσκόλεψε, καθώς από τη φύση του το παιχνίδι μας περιέχει ένα διακριτό χώρο κίνησης, αυτόν του πλέγματος του ταμπλό (board grid). Έτσι μπορέσαμε να δημιουργήσουμε σαν χάρτη (AreaMap) ένα τετράγωνο πλέγμα, τον οποίο χρησιμοποιεί ο αλγόριθμος A* και περιέχει εμπόδια στα σημεία που δεν έχει πρόσβαση ο παίχτης καθώς και στα σημεία που βρίσκονται άλλοι παίχτες. Ελεύθερα σημεία θεωρήθηκαν αυτά, στα οποία ο παίχτης μπορεί να κινηθεί, δηλαδή τα πλακάκια του δαπέδου καθώς και τα σημεία μέσα από τις πόρτες των δωματίων (σχήμα 5.2).

Όλα τα παραπάνω χρησιμοποιούνται σε μια ευρύτερη διαδικασία, που υπολογίζει το συντομότερο μονοπάτι ενός παίχτη, η οποία υλοποιείται μέσω μιας συνάρτησης. Η συνάρτηση αυτή έχει ως ορίσματα τα σημεία έναρξης ενός παίχτη (μπορεί να είναι περισσότερα από

ένα, αν βρίσκεται σε δωμάτιο με πολλές πόρτες) και τα σημεία στόχου, τα οποία αποτελούν τις πόρτες όλων των δωματίων που δεν έχει αποκλείσει ο παίχτης. Τα στάδια αυτής της συνάρτησης είναι:

1. Μετριέται το πλήθος των σημείων εισόδου και αρχικοποιούνται οι λίστες και οι μεταβλητές που θα χρησιμοποιηθούν.

2. Δημιουργείται ένας πίνακα που αποτελεί τον χάρτη του πλέγματος. Τοποθετούνται ως εμπόδια αρχικά όλα τα σημεία και αφαιρούνται αυτά που είναι μπροστά από τις πόρτες και όλα τα ελεύθερα πλακάκια του δαπέδου (αλγόριθμος 5.21).

Αλγόριθμος 5.21: Δημιουργία χάρτη πλέγματος

```
        //initialize all points obstacles
        for (int i = 0; i < 25; i++){
            for (int j = 0; j < 24; j++){
                obstacle[i, j] = 1;
            }
        }

        //make the appropriate tiles non-obstacle
        for (int i = 0; i < tiles.Length; i++) {
            if (tiles[i].Contains("Floor"))
                obstacle[points[i].x, points[i].y] = 0;
            else if (tiles[i].Contains("Tile")) {
    if(!GameObject.Find("Tiles").transform.Find(tiles[i]).GetComponent<TilePress↩
        >().occupied)
                    obstacle[points[i].x, points[i].y] = 0;
            }
            else if (tiles[i].Contains("initial")){
                obstacle[points[i].x, points[i].y] = 0;
            }
        }
```

3. Εκτέλεση του αλγορίθμου A* με σκοπό την εύρεση του συντομότερου μονοπατιού από κάθε σημείο έναρξης του παίχτη προς κάθε πόρτα των δωματίων που δεν έχει αποκλείσει (αλγόριθμος 5.22). Τα παραπάνω μονοπάτια αποθηκεύονται σε μια λίστα.

Αλγόριθμος 5.22: Εύρεση ελάχιστου μονοπατιού για κάθε δωμάτιο

```
        numOfStarts = -1;
        for (int i = 0; i < tiles.Length; i++){
            if (start.Equals(tiles[i])){ //find the each element of start
                numOfStarts++;
                foreach (string room in rooms){ //take the paths to all rooms
                    for (int j = 0; j < tiles.Length; j++) { //for each room
                        if (room.Equals(tiles[j])) { //find each element of room
                            //create map and initialize a star
                            AreaMap map = new AreaMap(25, 24, obstacle);
                            AStar astar = new AStar(map, new ManhattanDistance())↩
                                ;

                            //run astar algorithm to calculate the path
                            List<Point> path = astar.calcShortestPath(points[i].x↩
                                , points[i].y, points[j].x, points[j].y);
                            roomPaths[numOfStarts].Add(path);
                        }
                    }
                }
            }
        }
```

4. Έλεγχος αν υπάρχει έστω και ένα ελάχιστο μονοπάτι για κάποιο δωμάτιο. Αν ναι προχωράει στο επόμενο στάδιο. Αν όχι, τότε ο παίκτης θεωρείται μπλοκαρισμένος προς κάθε δυνατό δωμάτιο. Υπάρχει όμως περίπτωση ο παίχτης με τη ζαριά που έχει να μπορεί να κινηθεί. Οπότε υπολογίζεται ένα τυχαίο μονοπάτι χρησιμοποιώντας την συνάρτηση που ελέγχει αν ο παίχτης είναι μπλοκαρισμένος.

5. Αν ο παίχτης δεν είναι μπλοκαρισμένος υπολογίζεται το συντομότερο μονοπάτι στο δωμάτιο εκείνο το οποίο έχει εκτιμήσει ως καλύτερο (αλγόριθμος 5.23). Ο τρόπος με τον οποίον γίνεται η εκτίμηση περιγράφεται παρακάτω [υποενότητα 5.11.2].

Αλγόριθμος 5.23: Εύρεση ελάχιστου μονοπατιού για το εκτιμώμενο καλύτερο δωμάτιο

```
            for (int i = 0; i < roomPaths.Length; i++){
                for (int j = 0; j < roomPathsNames.Length; j++) {
                    if (goalList.Contains(roomPathsNames[j])) { //if room exist ↩
                        in goalList
                        if (roomPaths[i][j] != null) { //if path isn't null
                            if (goalUtil[goalList.IndexOf(roomPathsNames[j])] > ↩
                                maxUtil)
                            {
                                shortestPathLength = 200;
```

```
                            if (roomPaths[i][j].Count < shortestPathLength)
                            { //if it is the shortest path

                                shortestPathPoints = roomPaths[i][j];
                                shortestPathLength = shortestPathPoints.Count↩
                                    ; //make the length equal to the new path↩
                                     length
                                maxUtil = goalUtil[goalList.IndexOf(↩
                                    roomPathsNames[j])];
                            }
                        }
                        else if (goalUtil[goalList.IndexOf(roomPathsNames[j])↩
                            ] == maxUtil)
                        {
                            if (roomPaths[i][j].Count < shortestPathLength)
                            { //if it is the shortest path

                                shortestPathPoints = roomPaths[i][j];
                                shortestPathLength = shortestPathPoints.Count↩
                                    ; //make the length equal to the new path↩
                                     length
                                maxUtil = goalUtil[goalList.IndexOf(↩
                                    roomPathsNames[j])];
                            }
                        }
                    }
                }
            }
        }
```

6. Επιστρέφει το μονοπάτι στον διαχειριστή του παίχτη για να εκτελέσει ο παίχτης την κίνησή του.

Τελευταίο μέρος ήταν η βελτίωση του αλγορίθμου A*. Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο [υποενότητα 2.12.3], ένας έξυπνος τρόπος για βελτίωση του χρόνου εκτέλεσης του αλγορίθμου, ήταν η ταξινόμηση της ανοιχτής λίστας με χρήση της introsort. Δεν χρειαζόταν κάποια ιδιαίτερη υλοποίηση στον αλγόριθμό μας, αφού ο αλγόριθμος της ταξινόμησης introsort υπάρχει ήδη υλοποιημένος στις βιβλιοθήκες των δομών δεδομένων της *C#*. Ο αλγόριθμος της ταξινόμησης καταλάβαινε αυτόματα τη συνάρτηση σύγκρισης των κόμβων που είχαμε υλοποιήσει· δεν χρειαζόταν παρά μόνο μια κλήση της συνάρτησης Sort() για την εκτέλεση της ταξινόμησης στη λίστα μας.

5.11.2 Λήψη Απόφασης (Decision Making)

Πέρα από το να κινείται έξυπνα, ένας αυτόνομος παίχτης πρέπει και να αντιδράει έξυπνα στα διάφορα γεγονότα που συμβαίνουν κατά τη διάρκεια του παιχνιδιού. Στη δική μας περίπτωση, ο παίχτης πρέπει μέσα από εύστοχες υποθέσεις να αποκλείσει όσο το δυνατόν πιο γρήγορα όλα τα ενδεχόμενα, και να φτάσει στη λύση του μυστηρίου. Όπως είδαμε και στην προηγούμενη υποενότητα [υποενότητα 5.11.1], για να το πετύχει αυτό ο παίχτης κάνει κάποιες εκτιμήσεις στα διάφορα ενδεχόμενα και ανάλογα με τις εκτιμήσεις παίρνει την κατάλληλη απόφαση για το πως θα αντιδράσει. Οι εκτιμήσεις γίνονται μέσω συναρτήσεων χρησιμότητας (utility functions).

Η υλοποίηση του αλγορίθμου που διαχειρίζεται τις συναρτήσεις χρησιμότητας έγινε μέσα στον διαχειριστή του παίχτη. Η συγκεκριμένη υλοποίηση αποτελείται από δύο φάσεις. Στην πρώτη φάση ο παίχτης εκτιμά, βάσει των γεγονότων του παιχνιδιού και μέσω των συναρτήσεων χρησιμότητας, τα διάφορα ενδεχόμενα. Στην δεύτερη φάση ο παίχτης αποφασίζει να χρησιμοποιήσει τα ενδεχόμενα βάσει των συναρτήσεων χρησιμότητας. Η υλοποίηση διαφέρει επίσης ανάλογα με το επίπεδο του παίχτη. Για να υπάρχει ενδιαφέρον και ποικιλία στο παιχνίδι, δημιουργήθηκαν τρία επίπεδα δυσκολίας: εύκολο (easy), μέτριο (moderate) και δύσκολο (hard).

Οι συναρτήσεις που δημιουργήθηκαν για την παραπάνω υλοποίηση, ανάλογα με το επίπεδο δυσκολίας του παίχτη, είναι:

- **Σύνθεση Υπόθεσης:** Εδώ ο παίχτης επιλέγει ποιόν ύποπτο και ποιό όπλο θα χρησιμοποιήσει στην υπόθεση του. Στην περίπτωση του εύκολου επιπέδου, ο παίχτης τα επιλέγει τυχαία. Στις περιπτώσεις του μέτριου και του δύσκολου επιπέδου, ο παίχτης επιλέγει αυτόν τον ύποπτο και αυτό το όπλο που έχουν την μεγαλύτερη τιμή στην συνάρτηση χρησιμότητας. Το δωμάτιο της υπόθεσης είναι πάντα το δωμάτιο στο οποίο βρίσκεται ο παίχτης. Στην περίπτωση του εύκολου επιπέδου, ο παίχτης έχει επιλέξει να κινηθεί, βάσει του αλγορίθμου που περιγράψαμε στην προηγούμενη υποενότητα [υποενότητα 5.11.1], στο δωμάτιο εκείνο που βρίσκεται πιο κοντά του. Στην περίπτωση του μέτριου και του δύσκολου επιπέδου, ο παίχτης επιλέγει να κινηθεί στο δωμάτιο το οποίο έχει την μεγαλύτερη τιμή στη συνάρτηση χρησιμότητας.
- **Εμφάνιση κάρτας:** Ο παίκτης επιλέγει να εμφανίσει μια κάρτα για να απορρίψει την υπόθεση ενός άλλου παίχτης. Σε περίπτωση που ο παίχτης έχει πάνω από μια κάρτες

που αντιστοιχούν στην υπόθεση, τότε, αν είναι παίχτης εύκολου επιπέδου επιλέγει τυχαία την πρώτη, ενώ σε περίπτωση που είναι μέτριου ή δύσκολου επιλέγει αυτήν με τη μεγαλύτερη τιμή στη συνάρτηση χρησιμότητας (αλγόριθμος 5.24). Ο λόγος που επιλέγεται η μεγαλύτερη τιμή στην συνάρτηση χρησιμότητας εξηγείται παρακάτω 5.11.2.

Αλγόριθμος 5.24: Εμφάνιση κάρτας για απόρριψη της υπόθεσης ενός παίχτη, σε μέτριο και δύσκολο επίπεδο τεχνητής νοημοσύνης

```
/**show card for clever agents*/
private string showCardClev(string[] suggestion){

       float max = int.MinValue;
 string card = null;

 //return the card with the lowest utility
 for(int i=0; i<cards.Count; i++){
  if(cards[i] == suggestion[1] || cards[i] == suggestion[2] || cards[i] == ↩
      suggestion[3]){
   if(cardsUtil[i] > max){
    max = cardsUtil[i];
    card = cards[i];
   }
  }
 }

 return card;
}
```

- **Απάντηση Υπόθεσης:** Όταν ο παίχτης κάνει μια υπόθεση, του έρχεται η απάντηση από τον διαχειριστή του παιχνιδιού. Η απάντηση περιέχει την κάρτα που απορρίπτει την υπόθεση του, καθώς και τον παίχτη που του την έδειξε. Αρχικά ο παίχτης, όπως αναφέραμε σε προηγούμενη ενότητα [ενότητα 5.10], υπολογίζει το βάθος του παίχτη ο οποίος του έδειξε την κάρτα. Κατόπιν αφαιρεί την κάρτα που του έδειξε από τα ενδεχόμενά του. Αν ο παίχτης είναι εύκολου επιπέδου δεν κάνει κάτι περαιτέρω. Σε περίπτωση που είναι μέτριου ή δύσκολου επιπέδου, θεωρεί ότι πρέπει να εκτιμήσει παραπάνω τις άλλες δύο κάρτες που χρησιμοποίησε στην υπόθεση του και δεν απορρίφθηκαν. Για να τις υπερεκτιμήσει, προσθέτει στη συνάρτηση χρησιμότητας κάθε κάρτας μια συγκεκριμένη τιμή. Η τιμή αυτή εξαρτάται από το βάθος του παίχτη και το

πλήθος των παιχτών που παίζουν. Έτσι μπορεί να υπολογίσει, κατά μέσο όρο, πόσες κάρτες πέρασαν μέχρι να βρεθεί μια για να του απορρίψει την υπόθεση. Αυτή την τιμή προσθέτει στις συναρτήσεις χρησιμότητας των καρτών (αλγόριθμος 5.25).

Αλγόριθμος 5.25: Εκτίμηση ενδεχομένων ενός παίχτη μέτριου ή δύσκολου επιπέδου στην απάντηση της υπόθεσής του

```
//change the utility to the other two
for(int i=0; i<suggestion.Length; i++){

 //for each card in suggestion and for each number of players
 if(GameObject.Find("cluedo_house").GetComponent<GameController>().↩
     enabledPlayers.Length == 3){
  if(suspects.Contains(suggestion[i]))
   suspectsUtil[suspects.IndexOf(suggestion[i])] += (depth * 6f)/2;
  else if(weapons.Contains(suggestion[i]))
   weaponsUtil[weapons.IndexOf(suggestion[i])] += (depth * 6f)/2;
  else if (rooms.Contains(suggestion[i]))
   roomsUtil[rooms.IndexOf(suggestion[i])] += (depth * 6f)/2;
 }
 else if(GameObject.Find("cluedo_house").GetComponent<GameController>().↩
     enabledPlayers.Length == 4){
  if(suspects.Contains(suggestion[i]))
   suspectsUtil[suspects.IndexOf(suggestion[i])] += (depth * 4.5f)/2;
  else if(weapons.Contains(suggestion[i]))
   weaponsUtil[weapons.IndexOf(suggestion[i])] += (depth * 4.5f)/2;
  else if (rooms.Contains(suggestion[i]))
   roomsUtil[rooms.IndexOf(suggestion[i])] += (depth * 4.5f)/2;
 }
 else if(GameObject.Find("cluedo_house").GetComponent<GameController>().↩
     enabledPlayers.Length == 5){
  if(suspects.Contains(suggestion[i]))
   suspectsUtil[suspects.IndexOf(suggestion[i])] += (depth * 3.6f)/2;
  else if(weapons.Contains(suggestion[i]))
   weaponsUtil[weapons.IndexOf(suggestion[i])] += (depth * 3.6f)/2;
  else if (rooms.Contains(suggestion[i]))
   roomsUtil[rooms.IndexOf(suggestion[i])] += (depth * 3.6f)/2;
 }
 else if(GameObject.Find("cluedo_house").GetComponent<GameController>().↩
     enabledPlayers.Length == 6){
  if(suspects.Contains(suggestion[i]))
   suspectsUtil[suspects.IndexOf(suggestion[i])] += (depth * 3f)/2;
  else if(weapons.Contains(suggestion[i]))
   weaponsUtil[weapons.IndexOf(suggestion[i])] += (depth * 3f)/2;
  else if (rooms.Contains(suggestion[i]))
   roomsUtil[rooms.IndexOf(suggestion[i])] += (depth * 3f)/2;
 }
```

```
}
```

- **Κοινοποίηση Υπόθεσης:** Όπως ήδη έχουμε αναφέρει, όταν ένας παίχτης κάνει μια υπόθεση την κοινοποιεί στους άλλους παίχτες. Έτσι, κάθε παίχτης χρησιμοποιεί τις υποθέσεις των άλλων παιχτών για να εκτιμήσει τα ενδεχόμενά του. Στην περίπτωση που ο παίχτης είναι εύκολου επιπέδου δεν εκτιμάει κανένα ενδεχόμενο. Το μόνο που κάνει είναι να αφαιρεί μια κάρτα από τα ενδεχόμενά του σε περίπτωση που κάποιος άλλος παίχτης έκανε μια υπόθεση και ο πρώτος γνωρίζει τις δύο από αυτές επειδή τις έχει στο χέρι του και ένας τρίτος έχει απορρίψει την υπόθεση του δεύτερου.

  Όταν ο παίχτης είναι μέτριου επιπέδου εκτιμάει τα ενδεχόμενα με τον παρακάτω τρόπο (αλγόριθμος 5.26):

  - Άμα γνωρίζει δύο από τις κάρτες που περιέχονται στην υπόθεση του άλλου παίχτη, αφαιρεί τιμή 3 από την άγνωστη κάρτα σε περίπτωση που αυτός απέρριψε την υπόθεση ή 5 σε περίπτωση που δεν την απέρριψε κανένας.
  - Άμα γνωρίζει μια κάρτα από αυτές που περιέχονται στην υπόθεση του άλλου παίχτη, αφαιρεί τιμή 1 από κάθε κάρτα σε περίπτωση που την υπόθεση την απέρριψε κάποιος τρίτος, 2 σε περίπτωση που την απέρριψε αυτός ή 5 σε περίπτωση που δεν την απέρριψε κανένας.
  - Άμα δεν γνωρίζει καμία από τις κάρτες που περιέχονται στην υπόθεση του άλλου παίχτη, αφαιρεί τιμή 1 από κάθε κάρτα σε περίπτωση που απέρριψε την υπόθεση κάποιος τρίτος ή 5 στην περίπτωση που δεν την απέρριψε κανένας.

Αλγόριθμος 5.26: Εκτίμηση ενδεχομένων ενός παίχτη μέτριου επιπέδου στην υπόθεση ενός άλλου παίχτη

```
        //for each case change utilities or remove cards
        if (numOfCards == 2){
            if (IShow == "Other"){
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil.RemoveAt(suspects.IndexOf(suggestion[i↩
                            ]));
                        suspects.Remove(suggestion[i]);
```

```
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil.RemoveAt(weapons.IndexOf(suggestion[i]))↩
                            ;
                        weapons.Remove(suggestion[i]);
                    }
                    else if (rooms.Contains(suggestion[i]))  {
                        roomsUtil.RemoveAt(rooms.IndexOf(suggestion[i]));
                        rooms.Remove(suggestion[i]);
                    }
                }
            }
            else if (IShow == "Me") {
                for (int i = 0; i < suggestion.Length; i++) {
                    if (suspects.Contains(suggestion[i]))  {
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 3f;
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 3f;
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 3f;
                    }
                }
            }
            else {
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 5f;
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 5f;
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 5f;
                    }
                }
            }
        }
        else if (numOfCards == 1){
            if (IShow == "Other"){
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 1f;
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 1f;
                    }
                    else if (rooms.Contains(suggestion[i])) {
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 1f;
                    }
```

```
                }
            }
            else if (IShow == "Me") {
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 2f;
                    }
                    else if (weapons.Contains(suggestion[i])) {
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 2f;
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 2f;
                    }
                }
            }
            else{
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 5f;
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 5f;
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 5f;
                    }
                }
            }

        }
        else if (numOfCards == 0){
            if (IShow == "Other"){
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 1f;
                    }
                    else if (weapons.Contains(suggestion[i])){
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 1f;
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 1f;
                    }
                }
            }
            else {
                for (int i = 0; i < suggestion.Length; i++){
                    if (suspects.Contains(suggestion[i])){
                        suspectsUtil[suspects.IndexOf(suggestion[i])] -= 5f;
                    }
                    else if (weapons.Contains(suggestion[i])) {
                        weaponsUtil[weapons.IndexOf(suggestion[i])] -= 5f;
```

```
                    }
                    else if (rooms.Contains(suggestion[i])){
                        roomsUtil[rooms.IndexOf(suggestion[i])] -= 5f;
                    }
                }
            }
        }
    }
```

Όταν ο παίχτης είναι δύσκολου επιπέδου εκτιμάει τα ενδεχόμενα με τον ίδιο τρόπο, όπως στην περίπτωση του μέτριου επιπέδου, μόνο που αντί να αφαιρεί προσθέτει τις τιμές.

Επίσης στις περιπτώσεις του μέτριου και δύσκολου επιπέδου, ο παίχτης πέρα των ενδεχομένων, εκτιμά και τις κάρτες που έχει στο χέρι του, που χρησιμοποιεί για να απορρίψει κάποια υπόθεση ενός άλλου παίχτη. Όταν ένας παίχτη κάνει μια υπόθεση, κατά την κοινοποίησή της, οι υπόλοιποι παίχτες αυξάνουν κατά 1 την τιμή στη συνάρτηση χρησιμότητας των καρτών που έχουν στα χέρια τους και ανήκουν στην υπόθεση που έγινε. Για αυτό το λόγο, ένας παίχτης, κατά την εμφάνιση μιας κάρτας για την απόρριψη της υπόθεσης ενός άλλου παίχτη, χρησιμοποιεί την κάρτα αυτή με τη μεγαλύτερη τιμή· γιατί είναι αυτή που έχει χρησιμοποιηθεί στις περισσότερες υποθέσεις.

Η φιλοσοφία που χρησιμοποιήθηκε στις παραπάνω συναρτήσεις χρησιμότητας υλοποιήθηκε μετά από σκέψη και σχεδιασμό. Εκτιμήθηκε ότι για την απάντηση της υπόθεσης ενός παίχτη, πρέπει τα ενδεχόμενα που δεν απορρίφθηκαν να έχουν μεγαλύτερη σημασία για τον παίχτη. Έτσι τους δόθηκε μια αύξηση στη συνάρτηση χρησιμότητάς τους, ώστε να χρησιμοποιηθούν σε επόμενη υπόθεση. Επίσης θεωρήσαμε ότι μεγάλο ρόλο παίζει και κατά πόσο εύκολα απορρίφθηκε μια υπόθεση. Μια υπόθεση που απορρίφθηκε από τον πρώτο παίχτη για παράδειγμα, έχει μικρότερη αξία από μια υπόθεση που απορρίφθηκε στον τρίτο παίχτη· γιατί οι κάρτες των παιχτών που έχουν προσπελαστεί είναι περισσότερες οπότε και τα ενδεχόμενα που δεν απορρίφθηκαν πιο πιθανά να ανήκουν στη λύση του μυστηρίου.

Αντίστοιχη μελέτη έγινε και στη συνάρτηση χρησιμότητας όταν κοινοποιείται μια υπόθεση στους άλλους παίχτες. Μελετήθηκαν τέσσερα διαφορετικά μοντέλα συναρτήσεων. Τα δύο πρώτα είναι αυτά που χρησιμοποιήθηκαν στους παίχτες του μέτριου και του δύσκολου

επιπέδου. Τα άλλα δύο, είχαν την ίδια λογική με την μόνη διαφορά ότι πρόσθεταν ή αφαιρούσαν τιμή 1 αντίστοιχα, σε όλες τις παραπάνω περιπτώσεις που περιγράψαμε. Τα μοντέλα αυτά υλοποιήθηκαν σε ένα ξεχωριστό project και προσομοιώθηκαν για χίλια παιχνίδια και τέσσερις διαφορετικούς παίχτες (ένας για κάθε μοντέλο). Επίσης υπολογίσθηκε ο μέσος όρος των υποθέσεων που έγιναν από όλους τους παίχτες πριν βρει ο συγκεκριμένος παίχτης τη λύση (πίνακας 5.1). Σκοπός μας ήταν να δούμε ποιά μοντέλα ήταν αυτά που κέρδιζαν τις περισσότερες φορές. Είδαμε μια αρκετά αισθητή διαφορά ανάμεσα στο πρώτο και στο δεύτερο μοντέλο που χρησιμοποιήσαμε στην υλοποίησή μας.

Πίνακας 5.1: Αποτελέσματα μοντέλων για τις συναρτήσεις χρησιμότητας

Μοντέλο	Νίκες (%)	Μ.Ο υποθέσεων
# 1	55.7	18.02693
# 2	17.9	19.49721
# 3	13.0	18.96923
# 4	13.4	18.65672

Κεφάλαιο 6

Αξιολόγηση Συστήματος

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε τον τρόπο με τον οποίο αξιολογήθηκε η εφαρμογή του παιχνιδιού από διάφορους χρήστες. Αναφέρουμε τα αποτελέσματα που πήραμε από την αξιολόγηση καθώς και τα συμπεράσματα που προέκυψαν.

6.2 Μέθοδος Αξιολόγησης

Η μέθοδος που χρησιμοποιήθηκε για να αξιολογηθεί το παιχνίδι μας είναι αυτή των ερωτηματολογίων χρηστικότητας QUIS (Questionnaire for User Interaction Satisfaction) [64]. Οι χρήστες που δοκίμασαν το παιχνίδι απάντησαν σε ερωτήσεις που αφορούσαν την εμφάνιση, τη διαδραστικότητα, την λειτουργικότητα, την εκμάθηση, τα πολυμέσα καθώς και την εγκατάστασή του.

Στη διαδικασία αξιολόγησης συμμετείχαν 10 χρήστες (7 άντρες και 3 γυναίκες), ηλικίας από 23 έως 30 ετών, διαφορετικών επαγγελμάτων, επιπέδου μόρφωσης και εξοικείωσης με ψηφιακές εφαρμογές.

Το ερωτηματολόγιο περιελάμβανε 36 ερωτήσεις (προσαρμοσμένο στη δική μας εφαρμογή), χωρισμένες σε 7 διαφορετικές θεματικές ενότητες. Οι απαντήσεις δίνονταν με κλίμακα από το 1 έως το 9, συμπεριλαμβανομένης της απάντησης "δεν ξέρω-δεν απαντώ", και ο χρήστης είχε επιπλέον το δικαίωμα σε κάθε ενότητα να παραθέσει κάποιο σχόλιο. Παρατίθεται το

ερωτηματολόγιο που χρησιμοποιήθηκε:

Ερωτηματολόγιο Αξιολόγησης Διαλογικής Χρήσης Συστήματος
(QUIS - Πανεπιστήμιο του Maryland, 1997)

Αριθμός ερ/γίου: _____ Σύστημα: _____ Ηλικία: _____
Φύλο: Άρρεν ____ Θήλυ ____

ΜΕΡΟΣ 1: Γενική εντύπωση του χρήστη

Παρακαλώ κυκλώστε τους αριθμούς που αντικατοπτρίζουν ακριβέστερα τις εντυπώσεις σας από τη χρήση αυτού του υπολογιστικού συστήματος. Δεν ξέρω/δεν απαντώ = NA.

1.1 Γενική αντίδραση του συστήματος:

απαράδεκτη υπέροχη
1 2 3 4 5 6 7 8 9 NA

1.2 Γενική αντίδραση του συστήματος:

μπερδεύει ικανοποιεί
1 2 3 4 5 6 7 8 9 NA

1.3 Γενική αντίδραση του συστήματος:

χαζό ευχάριστο
1 2 3 4 5 6 7 8 9 NA

1.4 Γενική αντίδραση του συστήματος:

δύσκολο εύκολο
1 2 3 4 5 6 7 8 9 NA

1.5 Γενική αντίδραση του συστήματος:

άκαμπτο ευέλικτο
1 2 3 4 5 6 7 8 9 NA

ΜΕΡΟΣ 2: Οθόνη

2.1 Ο σχεδιασμός της οθόνης βοήθησε:

2.1.1 Ο σχεδιασμός της οθόνης βοήθησε:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

2.1.2 Ποσότητα πληροφορίας που εμφανιζόταν στην οθόνη:

ανεπαρκής επαρκής
1 2 3 4 5 6 7 8 9 NA

2.1.3 Δόμηση της πληροφορίας που εμφανιζόταν στην οθόνη:

χαοτική οργανωμένη
1 2 3 4 5 6 7 8 9 NA

2.2 Αλληλουχία οθονών:

2.2.1 Αλληλουχία οθονών:

μπερδεμένη σαφής
1 2 3 4 5 6 7 8 9 NA

2.2.2 Επόμενη οθόνη στη σειρά:

απρόβλεπτη προβλέψιμη
1 2 3 4 5 6 7 8 9 NA

2.2.3 Επιστροφή στην προηγούμενη οθόνη:

αδύνατη εύκολη
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για τη δομή των οθονών εδώ:

ΜΕΡΟΣ 3: Ορολογία και επικοινωνία με το σύστημα

3.1 Τα μηνύματα εμφανίζονται στην οθόνη με:

3.1.1 Τα μηνύματα εμφανίζονται στην οθόνη με:

ασυνέπεια συνέπεια
1 2 3 4 5 6 7 8 9 NA

3.2 Τα μηνύματα που εμφανίζονται στην οθόνη είναι:

3.2.1 Τα μηνύματα που εμφανίζονται στην οθόνη είναι:

μπερδεμένα σαφή
1 2 3 4 5 6 7 8 9 NA

3.3 Ο υπολογιστής σας ενημερώνει για το τι κάνει:

3.3.1 Ο υπολογιστής σας ενημερώνει για το τι κάνει:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

3.3.2 Εκτελώντας μια κίνηση οδηγούμαστε σε προβλέψιμο αποτέλεσμα:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

3.3.3 Καθυστερήσεις:

απαράδεκτες αποδεκτές
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την ορολογία και την επικοινωνία με το σύστημα:

ΜΕΡΟΣ 4: Εκμάθηση χρήσης

4.1 Η εκμάθηση χρήσης του συστήματος είναι:

4.1.1 Η εκμάθηση χρήσης του συστήματος είναι:

δύσκολη εύκολη
1 2 3 4 5 6 7 8 9 NA

4.1.2 Χρόνος για την εκμάθηση χρήσης του συστήματος είναι:

λίγος πολύς
1 2 3 4 5 6 7 8 9 NA

4.2 Η εξερεύνηση των δυνατοτήτων με τη μέθοδο "προσπάθειας και λάθους (trial and error)":

4.2.1 Η εξερεύνηση των δυνατοτήτων με τη μέθοδο "προσπάθειας και λάθους (trial and error)":

απογοητεύει αποδίδει
1 2 3 4 5 6 7 8 9 NA

4.3 Οι εργασίες γίνονται σε λογική αλληλουχία:

4.3.1 Οι εργασίες γίνονται σε λογική αλληλουχία:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

4.3.2 Τα βήματα για την ολοκλήρωση της εργασίας ακολουθούν μια λογική σειρά:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την εκμάθησης της χρήσης εδώ:

ΜΕΡΟΣ 5: Δυνατότητες του συστήματος

5.1 Η ταχύτητα του συστήματος είναι:

5.1.1 Η ταχύτητα του συστήματος είναι:

ικανοποιητική πολύ αργή
1 2 3 4 5 6 7 8 9 NA

5.2 Το σύστημα είναι σταθερό:

5.2.1 Το σύστημα είναι σταθερό:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

5.2.2 Χειρισμοί-Λειτουργίες:

αναξιόπιστα αξιόπιστα
1 2 3 4 5 6 7 8 9 NA

5.3 Η ευκολία χειρισμού εξαρτάται από την εμπειρία του χρήστη:

5.3.1 Η ευκολία χειρισμού εξαρτάται από την εμπειρία του χρήστη:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για τις δυνατότητες του συστήματος εδώ:

_____

_____

_____

ΜΕΡΟΣ 6: Πολυμέσα

6.1 Η ποιότητα των εικόνων/φωτογραφιών είναι:

κακή καλή
1 2 3 4 5 6 7 8 9 NA

6.1.1 Εικόνες/Φωτογραφίες:

θολές καθαρές
1 2 3 4 5 6 7 8 9 NA

6.1.2 Η φωτεινότητα της εικόνας/φωτογραφίας:

σκοτεινή φωτεινή
1 2 3 4 5 6 7 8 9 NA

6.2 Η ηχητική απόδοση είναι:

6.2.1 Η ηχητική απόδοση είναι:

ασταθής εύρυθμη
1 2 3 4 5 6 7 8 9 NA

6.2.2 Η ηχητική απόδοση είναι:

αλλοιωμένη ξεκάθαρη
1 2 3 4 5 6 7 8 9 NA

6.3 Τα χρώματα που χρησιμοποιούνται είναι:

αφύσικα φυσικά
1 2 3 4 5 6 7 8 9 NA

6.3.1 Εικόνες/Φωτογραφίες:

ανεπαρκής επαρκής
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για τα πολυμέσα εδώ:

ΜΕΡΟΣ 7: Εγκατάσταση συστήματος

7.1 Η ταχύτητα της εγκατάστασης του συστήματος είναι:

αργή γρήγορη
1 2 3 4 5 6 7 8 9 NA

7.2 Η εξατομίκευση είναι:

δύσκολη εύκολη
1 2 3 4 5 6 7 8 9 NA

7.3 Πληροφορείται ο χρήστης για την πρόοδό της:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

7.4 Δίνει εποικοδομητικές εξηγήσεις όταν συμβαίνουν αποτυχίες:

ποτέ πάντα
1 2 3 4 5 6 7 8 9 NA

Παρακαλώ γράψτε τα σχόλιά σας για την εγκατάσταση του συστήματος εδώ:

6.3 Στόχος και Αποτελέσματα

Ο στόχος της παραπάνω αξιολόγησης ήταν πρώτα από όλα να εξαχθούν κάποια αποτελέσματα όσον αφορά τη λειτουργία, τις δυνατότητες και τα γραφικά του παιχνιδιού και δεύτερον να βγουν κάποια συμπεράσματα που θα μας βοηθήσουν σε τυχόν μελλοντική του βελτίωση.

Τα αποτελέσματα που εξήχθησαν φαίνονται στους παρακάτω πίνακες. Κάθε πίνακας περιέχει στις ερωτήσεις κάθε μέρους, τον μέσο όρο των βαθμολογιών που προέκυψαν από τα ερωτηματολόγια με άριστα το 9.

Πίνακας 6.1: Αποτελέσματα ερωτηματολογίου: Μέρος 1

Ερώτηση	Μ.Ο βαθμολογίας
1.1	7.9
1.2	7.8
1.3	7.7
1.4	7.2
1.5	7.1

Πίνακας 6.2: Αποτελέσματα ερωτηματολογίου: Μέρος 2

Ερώτηση	Μ.Ο βαθμολογίας
2.1.1	7.4
2.1.2	8.5
2.1.3	8.6
2.2.1	8.6
2.2.2	8.6
2.2.3	7.5

Πίνακας 6.3: Αποτελέσματα ερωτηματολογίου: Μέρος 3

Ερώτηση	Μ.Ο βαθμολογίας
3.1.1	8.8
3.2.1	8.9
3.3.1	9
3.3.2	8.8
3.3.3	8.2

Πίνακας 6.4: Αποτελέσματα ερωτηματολογίου: Μέρος 4

Ερώτηση	Μ.Ο βαθμολογίας
4.1.1	7.7
4.1.2	2.7
4.2.1	8.7
4.3.1	8.8
4.3.2	8.8

Πίνακας 6.5: Αποτελέσματα ερωτηματολογίου: Μέρος 5

Ερώτηση	Μ.Ο βαθμολογίας
5.1.1	2.2
5.2.1	7.7
5.2.2	7.6
5.3.1	4.3

Πίνακας 6.6: Αποτελέσματα ερωτηματολογίου: Μέρος 6

Ερώτηση	Μ.Ο βαθμολογίας
6.1	7.4
6.1.1	8.1
6.1.2	7.5
6.2.1	8.5
6.2.2	8.4
6.3	8
6.3.1	7.9

Πίνακας 6.7: Αποτελέσματα ερωτηματολογίου: Μέρος 7

Ερώτηση	Μ.Ο βαθμολογίας
7.1	8.8
7.2	8.4
7.3	8.4
7.4	8.7

6.4 Συμπεράσματα

Βάσει των βαθμολογιών των αποτελεσμάτων, καθώς και των σχολίων που υπήρξαν στα ερωτηματολόγια των χρηστών που δοκίμασαν το παιχνίδι, εξήχθησαν τα παρακάτω συμπεράσματα:

- **Μέρος 1:** Οι χρήστες σε γενικές γραμμές έμειναν αρκετά ικανοποιημένοι από την εφαρμογή. Θεώρησαν αρκετά καλή την απόδοση του παιχνιδιού καθώς και ότι ήταν διασκεδαστικό και ευχάριστο. Επίσης, εκτίμησαν ότι ήταν ένα παιχνίδι ικανοποιητικής δυσκολίας, χωρίς να είναι ανέφικτο κάποιος να κερδίσει και ταυτόχρονα όχι υπερβολικά εύκολο, κάτι που θα το έκανε ανιαρό. Τέλος παρείχε αρκετή ευελιξία, ώστε να μπορεί ο κάθε χρήστης να το εξατομικεύει βάσει των απαιτήσεών του.
- **Μέρος 2:** Από πλευρά οθόνης οι χρήστες ήταν ευχαριστημένοι σε μεγάλο βαθμό. Ο τρόπος σχεδιασμού της οθόνης τους βοήθησε αρκετά κατά τη διάρκεια του παιχνιδιού, καθώς και οι πληροφορίες που εμφανίζονταν ήταν επαρκείς και πλήρεις. Δεν υπήρξε κάτι που να τους μπέρδευε ιδιαίτερα στον τρόπο σχεδιασμού της οθόνης. Τα αρνητικά σχόλια που υπήρξαν όσον αφορά την οθόνη είναι ότι θα μπορούσαν τα σημεία του τοίχου που βρίσκονται μπροστά από τον παίκτη να είναι διαφανή, ώστε να μπορεί ο παίχτης να κινείται πιο εύκολα, καθώς και ότι θα μπορούσε να υπάρχει μια μπάρα στο πλάι της οθόνης που να ενημερώνει τον χρήστη ποιός παίχτης είναι ενεργός κάθε φορά.
- **Μέρος 3:** Η υψηλή βαθμολογία των χρηστών στο τρίτο μέρος, δείχνει ότι έμειναν αρκετά ικανοποιημένοι σε σχέση με την επικοινωνία και την διαδραστικότητα μεταξύ του συστήματος και του χρήστη. Τα μηνύματα ήταν αρκετά, σαφή και ενημέρωναν πλήρως τον χρήστη για το κάθε γεγονός που λάμβανε χώρα κατά τη διάρκεια του παιχνιδιού. Η ροή του παιχνιδιού δεν μπέρδευε και ήταν λογική, χωρίς να υπάρχνουν καθυστερήσεις. Αξιοσημείωτα σχόλια είναι ότι πρώτον, θα μπορούσε να υπάρχει μια επιλογή για ταχεία προσπέλαση (fast-forward) των γύρων των αντιπάλων ή παράβλεψή τους. Δεύτερον τα ενημερωτικά μηνύματα ήταν αρκετά, ώστε πολλές φορές να κουράζουν τον χρήστη με αποτέλεσμα να αποπροσανατολίζεται και να μην γνωρίζει που βρίσκεται ο κάθε παίχτης.
- **Μέρος 4:** Τα αποτελέσματα του τέταρτου μέρους έδειξαν ότι η εκμάθηση του χρήστη ήταν σχετικά εύκολη και απαιτούσε ελάχιστο χρόνο. Το μόνο πράγμα που ίσως να δυσκόλεψε τους χρήστες είναι το μεγάλο πλήθος κανόνων στο παιχνίδι. Όλες οι εργασίες στο παιχνίδι ακολούθησαν λογική σειρά. Είδαμε τέλος, ότι σε περίπτωση που οι χρήστες δεν γνώριζαν κάτι, η μέθοδος "προσπάθειας και λάθους (trial and error)" είχε αποτέλεσμα. Βάσει των σχολίων, η μέθοδος χρησίμευσε στον εντοπισμό του τρόπου που κινείται ο παίχτης, καθώς και στο πως λειτουργούν τα κουμπιά εναλλαγής (toggle buttons) στο σημειωματάριο (notebook) του χρήστη.

- **Μέρος 5:** Οι χρήστες, βάσει των βαθμολογιών τους, βρήκαν αρκετά γρήγορη την ταχύτητα του παιχνιδιού. Ήταν σε γενικές γραμμές ικανοποιημένοι τόσο από τον χειρισμό του, όσο και από την σταθερότητα του συστήματος. Θεωρήθηκε ότι η ευκολία χειρισμού του παιχνιδιού εξαρτάται κατά ένα ποσοστό στην εμπειρία του χρήστη. Αυτό οφείλεται στην καλή γνώση των κανόνων του παιχνιδιού, σε προγενέστερη εμπειρία του χρήστη με το επιτραπέζιο, καθώς και στην εξοικείωσή του με το ψηφιακό παιχνίδι. Εδώ δεν υπήρξαν ιδιαίτερα σχόλια των χρηστών.

- **Μέρος 6:** Αρκετά ευχαριστημένοι φάνηκαν οι χρήστες όσον αφορά και τα πολυμέσα του παιχνιδιού. Οι ήχοι και η απόδοσή τους ήταν εξαιρετική. Η ποιότητα των εικόνων ήταν αρκετά καλή όπως και η φωτεινότητα τους σε φυσιολογικό επίπεδο. Κάποια σχόλια μίλαγαν για μερικές, λίγο θολές εικόνες που θα μπορούσαν να βελτιωθούν. Τέλος η ποσότητα και η ποιότητα των χρωμάτων στα γραφικά ήταν αρκετά ικανοποιητική με σχόλια που αφορούσαν ίσως σε μια βελτίωση στη ρεαλιστικότητά τους.

- **Μέρος 7:** Τέλος η εγκατάσταση του παιχνιδιού ήταν εύκολη, γρήγορη χωρίς σφάλματα, μέσα από έναν εύχρηστο εγκαταστάτη (installer) που παρείχε επιλογές στον χρήστη.

Κεφάλαιο 7

Αποτελέσματα και Μελλοντικές Επεκτάσεις

7.1 Εισαγωγή

Στο παρόν και τελευταίο κεφάλαιο, γίνεται μια σύντομη αναφορά των όσων υλοποιήθηκαν στη διπλωματική αυτή εργασία. Παραθέτονται βελτιώσεις που έγιναν στην εφαρμογή μας, βάσει των αποτελεσμάτων που εξήχθησαν, και τέλος αναφέρονται τυχόν μελλοντικές επεκτάσεις που μπορούν να γίνουν στο παιχνίδι.

7.2 Στόχος και Αποτελέσματα

Όπως ήδη έχει αναφερθεί, στόχος της παρούσας διπλωματικής εργασίας ήταν η σχεδίαση μιας τρισδιάστατης ψηφιακής εφαρμογής, που θα προσομοιώνει το επιτραπέζιο παιχνίδι *Cluedo*. Το γραφικό περιβάλλον του παιχνιδιού σχεδιάστηκε εξ ολοκλήρου στο λογισμικό τρισδιάστατων γραφικών *Autodesk 3ds Max* και το παιχνίδι υλοποιήθηκε στην μηχανή παιχνιδιών *Unity 3D*. Τέλος στο παιχνίδι ενσωματώθηκαν αυτόνομοι αντίπαλοι, οι οποίοι υλοποιήθηκαν μέσω μιας τεχνητής νοημοσύνης.

Βάσει των αποτελεσμάτων των χρηστών και των δοκιμών που έγιναν τόσο από εμάς όσο και από τον δοκιμαστή του παιχνιδιού, εντοπίστηκαν κάποιες βελτιώσεις οι οποίες ήταν εφικτές να γίνουν στα πλαίσια της διπλωματικής εργασίας. Παρατίθενται οι βελτιώσεις:

- Λόγω του ιδιαίτερου χειρισμού του παίχτη και της κάμεράς του, ήταν κάποιες φορές

δύσκολο ο χρήστης να κινήσει τον παίχτη του μέσα στο σπίτι, ειδικά σε σημεία που ο διάδρομος ήταν στενός. Αυτό οφειλόταν στο κεντρικό δωμάτιο της καταπακτής, στο οποίο υποτίθεται ότι βρίσκεται η λύση του μυστηρίου. Αυτό το δωμάτιο στην ουσία είναι άχρηστο. Οπότε αφαιρέθηκε από την γεωμετρία μας και τοποθετήθηκε το λογότυπο του *Cluedo* (το οποίο προηγουμένως βρισκόταν στην οροφή αυτού του δωματίου) κάτω στο πάτωμα. Έτσι ο χειρισμός του παίχτη έγινε ακόμα πιο εύκολος στον χρήστη.

- Μετά από διαδοχικές εκτελέσεις του παιχνιδιού, ανακαλύφθηκε ότι ο παίχτης μπορεί να μπλοκαριστεί σε ένα δωμάτιο και να μην μπορέσει στο γύρο του να βγει από αυτό. Έτσι δημιουργήθηκε κατάλληλος αλγόριθμος που ελέγχει αν ο χρήστης είναι μπλοκαρισμένος σε κάποιο δωμάτιο και σε περίπτωση που είναι τον ειδοποιεί με κατάλληλο μήνυμα.

- Έγιναν βελτιώσεις σε κάποιες φωτογραφίες του γραφικής διεπαφής ώστε να είναι καλύτερης ποιότητας. Αυτό έγινε στα πλαίσια που μπορούσαμε, καθώς δεν είχαμε τις φωτογραφίες σε ψηφιακή μορφή και ήταν αποτέλεσμα σάρωσης αυτών του επιτραπέζιου παιχνιδιού.

- Προσπαθήσαμε να βελτιώσουμε τα χρώματα του σπιτιού και ιδιαίτερα αυτά των τοίχων, ώστε να είναι πιο ρεαλιστικά. Τοποθετήσαμε υφές που ήταν πιο φυσικές και χρησιμοποιήσαμε διαφορετικούς τόνους χρωμάτων για να αποδώσουμε τις σκιάσεις των τοίχων.

- Τέλος, όπως και σε όλα τα βιντεοπαιχνίδια, μέσα από δοκιμές εντοπίστηκαν σφάλματα στον κώδικα του παιχνιδιού, τα οποία ονομάζονται "μικροβλάβες" (glitches) και τα οποία διορθώθηκαν. Παρ΄ όλα αυτά είναι πολύ πιθανό, με το πέρασμα του χρόνου και των συνεχών δοκιμών του παιχνιδιού από χρήστες, να εντοπιστούν και άλλα.

7.3 Μελλοντικές Επεκτάσεις και Βελτιώσεις

Μέσα από τα σχόλια των χρηστών, καθώς και από τη μελέτη του παιχνιδιού κατά τη διάρκεια υλοποίησής του, μπορέσαμε και εντοπίσαμε κάποιες επεκτάσεις οι οποίες μπορούν να βελτιωθούν μελλοντικά στο παιχνίδι και παρατίθενται για περαιτέρω μελέτη:

- Θα μπορούσε να επεκταθεί η γραφική επαφή του χρήστη κατά τη διάρκεια του παιχνιδιού με προσθήκη μιας μπάρας στο πλάι της οθόνης, που θα ενημερώνει ποιός παίχτης παίζει κάθε φορά ή ποιός παίχτης καλείται να δείξει μια κάρτα για να απορρίψει μια υπόθεση.

- Θα μπορούσαν να αφαιρεθούν τα μηνύματα εισόδου και εξόδου ενός παίχτης από ένα δωμάτιο. Ο στόχος θα ήταν να μην κουράζουν το χρήστη με μια πληροφορία που ήδη δίνεται φωνητικά. Ο λόγος που τοποθετήθηκαν αυτά τα μηνύματα, εκτός από το να ενημερώνουν τον χρήστη, είχαν σκοπό να κρύβουν την κίνηση που κάνει ο χρήστης κατά την είσοδο και την έξοδό του από ένα δωμάτιο. Επειδή οι χρήστες έχουν πάγιες θέσεις εντός των δωματίων, μόλις βρεθούν μπροστά στην πόρτα του δωματίου, μεταφέρονται αυτομάτως στις θέσεις τους. Αν αφαιρούνταν τα μηνύματα θα φαινόταν άσχημο ο παίχτης εκεί που βρίσκεται σε ένα σημείο, να πετάγεται μονομιάς σε ένα άλλο.

  Για να αφαιρεθούν αυτά τα μηνύματα και ταυτοχρόνως να φαίνεται ωραία η κίνηση των παιχτών, πρέπει να δημιουργηθεί ένα πλέγμα (grid) σε κάθε δωμάτιο έτσι ώστε να μπορούν να κινηθούν οι παίχτες μέσα σε αυτό. Ταυτόχρονα θα πρέπει να εκτελείται ο αλγόριθμος της εύρεσης βέλτιστου μονοπατιού (path planning) και εντός των δωματίων. Παρ΄ όλα αυτά λόγω της στενότητας χώρου μέσα στα δωμάτια, δεν ξέρουμε κατά πόσο θα ήταν εφικτή μια τέτοια υλοποίηση.

- Στην αρχή του παιχνιδιού, θα μπορούσε να δημιουργηθεί ένα ενημερωτικό βίντεο, το οποίο θα καθοδηγούσε τον χρήστη για το πως να κινεί τον παίχτη του.

- Θα μπορούσαν να γίνουν αλλαγές στα γραφικά ώστε να μπορεί ο χρήστης να κινήσει πιο εύκολα τον παίχτη του. Επειδή η κάμερα έχει ήδη πολλές δυνατότητες, δεν μπορεί να επεκταθεί περαιτέρω. Έτσι η δημιουργία διαφανών τοίχων στο σημείο που βρίσκεται ο παίχτης θα διευκόλυνε τον χρήστη να βλέπει καλύτερα και να συνεπώς να μετακινήσει με μεγαλύτερη ευκολία τον παίχτη του.

- Θα μπορούσε το παιχνίδι να έχει περισσότερους του ενός χρήστη που παίζει είτε να παίζεται διαδικτυακά με πολλού χρήστες.

- Αν και η τεχνητή νοημοσύνη των αντιπάλων είναι ήδη αρκετά δυνατή, θα μπορούσαν οι αντίπαλοι να επιλέγουν πιο έξυπνα τα δωμάτια που θα κάνουν τις υποθέσεις τους

και να έχουν τη δυνατότητα, κάποιες φορές, να παραπλανούν με τις υποθέσεις τους, τους υπόλοιπους παίχτες.

7.4 Επίλογος

Κλείνοντας αυτή την διπλωματική εργασία, είναι σημαντικό να αναφερθεί ότι ήταν μια σημαντική ευκαιρία η γνωριμία εκ των έσω του τρόπου με τον οποίον υλοποιείται ένα τρισδιάστατο παιχνίδι για υπολογιστές. Μετά από κάτι τέτοιο, ο καθένας βλέπει και εκτιμάει διαφορετικά ένα βιντεοπαιχνίδι. Η διαδικασία τόσο του σχεδιασμού, όσο και της υλοποίησης του παιχνιδιού ήταν αρκετά δύσκολη και χρονοβόρα. Παρ΄ όλα αυτά η ευχαρίστηση που προσέφερε το παιχνίδι στους χρήστες, καθώς και η εξοικείωση με τα εργαλεία που χρησιμοποιούνται κυρίως για επαγγελματική χρήση, είναι ένα κέρδος.

Η υλοποίηση ενός παιχνιδιού και όλα αυτά με τα οποία σχετίζεται, δίνουν τη δυνατότητα στον σχεδιαστή για μελλοντική βελτίωση του παιχνιδιού, καθώς και για περαιτέρω εξέλιξη και πορεία στο χώρο των γραφικών υπολογιστών.-

Βιβλιογραφία

[1] Wikipedia: Cluedo — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Cluedo. 1

[2] Hasbro, Inc.: Clue Instructions (1986) http://www.hasbro.com/common/instruct/clueins.pdf. 2

[3] Unity: Unity — Physics (2012) http://docs.unity3d.com/Documentation/Manual/Physics.html. 7

[4] Wikipedia: Video game — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Video_game. 11

[5] Wikipedia: Video game console — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Video_game_console. 12

[6] Wikipedia: Arcade game — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Arcade_game. 12

[7] Wikipedia: Nim — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Nim. 13

[8] BNL: BNL|History:The First Video Game? (2013) http://www.bnl.gov/about/history/firstvideo.php. 13

[9] Wikipedia: Video game genres — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Video_game_genres. 14

[10] Wikipedia: List of 3d graphics libraries — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries. 19

[11] Wikipedia: Microsoft direct3d — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Direct3D. 20

[12] Wikipedia: Opengl — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/OpenGL. 20

[13] Wikipedia: X3d — Wikipedia, the free encyclopedia (2013) . 21

[14] Wikipedia: Comparison of opengl and direct3d — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D. 21

[15] Νικόλαος Φράγκος: Αναπτύξη τρισδιάστατου παιχνιδιού ανοιχτού περιβάλλοντος με αυτονομία αντιπάλων. Διπλωματική εργασία, Πολυτεχνείο Κρητης, Ελλάδα (2010) 22, 28, 39

[16] Θεοχάρης, Θ. και Μπεμ, Α.: Γραφικά Αρχές & Αλγόριθμοι. Εκδόσεις Συμμετρία (1999) 22

[17] Wikipedia: Computer graphics lighting — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Computer_graphics_lighting. 22

[18] Department of Computer Science — University of Illinois at Chicago: Intro to Computer Graphics: Lighting and Shading (2013) http://www.cs.uic.edu/~jbell/CourseNotes/ComputerGraphics/LightingAndShading.html. 22

[19] Wikipedia: Lightmap — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Lightmap. 24

[20] Wikipedia: Texture mapping — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Texture_mapping. 24

[21] Wikipedia: Bump mapping — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Bump_mapping. 24

[22] Wikipedia: Shader — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Shader. 24

[23] Wikipedia: List of common shading algorithms — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/List_of_common_shading_algorithms. 25

[24] Wikipedia: Animation — Wikipedia, the free encyclopedia (2013) http://el.wikipedia.org/wiki/Animation. 25

[25] Wikipedia: Game physics — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Game_physics. 26

[26] Wikipedia: 3d computer graphics software — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/3D_computer_graphics_software. 27

[27] Wikipedia: Autodesk 3ds max — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/3ds_Max. 27

[28] Wikipedia: Blender (software) — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Blender_%28software%29. 27

[29] Wikipedia: Autodesk maya — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Maya_%28software%29. 27

[30] Irrlicht: Irrlicht Engine — A free open source 3D engine (2013) http://irrlicht.sourceforge.net/. 28

[31] Ogre3D: About « OGRE — Open Source 3D Graphics Engine (2009) http://www.ogre3d.org/about. 29

[32] Wikipedia: List of game engines — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/List_of_game_engines. 29

[33] Wikipedia: Game engine — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Game_engine. 29

[34] Wikipedia: Integrated development environment — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Integrated_development_environment. 30

[35] Wikipedia: Unity (game engine) — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Unity_%28game_engine%29. 30

[36] Wikipedia: Unreal engine — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Unreal_Engine. 30

[37] Wikipedia: Cryengine — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/CryEngine. 31

[38] Yahoo!7 Answers: What does SDK stand for? — Yahoo!7 Answers (2010) http://au.answers.yahoo.com/question/index?qid=20100407191632AA3feCS. 31

[39] Wikipedia: Panda3d — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Panda3D. 31

[40] Wikipedia: Physics engine — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Physics_engine. 32

[41] Wikipedia: Physx — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/PhysX. 33

[42] Wikipedia: Artificial intelligence (video games) — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Artificial_intelligence_%28video_games%29. 34

[43] Wikipedia: Decision making — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Decision_making. 34

[44] Stuart Russell και Peter Norvig: Τεχνητή Νοημοσύνη Μια σύχρονη προσέγγιση. Κλειδάριθμος (2005) 35, 36

[45] Wikipedia: Pathfinding — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Pathfinding. 35

[46] Wikipedia: A* search algorithm — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/A*_search_algorithm. 35

[47] Department of Computer Science — Kent State Univeristy: Heap Sort (2013) http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/heapSort.htm. 39

[48] Microsoft: Array.Sort<T> Method (T[]) (System) (2013) http://msdn.microsoft.com/en-us/library/kwx6zbd4.aspx. 40

[49] Autodesk: 3ds Max Help: Extension for Autodesk 3ds Max 2013 (2013) http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-Help/index.html. 42

[50] 3dsmax-tutorial: List of Available Modifiers (2013) http://www.3dmax-tutorials.com/List_of_Available_Modifiers.html. 46

[51] Wikipedia: Vrml — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/VRML. 47

[52] Unity: Unity — Unity Manual (2012) http://docs.unity3d.com/Documentation/Manual/index.html. 48

[53] Unity: Unity — Reference Manual (2012) http://docs.unity3d.com/Documentation/Components/index.html. 49

[54] Wikipedia: Scripting language — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Scripting_language. 61

[55] Unity: Unity Script Reference: (2013) http://docs.unity3d.com/Documentation/ScriptReference/index.html. 62

[56] 3dlancer: 3dlancer.net — about 3D graphics (2011) http://3dlancer.net/. 70

[57] 3DModelFree: 3D Model Download,Free 3D Models Download (2013) http://www.3dmodelfree.com/. 70

[58] Archive3D: Free 35000+ 3D models. Download without registration (2007) http://archive3d.net/. 70

[59] Autodesk: Autodesk Project Pinocchio (2013) http://projectpinocchio.autodesk.com/. 73

[60] Wikipedia: Isometric projection — Wikipedia, the free encyclopedia (2013) http://en.wikipedia.org/wiki/Isometric_projection. 91

[61] Gamelix: Gamelix propels your ideas (2013) http://gamelix.com/resources/Unity3D%20Forum/AlphaSelfIllum.shader. 92

[62] Oddcast: Best Text-to-Speech Demo: Create Talking Avatars and Online Characters | SitePal TTS Demo (2013) http://www.oddcast.com/home/demos/tts/tts_example.php. 97

[63] Google: / - a-star-java - A* (A Star) algorithm implementation in java - Google Project Hosting (2013) http://code.google.com/p/a-star-java/source/browse/. 111

[64] Στάθης Μελέτιος: Ανάπτυξη διαδραστικού τρισδιάστατου παιχνιδιού βασισμένο στη φυσική. Διπλωματική εργασία, Πολυτεχνείο Κρήτης, Ελλάδα (2009) 127