ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Ανάπτυξη παιχνιδιού στρατηγικής με εκπαιδευτικό σκοπό

Φοιτητής: Αλέξανδρος Βεκίλογλου
ΑΜ: 46363

Επιβλέπων Καθηγητής

Δημήτριος Μετάφας, Επίκουρος Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΙΟΥΛΙΟΣ 2024

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Development of strategy game with educational purposes

Student: Alexandros Vekiloglou
Registration Number: 46363

Supervisor

Dimitrios Metafas
Assistant Professor

ATHENS-EGALEO, JULY 2024

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Δ. Μετάφας, Επίκουρος Καθηγητής	Μ. Ραγκούση, Καθηγήτρια	Αικ. Ζαχαριάδου, Καθηγήτρια
ΔΗΜΗΤΡΙΟΣ ΜΕΤΑΦΑΣ Digitally signed by ΔΗΜΗΤΡΙΟΣ ΜΕΤΑΦΑΣ Date: 2024.07.26 15:57:41 +03'00' (Υπογραφή)	MARIA RAGKOUSI Digitally signed by MARIA RAGKOUSI Date: 2024.07.26 17:52:43 +03'00' (Υπογραφή)	Aikaterini-Styliani Zachariadou Digitally signed by Aikaterini-Styliani Zachariadou Date: 2024.07.29 20:00:53 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Αλέξανδρος Βεκίλογλου του Ιωάννη, με αριθμό μητρώου 46363 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο Δηλών
Αλέξανδρος Βεκίλογλου

Υπογραφή φοιτητή

Περίληψη

Η πληροφορική έχει πλέον εισχωρήσει σε κάθε τομέα της ανθρώπινης δραστηριότητας. Πράγματι πλέον στη συντριπτική πλειονότητα των ενεργειών που εκτελούν οι άνθρωποι, η πληροφορική διαδραματίζει κάποιο ρόλο, είτε μικρό είτε και μεγαλύτερο. Και ο χώρος της εκπαίδευσης δεν θα μπορούσε να μείνει ανεπηρέαστος από τη τάση αυτή. Η μάθηση πλέον ενισχύεται από τα εργαλεία που η πληροφορική διαθέτει, διευκολύνοντας την αποστολή των εκπαιδευτικών αλλά και την εκπαιδευτική διαδικασία των μαθητών. Στη τελευταία αυτή διαδικασία σημαντικό ρόλο μπορούν να διαδραματίσουν τα παιχνίδια εκπαιδευτικού σκοπού.

Στόχο της παρούσας διπλωματικής εργασίας αποτελεί η σχεδίαση και ανάπτυξη ενός ψηφιακού εκπαιδευτικού παιχνιδιού σε τρισδιάστατο ψηφιακό περιβάλλον. Τα εκπαιδευτικά παιχνίδια έχει διαπιστωθεί ότι συμβάλλουν σημαντικά στην εκπαιδευτική διαδικασία, με έναν ευχάριστο και ελκυστικό για τους εκπαιδευόμενους τρόπο. Για το σκοπό αυτό θα πραγματοποιηθεί μια ανασκόπηση πεδίου σχετικά με το πώς μπορεί να συμβάλει στην εκπαιδευτική διαδικασία το παιχνίδι γενικά αλλά και πιο συγκεκριμένα το παιχνίδι στρατηγικής. Στη συνέχεια θα παρουσιαστεί το παιχνίδι στρατηγικής 7 Wonders το οποίο θα αποτελέσει και το αντικείμενο εστιασμού της παρούσας εργασίας καθώς μια έκδοση αυτού θα αναπτυχθεί στα πλαίσια της εργασίας αυτής. Η μηχανή η οποία θα χρησιμοποιηθεί για τη δημιουργία του παιχνιδιού είναι η Unity, η οποία θα παρουσιαστεί στο επόμενο κεφάλαιο της εργασίας. Ακολούθως θα παρουσιαστεί η ανάπτυξη της εφαρμογής και στη συνέχεια μια ενδεικτική παρουσίαση της εκτέλεσης της. Στο τελευταίο κεφάλαιο θα παρουσιαστούν τα αποτελέσματα της εργασίας, οι περιορισμοί καθώς και τα συμπεράσματα της έρευνας.

Ο συγγραφέας ευελπιστεί ότι μέσα από την ανάγνωση της εργασίας αυτής ο αναγνώστης θα αποκτήσει δύο βασικές δεξιότητες. Η πρώτη είναι η διαδικασία ανάπτυξης εφαρμογής μέσω της μηχανής Unity όπου ο χρήστης θα εξοικειωθεί τόσο με την χρήση της μηχανής και της γλώσσας προγραμματισμού C#, που αυτή χρησιμοποιεί, αλλά και με την κατασκευή συνθετότερων τρισδιάστατων αντικειμένων, μέσω της εφαρμογής Blender, διαδικασίας που η Unity από μόνη της δεν είναι ικανή να διεκπεραιώσει. Η άλλη δεξιότητα που θα αποκτήσει ο αναγνώστης είναι η σημασία του παιχνιδιού και πιο συγκεκριμένα του ψηφιακού παιχνιδιού στρατηγικής στην εκπαιδευτική διαδικασία. Έτσι, είτε ο αναγνώστης είναι μαθητής, είτε είναι εκπαιδευτικός, είτε γονιός, είτε απλά ένα άτομο που έρχεται σε επαφή με άτομα που εκπαιδεύονται, θα αποχτήσει ένα σημαντικό εργαλείο που θα διευκολύνει σε μεγάλο βαθμό το έργο του.

Λέξεις – κλειδιά

Εκπαιδευτικό παιχνίδι, παιχνίδι στρατηγικής, Blender, Unity 3D, C#, μηχανή παιχνιδιών, ανάπτυξη λογισμικού

Abstract

Computing has now permeated every field of human activity. Indeed, in most human activities, IT plays some role, sometimes small, others large. And the field of education could not remain unaffected by this trend. Learning is now enhanced by the tools that IT has available, facilitating the mission of teachers and the educational process of students. Educational games can play a significant role in this last process.

This thesis aims to study the effect of the digital educational game on the educational process. For this purpose, a field review will be carried out on how the game in general and more specifically the strategy game can contribute to the educational process. Then the strategy game 7 Wonders will be presented which will be the focus of this work, as a version of it will be developed within the framework of this thesis. The engine used to create the game is Unity, presented in the next chapter of the thesis. Next, the development of the application will be presented, followed by an indicative presentation of its execution. The last chapter will present the results of this thesis, the limitations and the research conclusions.

The author hopes that through reading this thesis the reader will acquire two basic skills. The first is the application development process through the Unity engine, where the user will become familiar both with the use of the engine and the C# programming language it uses, but also with the construction of more complex 3D objects, through the Blender application, a process that Unity alone it is not able to process. The other skill that the reader will acquire is the importance of the game and more specifically the digital strategy game in the educational process. Thus, whether the reader is a student, an educator, a parent, or simply a person who meets people being educated, he will gain a valuable tool that will facilitate his work.

Keywords

Education game, strategy game, Blender, Unity 3D, C#, game engine, software development

Περιεχόμενα

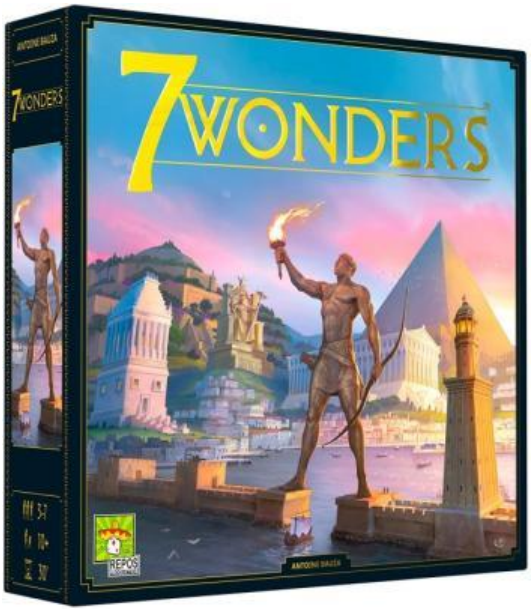
1 ΚΕΦΑΛΑΙΟ 1ο : Εισαγωγή σα παιχνίδια και την εκπαίδευση

Στο κεφάλαιο αυτό αρχικά εισάγεται το αντικείμενο, ο στόχος, η δομή και η καινοτομία της παρούσας διπλωματικής εργασίας.
Στη συνέχεια, επιχειρείται μια εισαγωγή στο θέμα παρουσιάζοντας στον αναγνώστη εισαγωγικές έννοιες σε μια προσπάθεια να αποκτήσει μια σφαιρική άποψη για το θέμα. Πιο συγκεκριμένα θα παρουσιαστεί ο όρος του παιχνιδιού, κάποια ιστορικά στοιχεία καθώς και οι διάφορες μορφές του. Έτσι ο αναγνώστης θα αποκτήσει ένα θεωρητικό υπόβαθρο για το θέμα το οποίο θα τον βοηθήσει να αντιληφθεί ευκολότερα τα αντικείμενα που θα παρουσιαστούν στην πορεία.

1.1 Η παρούσα διπλωματική εργασία

Στην παρούσα διπλωματική εργασία αντικείμενο είναι η σχεδίαση και ανάπτυξη σε τρισδιάστατο υπολογιστικό περιβάλλον ενός κλασσικού (μη ψηφιακού) επιτραπέζιου παιχνιδιού με εκπαιδευτικό χαρακτήρα.

Συγκεκριμένα, θα δημιουργηθεί μια ψηφιακή έκδοση του δημοφιλούς παιχνιδιού στρατηγικής 7 Wonders. Πρόκειται για ένα επιτραπέζιο παιχνίδι το οποίο δημιουργήθηκε από το Γάλλο σχεδιαστή παιχνιδιών Antoine Bauza το 2010 και κυκλοφόρησε από την εταιρία Repos Production. Έκτοτε έχει λάβει ιδιαίτερα μεγάλη αποδοχή από το καταναλωτικό κοινό, πολύ καλές κριτικές αλλά και διεθνή βραβεία όπως το Kennerspiel des Jahres το 2011.



Εικόνα 1: Το παιχνίδι 7 Wonders. Πηγή: Repos Production

Το παιχνίδι αναπαριστά αρχαίους πολιτισμούς και βασίζεται σε κάρτες μέσω των οποίων οι παίχτες χτίζουν και εξοπλίζουν τις πόλεις τους. Στο τέλος του κάθε γύρου, ο κάθε παίχτης αναμετριέται στρατιωτικά με τους γειτονικούς του και κερδίζει πόντους στις νίκες, ενώ οι ήττες του αφαιρούν πόντους. Παράλληλα η δημιουργία ορισμένων κατασκευών επιφέρει επιπλέον πόντους στο παίχτη

που τις κατασκευάζει. Στο τέλος του παιχνιδιού, ο παίχτης με τους περισσότερους πόντους, πρώτα στρατιωτικούς και ακολούθως κατασκευαστικούς, κερδίζει το παιχνίδι.

Πρόκειται για ένα παιχνίδι στρατηγικής καθώς ο παίχτης θα πρέπει να καταστρώσει ένα σχέδιο και να το ακολουθήσει ώστε να καταφέρει να κερδίσει τους αντιπάλους του. Πιο συγκεκριμένα ο παίχτης θα πρέπει να διαχειριστεί τους διαθέσιμους πόρους που του παρέχουν οι κάρτες που τυχαία παίρνει όσο πιο αποδοτικά ώστε να γίνει ισχυρότερος από τους αντιπάλους του και να καταφέρει να τους κερδίσει τόσο στο πεδίο της μάχης άλλα όσο και στο κατασκευαστικό. Παράλληλα διαθέτει και χρήματα είτε από τις κάρτες που λαμβάνει – είτε από την πώληση των πόρων που διαθέτει, τα οποία μπορεί να χρησιμοποιήσει για να αγοράσει τους πόρους που χρειάζεται από τους γείτονες του. Φυσικά δίνοντας χρήματα στους γείτονες του τους επιτρέπει να ενισχυθούν και αυτοί με τη σειρά τους οπότε και αυτό θα πρέπει να γίνει κατόπιν σκέψης.

Δεδομένου ότι το παιχνίδι αποτελεί ένα παιχνίδι στρατηγικής, εύκολα ο αναγνώστης καταλήγει στο συμπέρασμα ότι πρόκειται για εκπαιδευτικό παιχνίδι καθώς, όπως αναφέρθηκε και σε προηγούμενη παράγραφο, τα παιχνίδια στρατηγικής ωθούν τους παίχτες να σκέφτονται προτού αντιδράσουν. Στο παιχνίδι, ο παίχτης, μόλις πάρει μια κάρτα έχει τον απαιτούμενο χρόνο να σκεφτεί και να επιλέξει ποια χρήση αυτής θα ήταν η πιο συμφέρουσα και ακολούθως να την πραγματοποιήσει. Συνεπώς το παιχνίδι οξύνει την κριτική σκέψη, τη λήψη αποφάσεων και τον σχεδιασμό μελλοντικών ενεργειών, εξασκώντας με τον τρόπο αυτό δεξιότητες όπως την επίλυση προβλημάτων, τη στρατηγική σκέψη, τη διαχείριση των διαθέσιμων πόρων καθώς και άλλους.

Πέρα όμως από τα γενικά οφέλη που παρέχουν τα παιχνίδια στρατηγικής το συγκεκριμένο, επειδή έχει ως θέμα τα 7 θαύματα του κόσμου, παρουσιάζει στους παίκτες του και σημαντικά ιστορικά θέματα. Πιο συγκεκριμένα οι παίχτες θα γνωρίσουν τα 7 θαύματα του κόσμου, τους πολιτισμούς οι οποίοι τα δημιούργησαν, την ιστορία τους, την τοποθεσία τους καθώς και πολλά άλλα. Με τον τρόπο αυτό οι παίχτες συνδέουν τη γνώση που απέκτησαν στο σχολείο μέσα από τα διάφορα βιβλία ιστορίας με το παρόν, μέσα από ένα διασκεδαστικό και ευχάριστο παιχνίδι. Έτσι, τα ιστορικά στοιχεία που λαμβάνουν, αποκτούν ένα νόημα και γίνονται τμήμα της καθημερινότητας τους.

Η δομή της παρούσας εργασίας είναι η εξής:

Στο υπόλοιπο του 1ου κεφαλαίου παρουσιάζεται μια σύντομη ανάλυση του παιχνιδιού ως έννοια και οι κατηγορίες των ηλεκτρονικών παιχνιδιών.

Το 2ο κεφάλαιο επικεντρώνεται στην περιγραφή του επιτραπέζιου 7 wonders από τα περιεχόμενα και το στήσιμό του, μέχρι και τον τρόπο με τον οποίο παίζεται.

Στο 3ο κεφάλαιο βρίσκεται η παρουσίαση βασικών μηχανών δημιουργίας βιντεοπαιχνιδιών και η περιγραφή της unity συνοδευόμενη από έναν σύντομο οδηγό εγκατάστασης και μια γνωρημία με το περιβάλλον της.

Στο 4ο κεφάλαιο παρουσιάζεται η αρχιτεκτονική του κώδικα σε C# που γράφτηκε για το παιχνίδι.

Στο 5ο κεφάλαιο, ο αναγνώστης, θα βρει την ανάπτυξη της εφαρμογής, συμπεριλαμβανομένης της κατασκευής και της εισαγωγής των μοντέλων στη unity και του στησίματος του παιχνιδιού.

Στο 6ο κεφάλαιο παρουσιάζεται η εκτέλεση του παιχνιδιού στο περιβάλλον της unity.

Η εργασία κλείνει με τα Συμπεράσματα και προτάσεις για περαιτέρω εξέλιξη και έρευνα.

Η καινοτομία της παρούσας διπλωματικής εργασίας βρίσκεται στο γεγονός ότι μέχρι σήμερα δεν υπάρχουν διαθέσιμα πολλά βιντεοπαιχνίδια βασισμένα σε επιτραπέζια στρατηγικής όπως το 7

wonders με πλήρη τεκμηρίωση του τρόπου ανάπτυξής τους. Για το λόγο αυτό, στη διπλωματική αυτή υπάρχει υπάρχει λεπτομερής ανάλυση του τρόπου με τον οποίο δημιουργήθηκε το συγκεκριμένο παιχνίδι, αρκετά ικανή να βοηθήσει κάποιον να κατασκευάσει ένα παιχνίδι με αντίστοιχους μηχανισμούς και τρόπο παιξίματος.

1.2 Παιχνίδι

Αναζητώντας κανείς τον όρο σε μία δημοφιλή μηχανή αναζήτησης το πρώτο αποτέλεσμα που θα λάβει είναι ο ιστότοπος της Wikipedia (Wikipedia, 2024), η οποία αναφέρει ότι ως παιχνίδι αποκαλείται η δομημένη δραστηριότητα η οποία διεξάγεται με σκοπό τη ψυχαγωγία ή ασκείται ως εκπαιδευτικό εργαλείο. Από την άλλη πλευρά στη διεθνή βιβλιογραφία ο αναγνώστης θα εντοπίσει πληθώρα αναφορών μεταξύ των οποίων ξεχωρίζει αυτή του Freud (Freud, 1955) ο οποίος περιγράφει το παιχνίδι ως ένα μέσο με το οποίο το παιδί εξωτερικεύει τόσο τον συναισθηματικό του κόσμο όσο και τους φόβους αλλά και τις αδυναμίες του. Παράλληλα μέσα από το παιχνίδι μπορούν να απελευθερωθούν ορισμένα τραύματα αλλά και να βρει το παιδί ανακούφιση από το ψυχικό άγχος. Τέλος αναφέρει ότι, μεταξύ των υπολοίπων ωφελειών, το παιχνίδι προσφέρει στο παιδί και ασφάλεια ώστε να μπορεί να ξεφύγει από την σκληρή πραγματικότητα που ενδεχομένως να βιώνει, τις μη αποδεκτές συμπεριφορές αλλά και τις εμπειρίες από τις οποίες ενδεχομένως να απειλείται. Ο Winnicott (Winnicott, 1971) από την άλλη πλευρά εστιάζει στην κοινωνικοποίηση του παιδιού η οποία ενισχύεται μέσα από το παιχνίδι, καθώς και στην εξωτερίκευση συναισθημάτων (θαλπωρής, φόβου, χαράς, επιθετικότητας, κ.α.) η οποία πραγματοποιείται σε ένα πλαίσιο στο οποίο δεν υπάρχει η τιμωρία.

Φυσικά το παιχνίδι δεν αποτελεί χαρακτηριστικό της σύγχρονης εποχής καθώς ανασκαφές σε πληθώρα αρχαίων πολιτισμών όπως στη Μεσοποταμία, στην Αίγυπτο, στην Ινδία, στη Κίνα, στο Βυζάντιο και φυσικά και στην Ελλάδα έχουν εντοπισθεί διάφορα αντικείμενα όπως μινιατούρες, άλλοτε κεραμικές και άλλοτε μεταλλικές, οι οποίες πιθανολογείται ότι αποτελούσαν τα παιχνίδια των παιδιών της εποχής. Παράλληλα έχουν εντοπισθεί και αρκετές ζωγραφιές οι οποίες απεικονίζουν ανθρώπους κατά την διεξαγωγή παιχνιδιών (Frost, 2009). Πιο συγκεκριμένα στην Ελλάδα το παιχνίδι ήταν συνυφασμένο με την παιδική ηλικία και για το λόγο αυτό οι λέξεις *παίγνιον* αλλά και *παῖς* έχουν κοινή ρίζα. Στη συνέχεια το παιχνίδι επεκτάθηκε καθώς συμμετείχαν και άτομα μεγαλύτερης ηλικίας. Η πρώτη μάλιστα καταγεγραμμένη αναφορά του όρου συναντιέται στην Οδύσσεια του Ομήρου όπου ο Οδυσσέας, μεταμφιεσμένος σε ζητιάνο λαμβάνει μέρος στα παιχνίδια του βασιλιά των Φαιάκων Αλκίνοου. Τα παιχνίδια περιλάμβαναν αθλητικές δραστηριότητες όπως το τρέξιμο, η πάλη και το πέταγμα του δίσκου. Μετά από την Ομηρική εποχή (1100-323 π.Χ.) και πιο συγκεκριμένα στην Ολυμπιακή περίοδο (776 π.Χ.) συναντώνται αναφορές σε έργα του Πλάτωνα αλλά και του Αριστοτέλη οι οποίες εστιάζουν στη συμβολή των παιχνιδιών στη σωματική, ηθική καθώς και την ψυχική ανάπτυξη των παιδιών. Κατά τη σκοτεινή περίοδο του Μεσαίωνα ($5^{ος}$ – $15^{ος}$ αιώνας) το παιχνίδι παραγκωνίστηκε για να ενισχυθεί εκ νέου στα χρόνια που ακολούθησαν.

1.2.1 Κατηγορίες παιχνιδιού

Το παιχνίδι μπορεί να διαχωριστεί σε διάφορες κατηγορίες ανάλογα με τον τρόπο με τον οποίο ό ή οι παίχτες μπορούν να το εκτελέσουν. Πιο συγκεκριμένα το παιχνίδι μπορεί να είναι:

- Ατομικό

Το ατομικό παιχνίδι είναι το παιχνίδι το οποίο το παιδί το παίζει μόνο του και συνήθως αποτελεί και το πρώτο παιχνίδι στο οποίο ασχολείται το παιδί στα πρώτα του βήματα. Μέσα από το είδος αυτό του παιχνιδιού το παιδί συνειδητοποιεί την ύπαρξη του αλλά και την προσωπική του αξία. Αναπτύσσει διάφορες δεξιότητες όπως την υπομονή, την εφευρετικότητα αλλά και την δημιουργικότητα. Παράλληλα εξερευνά τις φυσικές αλλά και τις πνευματικές του ιδιότητες, απελευθερώνεται και δημιουργεί τις διάφορες εμπειρίες της καθημερινότητας ελεύθερα (Κάππας, 2005).

- Ομαδικό

Το ομαδικό από την άλλη πλευρά παίζεται με περισσότερους από ένα παίκτες. Με τον τρόπο αυτό τα παιδιά κοινωνικοποιούνται καθώς συναναστρέφονται με άλλα συνομήλικά παιδιά, πράγμα που έχει σημαντική επίδραση στην ανάπτυξη τους. Πολλαπλά τα οφέλη της δραστηριότητας αυτής για αυτά καθώς μέσω αυτής το κάθε ένα παιδί ξεχωριστά εξετάζει τις δυνάμεις του καθώς συγκρίνεται με τα υπόλοιπα παιδιά του παιχνιδιού, ανταγωνίζεται και επιβραβεύεται τονίζοντας την αυτοπεποίθηση του. Παράλληλα ανακαλύπτει αλλά και εξωτερικεύει τα αρνητικά ένστικτα του αποκτώντας αυτογνωσία. Τέλος συνειδητοποιεί τα όρια των δυνάμεων του, της αντοχής αλλά και της απόδοσης του (Αντωνιάδης, 1994).

- Παράλληλο

Στη κατηγορία αυτή εντάσσεται το παιχνίδι κατά το οποίο το παιδί παίζει μόνο του αλλά σε κοντινή απόσταση από άλλα παιδία ή χρησιμοποιεί παρόμοια υλικά. Με τον τρόπο αυτό το παιδί εμπνέεται από το παιχνίδι των άλλων παιδιών και εμπλουτίζει τη σκέψη του τονίζοντας την κοινωνικοποίηση του (Feldman, 2010).

- Αυθόρμητο

Το παιχνίδι αυτό αποτελεί ουσιαστικά τη δραστηριότητα μέσα από την οποία τα παιδιά βλέπουν αλλά και αναφέρονται στο κόσμο. Κατά τη διάρκεια του παιχνιδιού τα παιδιά δημιουργούν φανταστικές δημιουργίες οι οποίες αποτελούν τη βασική μετάβαση στην εξέλιξη του παιδιού. Μέσω αυτής το παιδί θα καταλάβει τη σύνδεση των συμβόλων, των αντικειμένων αλλά και των διαφόρων πράξεων που παρουσιάζονται στον εξωτερικό κόσμο. Παράλληλα μέσα από το παιχνίδι αυτό θα εξωτερικευτεί ο χαρακτήρας καθώς και η προσωπικότητα του κάθε παιδιού. Κατά τη διάρκεια του παιχνιδιού τα παιδιά θα υλοποιήσουν τις ξεχωριστές τους ατομικές ανάγκες αλλά και τα ενδιαφέροντα καθώς και θα ανακαλύψουν το εξωτερικό τους περιβάλλον. Τέλος το αυθόρμητο παιχνίδι επιτρέπει στα παιδιά να επικοινωνήσουν ελεύθερα με μικρό αριθμό ατόμων (Κάππας, 2005).

1.2.2 Το εκπαιδευτικό παιχνίδι

Ο όρος περιλαμβάνει τρείς βασικές κατηγορίες από τις οποίες ξεχωρίζει η ψυχαγωγική εκπαίδευση η οποία περιλαμβάνει ένα σύνολο στρατηγικών, μεθοδολογιών και προσπαθειών οι οποίες έχουν σαν σκοπό να αποκτήσει η διαδικασία της μάθησης ένα πιο διασκεδαστικό χαρακτήρα, εντός της σχολικής αίθουσας ή και εκτός, με τη χρήση των ψηφιακών μέσων (όπως ο ήχος, τα κινούμενα σχέδια, το βίντεο, τις εικόνες κλπ.) ή και άλλων τεχνικών. Στα πλαίσια της ψυχαγωγικής εκπαίδευσης μπορούν να χρησιμοποιηθούν εκπαιδευτικά παιχνίδια, εκπαιδευτική τηλεόραση, μουσική, ταινίες καθώς και εφαρμογές εκπαιδευτικού χαρακτήρα. Στόχος της όλης προσπάθειας η έλκυση του ενδιαφέροντος των μαθητών, η ανάλυση και η αξιολόγηση των εκπαιδευτικών θεμάτων με πιο ευχάριστους αλλά και ψυχαγωγικούς τρόπους (Aksakal, 2015).

Η δεύτερη κατηγορία που θα παρουσιαστεί στη παρούσα εργασία είναι η μάθηση βασισμένη στο παιχνίδι η οποία και αποτελείται από τα παιχνίδια σοβαρού σκοπού. Τα παιχνίδια αυτά σχεδιάζονται και κατασκευάζονται με αρχικό στόχο να εκπαιδεύσουν τους παίχτες τους πάνω σε κάποιο συγκεκριμένο θέμα και σαν δευτερεύον στόχο να διασκεδάσουν τους παίχτες. Μπορούν να χρησιμοποιηθούν είτε από τα επίσημα εκπαιδευτικά περιβάλλοντα όπως το σχολείο αλλά και το πανεπιστήμιο, είτε από ανεπίσημα όπως σε μουσεία, εκθέσεις διαφόρων ειδών καθώς και αλλού. Φυσικά δεν είναι λίγες οι περιπτώσεις όπου τα λογισμικά αυτά είναι αρκετά ικανά ώστε να σταθούν μόνα τους και να εκπαιδεύσουν το άτομο με πλήρη απουσία εκπαιδευτικού (Prensky, 2007).

Μια επιπλέον κατηγορία που αξίζει να αναφερθεί είναι η ψηφιακή εκπαίδευση η οποία στη διεθνή βιβλιογραφία εντοπίζεται με τον όρο E-learning και αναφέρεται στην εκπαίδευση η οποία πραγματοποιείται μέσω της ψηφιακής τεχνολογίας. Κατά την εκπαίδευση αυτή μπορούν να χρησιμοποιηθούν ποικίλες μορφές ηλεκτρονικών μέσων, τεχνολογιών, αλλά και εφαρμογών εκπαίδευσης οι οποίες θα είναι είτε υποβοηθούμενες είτε θα βασίζονται στους ηλεκτρονικούς υπολογιστές κάθε μορφής (προσωπικός υπολογιστής, System on Chip, tablet, έξυπνο κινητό τηλέφωνο, κλπ.). Η εκπαίδευση αυτή μπορεί να πραγματοποιηθεί είτε τοπικά με υλικό που θα βρίσκεται σε κάποιο ψηφιακό μέσο (υπολογιστή, cdrom, USB flash, κλπ.) είτε και απομακρυσμένα με το υλικό να διανέμεται μέσω δικτύου (Fullerton, 2008).

Η παιχνιδοποίηση, η οποία στη διεθνή βιβλιογραφία εντοπίζεται με τον όρο Gamification αποτελεί μια επίσης αξιοσημείωτη κατηγορία, η οποία περιλαμβάνει τη χρήση στοιχείων των ψηφιακών παιχνιδιών και όχι ολοκληρωμένων εφαρμογών, με στόχο την ενίσχυση της εκπαιδευτικής διαδικασίας. Η διαδικασία επιτυγχάνεται με την εφαρμογή τεχνικών οι οποίες εφαρμόζονται κατά τον σχεδιασμό αλλά και την ανάπτυξη των εφαρμογών (Sebastian Deterding, 2011).

1.3 Το ηλεκτρονικό – ψηφιακό παιχνίδι

Το ηλεκτρονικό παιχνίδι από την άλλη πλευρά είναι αυτό που παίζεται με τη βοήθεια του ηλεκτρονικού υπολογιστή ή με κάποιο αντίστοιχο ηλεκτρονικό μέσο. Καθώς η τεχνολογία επέτρεψε την κατασκευή των ψηφιακών παιχνιδιών τα σύγχρονα χρόνια το ψηφιακό παιχνίδι αποτελεί ένα σχετικά πρόσφατο επίτευγμα. Στο σχετικά σύντομο χρονικό διάστημα της ζωής του έχουν χρησιμοποιηθεί διάφοροι όροι για να το περιγράψουν όπως βιντεοπαιχνίδι (video game), παιχνίδι υπολογιστή (computer game), ηλεκτρονικό παιχνίδι (electronic game) καθώς και άλλα. Αξίζει να αναφερθεί ότι ο όρος βιντεοπαιχνίδι εμφανίστηκε πρώτος καθώς οι πρώτες μηχανές ψηφιακών παιχνιδιών δεν διέθεταν οθόνη σε μια προσπάθεια να κρατήσουν οι κατασκευαστές το κόστος της συσκευής χαμηλό. Αντί αυτού συνδέονταν στην συσκευή τηλεόρασης που υπήρχε σε κάθε σπίτι και πιο συγκεκριμένα στην υποδοχή του βίντεο. Για το λόγο αυτό ονομάστηκαν και βιντεοπαιχνίδια. Στη συνέχεια, με την επικράτηση των υπολογιστών μετονομάστηκαν σε παιχνίδια ηλεκτρονικού υπολογιστή αφού πλέον παιζόταν μέσω αυτού.

Η πρώτη συσκευή που θεωρείται ηλεκτρονικό παιχνίδι, όχι ψηφιακό όμως καθώς ήταν αναλογική, πατενταρίστηκε στις Ηνωμένες Πολιτείες Αμερικής στις 25 Ιανουαρίου του 1947. Εμπνευσμένη από τις συσκευές ραντάρ διέθετε ένα καθοδικό σωλήνα ο οποίος εμφάνιζε ένα τόξο και επέτρεπε στον παίχτη να κατευθύνει ένα πύραυλο προς συγκεκριμένους στόχους.

Εικόνα 2: Ένα από τα πρώτα ηλεκτρονικά παιχνίδια. Πηγή: Wikipedia

Το ψηφιακό παιχνίδι υπάρχει μέσα σε ένα εικονικό περιβάλλον, όχι στον πραγματικό κόσμο. Για το σκοπό αυτό είναι σκόπιμο να παρουσιαστεί αρχικά ο όρος αυτός και ακολούθως το ψηφιακό παιχνίδι. Πιο συγκεκριμένα το εικονικό περιβάλλον αποτελεί ένα τεχνητό περιβάλλον που κατασκευάζεται μέσω λογισμικού στοχεύοντας να δημιουργήσει στο παίχτη την ψευδαίσθηση ότι βρίσκεται σε κάποιο πραγματικό περιβάλλον (Kuo-Ting Huang, 2019). Μάλιστα, όσο καλύτερο είναι το τεχνητό περιβάλλον τόσο αυξάνεται η αίσθηση της αληθοφάνειας του χρήστη και το παιχνίδι γίνεται καλύτερο. Τα λογισμικά χρησιμοποιούν οπτικούς μηχανισμούς, ακουστικούς αλλά και απτικούς, προσφέροντας στον παίχτη αισθητηριακές εμπειρίες. Με τον τρόπο αυτό ο παίχτης μπορεί να κινηθεί ελεύθερα σε ένα δισδιάστατο ή και τρισδιάστατο εικονικό κόσμο, να τον εξερευνήσει και αλληλεπιδράσει με αυτόν αλλά και με τα αντικείμενα που υπάρχουν μέσα σε αυτόν. Παράλληλα υπάρχει η δυνατότητα το ίδιο ψηφιακό παιχνίδι να παίζεται από πολλούς παίχτες οι οποίο να εμφανίζονται στον ίδιο τεχνητό κόσμο και να αλληλεπιδρούν μεταξύ τους. Το περιβάλλον αυτό παρουσιάζεται στον παίχτη μέσω μιας οπτικοακουστικής συσκευής, ο παίχτης αλληλεπιδρά με αυτό με κάποια συσκευή εισόδου ώστε να ακολουθήσει την ιστορία που αποτελεί το ψηφιακό παιχνίδι (Esposito, 2005). Πιο αναλυτικά στην εγκυκλοπαίδεια της πληροφορικής και των τεχνολογιών (Kamisah Osman, 2018) ως ψηφιακό παιχνίδι ορίζεται κάθε παιχνίδι το οποίο:

- Προκειμένου να εκτελεστεί απαιτείται η χρήση κάποιας ηλεκτρονικής συσκευής.
- Προϋποθέτει κάποιο είδος αλληλεπίδρασης του παίχτη με την ηλεκτρονική συσκευή μέσω κάποιας συσκευής η οποία συνδέεται με την ηλεκτρονική συσκευή και η αλληλεπίδραση αυτή παρουσιάζεται στον παίχτη μέσω κάποιας οθόνης.
- Η λειτουργία του παιχνιδιού θα πρέπει σαφώς να καθορίζεται από συγκεκριμένους κανόνες.
- Μπορεί να εκτελείται είτε τοπικά είτε και απομακρυσμένα, μέσω διαδικτύου.
- Μπορεί να εκτελεστεί είτε από ένα είτε και από περισσότερους παίκτες.

Εικόνα 3: Βιντεοπαιχνίδι Ping - Pong. Πηγή: Wikipedia

1.4 Κατηγορίες παιχνιδιών

1.4.1 Παιχνίδια δράσης

Τα ψηφιακά παιχνίδια, αλλά και τα παιχνίδια γενικότερα, ανάλογα με το περιβάλλον στο οποίο εκτελούνται αλλά και το περιεχόμενο τους μπορούν να οργανωθούν σε διάφορες κατηγορίες. Οι πλέον αντιπροσωπευτικές αλλά και επικρατέστερες παρουσιάζονται στην παρούσα παράγραφο με έμφαση προφανώς στα ψηφιακά παιχνίδια. Η παρουσίαση θα ξεκινήσει με τη κατηγορία των παιχνιδιών δράσης στα οποία ο κάθε παίχτης παρουσιάζεται στον εικονικό κόσμο του παιχνιδιού μέσω μιας εικονικής αναπαράστασης, ενός χαρακτήρα, τον οποίο δύναται να επιλέγει ο ίδιος, μέσω διαφόρων επιλογών που του δίνονται, είτε και να δημιουργήσει το χαρακτήρα που επιθυμεί. Στη διεθνή βιβλιογραφία ο εικονικός χαρακτήρας αυτός ονομάζεται avatar και μέσω αυτού αλληλεπιδρά με το εικονικό περιβάλλον του παιχνιδιού, τους άλλους παίκτες – αν φυσικά υπάρχουν και προσπαθεί να εκτελέσει μια συγκεκριμένη αποστολή, ακολουθώντας τους κανόνες του παιχνιδιού. Μέσα από τα παιχνίδια της κατηγορίας αυτής, ο παίχτης βελτιώνει τα αντανακλαστικά και την ταχύτητα του στη προσπάθεια του να εξερευνήσει τον εικονικό κόσμο και να λύσει τα προβλήματα που του ανατίθενται. Θα πρέπει να είναι ικανός να λαμβάνει γρήγορα αποφάσεις, σύμφωνα με το επίπεδο δυσκολίας του παιχνιδιού, το οποίο επίπεδο αυξάνεται καθώς το παιχνίδι εξελίσσεται. Παράλληλα, τα παιχνίδια αυτά συνδυάζουν χαρακτηριστικά όπως η προσοχή, η εναλλαγή της εστιασμένης και διαιρεμένης προσοχής, ο σχεδιασμός των κινήσεων μέσα στο παιχνίδι που θα επιτρέψουν τον παίχτη να επιτύχει το στόχο του, καθώς και άλλα. Τα χαρακτηριστικά αυτά εξασκούν τον εγκέφαλο στη διαδικασία της μάθησης αλλά και τον διευκολύνουν (Gambacorta C, 2018). Τον τίτλο του πρώτου ψηφιακού παιχνιδιού της κατηγορίας αυτής τον φέρει το Space Invaders, το οποίο αναπτύχθηκε από τον Τομοχίρο Νισικάντο ο οποίος το διάστημα αυτό εργαζόταν για την εταιρία Taito η οποία είχε

έδρα στην Ιαπωνία. Στο παιχνίδι ο παίχτης κινούσε μέσω ενός μοχλού ένα κανόνι οριζόντια και πυροβολούσε τους εξωγήινους που του επετίθεντο. Τα πλέον γνωστά παιχνίδια της κατηγορίας είναι το Packman, το Quake, το Unreal Tournament, καθώς και άλλα.

1.4.2 Παιχνίδια στρατηγικής

Τα παιχνίδια στρατηγικής, από την άλλη πλευρά, όπως και το όνομα τους ορίζει, απαιτούν από τον παίχτη να καταστρώσει μια στρατηγική η οποία θα του επιτρέψει να επιτύχει το στόχο του και να κερδίσει τους αντιπάλους του. Ο σχεδιασμός της στρατηγικής απαιτεί κριτική σκέψη, λήψη αποφάσεων και σχεδιασμό κινήσεων αναπτύσσοντας με τον τρόπο αυτό δυνατότητες όπως την εύρεση λύσεων σε προβλήματα, τη στρατηγική σκέψη, τη διαχείριση των διαθέσιμων πόρων καθώς και άλλα. Το στοιχείο ακριβώς αυτό είναι και το βασικό που διαφοροποιεί τα παιχνίδια της κατηγορίας αυτής από αυτά των υπολοίπων καθώς εδώ απαιτείται σκέψη και όχι γρήγορες αποφάσεις καθώς αυτές δεν εξασφαλίζουν την επιτυχία. Ο παράγοντας τύχη δεν έχει την ίδια βαρύτητα, όπως και στα άλλα παιχνίδια καθώς όλοι οι παίχτες ξεκινούν από μια κοινή βάση πάνω στην οποία θα πρέπει να αναπτύξουν την στρατηγική τους προκειμένου να επικρατήσουν έναντι των αντιπάλων τους. Από τα πλέον γνωστά ψηφιακά παιχνίδια της κατηγορίας είναι το Word Of Warcraft, το Age of Empires καθώς και άλλα.

1.4.3 Παιχνίδια περιπέτειας

Μία εξίσου αξιοσημείωτη κατηγορία είναι αυτή που περιλαμβάνει τα παιχνίδια περιπέτειας και στη διεθνή βιβλιογραφία εντοπίζεται με τον όρο Adventure Games. Στη κατηγορία αυτή περιλαμβάνονται τα παιχνίδια στα οποία ο παίχτης πρωταγωνιστεί σε μια διαδραστική ιστορία η οποία συνήθως λαμβάνει χώρα σε ένα εικονικό κόσμο από συνδεμένα δωμάτια ή σκηνές. Η ιστορία αυτή περιλαμβάνει την εξερεύνηση του εικονικού περιβάλλοντος για τον εντοπισμό αντικειμένων, την επίλυση διαφόρων γρίφων, την πραγματοποίηση διαφόρων διαλόγων και αποστολών σε μια προσπάθεια να εκτελεστεί η κάθε διαφορετική αποστολή του παιχνιδιού. Και στη περίπτωση αυτή τα γρήγορα αντανακλαστικά δεν εξασφαλίζουν την επιτυχία καθώς μεγαλύτερο ρόλο διαδραματίζει η σκέψη και οι προσεκτικά σχεδιασμένες κινήσεις. Μπορεί να παίζεται από ένα παίχτη ή και από περισσότερους είτε τοπικά είτε μέσω δικτύου. Τέλος, σε αρκετές περιπτώσεις η ιστορία στην οποία βασίζεται το παιχνίδι αποτελεί αντιγραφή ή παραλλαγή κάποιο σεναρίου γνωστής ταινίας. Οι πλέον γνωστές περιπτώσεις της κατηγορίας αυτής είναι το Mystery House καθώς και το Walking Dead.

1.4.4 Παιχνίδια ρόλων

Στην επόμενη κατηγορία περιλαμβάνονται τα παιχνίδια στα οποία ο παίχτης εκτελεί ένα συγκεκριμένο ρόλο αναλαμβάνοντας την ευθύνη για τις πράξεις του και τις αποφάσεις του. Ανάλογα με τους κανόνες του παιχνιδιού οι πράξεις ή οι αποφάσεις είτε πετυχαίνουν είτε αποτυγχάνουν να πετύχουν τους προκαθορισμένους στόχους και από αυτό κρίνεται η επιτυχία ή η αποτυχία του παίχτη (Renu Mary Daniel, 2016). Ο ρόλος που θα επιλέξει ο παίχτης έχει συγκεκριμένα χαρακτηριστικά τα οποία τον διαφοροποιούν από τους υπόλοιπους του παιχνιδιού, συγκεκριμένες ιδιότητες, απασχόληση ή και φύλο. Τα χαρακτηριστικά αυτά ο παίχτης μπορεί να τα εξελίσσει καθώς το παιχνίδι προχωρά και με τον τρόπο αυτό να γίνεται καλύτερος από τους υπόλοιπους συναγωνιζόμενους. Η υιοθέτηση αυτή των διαφορετικών ρόλων προσδίδει στο παίχτη μια ελευθερία, καθώς μέσα από το παιχνίδι μπορεί να μεταμορφωθεί και κάτι διαφορετικό.

1.4.5 Παιχνίδια προσομοίωσης

Τα παιχνίδια προσομοίωσης μπορούν να χρησιμοποιηθούν όχι μόνο για σκοπούς διασκέδασης αλλά και για εκπαιδευτικούς σκοπούς. Κατά τη διεξαγωγή του παιχνιδιού ο χρήστης αποκτά κάποιες συγκεκριμένες δεξιότητες τις οποίες απαιτεί το επιστημονικό πεδίο στο οποίο εξελίσσεται το παιχνίδι. Ο βαθμός κατάκτησης της δεξιότητας καθορίζει την επίδοση του παίχτη στο παιχνίδι ενώ παράλληλα ρόλο μπορούν να παίζουν και η τύχη, η στρατηγική του παίχτη, η ικανότητα στη λήψη αποφάσεων, καθώς και άλλα. Στο παιχνίδι μπορεί να χρησιμοποιούνται τόσο εικονικοί όσο και πραγματικοί κόσμοι, όπου στη περίπτωση των πραγματικών μπορεί να έχουν εφαρμογή οι νόμοι της φυσικής καθώς και οι διάφοροι περιορισμοί της καθημερινότητας. Με τον τρόπο αυτό ο παίχτης μπορεί να εκπαιδευτεί σε κάποιο συγκεκριμένο επιστημονικό πεδίο μέσα σε ένα ασφαλές περιβάλλον που μοιάζει με το πραγματικό. Από τα πλέον γνωστά παιχνίδια της κατηγορίας αυτής είναι το Sims και το Flight Simulator.

1.4.6 Παιχνίδια σοβαρού σκοπού

Η συγκεκριμένη παρουσίαση θα κλίσει με τα παιχνίδια σοβαρού σκοπού τα οποία καλούνται σοβαρά καθώς ο στόχος τους είναι η εκπαίδευση του παίχτη ενώ η διασκέδαση του έρχεται σε δεύτερη μοίρα. Πιο συγκεκριμένα τα παιχνίδια αυτά σχεδιάζονται και κατασκευάζονται με σκοπό όχι τη διασκέδασή αλλά την εκπαίδευση πάνω σε ένα συγκεκριμένο επιστημονικό πεδίο με διασκεδαστικό τρόπο. Το παιχνίδι δηλαδή διατηρεί τα παιγνιώδη στοιχεία, παραπέμπει σε παιχνίδι αλλά ο στόχος του είναι εκπαιδευτικός (Kiryakova, 2014). Απευθύνονται τόσο σε ενήλικες αλλά και σε άτομα νεαρής ηλικίας καθώς πλέον μπορεί κανείς να εντοπίσει στο εμπόριο προϊόντα για όλες τις ηλικίες. Το παιχνίδι χαρακτηρίζεται από συγκεκριμένου μαθησιακούς στόχους, παρέχει ανατροφοδότηση στο παίχτη σχετικά με την επίδοση του η οποία μπορεί να γνωστοποιείται και στον εκπαιδευτικό αλλά και στους υπόλοιπους εκπαιδευόμενους. Με τον τρόπο αυτό ο εκπαιδευτής μπορεί να έχει πλήρη εικόνα της επίδοσης του κάθε μαθητή ξεχωριστά ενώ παράλληλα οι μαθητές συναγωνίζονται μεταξύ τους ώστε να πετύχουν την μέγιστη επίδοση και να εισπράξουν την καθορισμένη επιβράβευση. Οι εκπαιδευόμενοι απολαμβάνουν περισσότερο τη μάθηση καθώς αυτή πραγματοποιείται μέσα σε ένα περιβάλλον που ενισχύει το ενδιαφέρον του μαθητή καθώς επιτρέπει περισσότερες αισθήσεις να συμμετάσχουν στη διαδικασία της εκπαίδευσης. Πιο συγκεκριμένα, με τη βοήθεια της τεχνολογίας, ο εκπαιδευόμενος δεν περιορίζεται μόνο στην ανάγνωση της γνώσης αλλά μπορεί να παρακολουθεί εικόνες και βίντεο από το εκπαιδευτικό θέμα, να ακούει πληροφορίες για το θέμα, αποκτώντας με τον τρόπο αυτό μια πιο ουσιαστική εικόνα του γνωστικού αντικειμένου. Παράλληλα η μάθηση βελτιώνεται καθώς το παιχνίδι αποτελεί ένα μέσο εξάσκησης της αποκτηθείσας γνώσης, συνδέοντας την με την προγενέστερη (Richard N. Van Eck, 2017). Από τα πλέον γνωστά εκπαιδευτικά λογισμικά για την εκμάθηση του προγραμματισμού σε παιδιά αποτελεί το Scratch.

Εικόνα 4: Εφαρμογή Scratch. Πηγή: stem.edu.gr

2 ΚΕΦΑΛΑΙΟ 2ο : Περιγραφή παιχνιδιού

Στο κεφάλαιο αυτό θα γίνει μια αναλυτική περιγραφή του παιχνιδιού στην οποία θα αναλυθούν οι κανόνες του. Στόχος είναι να εξοικειωθεί ο αναγνώστης με τους όρους του παιχνιδιού, τα αντικείμενα του, τον τρόπο που αυτά τοποθετούνται στο τραπέζι καθώς και τον τρόπο που παίζεται το παιχνίδι. Μέσα από τη παρουσίαση αυτή ο αναγνώστης θα αποκτήσει τη βασική γνώση που απαιτείται ώστε να μπορεί να παρακολουθήσει την παρουσίαση που ακολουθεί. Οι πληροφορίες που παρουσιάζονται είναι αναρτημένες στον επίσημο ιστότοπο της εταιρίας και πιο συγκεκριμένα στη διεύθυνση https://www.prod.com/en/games/7-wonders.

2.1 Τα αντικείμενα του παιχνιδιού

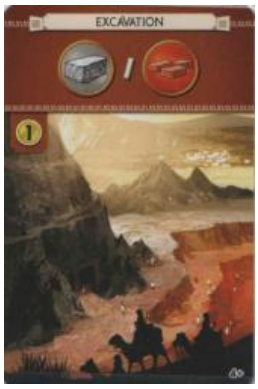
2.1.1 Οι κάρτες

Όπως αναφέρθηκε και σε προηγούμενη παράγραφο το παιχνίδι βασίζεται σε κάρτες οι οποίες αναπαριστούν τις διάφορες κατασκευές τις οποίες μπορεί να κατασκευάσει ο κάθε παίχτης στην πόλη του. Τέτοιες κατασκευές είναι ένα ορυχείο από το οποίο θα παράγει κάποιο πόρο όπως κάποιο μέταλλο για παράδειγμα, ένα εργοτάξιο κοπής δέντρων από όπου θα παίρνει ξυλεία, σχολείο, λουτρό, αγορά κλπ. Το παιχνίδι εκτελείται χρονολογικά σε τρείς χρονικές περιόδους και η κάθε κάρτα αντιστοιχεί σε συγκεκριμένη χρονική περίοδο. Υπάρχουν συνολικά 148 κάρτες εκ των οποίων 49 αντιστοιχούν στη πρώτη περίοδο, 49 στη 2η και τέλος 50 στη 3η. Παράλληλα οι κάρτες ομαδοποιούνται με βάση το χρώμα τους. Συνεπώς υπάρχουν

1. Οι καφέ κάρτες οι οποίες αντιστοιχούν σε κατασκευές οι οποίες παρέχουν βασικούς πόρους όπως πηλό, μάρμαρο, σιδηρομετάλλευμα και ξύλο.
2. Οι γκρι κάρτες αντιστοιχούν σε κατασκευές που παρέχουν γυαλί, ύφασμα και πάπυρο.
3. Οι μπλε κάρτες παρέχουν πόντους οι οποίοι αθροίζονται στο τέλος του παιχνιδιού και ο παίχτης με τους περισσότερους κερδίζει.
4. Οι πορτοκαλί κάρτες επιτρέπουν τις εμπορικές δραστηριότητες ενώ οι κόκκινες αυξάνουν τη στρατιωτική ισχύς.
5. Οι πράσινες αντιστοιχούν στις κατασκευές οι οποίες επιτρέπουν τις επιστημονικές ανακαλύψεις στους τομείς της μηχανικής, των γραμμάτων αλλά και των αριθμών.
6. Τέλος, οι μοβ κάρτες παρέχουν και αυτές πόντους που καθορίζουν τη νίκη αλλά υπό συγκεκριμένες συνθήκες.

Εικόνα 5: Η πίσω όψη της κάρτας. Πηγή: 7-wonders

Στην αρχή του παιχνιδιού οι κάρτες τοποθετούνται στο τραπέζι με την πίσω όψη πάνω ώστε να μην είναι ορατό το περιεχόμενο της κάρτας. Τοποθετούνται σε στοίβες ανάλογα με την χρονική περίοδο στην οποία αναφέρονται. Έτσι η πίσω όψη της κάρτας αναγράφει την περίοδο στην οποία ανήκει (περίοδος Ι, περίοδος ΙΙ, περίοδος ΙΙΙ) και στο κάτω μέρος αυτής ένα σύμβολο που δείχνει προς πια κατεύθυνση θα πρέπει να κινηθεί η κάρτα. Πιο συγκεκριμένα ο παίχτης μόλις ολοκληρώσει το παίξιμο μια κάρτας περνάει τις υπόλοιπες που του αναλογούν σε ένα από τους διπλανούς του. Το αν θα πάει στον αριστερό ή στο δεξί αυτό καθορίζεται από το σύμβολο αυτό της κάρτας.



Εικόνα 6: Καφέ κάρτα. Πηγή: 7-wonders

Στην μπροστά όψη της κάρτας και με κατεύθυνση από πάνω προς τα κάτω ο αναγνώστης αρχικά θα εντοπίσει το όνομα της κάρτας. Κάτω από το όνομα η κάρτα έχει μια περιοχή στην οποία εμφανίζονται οι πόροι ή ο πόρος ο οποίος παρέχει η κάρτα. Στα δεξιά της περιοχής αυτής εμφανίζονται τα σύμβολα αλυσίδας της κάθε κάρτας στη περίπτωση που η κάρτα έχει τέτοια. Κάτω από τις περιοχές αυτές και αριστερά είναι η ζώνη κόστους της κάρτας όπου αναγράφεται το κόστος της συγκεκριμένης κατασκευής. Τέλος στο κάτω μέρος και αριστερά εμφανίζεται ένας αριθμός που αντιστοιχεί στον ελάχιστο αριθμό παιχτών για τους οποίους είναι κατάλληλη η κάρτα.

2.1.2 Οι κάρτες θαύματα

Το παιχνίδι περιέχει 7 κάρτες θαύματα και κάθε παίχτης παίρνει από μία. Η κάθε κάρτα παρουσιάζει ένα από τα 7 θαύματα του κόσμου και μπορεί να χρησιμοποιηθεί είτε από τη μία πλευρά της είτε από την άλλη. Η διαφορά της μιας όψης από την άλλη είναι ότι στη μία το θαύμα παρουσιάζεται πως είναι την ημέρα ενώ η άλλη πλευρά το παρουσιάζει τις νυχτερινές ώρες. Με τον τρόπο αυτό ο παίχτης αποκτά μια πιο ολοκληρωμένη εικόνα του θαύματος καθώς η εικόνα τους διαφέρει.

Εικόνα 7: Κάρτα θαύματος. Πηγή: 7-Wonders

Στην πάνω αριστερή πλευρά της κάρτας εμφανίζεται ο πόρος ο οποίος προσφέρεται από την κάρτα. Στην απέναντι αριστερή πλευρά αναγράφεται το όνομα της κάρτας και δίπλα του το σύμβολο του ήλιου ή του φεγγαριού, που αντιστοιχούν στην όψη της ημέρας ή της νύχτας. Στη κάτω πλευρά της κάρτας παρατίθενται δύο, τρία ή τέσσερα κατασκευαστικά στάδια και για κάθε ένα από αυτά παρουσιάζεται το κόστος του αλλά και το τι προσφέρει στον παίχτη αν η συγκεκριμένη κατασκευή ολοκληρωθεί.

2.1.3 Μάρκες στρατιωτικών συγκρούσεων

Το παιχνίδι περιλαμβάνει 48 μάρκες εκ των οποίων οι 24 αναπαριστούν στρατιωτική ήττα και οι υπόλοιπες 24 τη νίκη. Οι τελευταίες μοιράζονται στις τρείς περιόδους από 8 σε κάθε περίοδο. Όταν ένας παίχτης είναι στρατιωτικά πιο ισχυρός από το γειτονικό του παίρνει τη μάρκα που αντιστοιχεί στη περίοδο, ενώ όταν είναι πιο αδύναμος παίρνει τη μάρκα -1. Όταν υπάρχει ισοπαλία κανείς δεν λαμβάνει μάρκα.

Εικόνα 8: Οι μάρκες του παιχνιδιού. Πηγή: 7 - Wonders

2.1.4 Νομίσματα

Τα νομίσματα αποτελούν τα χρήματα που υπάρχουν στο παιχνίδι. Όταν το παιχνίδι ξεκινά κάθε παίχτης έχει 3 νομίσματα. Για την δημιουργία των διαφόρων κατασκευών συχνά απαιτούνται νομίσματα τα οποία ο παίχτης μπορεί να αποκτήσει πουλώντας πόρους στους γείτονες του.

Εικόνα 9: Τα νομίσματα του παιχνιδιού. Πηγή: 7-wonders

2.1.5 Έγγραφα

Στο κουτί του παιχνιδιού περιλαμβάνονται και τρία έγγραφα εκ των οποίων το ένα παρουσιάζει τα σύμβολα του παιχνιδιού ώστε οι παίχτες να έχουν μια αναφορά του τι αναπαριστά το κάθε σύμβολο, ένα έγγραφο πάνω στο οποίο αθροίζονται οι πόντοι του κάθε παίχτη και τέλος ένα έγγραφο που εξηγεί τις κάρτες καθώς και τις συνδέσεις μεταξύ τους.

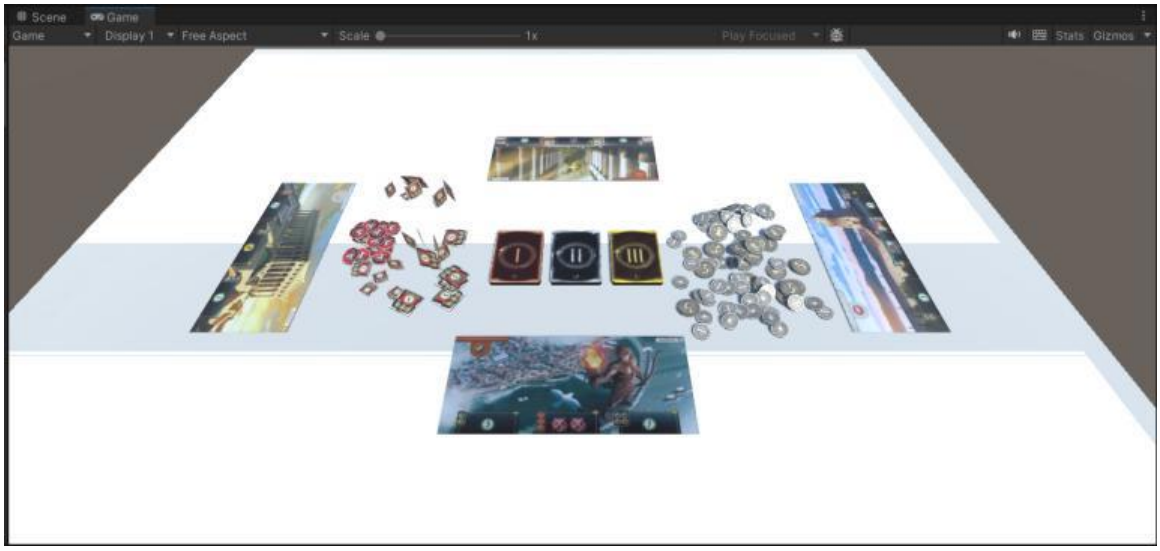
2.2 Στήσιμο του παιχνιδιού

2.2.1 Επιλογή καρτών περιόδων

Πρώτο βήμα για την έναρξη του παιχνιδιού η διαχώριση των καρτών των τριών περιόδων. Αφού καθοριστεί ο αριθμός των παιχτών θα πρέπει να αφαιρεθούν οι κάρτες που προορίζονται για περισσότερους παίχτες από αυτούς που συμμετέχουν στο παιχνίδι και κατόπιν οι κάρτες που απομένουν να ανακατευτούν και να στοιβαχθούν ανά περίοδο. Για παράδειγμα, αν στο παιχνίδι συμμετέχουν 4 παίχτες θα πρέπει να μείνουν στις στοίβες οι κάρτες που έχουν 3+ και 4+. Τέλος, στη στοίβα με τις κάρτες της 3^{ης} περιόδου θα πρέπει να προστεθούν μωβ κάρτες. Ο αριθμός τους θα είναι όσο ο αριθμός των παιχτών + 2 και θα πρέπει να επιλεγούν τυχαία. Οι υπόλοιπες δεν θα χρησιμοποιηθούν στο παιχνίδι.

2.2.2 Παραλαβή καρτών και χρημάτων

Το επόμενο βήμα προτού το παιχνίδι ξεκινήσει είναι να παραλάβει ο κάθε παίχτης τα υπόλοιπα αντικείμενα που θα χρειαστεί. Αρχικά θα πρέπει να διαλέξει την κάρτα του θαύματος που θα χρησιμοποιήσει και ακολούθως να επιλέξει όψη. Στη συνέχεια λαμβάνει τρία νομίσματα της μιας μονάδας από τη στοίβα των νομισμάτων και τα τοποθετεί πάνω στη κάρτα θαύματος, στη δεξιά της πλευρά. Οι μάρκες, οι κάρτες και τα χρήματα θα πρέπει να τοποθετηθούν πάνω στο τραπέζι όπως αναπαρίστανται στην επόμενη εικόνα.



Εικόνα 10: Το στήσιμο του παιχνιδιού για 4 παίκτες. Διακρίνονται οι μάρκες στο αριστερό μέρος, οι κάρτες των περιόδων στοιβαγμένες και τέλος τα κέρματα. Πηγή: Αλέξανδρος Βεκίλογλου

2.3 Εκτέλεση του παιχνιδιού

Πρώτη κίνηση για την έναρξη του παιχνιδιού αποτελεί το μοίρασμα των καρτών της αντίστοιχης περιόδου στους παίχτες. Πιο συγκεκριμένα ένας παίχτης θα πάρει τη στοίβα των καρτών της πρώτης περιόδου και θα μοιράσει κυκλικά από 7 κάρτες σε κάθε ένα παίχτη. Οι κάρτες μοιράζονται με την πίσω όψη από πάνω ώστε να μην είναι ορατό το περιεχόμενο της κάρτας. Κατόπιν ο κάθε παίχτης τοποθετεί τις κάρτες που του αναλογούν στην αριστερή πλευρά της κάρτας θαύματος που έχει μπροστά του.

Η κάθε μια από τις 3 φάσεις του παιχνιδιού εξελίσσεται σε 6 γύρους. Πιο συγκεκριμένα σε κάθε γύρο όλοι οι παίχτες ταυτόχρονα διαλέγουν μία κάρτα από τη στοίβα τους, την παίζουν και ακολούθως δίνουν στο διπλανό τους τις υπόλοιπες. Αν η στοίβα με τις κάρτες θα μεταφερθεί στο δεξί παίχτη ή στον αριστερό αυτό καθορίζεται από το σύμβολο που υπάρχει στο κάτω μέρος της πίσω όψης της κάρτας. Συνεπώς μετά από 6 γύρους ο κάθε παίχτης θα λάβει δύο κάρτες από το γειτονικό του. Από αυτές θα διαλέξει τη μία να παίξει και την άλλη θα την τοποθετήσει με την πίσω όψη πάνω στο κενό ανάμεσα στις μάρκες και στα νομίσματα, δίπλα στις στοίβες με τις κάρτες των περιόδων. Όλες οι κάρτες που δεν χρησιμοποιούνται θα τοποθετούνται πάνω από αυτή με την πίσω όψη πάνω.

2.3.1 Το παίξιμο της κάρτας

Μόλις οι παίχτες αποφασίσουν ποια κάρτα θα επιλέξουν από τη στοίβα τους μπορούν να εκτελέσουν μία από τις ακόλουθες ενέργειες.

2.3.1.1 Δημιουργία κατασκευής

Η πλέον συνηθισμένη ενέργεια αποτελεί την δημιουργία της κατασκευής που προσφέρει η κάρτα. Μόλις η κατασκευή ολοκληρωθεί η κάρτα τοποθετείται σε συγκεκριμένη θέση στη κάρτα θαύματος και παραμένει εκεί, παρέχοντας τον πόρο που διαθέτει καθ' όλη τη διάρκεια του παιχνιδιού. Για να δημιουργηθεί όμως η κατασκευή θα πρέπει να διευκρινιστεί αν ο παίχτης διαθέτει τους απαραίτητους για τη κατασκευή πόρους.

Πιο συγκεκριμένα, όπως παρουσιάστηκε και σε προηγούμενη παράγραφο, η κάθε κάρτα έχει μια περιοχή στην οποία παρουσιάζεται το κόστος της. Οι κάρτες που υπάρχουν στο παιχνίδι μπορεί να είναι δωρεάν, να μην έχουν κάποιο κόστος δηλαδή, να έχουν κόστος χρηματικό, να έχουν κάποιο

πόρο σαν κόστος και τέλος να μην έχουν κόστος μέσω των αλυσίδων. Συνεπώς αν στη περιοχή του κόστους της κάρτας υπάρχει ένα ή περισσότερα νομίσματα τότε για να δημιουργηθεί η κατασκευή θα πρέπει ο παίχτης να διαθέτει το σύνολο των κερμάτων που αναφέρονται. Αν τα διαθέτει τότε θα πρέπει να τα μεταφέρει από την κάρτα θαύματος που βρίσκονται στη στοίβα με τα νομίσματα που υπάρχει πάνω στο τραπέζι.

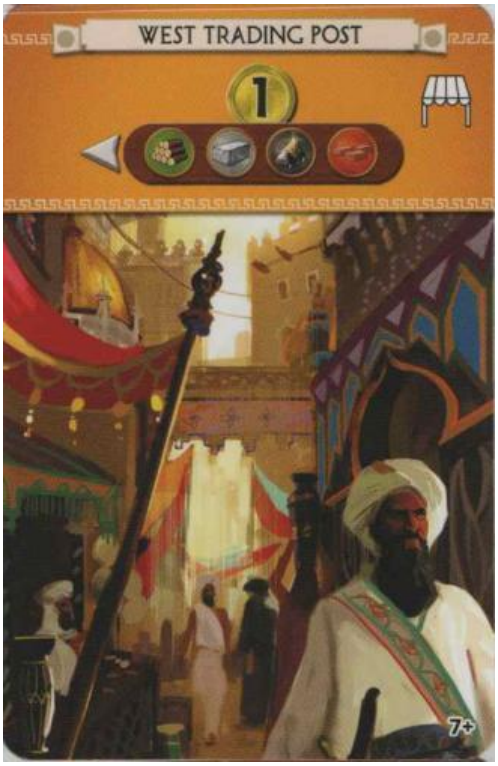
Στη περίπτωση που στη περιοχή του κόστους της κάρτας υπάρχει ένα η περισσότερα σύμβολα πόρου τότε ο παίχτης θα πρέπει να διαθέτει τους πόρους για τη κατασκευή. Στην πρώτη περίοδο οι περισσότερες κάρτες δεν απαιτούν πόρους συνεπώς ο παίχτης μπορεί να δημιουργήσει τις κατασκευές. Παράλληλα και η κάρτα θαύματος προσφέρει κάποιο πόρο τον οποίο ο παίχτης μπορεί να χρησιμοποιήσει για να δημιουργήσει την κατασκευή. Κάθε σύμβολο πόρου αντιστοιχεί σε μία μονάδα, συνεπώς αν κάποια κάρτα έχει στη περιοχή του κόστους δύο σύμβολα από ένα ή περισσότερους πόρος τότε απαιτεί αντίστοιχη ποσότητα από τους διαθέσιμους πόρους. Για παράδειγμα αν κάποιος παίχτης έχει επιλέξει την κάρτα της Εφέσου η οποία διαθέτει μια μονάδα πάπυρου τότε μπορεί ο συγκεκριμένος παίχτης να δημιουργήσει τις κατασκευές που απαιτούν μια μονάδα από τον πόρο αυτό. Πέρα από τον πόρο που προσφέρει η κάρτα θαύματος, πόρους προσφέρουν και οι κατασκευές που ενδεχομένως να έχει ήδη δημιουργήσει ο παίχτης από προηγούμενους γύρους. Πιο συγκεκριμένα η κάθε κατασκευή που δημιουργείται η αντίστοιχη κάρτα τοποθετείται στη καθορισμένη θέση της κάρτας θαύματος, όπως στην επόμενη εικόνα.

Εικόνα 11: Η κάθε κάρτα που κατασκευάζεται τοποθετείται κάτω από την κάρτα θαύματος στην αριστερή πλευρά της με τέτοιο τρόπο ώστε να είναι εμφανής ο πόρος ο οποίος παρέχει. Πηγή: Αλέξανδρος Βεκίλογλου

Με τον τρόπο αυτό ο κάθε παίχτης μπορεί εύκολα να γνωρίζει ποιους πόρους παράγει η πόλη του. Συνεπώς, για κάθε νέα κατασκευή που θέλει να φτιάξει ελέγχει τις κάρτες που υπάρχουν στην πόλη του και καθορίζει αν μπορεί ή όχι να δημιουργήσει τη νέα κατασκευή.

Στη περίπτωση που δεν διαθέτει τους απαιτούμενους πόρους μπορεί να τους αγοράσει από τους γειτονικούς του παίχτες, αυτόν που βρίσκεται δεξιά του και αυτόν που βρίσκεται αριστερά του. Για κάθε πόρο που αγοράζει θα πρέπει να καταβάλει δύο νομίσματα του ενός στο συγκεκριμένο παίχτη ή στους παίχτες, εκτός και αν ο παίχτης έχει δημιουργήσει κατασκευή κίτρινης κάρτας η οποία του δίνει τη δυνατότητα να αγοράσει συγκεκριμένο πόρο αντί ενός νομίσματος. Ο παίχτης ο οποίος διαθέτει τον πόρο δεν μπορεί να αρνηθεί την πώληση ενώ παράλληλα τον πόρο που πουλά μπορεί να τον χρησιμοποιήσει και ο ίδιος, δεν τον επηρεάζει η πώληση δηλαδή. Μοναδικός περιορισμός αποτελεί ο κανόνας ότι τον ίδιο πόρο ένας παίχτης μπορεί να τον αγοράσει μια φορά ανά γύρο. Δηλαδή αν ένας παίχτης επιθυμεί να δημιουργήσει μια κατασκευή η οποία απαιτεί για παράδειγμα δύο μονάδες ξύλου και ένας γείτονας διαθέτει μόνο μια κατασκευή που παρέχει μια μονάδα ξύλου, ο παίχτης δεν μπορεί να αγοράσει τη μία μονάδα δύο φορές ώστε να ολοκληρώσει τη κατασκευή του. Αν όμως ο άλλος γείτονας του διαθέτει και εκείνος πόρο που παρέχει ξύλο τότε μπορεί να αγοράσει τη μια μονάδα από τον ένα και την άλλη από τον άλλο. Τέλος ο πόρος ο οποίος αγοράστηκε

μπορεί μόνο να χρησιμοποιηθεί μια φορά στον τρέχων γύρο. Δεν υπάρχει για τους επόμενους, με την έννοια ότι καταναλώθηκε στην κατασκευή. Φυσικά η κάθε κατασκευή παρέχει τους πόρους της για όλη τη διάρκεια του παιχνιδιού. Τέλος οι πόροι που μπορούν να αγοραστούν είναι αυτοί των καφέ και των γκρι καρτών. Δεν μπορούν να αγοραστούν πόροι των κίτρινων καρτών ή οι πόροι που παρέχονται από τις κάρτες θαυμάτων.



Εικόνα 12: Η συγκεκριμένη κάρτα επιτρέπει την αγορά των πόρων που παρουσιάζονται από τον γείτονα στα αριστερά, εφόσον τον διαθέτει, με κόστος ενός νομίσματος αντί των δύο. Πηγή: Αλέξανδρος Βεκίλογλου

Τελευταία δυνατότητα αποτελεί η δωρεάν δημιουργία της κατασκευής μέσω των αλυσίδων. Η δυνατότητα αυτή παρέχεται από την δεύτερη περίοδο και μετά όπου αν έχει δημιουργηθεί μια κατασκευή η οποία συνδέεται με μια άλλη τότε το κόστος της δεύτερης το καλύπτει η πρώτη.

Εικόνα 13: Αν έχει κατασκευαστεί η κάρτα 'Θέατρο' τότε η κάρτα 'Κήποι' μπορεί να κατασκευαστεί χωρίς να καταβληθούν οι πόροι που απαιτεί. Πηγή: Αλέξανδρος Βεκίλογλου

2.3.1.2 Κατασκευή μέρους-κομματιού

Ο παίχτης έχει τη δυνατότητα να κατασκευάσει ένα μέρος-στάδιο του θαύματος. Για το σκοπό αυτό μπορεί να χρησιμοποιήσει οποιαδήποτε κάρτα και θα πρέπει να διαθέτει τους πόρους που απαιτεί η συγκεκριμένη φάση κατασκευής. Η κατασκευή σταδίου του θαύματος περιλαμβάνει τρεις φάσεις οι οποίες μπορούν να κατασκευαστούν σε οποιαδήποτε περίοδο. Μοναδικός περιορισμός η σειρά, δηλαδή ο παίχτης θα ξεκινήσει τη κατασκευή από την φάση που βρίσκεται στα αριστερά και κατόπιν να συνεχίσει στην επόμενη. Όταν κατασκευάσει μια φάση τοποθετεί την κάρτα στην φάση αυτή κάτω από τη κάρτα θαύμα με τέτοιο τρόπο ώστε να είναι ορατή η μισή κάρτα. Από τη στιγμή που ολοκληρώσει τη φάση ο πόρος που παρέχει η φάση είναι διαθέσιμος για όλη την υπόλοιπη διάρκεια του παιχνιδιού.

2.3.1.3 Πώληση της κάρτας

Στη περίπτωση που ο παίχτης δεν έχει τη δυνατότητα να δημιουργήσει τη κατασκευή καθώς είτε δεν διαθέτει ο ίδιος τους απαραίτητους πόρους (χρηματικούς ή υλικούς) ούτε οι γείτονες του, ή δεν επιθυμεί να δημιουργήσει τη συγκεκριμένη κατασκευή διότι μπορεί ήδη να διαθέτει τον πόρο που παρέχει, έχει τη δυνατότητα να πουλήσει την κάρτα. Στη περίπτωση αυτή τοποθετεί την κάρτα με την πίσω όψη πάνω στο σημείο του τραπεζιού όπου αφήνεται και η τελευταία κάρτα του γύρου και παίρνει ως αντάλλαγμα τρία κέρματα από το σωρό και τα τοποθετεί στην δεξιά πλευρά της κάρτας θαύματος, μαζί με τα υπόλοιπα, εφόσον διαθέτει. Με τον τρόπο αυτό ο παίχτης μπορεί να αναπτύξει τη στρατηγική του έχοντας πλήρη ελευθερία κινήσεων και επιλέγοντας κάθε φορά να πράξει την ενέργεια που εκείνος κρίνει ότι είναι η καταλληλότερη. Έτσι, μπορεί για παράδειγμα να ανταλλάξει μια κάρτα με χρήματα και να αγοράσει τον πόρο που επιθυμεί, στον επόμενο γύρο, ώστε να δημιουργήσει κάποια κατασκευή που θεωρεί πιο χρηστική.

2.3.2 Ολοκλήρωση του γύρου

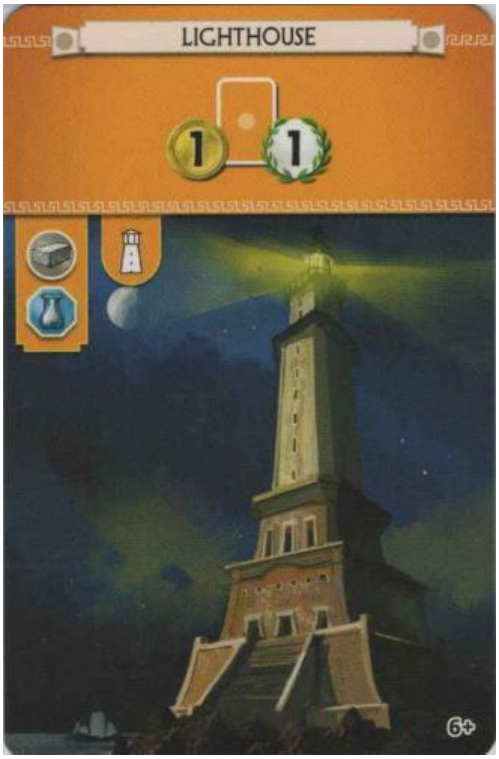
Μόλις ο παίχτης παίξει την κάρτα του παραδίδει τις υπόλοιπες κάρτες του στο γείτονα του και λαμβάνει από τον άλλο γείτονα τις δικές του. Επιλέγει εκ νέου μια κάρτα, την παίζει και περνά τις υπόλοιπες στο γείτονα για τον επόμενο γύρο. Η διαδικασία επαναλαμβάνεται για έξι συνολικά γύρους και ολοκληρώνεται με την απόρριψη της τελευταίας κάρτας. Όταν όλοι οι παίχτες παίξουν και την τελευταία τους κάρτα και απορρίψουν την άλλη τότε είναι η ώρα της στρατιωτικής αναμέτρησης η οποία και θα κρίνει τον νικητή του γύρου. Για το σκοπό αυτό ο κάθε παίχτης μετρά τα στρατιωτικά σύμβολα που περιέχονται στις κάρτες που έχει τοποθετήσει στη κάρτα θαύμα, στις κατασκευές του δηλαδή. Ακολούθως συγκρίνει τον αριθμό αυτό με τον αντίστοιχο των δύο γειτόνων του ξεχωριστά. Έτσι για κάθε γείτονα, στη περίπτωση που ο συνολικός αριθμός των στρατιωτικών συμβόλων είναι ίσος με αυτόν του γείτονα τότε υπάρχει ισοπαλία. Αν ο ένας έχει περισσότερα από τον άλλο τότε αυτός με τα περισσότερα λαμβάνει μια μάρκα της στρατιωτικής νίκης που αντιστοιχεί στη συγκεκριμένη περίοδο. Αντίθετα, ο παίχτης με τα λιγότερα λαμβάνει μια μάρκα στρατιωτικής ήττας, συγκεκριμένα αυτή με το -1. Ακολούθως ο παίχτης αναμετριέται με τον άλλο γείτονα του. Τέλος στη περίπτωση ισοπαλίας κανένας από τους δύο παίχτες δεν παίρνει κάτι.

2.4 Ολοκλήρωση του παιχνιδιού

Μετά το τέλος και των τριών φάσεων και την στρατιωτική αναμέτρηση της τρίτης φάσης το παιχνίδι ολοκληρώνεται και ξεκινά η διαδικασία που θα αναδείξει το νικητή. Αυτός καθορίζεται από τον αριθμό των νικητήριων πόντων που κάθε παίχτης έχει κερδίσει. Οι πόντοι αυτοί συγκεντρώνονται από διαφορετικά σημεία και για τη διευκόλυνση της διαδικασίας χρησιμοποιείται το αντίστοιχο έγγραφο πάνω στο οποίο ο κάθε παίχτης θα συμπληρώσει και θα αθροίσει τους πόντους που έχει κερδίσει. Πιο συγκεκριμένα η πρώτη γραμμή αντιστοιχεί στους πόντους που δίνουν οι φάσεις του θαύματος που έχει κατασκευάσει, ακολούθως αυτά των νομισμάτων, των στρατιωτικών συγκρίσεων, κ.ο.κ.

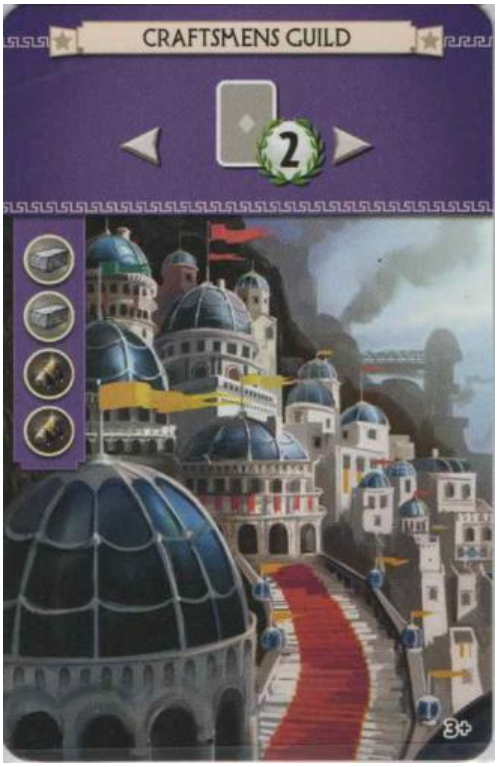
Εικόνα 14: Νικητήριος πόντος. Οι κάρτες χρώματος μπλε και μοβ παρέχουν τους πόντους αυτούς. Πηγή: Αλέξανδρος Βεκίλογλου

Συνεπώς αρχικά ο κάθε παίχτης αθροίζει τους νικητήριους πόντους που αναγράφονται στις φάσεις του θαύματος που έχει κατασκευάσει, πάνω στη κάρτα θαύματος που έχει. Ακολούθως σημειώνει ένα πόντο για κάθε τρία νομίσματα που έχει στη κατοχή του. Στη συνέχεια αθροίζει τους πόντους που αναγράφονται στις μπλε κάρτες που έχει τοποθετήσει στη κάρτα θαύμα του, στις κατασκευές δηλαδή που έχει δημιουργήσει. Οι πόντοι από τις στρατιωτικές αναμετρήσεις επίσης αθροίζονται και ισοδυναμούν με πόντους οι οποίοι θα μπορούν να είναι και αρνητικές τιμές. Ακολούθως αθροίζει αυτούς των κίτρινων καρτών σύμφωνα με τα σύμβολα που διευκρινίζονται στο πίνακα των συμβόλων. Πιο συγκεκριμένα η κάθε κίτρινη κάρτα παρουσιάζει ένα σύμβολο ο ρόλος του οποίου παρουσιάζεται στο φυλλάδιο επεξήγησης των συμβόλων.



Εικόνα 15: Αν κατασκευαστεί η συγκεκριμένη κάρτα ο παίχτης λαμβάνει ένα κέρμα και ένα νικητήριο πόντο για κάθε κίτρινη κάρτα που έχει κατασκευάσει. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως έρχεται η σειρά των πράσινων καρτών. Ειδικότερα για τις πράσινες κάρτες ο παίχτης θα πρέπει να ομαδοποιήσει τα ίδια σύμβολα και ανάλογα με τα πόσα σύμβολα υπάρχουν στην ομάδα λαμβάνει αντίστοιχο αριθμό πόντων. Πιο συγκεκριμένα αν διαθέτει ένα μόνο σύμβολο λαμβάνει ένα πόντο, τα δύο ίδια σύμβολα δίνουν 4 πόντους, τα 3 ίδια σύμβολα 9 πόντους, τα 4 ίδια σύμβολα 16 πόντους, τα 5 ίδια δίνουν 25 και τέλος τα 6 ίδια 36 πόντους. Επιπλέον για κάθε τρία διαφορετικά σύμβολα κερδίζει 7 πόντους. Τέλος αθροίζονται οι πόντοι που αναγράφονται στις μοβ κάρτες σύμφωνα και πάλι με την επεξήγηση των συμβόλων.



Εικόνα 16: Έχοντας κατασκευάσει ο παίχτης τη κάρτα αυτή αποκτά 2 πόντους για κάθε γκρι κάρτα που έχει κατασκευάσει. Πηγή: Αλέξανδρος Βεκίλογλου

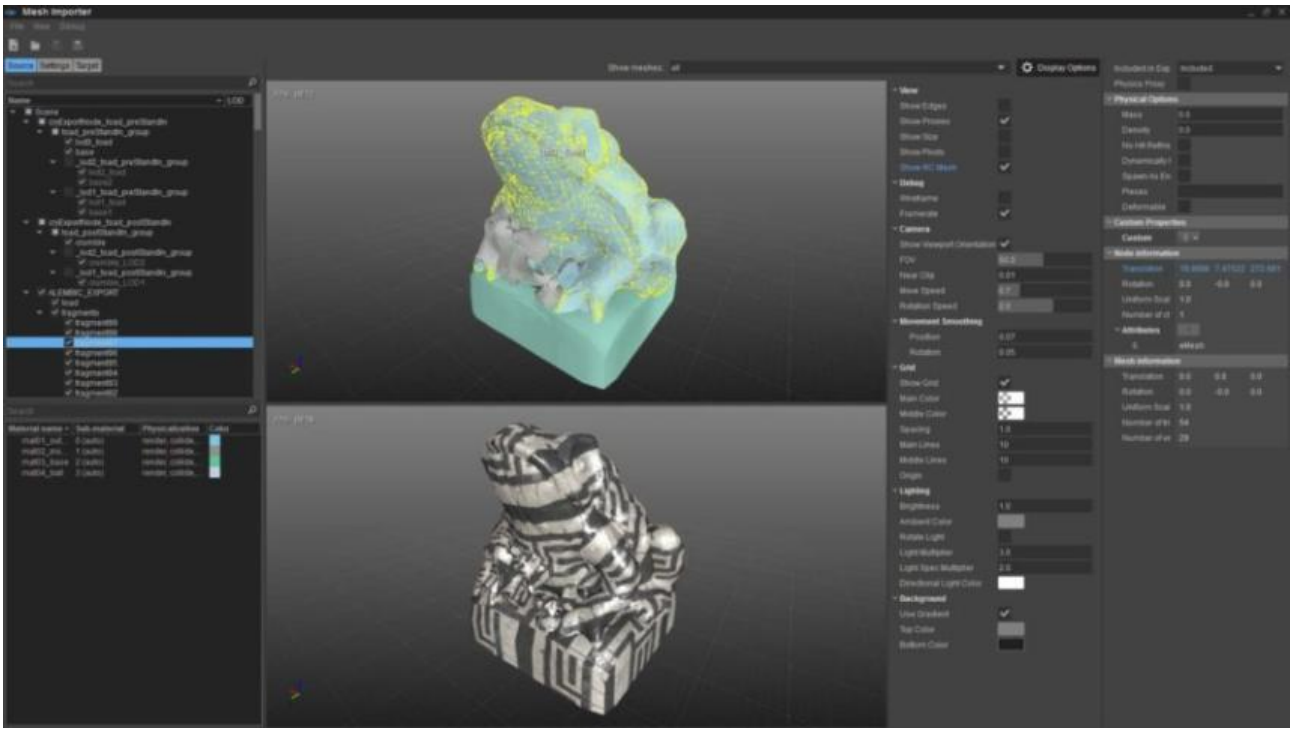
Ο παίχτης που συγκεντρώνει τους περισσότερους νικητήριους πόντους αναδεικνύεται νικητής.

3 ΚΕΦΑΛΑΙΟ 3ο : Περιγραφή της μηχανής Unity

Η κατασκευή μιας εφαρμογής παιχνιδιού για τον υπολογιστή απαιτεί γνώσεις σε ιδιαίτερα μεγάλο εύρος επιστημονικών πεδίων τα οποία δεν περιορίζονται μόνο γύρω από τον ηλεκτρονικό υπολογιστή αλλά επεκτείνονται και σε άλλους τομείς. Πιο συγκεκριμένα, ειδικά για τον ηλεκτρονικό υπολογιστή ο δημιουργός της εφαρμογής του παιχνιδιού θα πρέπει να έχει πλήρη εικόνα του τρόπου με τον οποίο λειτουργούν τα λειτουργικά συστήματα των διαφορετικών πλατφορμών στις οποίες θα λειτουργήσει η εφαρμογή (Windows, Macintosh, Linux, Android, iOS, κλπ.), τα περιφερικά που χρησιμοποιεί η κάθε πλατφόρμα (χειριστήρια, οθόνες αφής, κτλ.), του τρόπου με τον οποίο γίνεται η διαχείριση της μνήμης, των αρχείων, των δικαιωμάτων κλπ. Επιπλέον ο κατασκευαστής θα πρέπει, πέρα από τη συγγραφή του κώδικα να γνωρίζει πώς θα δημιουργήσει τα διάφορα μοντέλα, πώς θα κατασκευαστούν οι υφές τους, να έχει καλλιτεχνικές ικανότητες ώστε το παιχνίδι να είναι αισθητικά αποδεκτό κλπ. Εύκολα καθίσταται αντιληπτό στον αναγνώστη ότι η προσέγγιση αυτή δεν είναι αποδοτική καθώς απαιτεί η ομάδα ανάπτυξης του παιχνιδιού να αποτελείτε από μεγάλο αριθμό μελών τα οποία θα διαθέτουν το κάθε ένα μέλος ξεχωριστά τη δική του εξειδίκευση σε συγκεκριμένο τομέα. Ο μεγάλος αριθμός μελών από την άλλη πλευρά εγείρει επιπλέον προβλήματα επικοινωνίας, συγχρονισμού, εκτόξευση του κόστους κατασκευής κλπ.

Προς την αντιμετώπιση των θεμάτων αυτών οι σχεδιαστές λογισμικού δημιούργησαν τις μηχανές παιχνιδιών. Οι μηχανές αυτές, οι οποίες ουσιαστικά αποτελούν λογισμικά, αποτελούν ένα επίπεδο επικοινωνίας του κατασκευαστή με την πλατφόρμα, αποκρύπτοντας από αυτόν τις περιττές λεπτομέρειες. Με τον τρόπο αυτό ο κατασκευαστής δηλώνει για μια πλατφόρμα (ή πλατφόρμες) επιθυμεί να κατασκευάσει το λογισμικό του, γράφει τον απαιτούμενο κώδικα και η μηχανή αναλαμβάνει να εκτελέσει τις απαραίτητες λειτουργίες ώστε ο κώδικάς να μπορεί να εκτελεστεί στη κάθε πλατφόρμα. Με τον τρόπο αυτό ο κατασκευαστής αφοσιώνεται στο έργο του, δεν ασχολείται με θέματα που ξεφεύγουν από το αντικείμενο του με αποτέλεσμα να είναι αρκετά πιο αποδοτικός. Ο αριθμός των ατόμων που συμμετέχουν στην ομάδα ανάπτυξης μειώνεται σημαντικά και παραμένουν μόνο τα άτομα που απαιτούνται καθαρά για την δημιουργία του παιχνιδιού καθ’ εαυτού. Έτσι επιτυγχάνεται μια σημαντική μείωση της πολυπλοκότητας του όλου εγχειρήματος, του κόστους κατασκευής και παράλληλα και του χρόνου ολοκλήρωσης. Μεταξύ των πιο καθιερωμένων μηχανών παιχνιδιών ξεχωρίζουν η Cry Engine, η Unreal, η Unity, η Pygame καθώς και άλλες.

3.1 CryEngine



Εικόνα 17: Cry Engine. Πηγή: www.codemotion.com

Η μηχανή αποτελεί δημιούργημα της γερμανικής εταιρίας Crytec και πρωτοεμφανίστηκε τον Μάιο του 2002. Από τα πρώτα προϊόντα της αποτελεί το ιδιαίτερα δημοφιλές παιχνίδι Far Cry το οποίο αποτελούσε ένα παιχνίδι τύπου First Person Shooter. Πιο συγκεκριμένα το παιχνίδι παρουσίαζε τον παίχτη ως ένα ήρωα μέσα σε ένα εικονικό περιβάλλον με τέτοιο τρόπο ώστε ο παίχτης παρουσιαζόταν να είναι μέσα στο παιχνίδι όπως ακριβώς στο πραγματικό κόσμο. Διέθετε διάφορα όπλα, είχε τη δυνατότητα να εξοπλιστεί και με επιπλέον σε μια προσπάθεια να εκτελέσει διάφορες αποστολές. Το λογισμικό περιλαμβάνει ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών εξοπλισμένο με εργαλεία για την συγγραφή και αποσφαλμάτωση του κώδικα, την κατασκευή των μοντέλων, των σκηνών αλλά και των υλικών που θα χρειαστούν για να ντύσουν τα διάφορα αντικείμενα. Η εταιρία προσφέρει τη μηχανή δωρεάν για περιορισμένη χρήση. Πιο συγκεκριμένα η χρήση είναι δωρεάν αν το προϊόν δεν ξεπεράσει ως κέρδος τα 5000 δολάρια Αμερικής. Στη περίπτωση που ξεπεράσει το ύψος αυτό τότε θα πρέπει να καταβληθεί στην εταιρία το 5% των κερδών. Από το Απρίλιο του 2015 η μηχανή αποτελεί ιδιοκτησία της Amazon.

3.2 Unreal Engine

Εικόνα 18: Unreal Engine. Πηγή: academyofanimatedart.com

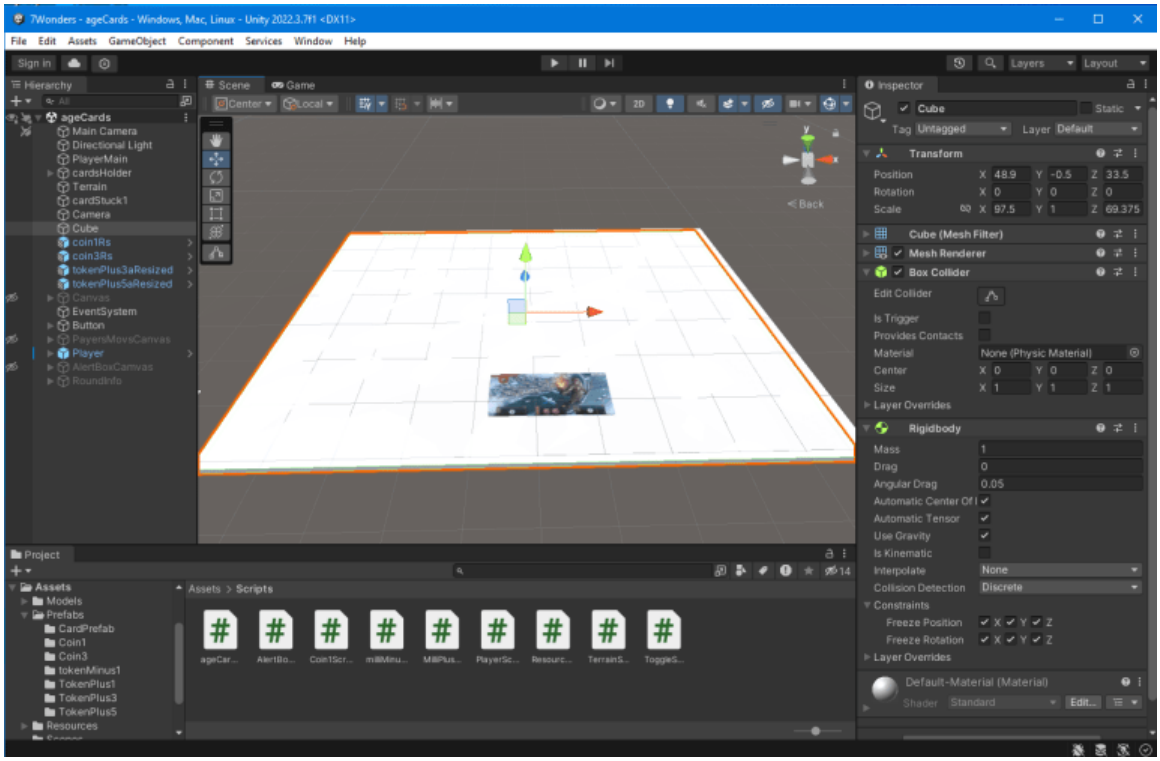
Η μηχανή αποτελεί αντίστοιχη πρόταση όπως αυτής που παρουσιάστηκε στη προηγούμενη παράγραφο και αποτελεί προϊόν της εταιρίας Epic Games. Αποτελεί παλαιότερη μηχανή στο χώρο από την CryEngine καθώς παρουσιάστηκε το 1998 με το πρώτο προϊόν της το Unreal το οποίο επίσης αποτελεί First Person Souter. Η μηχανή μπορεί να κατασκευάσει εφαρμογές για την πλειονότητα των επικρατέστερων πλατφόρμων όπως Windows, έξυπνα κινητά τηλέφωνα καθώς και παιχνιδομηχανές. Σαν γλώσσα προγραμματισμού πλέον χρησιμοποιεί τη C++ πράγμα που την καθιστά λιγότερο δημοφιλή από την Unity η οποία χρησιμοποιεί την πιο φιλική προς το χρήστη C#. Ενσωματώνει και αυτή, στο ολοκληρωμένο περιβάλλον της, τη πλειονότητα των εργαλείων που θα χρειαστεί η ομάδα ανάπτυξης για την κατασκευή της εφαρμογής. Από το 2015 και έπειτα είναι πλέον ανοιχτού κώδικα και είναι αυτός είναι διαθέσιμος από το δημοφιλές αποθετήριο κώδικα GitHub. Όπως και η Cry Engine, στη περίπτωση της επαγγελματικής χρήσης θα πρέπει να καταβληθεί στην εταιρία το 5% των κερδών ώστε η μηχανή να μπορεί να χρησιμοποιηθεί.

3.3 Pygame

Εικόνα 19: Pygame. Πηγή: Αλέξανδρος Βεκίλογλου

Η pygame διαφέρει από τις υπόλοιπες μηχανές που παρουσιάστηκαν καθώς επί της ουσίας δεν αποτελεί μηχανή αλλά μάλλον βιβλιοθήκη. Πιο συγκεκριμένα δεν αποτελεί ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού που περιέχει κειμενογράφο, μεταγλωττιστή, σχεδιαστικά εργαλεία κλπ., όπως τα υπόλοιπα λογισμικά που παρουσιάστηκαν, αλλά απλά μια συλλογή από αντικείμενα. Τη βιβλιοθήκη θα πρέπει να την εισάγει ο προγραμματιστής στο δικό του ολοκληρωμένο περιβάλλον ανάπτυξης (όπως το eclipse για παράδειγμα) το οποίο περιέχει όλα τα απαραίτητα εργαλεία (κειμενογράφο, μεταγλωττιστή, κλπ.) και κατόπιν να το χρησιμοποιήσει. Πρόκειται για μια δωρεάν βιβλιοθήκη η οποία μπορεί να λειτουργήσει σε όλες τις πλατφόρμες καθώς η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η Python. Στερείται βασικών εργαλείων που οι άλλες μηχανές παρέχουν όμως αποτελεί τη βάση που οι όλες οι μηχανές χρησιμοποιούν. Με άλλα λόγια αν κάποιος επιθυμεί να ασχοληθεί με την ανάπτυξη εφαρμογών τότε η συγκεκριμένη βιβλιοθήκη θα του παράσχει τη βασική γνώση που απαιτείται ενώ οι άλλες μηχανές συχνά την παρακάμπτουν.

3.4 Unity



Εικόνα 20: Unity Engine. Πηγή: Αλέξανδρος Βεκίλογλου

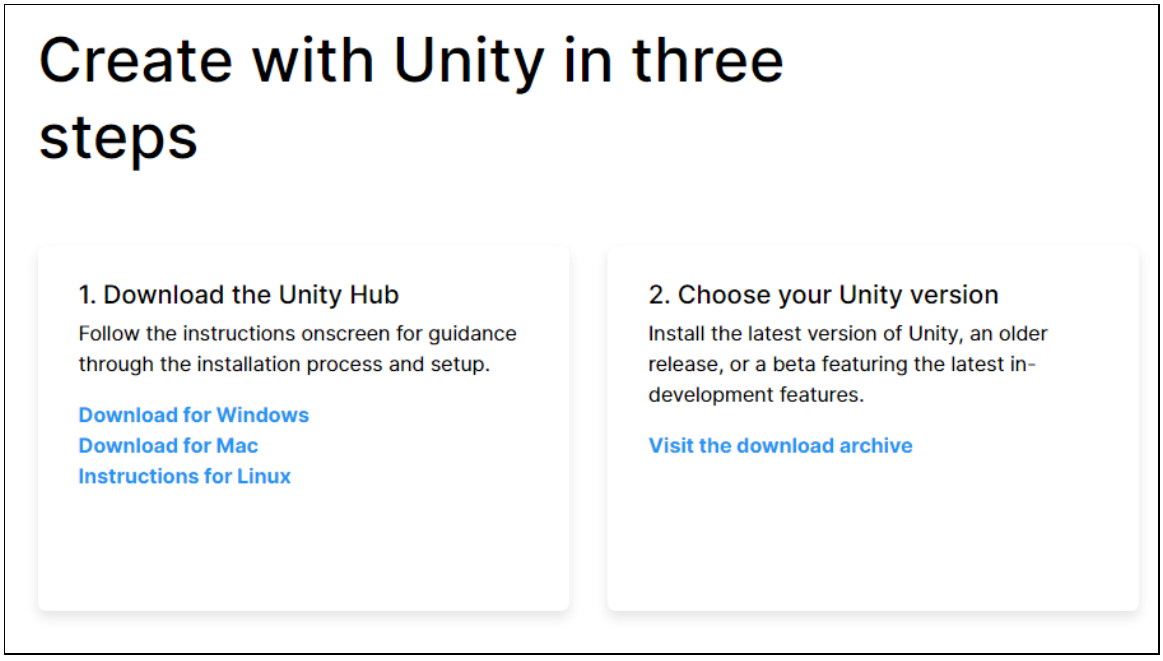
Μαζί με την Unreal αποτελούν δύο από τις πλέον καταξιωμένες μηχανές παιχνιδιών. Παρουσιάστηκε αρχικά το 2005 από την ομώνυμη εταιρία και από τότε έχει κερδίσει την αποδοχή μεγάλου αριθμού κατασκευαστών παιχνιδιών. Από τους βασικότερους λόγους που την καθιέρωσαν στη κατηγορία αυτή αποτελεί το γεγονός ότι είναι ιδιαίτερα φιλική προς το χρήστη, συγκεντρώνει μεγάλο αριθμό απαραίτητων εργαλείων που διευκολύνουν την αποστολή του και η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η C#. Η γλώσσα αυτή είναι απαλλαγμένη από την πολυπλοκότητα που χαρακτηρίζει την ανταγωνίστρια της C++, που χρησιμοποιεί η Unreal, με όλα τα μειονεκτήματα που συνοδεύει το γεγονός αυτό. Το γεγονός αυτό, ότι η μηχανή είναι πιο απλή στη χρήση την καθιστά

κατάλληλη για τη χρήση από ένα μόνο άτομο. Πιο συγκεκριμένα μέσω της μηχανής αυτής ακόμα και ένα άτομο μόνο μπορεί να αναπτύξει μια αξιοπρεπέστατη εφαρμογή, σε αντίθεση με τις άλλες μηχανές που απαιτούν πιο πολυπληθείς ομάδες όπου κάθε μέλος διαθέτει και ξεχωριστή εξειδίκευση. Τέλος υποστηρίζει την ανάπτυξη εφαρμογών για όλες τις πλατφόρμες και ακολουθεί το ίδιο μοντέλο χρέωσης με τις άλλες μηχανές που παρουσιαστήκαν στις προηγούμενες παραγράφους.

Η συγκεκριμένη μηχανή, για τους λόγους που προαναφέρθηκαν, επιλέχθηκε να χρησιμοποιηθεί για την ανάπτυξη της παρούσας εφαρμογής. Για το σκοπό αυτό στις επόμενες παραγράφους θα παρουσιαστούν τα απαραίτητα βήματα για την εγκατάσταση και χρήση της ώστε να εξοικειωθεί ο αναγνώστης με τα θέματα αυτά.

3.4.1 Λήψη και εγκατάσταση της μηχανής

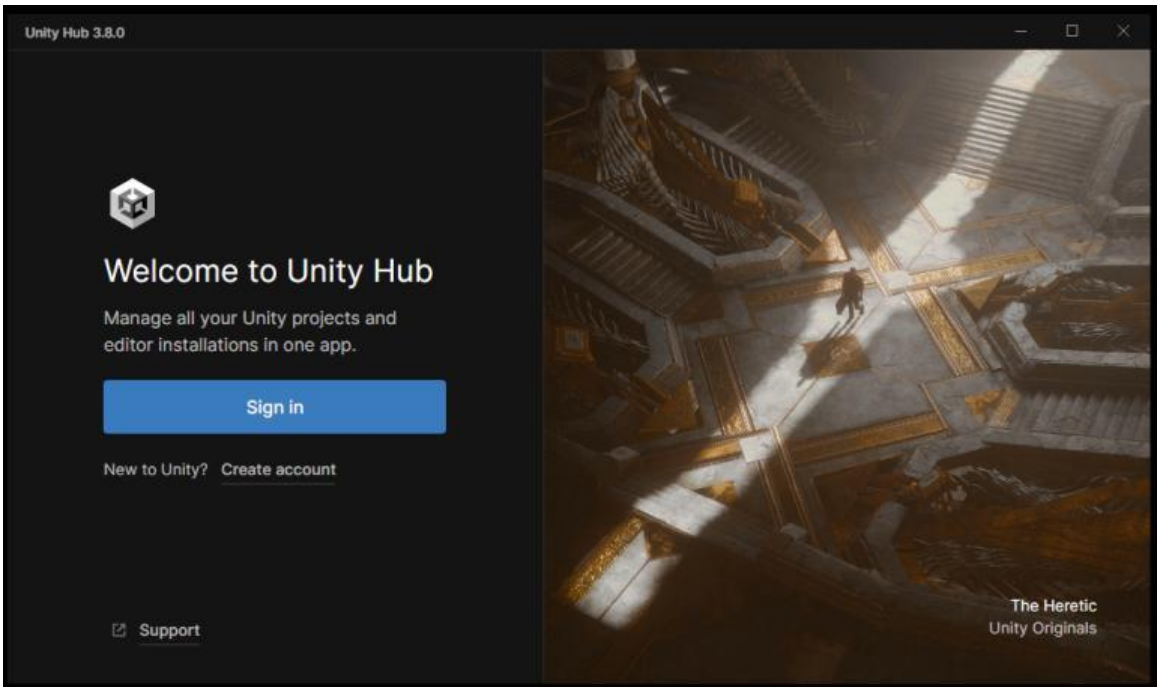
Η κατασκευή ενός παιχνιδιού περιλαμβάνει την δημιουργία των αντικειμένων που θα αποτελούν τους ήρωες και το περιβάλλον και ακολούθως την συγγραφή του απαραίτητου κώδικα ο οποίος θα δίνει ζωή στα αντικείμενα. Πιο συγκεκριμένα μέσω του κώδικα ο προγραμματιστής μπορεί να περιγράψει στην εφαρμογή τι θα κάνει το κάθε αντικείμενο, πχ ένα όπλο να εκτοξεύει ένα βλήμα όταν πατιέται το αριστερό κουμπί του ποντικιού, κλπ. Η κατασκευή των μοντέλων και των λοιπών αντικειμένων του παιχνιδιού υλοποιείται μέσα από το ολοκληρωμένο περιβάλλον της Unity ενώ η συγγραφή του κώδικα γίνεται μέσα από διαφορετικό ολοκληρωμένο περιβάλλον ανάπτυξης. Η Unity προτείνει να χρησιμοποιείται το Visual Studio της Microsoft. Για το σκοπό αυτό προκειμένου ο χρήστης να εγκαταστήσει τη μηχανή στον υπολογιστή του εγκαθιστά ένα ενδιάμεσο λογισμικό, το Unity Hub, το οποίο αναλαμβάνει να εγκαταστήσει τη Unity μηχανή, το Visual Studio καθώς και τα υπόλοιπα συνοδευτικά λογισμικά, ενώ παράλληλα αναλαμβάνει και την κατάλληλη ενημέρωση τους όποτε αυτό κρίνεται απαραίτητο.



Εικόνα 21: Λήψη του Unity Hub. Πηγή: https://unity.com/download#how-get-started

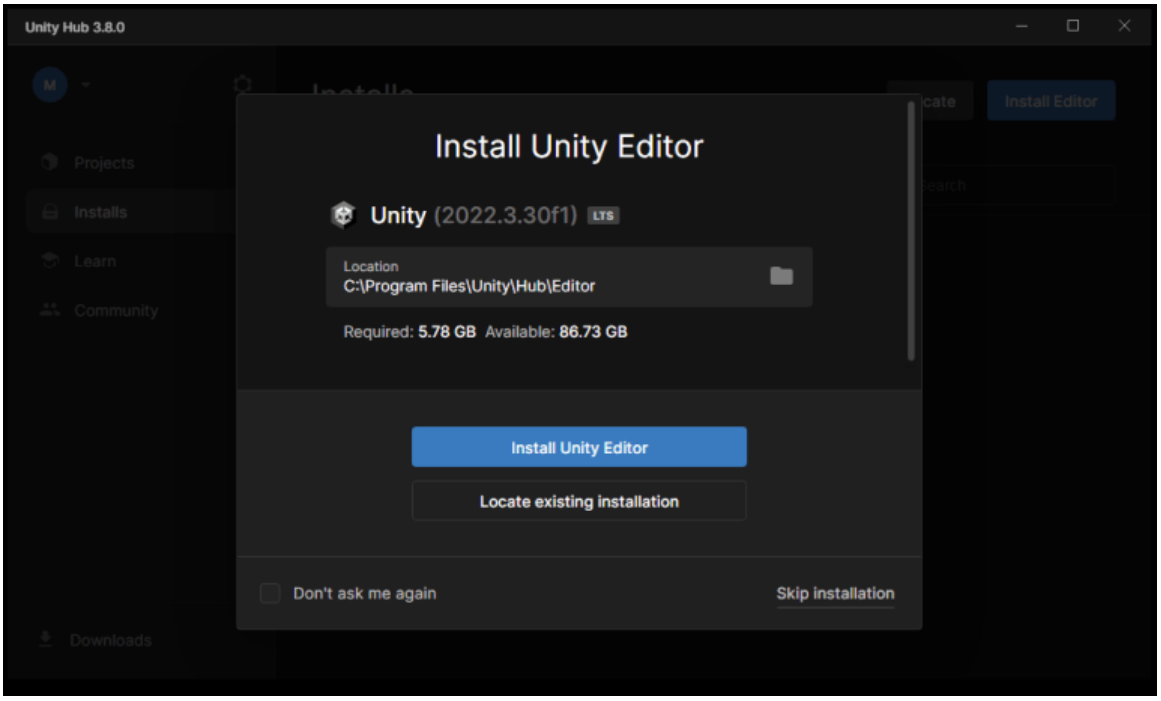
Συνεπώς για την εγκατάσταση του λογισμικού ο χρήστης θα επισκεφθεί τον επίσημο ιστότοπο της εταιρίας Unity.com και από εκεί θα μεταβεί στην ενότητα λήψεις (Downloads). Από εκεί θα κατεβάσει την έκδοση της εφαρμογής Unity Hub που είναι κατάλληλη για την πλατφόρμα του. Μετά

τη λήψη του αρχείου ο χρήστης θα προχωρήσει στην εγκατάσταση του. Μόλις η εγκατάσταση ολοκληρωθεί η εφαρμογή ξεκινά και ζητά από το χρήστη να δημιουργήσει λογαριασμό ή να εισάγει τα στοιχεία του εφόσον διαθέτει.

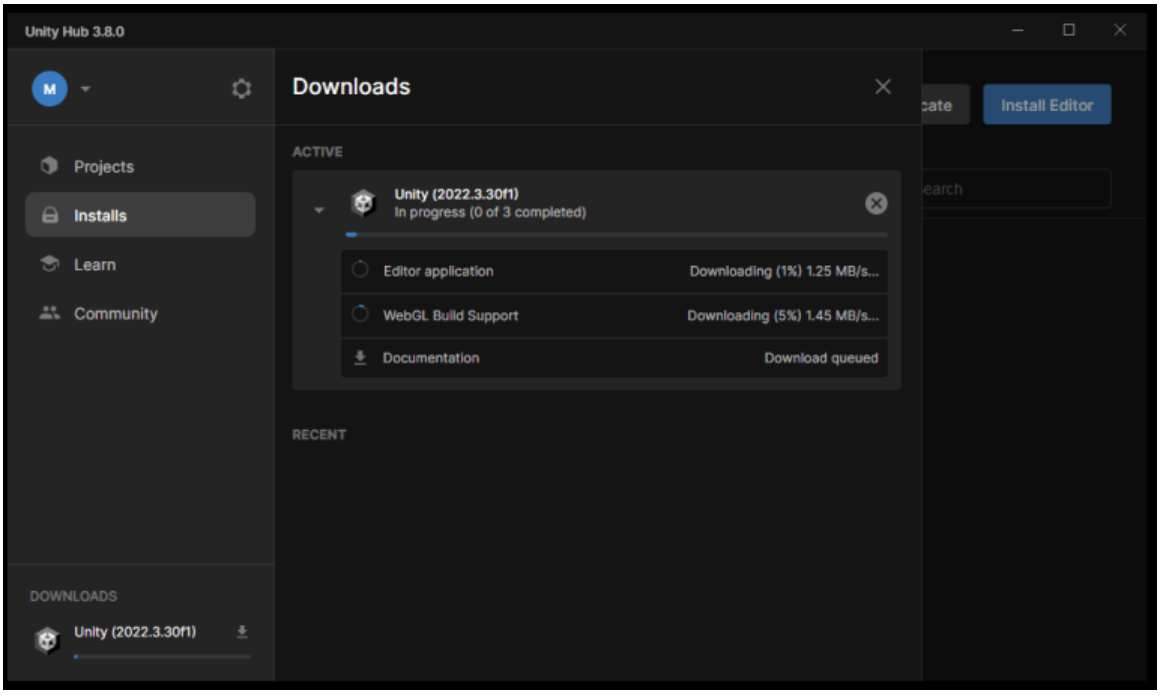


Εικόνα 22: Unity Hub. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις η σύνδεση του χρήστη ολοκληρωθεί η εφαρμογή ξεκινά την εγκατάσταση των απαιτούμενων λογισμικών. Αρχικά εγκαθιστά τον επεξεργαστή κειμένου (Editor).

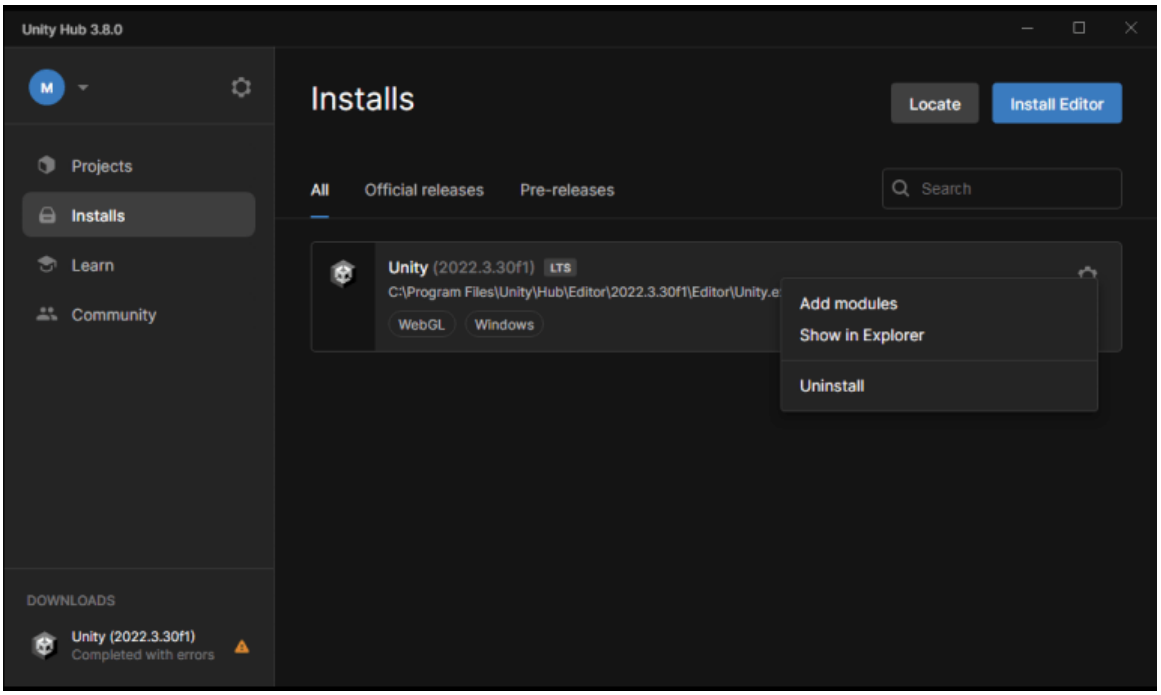


Εικόνα 23: Επιλογή εγκατάστασης επεξεργαστή κειμένου. Πηγή: Αλέξανδρος Βεκίλογλου

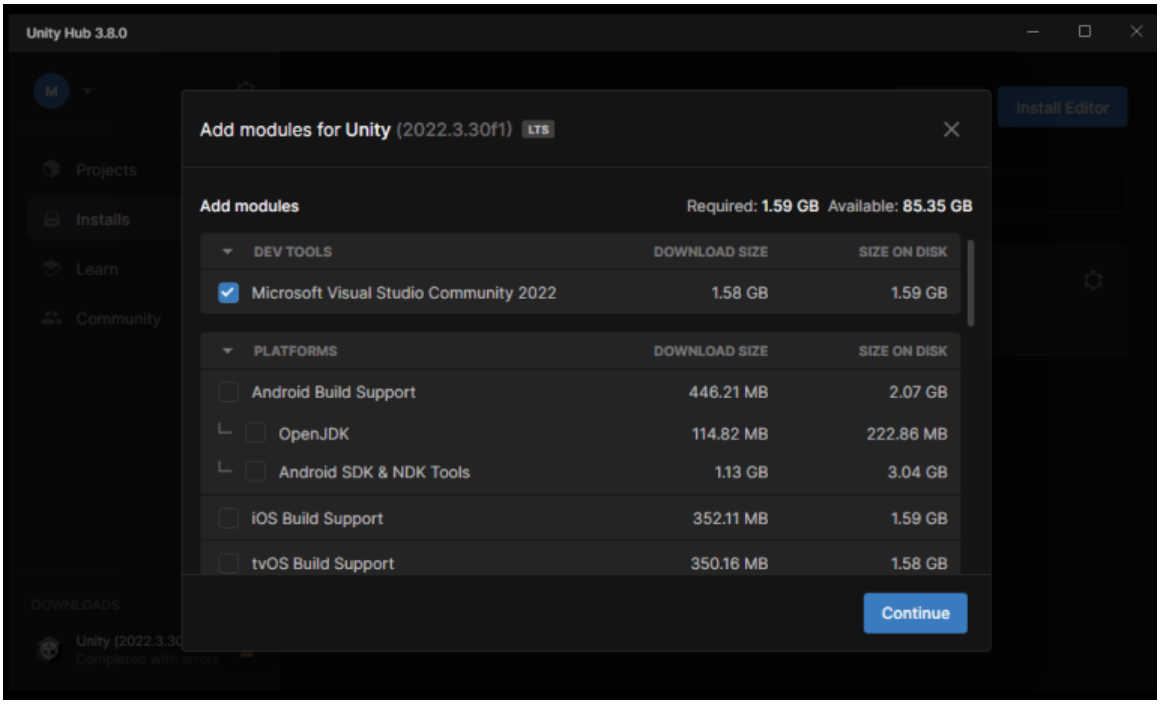


Εικόνα 24: Εγκατάσταση λογισμικού. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις ολοκληρωθεί η εγκατάσταση του λογισμικού ο χρήστης θα πρέπει να εγκαταστήσει το ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού Visual Studio το οποίο θα χρησιμοποιήσει για τη σύνταξη του κώδικά. Η έκδοση του λογισμικού εξαρτάται από τον editor της Unity που έχει εγκατασταθεί και για το σκοπό αυτό ο χρήστης θα πρέπει να έχει επιλέξει την έκδοση της Unity που έχει εγκατασταθεί από την καρτέλα Installs, από εκεί να επιλέξει το κουμπί με το γρανάζι που εμφανίζει το αναδυόμενο μενού. Από εκεί θα επιλέξει Install Modules ώστε να εγκαταστήσει τα λογισμικά που επιθυμεί και που αρμόζουν στην έκδοση της Unity που έχει εγκαταστήσει.

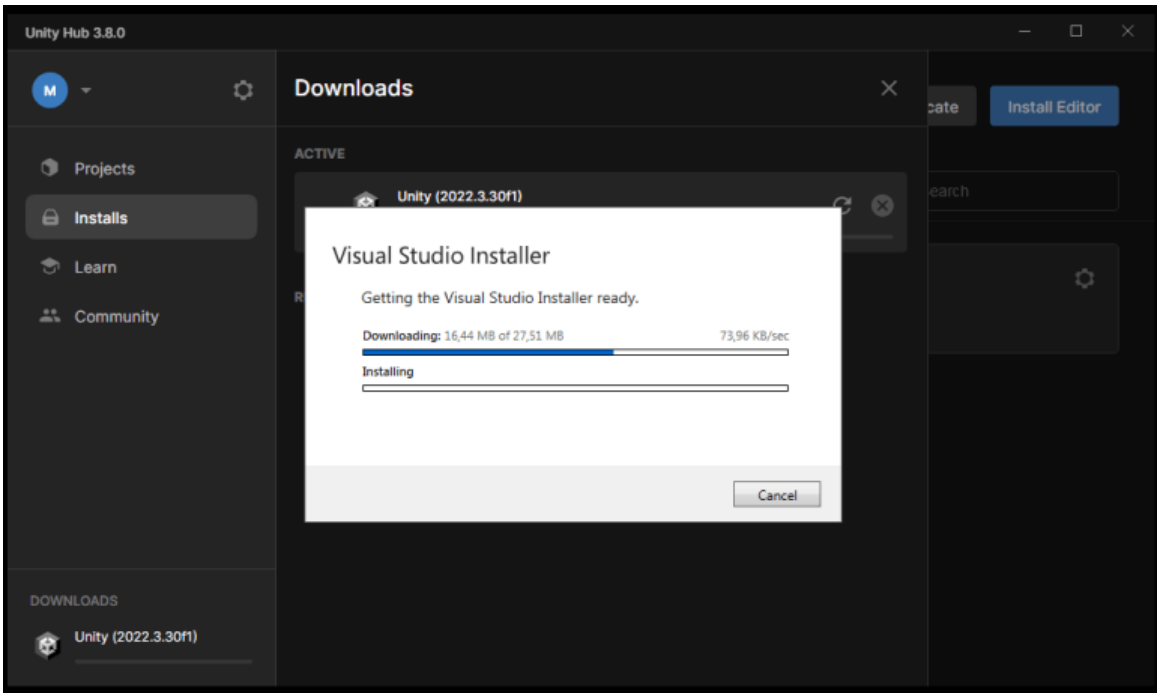


Εικόνα 25: Εγκατάσταση συνοδευτικού λογισμικού. Πηγή: Αλέξανδρος Βεκίλογλου

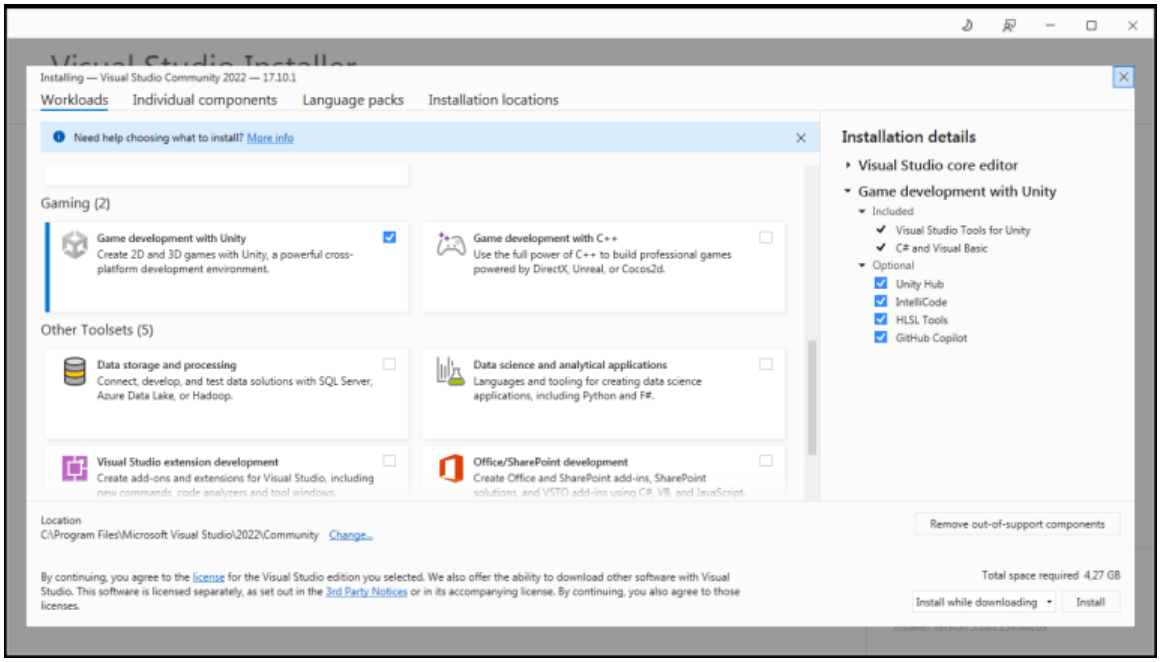


Εικόνα 26: Επιλογή των συνοδευτικών λογισμικών. Πηγή: Αλέξανδρος Βεκίλογλου

Όπως αναφέρθηκε και σε προηγούμενη παράγραφο η Unity υποστηρίζει πληθώρα πλατφόρμων. Ο προγραμματιστής δηλαδή μπορεί να αναπτύξει την εφαρμογή του, πέρα από Windows και για Android, iOS κλπ. Από την επιλογή αυτή θα επιλέξει το κατάλληλο περιβάλλον ανάπτυξης για την κάθε πλατφόρμα. Για παράδειγμα αν επιθυμεί να αναπτύξει εφαρμογή για το λειτουργικό Android θα πρέπει να επιλέξει και το Android Build Support.



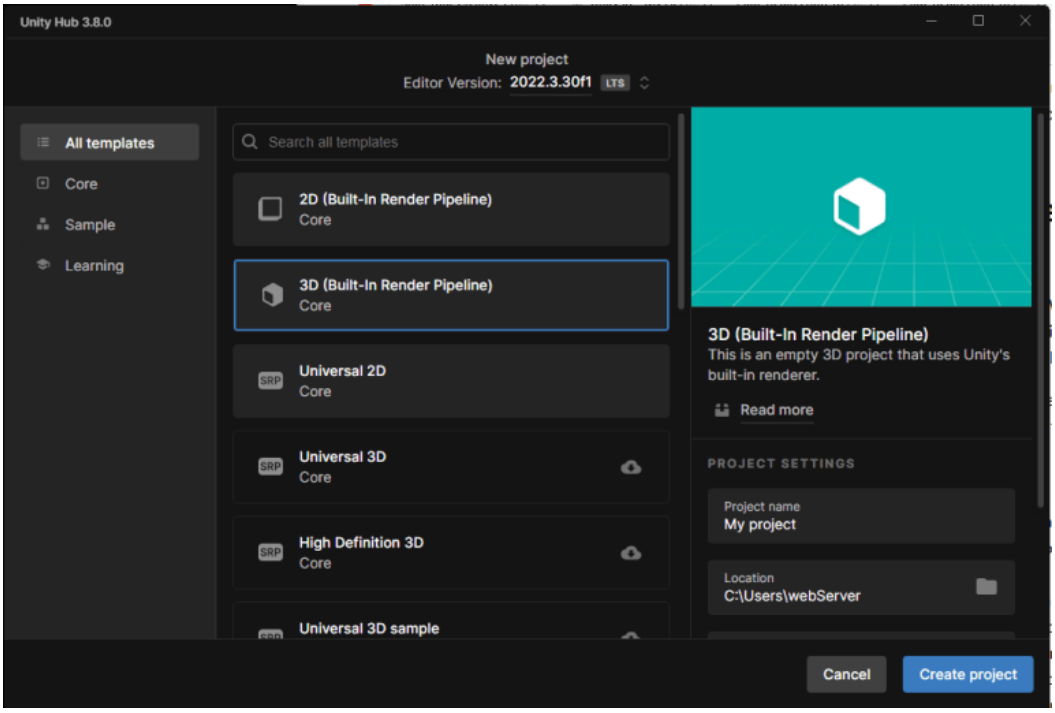
Εικόνα 27: Εγκατάσταση του Microsoft Visual Studio. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 28: Εγκατάσταση του συνοδευτικού λογισμικού για το Microsoft Visual Studio. Πηγή: Αλέξανδρος Βεκίλογλου

Όταν ολοκληρωθεί η εγκατάσταση του Visual Studio εμφανίζεται το παράθυρο που επιτρέπει στο χρήστη να εγκαταστήσει το συνοδευτικό λογισμικό που απαιτείται για το Visual Studio. Πιο συγκεκριμένα, για να μπορέσει να επικοινωνήσει το Visual Studio με τη Unity θα πρέπει αυτό να διαθέτει το απαιτούμενο λογισμικό. Αυτό επιλέγεται μεταξύ των υπολοίπων τσεκάροντας το κουτάκι που υπάρχει στα δεξιά του κάθε πακέτου.

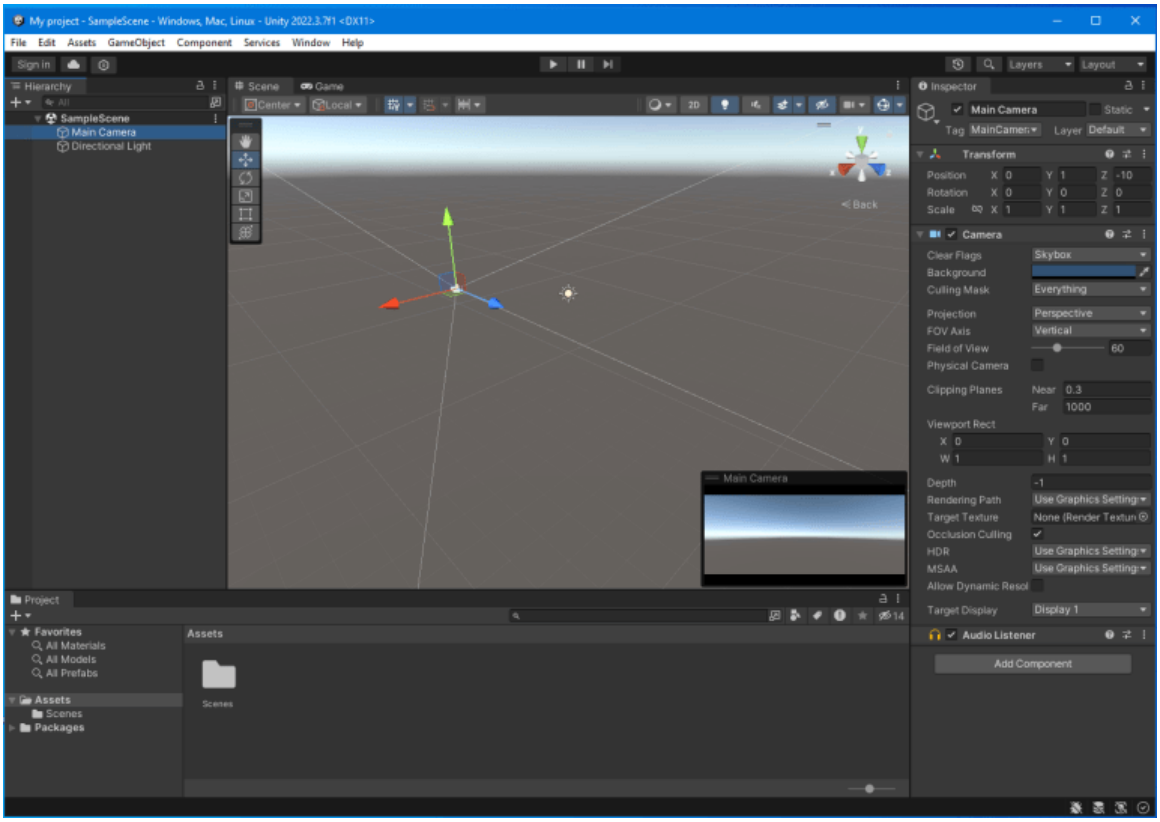
Μόλις η εγκατάσταση ολοκληρωθεί ο χρήστης είναι πλέον έτοιμος να ξεκινήσει ένα νέο Project.



Εικόνα 29: Δημιουργία νέου Project. Πηγή: Αλέξανδρος Βεκίλογλου

Για τη δημιουργία του νέου project ο χρήστης θα πρέπει να επιλέξει το είδος του. Μεταξύ των διαθέσιμων επιλογών είναι δισδιάστατη ή τρισδιάστατη εφαρμογή, εφαρμογή επαυξημένης πραγματικότητας, εικονικής καθώς και άλλα. Στη παρούσα εργασία θα δημιουργηθεί μια εφαρμογή τριών διαστάσεων κάνοντας χρήση του Render Pipeline της Unity καθώς είναι πιο απλός και παρόλο που παρέχει μειωμένη λειτουργικότητα σε σχέση με τον Universal οι δυνατότητες του είναι υπέρ αρκετές για την παρούσα εργασία.

3.4.2 Γνωριμία με το περιβάλλον

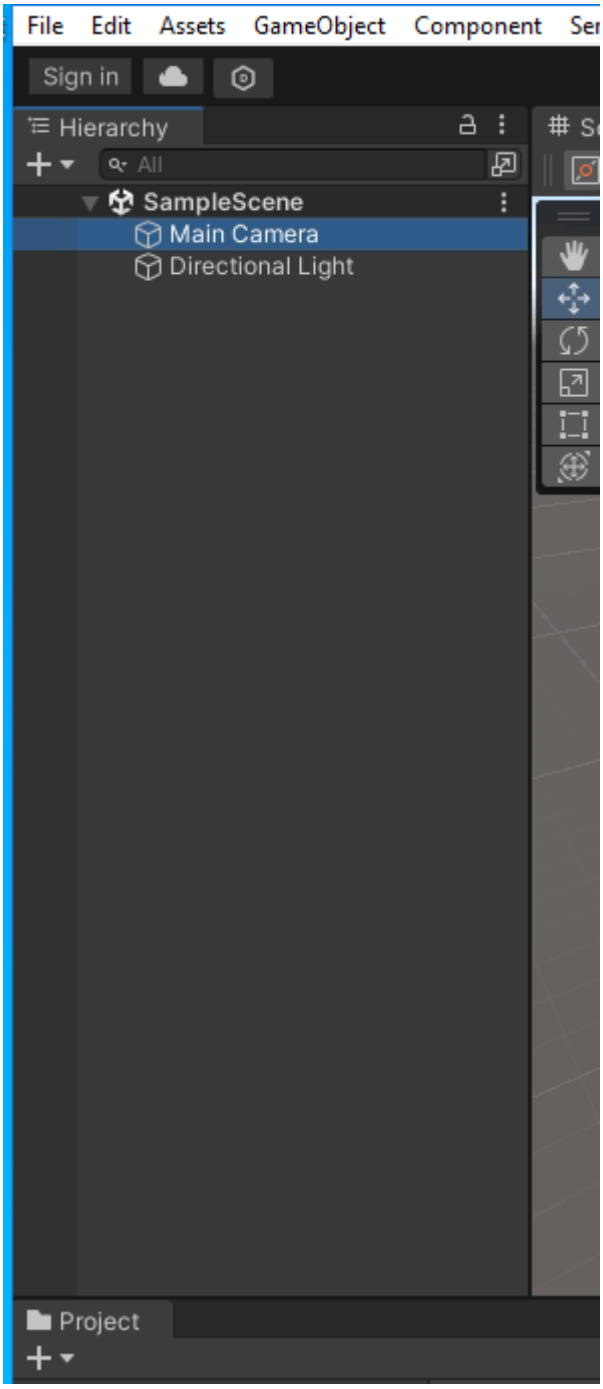


Εικόνα 30: Το περιβάλλον της Unity. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις η εφαρμογή ξεκινήσει εμφανίζει στο χρήστη το παράθυρο της προηγούμενης εικόνας. Στο περιβάλλον αυτό ο χρήστης θα διακρίνει πάνω αριστερά το μενού επιλογών και από κάτω στο μέσο του παραθύρου τα τρία κουμπιά που χρησιμοποιούνται για το παίξιμο της συγκεκριμένης σκηνής. Η υπόλοιπη οθόνη χωρίζεται σε 4 βασικά μέρη.

3.4.2.1 Η καρτέλα Ιεραρχίας (Hierarchy)

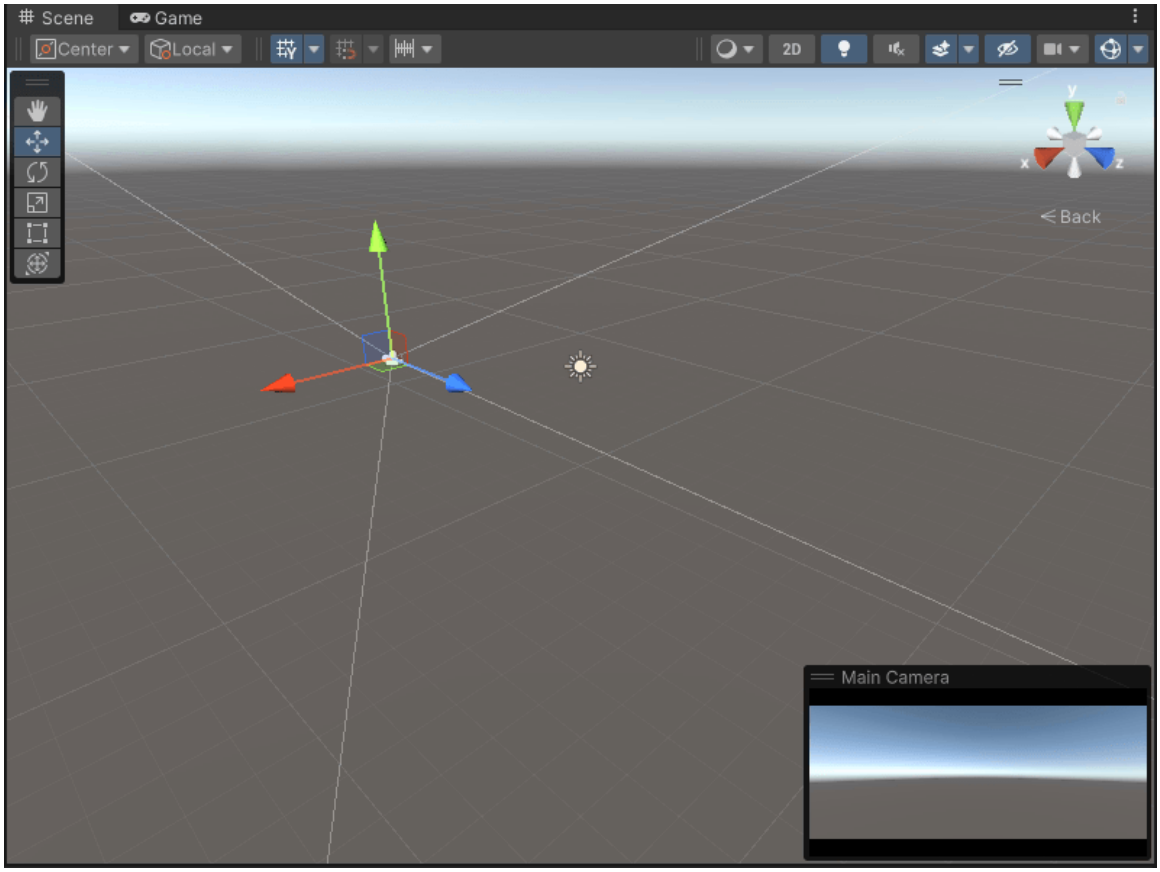
Στη καρτέλα αυτή εμφανίζονται τα αντικείμενα που υπάρχουν στη σκηνή. Αρχικά στη σκηνή η εφαρμογή δημιουργεί ένα αντικείμενο κάμερας το οποίο χρησιμοποιείται για να δείχνει την σκηνή και ένα φως τύπου κατευθυντικού, το οποίο φωτίζει τη σκηνή για να μπορεί να τη δει ο χρήστης. Είναι προφανές ότι τα αντικείμενα αυτά τα δημιουργεί μόνη της εφαρμογή διότι εάν δεν τα έφτιαχνε ο χρήστης δεν θα μπορούσε να δει τι έχει φτιάξει. Φυσικά ο χρήστης μπορεί να διαγράψει τα αντικείμενα αυτά αν δε τα χρειάζεται, να δημιουργήσει νέα κλπ. Όλα τα αντικείμενα ανήκουν σε μία σκηνή το όνομα της οποίας εμφανίζεται πάνω πάνω (SampleScene). Σαν σκηνή ορίζεται μία πίστα του παιχνιδιού, ένα ανεξάρτητο δηλαδή τμήμα της εφαρμογής.



Εικόνα 31: Η καρτέλα Hierarchy. Πηγή: Αλέξανδρος Βεκίλογλου

3.4.2.2 Η καρτέλα σκηνής (Scene)

Τα αντικείμενα που δημιουργεί ο χρήστης, για τη συγκεκριμένη σκηνή, εμφανίζονται στην καρτέλα της σκηνής. Από εκεί μπορεί να δει πως είναι τοποθετημένα στο χώρο, να τα επιλέξει και να εκτελέσει διάφορες ενέργειες όπως να τα μετακινήσει κλπ. Για να μπορεί να προσανατολιστεί στη καρτέλα πάνω δεξιά εμφανίζεται το σύστημα των αξόνων για να ξέρει ο χρήστης από πια πλευρά κοιτά τη σκηνή του. Στην επόμενη εικόνα ο αναγνώστης θα δει τη σκηνή στην οποία υπάρχει μία κάμερα (η οποία είναι και επιλεγμένη) και το φως που αναφέρθηκε στη προηγούμενη παράγραφο.

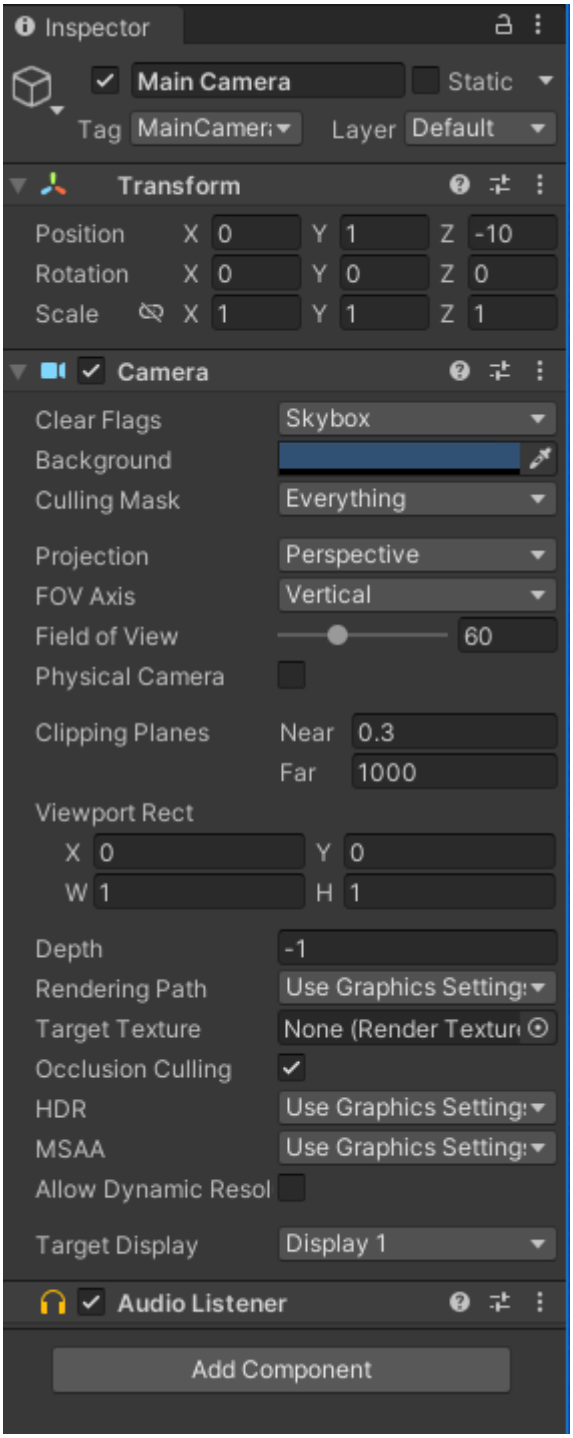


Εικόνα 32: Η καρτέλα Scene. Πηγή: Αλέξανδρος Βεκίλογλου

Δίπλα στη καρτέλα αυτή ο χρήστης θα εντοπίσει τη καρτέλα Game. Η καρτέλα αυτή εμφανίζει την εικόνα μέσα από τα 'μάτια' της κάμερας. Η διαφορά των δυο είναι ότι η κάμερα εμφανίζει τη σκηνή όπως τη βλέπει ο παίχτης, δηλαδή το τελικό προϊόν. Αντίθετα η καρτέλα Scene εμφανίζει την σκηνή χωρίς να έχει γίνει render.

3.4.2.3 Η καρτέλα Inspector

Το κάθε αντικείμενο που δημιουργείται (ή εισάγεται) μέσα στη σκηνή έχει κάποιες ιδιότητες. Κάποιες από αυτές είναι κοινές για όλα, όπως για παράδειγμα η θέση, ενώ κάποιες άλλες συνδέονται με το συγκεκριμένο αντικείμενο. Από την καρτέλα αυτή ο προγραμματιστής θα δώσει τιμές για τις ιδιότητες αυτές. Το περιεχόμενο της καρτέλας αυτής είναι άμεσα συνδεδεμένο με το αντικείμενο που έχει επιλεγεί. Έτσι αν ο προγραμματιστής επιλέξει την κάμερα, η καρτέλα θα εμφανίσει τις ιδιότητες της κάμερας. Αν επιλέξει το φως τότε το περιεχόμενο της θα ανανεωθεί και πλέον θα παρουσιάζει τις ιδιότητες του φωτός. Οι ιδιότητες που μοιράζονται σε όλα τα αντικείμενα, όπως η θέση τους, η περιστροφή τους και το μέγεθός τους σε σχέση με το αρχικό εμφανίζονται πάνω πάνω ενώ στη συνέχεια εμφανίζονται οι ιδιότητες που δεν είναι κοινές για το κάθε αντικείμενο. Στις ιδιότητες του κάθε αντικειμένου ο προγραμματιστής, πέρα από αυτές που ήδη υπάρχουν, μπορεί να προσθέσει επιπλέον. Για παράδειγμα, για ένα αντικείμενο τύπου σφαίρα μπορεί να προσθέσει ιδιότητα Rigidbody η οποία προσδίδει μάζα σε ένα αντικείμενο ώστε αυτό να έχει βαρύτητα και να πέφτει σαν να έχει βάρος.

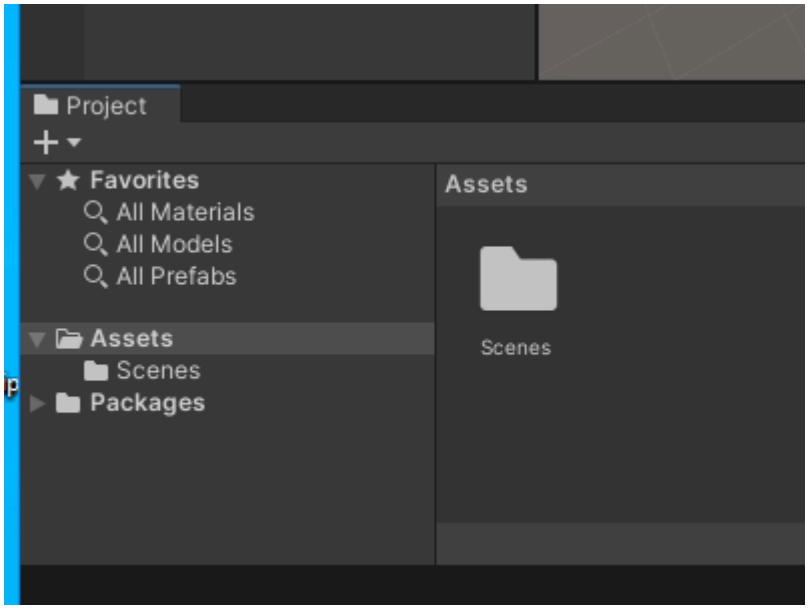


Εικόνα 33: Η καρτέλα Inspector του αντικειμένου κάμερα. Πηγή: Αλέξανδρος Βεκίλογλου

3.4.2.4 Η καρτέλα Project

Η κάθε εφαρμογή που δημιουργείται στον ηλεκτρονικό υπολογιστεί αποτελείται από διάφορα αντικείμενα. Έτσι και στη Unity η εφαρμογή θα αποτελείται από τα διάφορα μοντέλα, εικόνες που χρησιμοποιούνται για να δημιουργούν τα υλικά που θα 'βάφουν' τα μοντέλα, διάφορες παραλλαγές των ίδιων μοντέλων που πιθανόν να μοιράζονται και σε διαφορετικές σκηνές, αρχεία κώδικά καθώς και άλλα. Τα συστατικά αυτά, ο αριθμός των οποίων εύκολα καθίσταται μη διαχειρίσιμος, θα πρέπει να τακτοποιηθούν κάπως ώστε να μπορεί να τα χρησιμοποιήσει ο προγραμματιστής. Το ρόλο αυτό τον αναλαμβάνει η καρτέλα αυτή που εμφανίζει τους φακέλους της εφαρμογής όπου μέσα σε αυτούς ο χρήστης θα ταχτοποιήσει τα διάφορα συστατικά. Όταν η εφαρμογή δημιουργείται αρχικά περιέχει

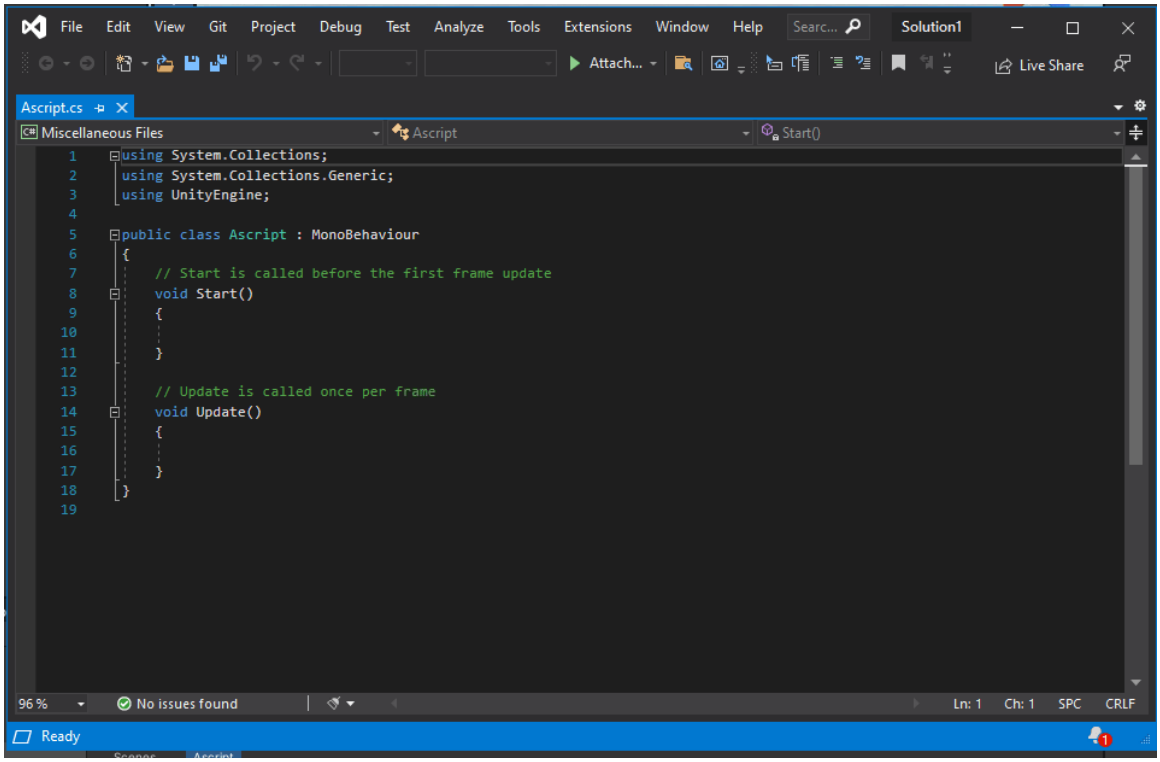
δύο φακέλους όπου ο Assets περιέχει ένα φάκελο για τις σκηνές και το Packages ο οποίος περιέχει τα συνοδευτικά προγράμματα.



Εικόνα 34: Η καρτέλα Project. Πηγή: Αλέξανδρος Βεκίλογλου

3.4.3 Η ιδιότητα Script

Μία επιπλέον ιδιαίτερα σημαντική ιδιότητα που είναι κοινή για όλα τα αντικείμενα αποτελεί το Script. Ουσιαστικά πρόκειται για ένα πρόγραμμα που 'χρησιμοποιεί' τα αντικείμενα της εφαρμογής. Για τη δημιουργία του ο χρήστης επιλέγει το αντικείμενο του οποίου επιθυμεί να αποτελέσει ιδιότητα το Script, επιλέγει 'Add Component' από την καρτέλα Inspector και ακολούθως επιλέγει New Script. Θα πρέπει να δώσει όνομα στο Script (το οποίο αποτελεί και τον τύπο του) και αμέσως αυτό εμφανίζεται και στη καρτέλα Inspector αλλά και στην Project. Κάνοντας διπλό κλικ στο Script στη καρτέλα Project ή πατώντας Open από την καρτέλα Inspector θα ανοίξει το Visual Studio και ο προγραμματιστής θα μπορεί να εισάγει τον κώδικά του στο αρχείο.



Εικόνα 35: Επεξεργασία του Script από το Visual Studio. Πηγή: Αλέξανδρος Βεκίλογλου

Το αρχείο αρχικά περιέχει τα απαραίτητα Imports ώστε μπορεί να χρησιμοποιήσει τα βασικά αντικείμενα και ακολούθως δηλώνει μια κλάση η οποία έχει το όνομα του αρχείου. Η κλάση περιέχει δύο συναρτήσεις οι οποίες δεν έχουν τιμή επιστροφής και δεν δέχονται παραμέτρους, την Start και την Update. Η πρώτη εκτελείται μόλις δημιουργηθεί το αντικείμενο του οποίου το Script αποτελεί ιδιότητα, συνεπώς μόνο μία φορά για όλη τη ζωή του αντικειμένου. Η Update από την άλλη πλευρά εκτελείται ανά καρέ, δηλαδή αρκετές φορές ανά δευτερόλεπτο. Με τον τρόπο αυτό ο προγραμματιστής στην Start τοποθετεί ενδεχομένως αρχικοποιήσεις αντικειμένων και μεταβλητών ενώ στην Update κώδικά που θα εκτελείται ανά καρέ. Φυσικά ο προγραμματιστής είναι ελεύθερος να συντάξει τις δικές του συναρτήσεις οι οποίες θα εκτελούνται είτε από τη Start είτε από την Update.

Στο σημείο αυτό αξίζει να αναφερθεί ότι επειδή η Update εκτελείται συνεχώς θα πρέπει να λαμβάνεται μέριμνα για την εκτέλεση απαιτητικών σε πόρους εργασιών εντός της. Για παράδειγμα η εκτέλεση ενός μεγάλου βρόχου for θα μπορούσε να έχει σαν αποτέλεσμα το παιχνίδι να μην έχει μια φυσιολογική ροή καθώς τα καρέ στα οποία εκτελείται η for θα εκτελούνται πιο αργά από τα υπόλοιπα. Αντί αυτού προτείνεται να χρησιμοποιείται μια μεταβλητή η οποία θα αυξάνεται σε κάθε καρέ ώστε η διεργασία να εκτελείται σε περισσότερα του ενός καρέ (αντί του να εκτελείται όλη σε ένα μόνο) με αποτέλεσμα να υπάρχει μια πιο φυσιολογική ροή στο παιχνίδι.

4 ΚΕΦΑΛΑΙΟ 4ο : Αρχιτεκτονική Κώδικα

Στο κεφάλαιο αυτό θα παρουσιαστεί η αρχιτεκτονική του κώδικα της εφαρμογής. Στόχος της συγκεκριμένης παρουσίασης είναι να καταστεί ευκολότερη η κατανόηση του προγράμματος, ο σχεδιασμός του και η δομή του ώστε να είναι εφικτή η υποστήριξη του από τους αναγνώστες.

4.1 Εισαγωγή

Στόχο του προγράμματος αποτελεί η κατασκευή μιας έκδοσης του δημοφιλούς παιχνιδιού 7 Wonders. Η εφαρμογή θα αποτελεί ένα παιχνίδι με κάρτες όπου κάθε παίχτης θα λαμβάνει τυχαία κάρτες, θα τις παίζει κερδίζοντας ή χάνοντας χρήματα με στόχο να συγκεντρώσει τους περισσότερους πόντους. Μέσα από το παιχνίδι αυτό ο παίχτης θα ενισχύσει την κριτική του σκέψη, θα ενισχύσει την ικανότητα του να οργανώνει στρατηγικές και να τις ακολουθεί. Παράλληλα θα διασκεδάζει παίζοντας ένα βραβευμένο παιχνίδι.

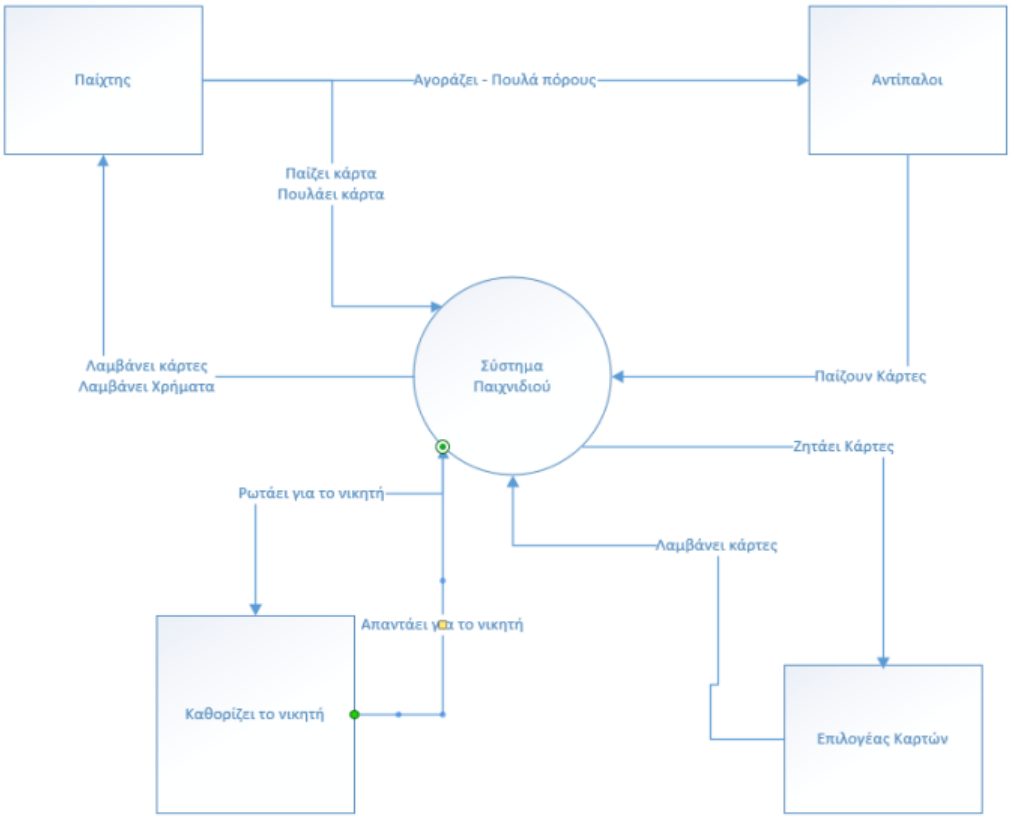
Στην παρούσα έκδοση η εφαρμογή καλύπτει εξολοκλήρου την πρώτη από τις τρεις χρονικές περιόδους του παιχνιδιού. Πιο συγκεκριμένα το παιχνίδι ξεκινά τοποθετώντας τις μάρκες, τα κέρματα και τις κάρτες πάνω σε ένα τραπέζι με έναν αρκετά αληθοφανή τρόπο. Ειδικά για τις μάρκες και τα κέρματα ‘πέφτουν’ πάνω στο τραπέζι και η όψη που θα βρίσκεται από πάνω καθώς και ο προσανατολισμός τους καθορίζεται με τυχαίο τρόπο. Ακολούθως κάθε παίχτης παίρνει με τυχαίο τρόπο επτά κάρτες. Ο βασικός παίχτης επιλέγει ποια θα παίξει ενώ οι υπόλοιποι παίζουν κάποια στην τύχη. Μόλις ο βασικός παίχτης την επιλέξει την κάρτα που θα παίξει η εφαρμογή εμφανίζει ένα παράθυρο διαλόγου όπου παρουσιάζει στο παίχτη τις διάφορες επιλογές του. Πιο συγκεκριμένα ο παίχτης μπορεί είτε να παίξει την κάρτα, είτε να κατασκευάσει μέρος του θαύματος είτε να πουλήσει την κάρτα. Για να την παίξει όμως θα πρέπει να διαθέτει τους απαραίτητους πόρους. Συνεπώς η εφαρμογή, κάθε φορά που ο παίχτης επιλέγει την κάρτα που θα παίξει εξετάζει εάν ο παίχτης διαθέτει τους απαραίτητους πόρους. Στη περίπτωση που δεν τους διαθέτει τους αναζητά στους γειτονικούς του. Ακολούθως του εμφανίζει ένα νέο παράθυρο όπου τον ενημερώνει για το ποιος γείτονας έχει ποιον πόρο, εφόσον φυσικά κάποιος ή κάποιοι γείτονες του διαθέτουν τον πόρο. Από εκεί διαλέγει και αγοράζει εφόσον διαθέτει αρκετά χρήματα. Μόλις η συναλλαγή ολοκληρωθεί τα χρήματα μεταφέρονται στον κατάλληλο παίχτη. Ταυτόχρονα τα κέρματα μεταφέρονται και στη κάρτα θαύματος του παίχτη. Η επιλογή κατασκευής μέρους του θαύματος δεν είναι διαθέσιμη στη παρούσα έκδοση, καθώς αποτελεί προαιρετική επιλογή και στο παιχνίδι, ενώ έχει υλοποιηθεί και η πώληση της κάρτας. Έτσι, στη περίπτωση που ο παίχτης δεν διαθέτει αρκετά χρήματα ή κανένας γείτονας δεν διαθέτει τους απαραίτητους πόρους για να τη κατασκευάσει μπορεί να πουλήσει την κάρτα πατώντας το κατάλληλο κουμπί. Άμεσα η κάρτα μεταφέρεται στην κατάλληλη θέση και ο παίχτης λαμβάνει το αντίτιμο. Η διαδικασία επαναλαμβάνεται 7 συνολικά φορές και στο τέλος η εφαρμογή καταλήγει στο νικητή αφού όλοι οι παίχτες αναμετρηθούν στρατιωτικά.

Τέλος, ο συγγραφέας, εκτιμά ότι η παρουσίαση αυτή θα φανεί χρήσιμη τόσο στους προγραμματιστές που θα θελήσουν να δημιουργήσουν ένα αντίστοιχο παιχνίδι ή σε εκείνους που θα θελήσουν να το επεκτείνουν ώστε να καλύψουν και τις υπόλοιπες περιόδους καθώς και σε σχεδιαστές παιχνιδιών αλλά και σε διαχειριστές έργων.

4.2 Διάγραμμα πλαισίου – Context Diagram

Στη επόμενη εικόνα παρουσιάζεται το Διάγραμμα πλαισίου το οποίο έχει σαν στόχο να δείξει με τρόπο απλό και κατανοητό τις βασικές διαδικασίες της εφαρμογής, τις οντότητες που συμμετέχουν και τέλος τις ροές των δεδομένων. Στόχος του διαγράμματος αυτού είναι να προσφέρει μια γενική εικόνα του τρόπου με τον οποίο οι διάφορες πληροφορίες κινούνται μέσα στο σύστημα και πώς αλληλεπιδρούν με τις υπόλοιπες εξωτερικές οντότητες.

Στο διάγραμμα αυτό παρουσιάζονται σαν οντότητες ο παίχτης, οι αντίπαλοι, το σύστημα επιλογής καρτών και το σύστημα επιλογής νικητή, γύρω από το βασικό σύστημα του παιχνιδιού. Όλες οι εξωτερικές οντότητες ενώνονται με την κεντρική με τα βέλη τα οποία αναπαριστούν την ροή των δεδομένων. Για παράδειγμα, όταν το παιχνίδι ξεκινά το σύστημα παιχνιδιού επικοινωνεί με τον επιλογέα καρτών και λαμβάνει για κάθε παίχτη μία κάρτα επιλεγμένη με τυχαίο τρόπο. Την κάρτα αυτή την αναθέτει σε ένα παίχτη και ακολούθως συνεχίζει στον επόμενο.



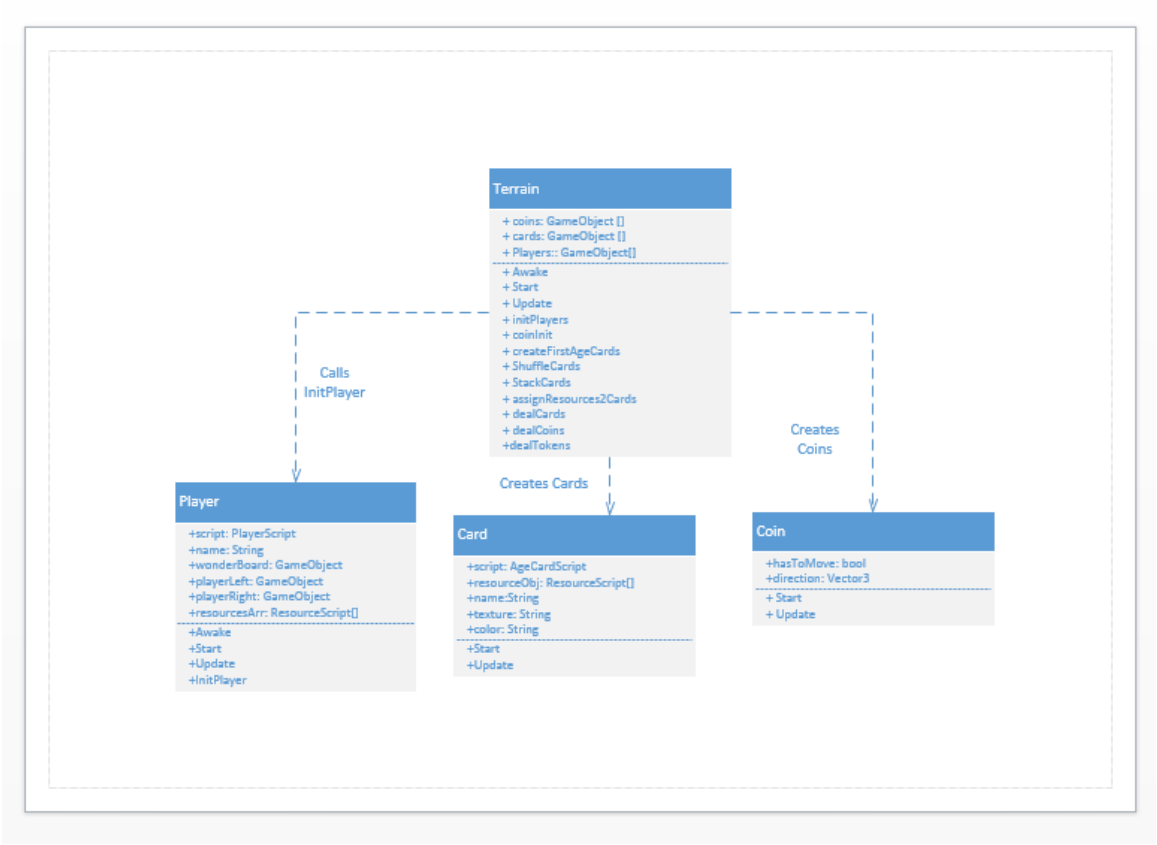
Εικόνα 36: Διάγραμμα Πλαισίου - Context Diagram. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συνέχεια θα παρουσιαστούν τα διαγράμματα κλάσεων και περιπτώσεων χρήσης της εφαρμογής. Για λόγους απλότητας τα διαγράμματα θα παρουσιαστούν ανά φάση.

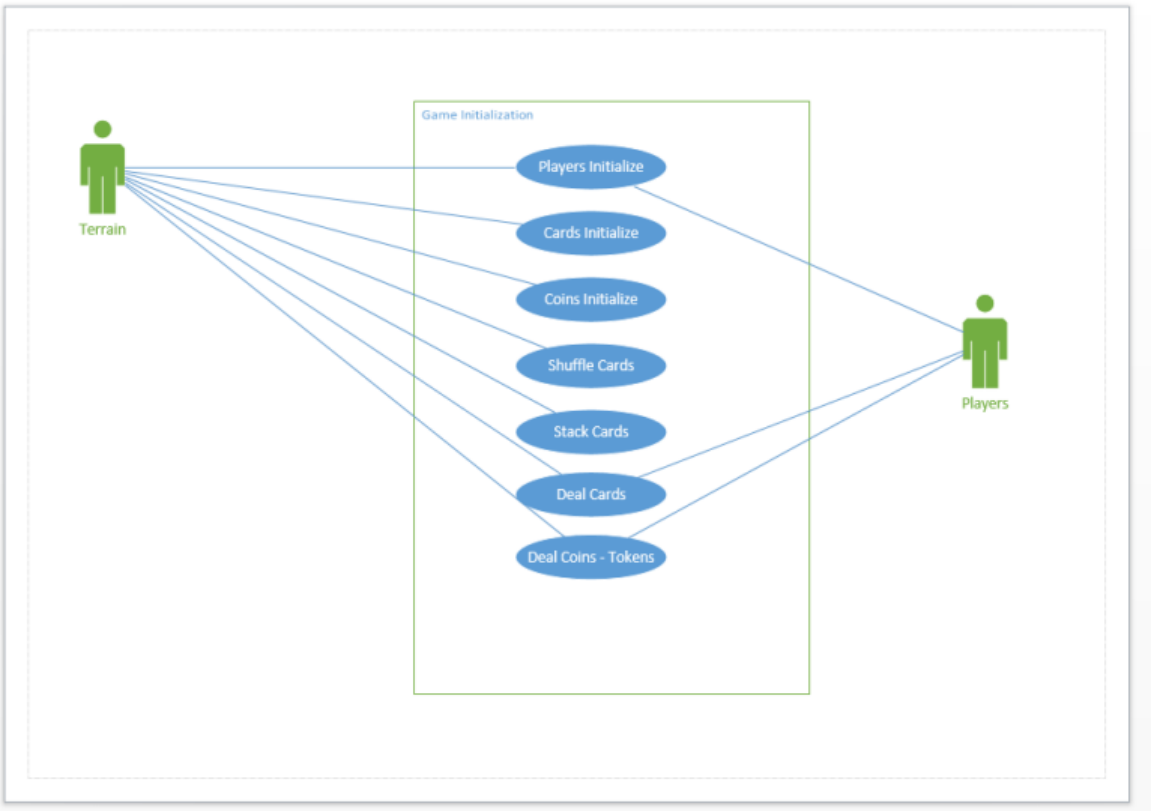
4.2.1 Αρχικοποίηση παιχνιδιού

Μόλις η εφαρμογή ξεκινά, αρχικοποιεί τους παίχτες και τους αναθέτει την κάρτα θαύματος. Στη συνέχεια θα δημιουργεί τα αντικείμενα κάρτες, κέρματα και τους πόντους των μαχών. Ακολούθως ανακατεύει τις κάρτες και μοιράζει στους παίχτες τις κάρτες, τα κέρματα αλλά και τους πόντους. Όλα

τα αντικείμενα, εκτός των πόντων των μαχών (τα tokens) αποτελούν αντικείμενα τύπου Prefabs τα οποία αρχικοποιούνται μέσω της instantiate του αντικειμένου object. Η instantiate καλείται από το Script που έχει ανατεθεί στο αντικείμενο terrain ανά περίπτωση. Πιο συγκεκριμένα για τις κάρτες θαύματος καλείται στην Start και τα υπόλοιπα αντικείμενα στις λοιπές συναρτήσεις. Στη συνέχεια δημιουργεί τα αντικείμενα, ανακατεύει τις κάρτες και τοποθετεί όλα τα αντικείμενα πάνω στο τραπέζι. Ακολούθως η εφαρμογή μοιράζει τα κέρματα και τα tokens στις κατάλληλες θέσεις των παιχτών. Τέλος μοιράζει τις κάρτες στους παίχτες.



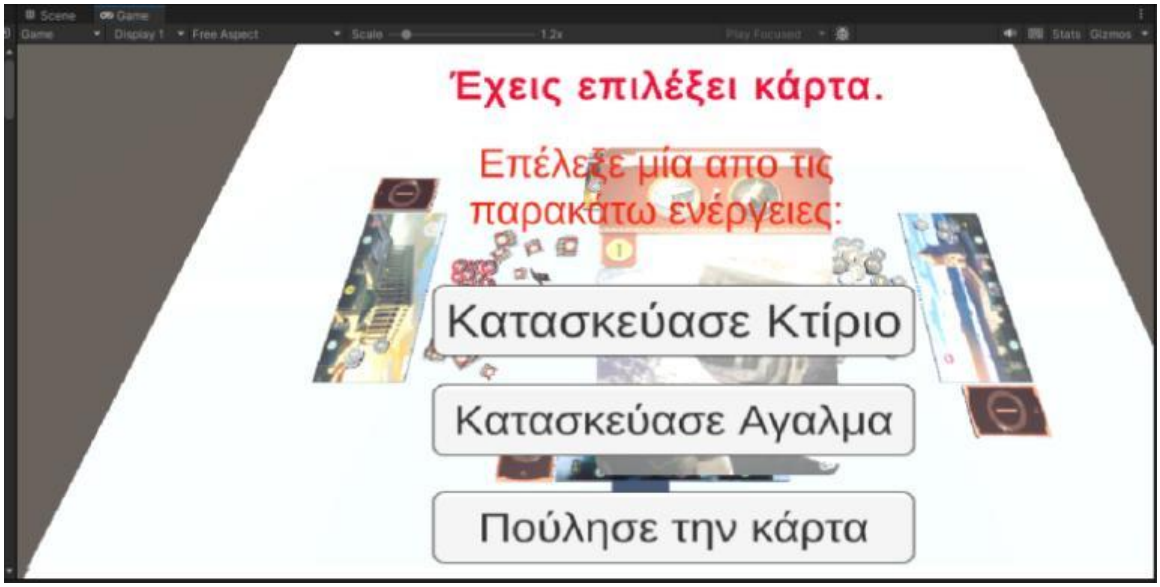
Εικόνα 37: Διάγραμμα κλάσεων αρχής παιχνιδιού. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 38: Διάγραμμα περιπτώσεων χρήσης. Πηγή: Αλέξανδρος Βεκίλογλου

4.2.2 Παίξιμο κάρτας

Μόλις ο κάθε παίχτης λάβει τις κάρτες του τότε οι κάρτες του βασικού παίχτη απλώνονται μπροστά του ώστε να μπορεί να τις δει. Κάθε φορά που τοποθετεί το δείκτη του ποντικιού πάνω σε μια κάρτα αυτή εμφανίζεται μεγαλύτερη στο κέντρο της οθόνης. Κάνοντας κλικ πάνω της η κάρτα επιλέγεται για να παιχτεί. Μόλις επιλεγεί η εφαρμογή εμφανίζει ένα παράθυρο διαλόγου το οποίο διαθέτει τρία κουμπιά με τις ισάριθμες επιλογές του παίχτη. Μπορεί είτε να δημιουργήσει την κατασκευή, είτε να κατασκευάσει μέρος του θαύματος (η επιλογή αυτή δεν διατίθεται στην παρούσα έκδοση) και τέλος να πουλήσει τη κάρτα.

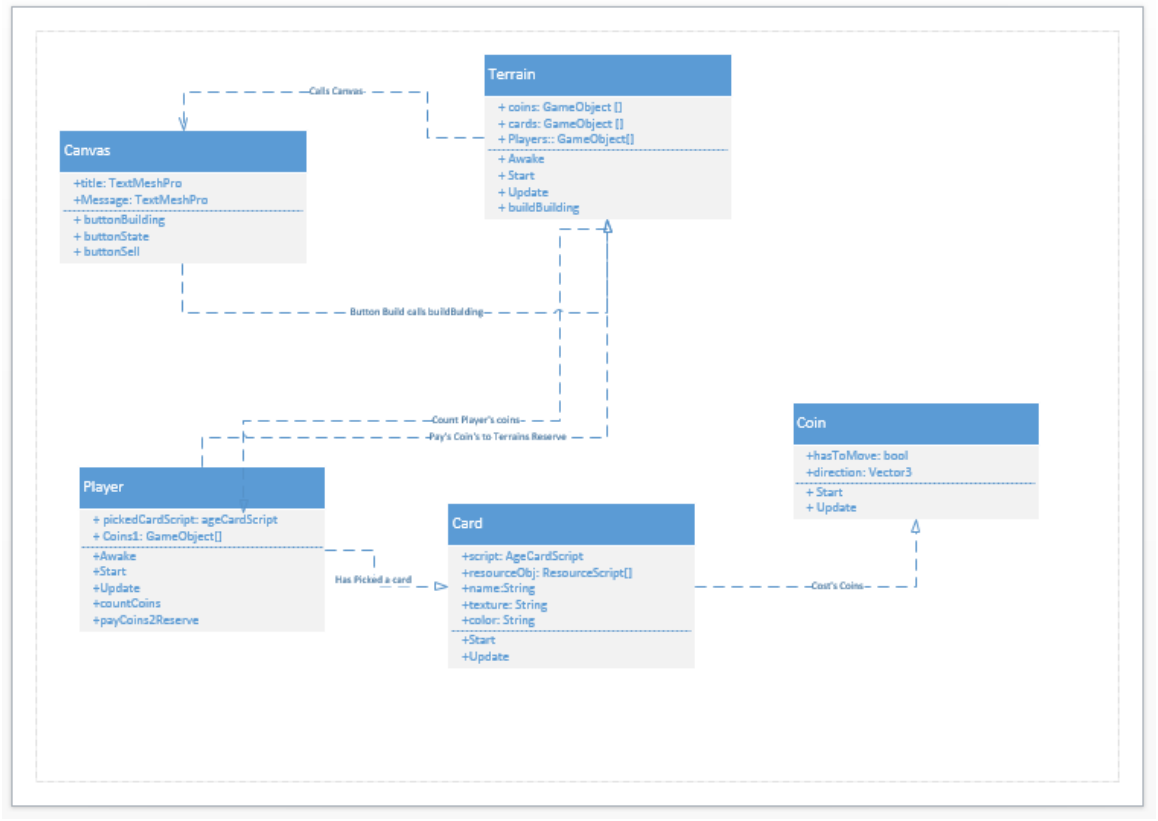


Εικόνα 39: GUI διαθέσιμων επιλογών. Πηγή: Αλέξανδρος Βεκίλογλου

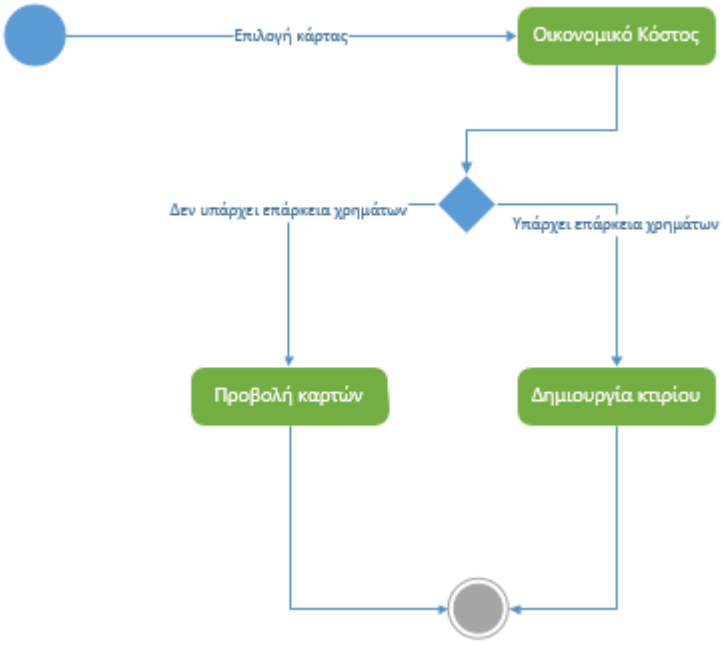
4.2.2.1 Κατασκευή κτιρίου

Πληρωμή κόστους

Για να δημιουργήσει ο παίχτης τη κατασκευή θα πρέπει να διαθέτει τους απαραίτητους πόρους είτε χρηματικούς είτε άλλους. Για το σκοπό αυτό όταν επιλέγεται το κουμπί 'Κατασκεύασε Κτίριο' η εφαρμογή ελέγχει τι πόρους απαιτεί η κάρτα. Αρχικά ελέγχει αν η κάρτα απαιτεί χρήματα και αν ναι ελέγχει αν ο παίχτης έχει αρκετά. Αν δεν διαθέτει αρκετά τον ενημερώνει και γυρνά στη προηγούμενη κατάσταση ώστε να διαλέξει μια νέα κάρτα από τις 7 που έχει και η διαδικασία επαναλαμβάνεται. Αν διαθέτει χρήματα εκτελείται η συνάρτηση που μεταφέρει τα κέρματα από το παίχτη στη στοίβα και η κάρτα προστίθεται στις κάρτες του.



Εικόνα 40: Διάγραμμα κλάσεων για τη πληρωμή του κόστους της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

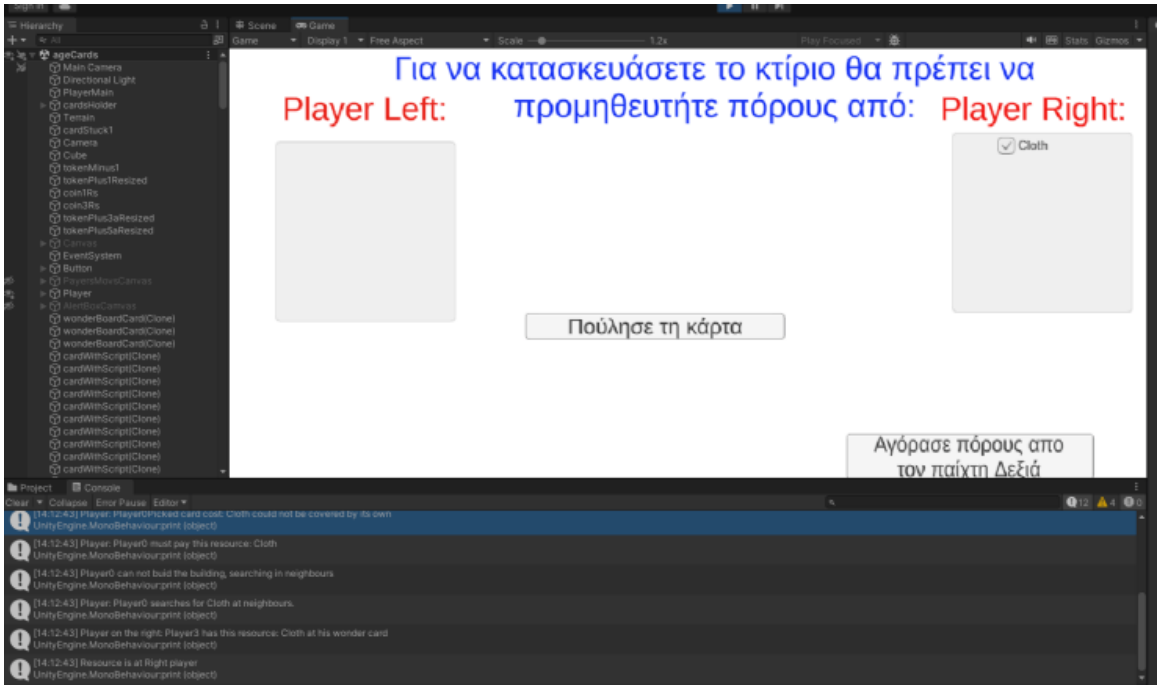


Εικόνα 41: Διάγραμμα καταστάσεων για την καταβολή του κόστους της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

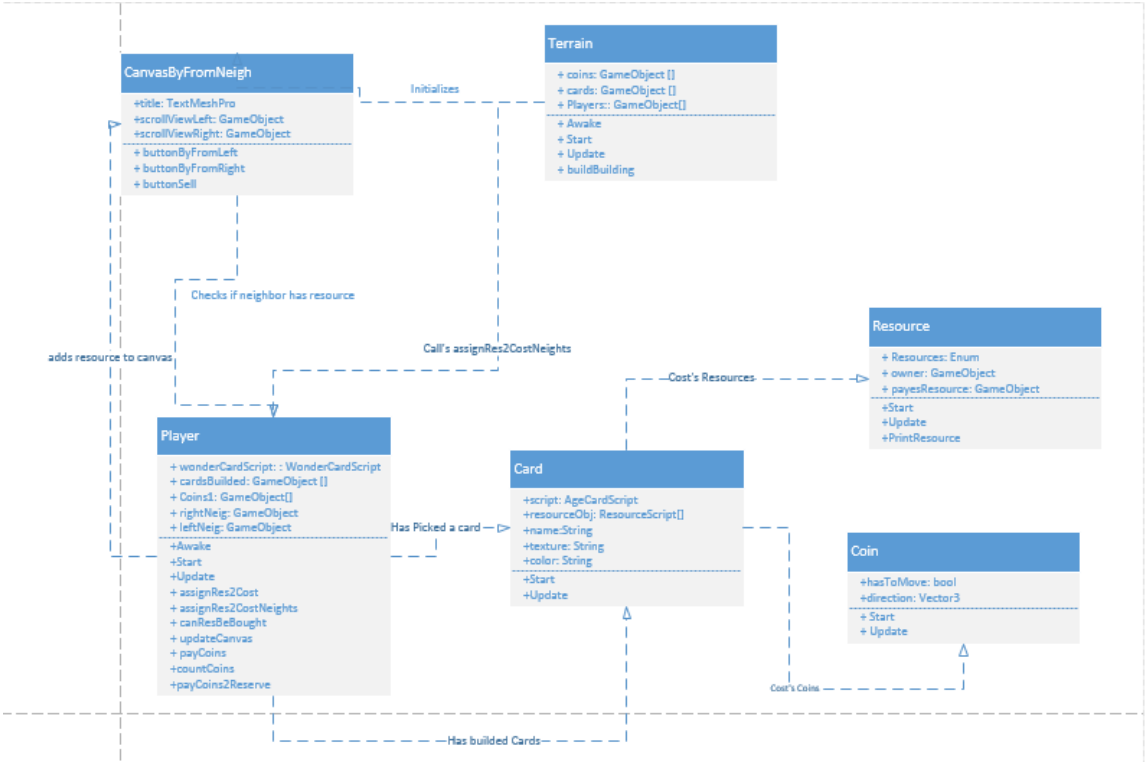
Χρήση πόρων

Οι κάρτες μπορούν να απαιτούν, πέρα από το κόστος σε χρήματα, κόστος σε πόρους. Η κάρτα θαύματος του κάθε παίχτη προσφέρει ένα πόρο. Καθώς ο παίχτης δημιουργεί κατασκευές αυτές

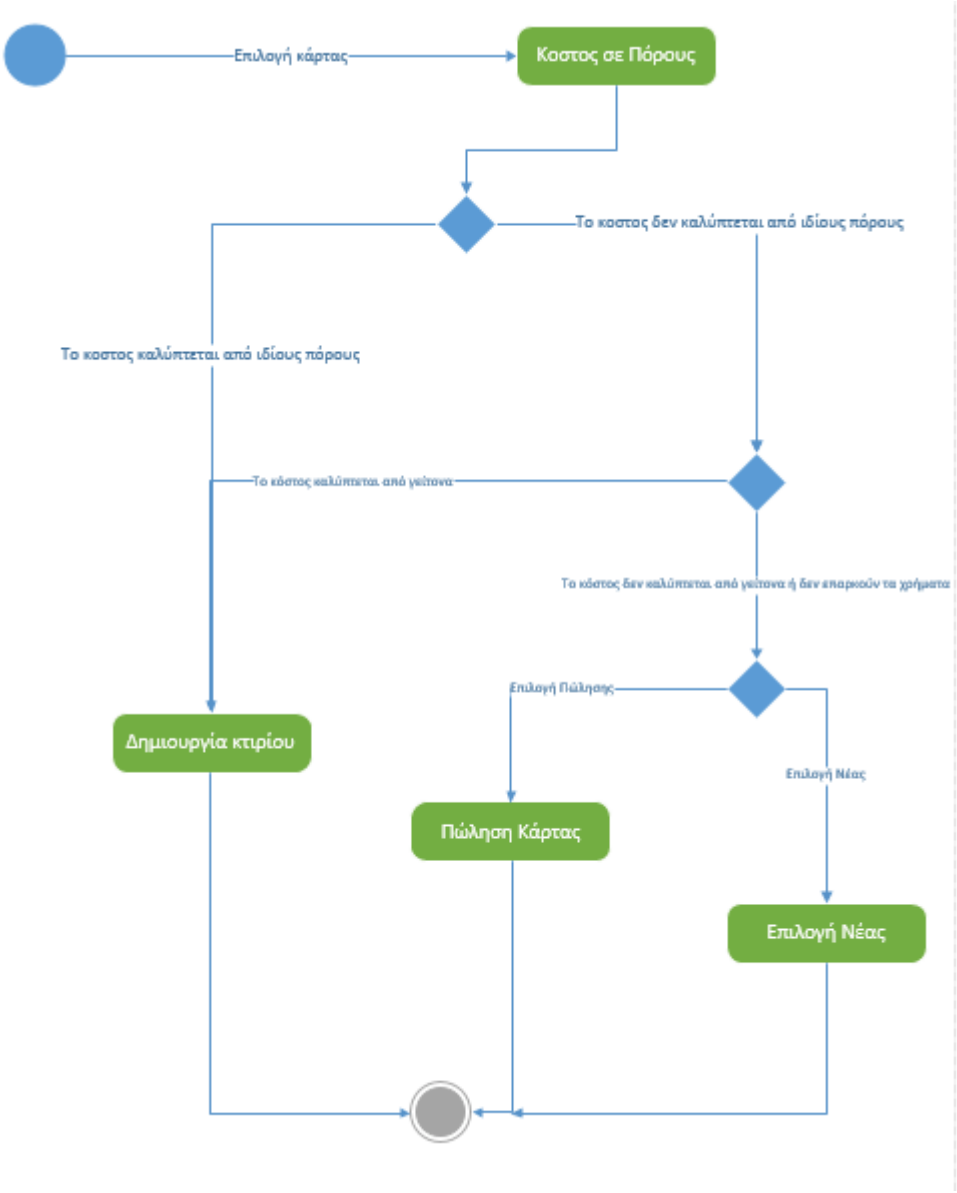
προσφέρουν πόρους. Συνεπώς για να κατασκευαστεί το κτίριο ελέγχεται αρχικά αν η κάρτα θαύματος διαθέτει τον πόρο. Αν όχι ελέγχεται αν τον διαθέτουν οι κάρτες που έχει ήδη κατασκευάσει ο παίχτης. Στη περίπτωση που ο πόρος καλύπτεται η κάρτα 'χτίζεται' και τοποθετείται στην κατάλληλη θέση στη κάρτα θαύματος. Σε διαφορετική περίπτωση αναζητείται ο πόρος στους γείτονες. Για τη διευκόλυνση του παίχτη η εφαρμογή εμφανίζει ένα νέο GUI το οποίο παρουσιάζει ποιος πόρος διατίθεται από ποιον γείτονα. Ο παίχτης επιλέγει τον πόρο που θέλει να αγοράσει και εφόσον διαθέτει τα απαραίτητα χρήματα ενεργοποιείται το κουμπί αγοράς. Διαφορετικά η εφαρμογή τον ενημερώνει ότι δεν διαθέτει το επαρκές υπόλοιπο. Αν μπορέσει να αγοράσει μεταφέρονται τα χρήματα στον γείτονα και η κάρτα χτίζεται. Διαφορετικά επιστρέφει στην αρχική κατάσταση επιλογής άλλης κάρτας και η διαδικασία επαναλαμβάνεται, ώστε ο παίχτης να επιλέξει κάρτα που χρειάζεται άλλους πόρους ώστε να μπορέσει να καλύψει το κόστος. Τέλος παρέχεται και η επιλογή πώλησης της κάρτας.



Εικόνα 42: Επιλογή πόρου και ενεργοποίηση του κουμπιού αγοράς εφόσον ο παίχτης διαθέτει επαρκές υπόλοιπο. Πηγή: Αλέξανδρος Βεκίλογλου



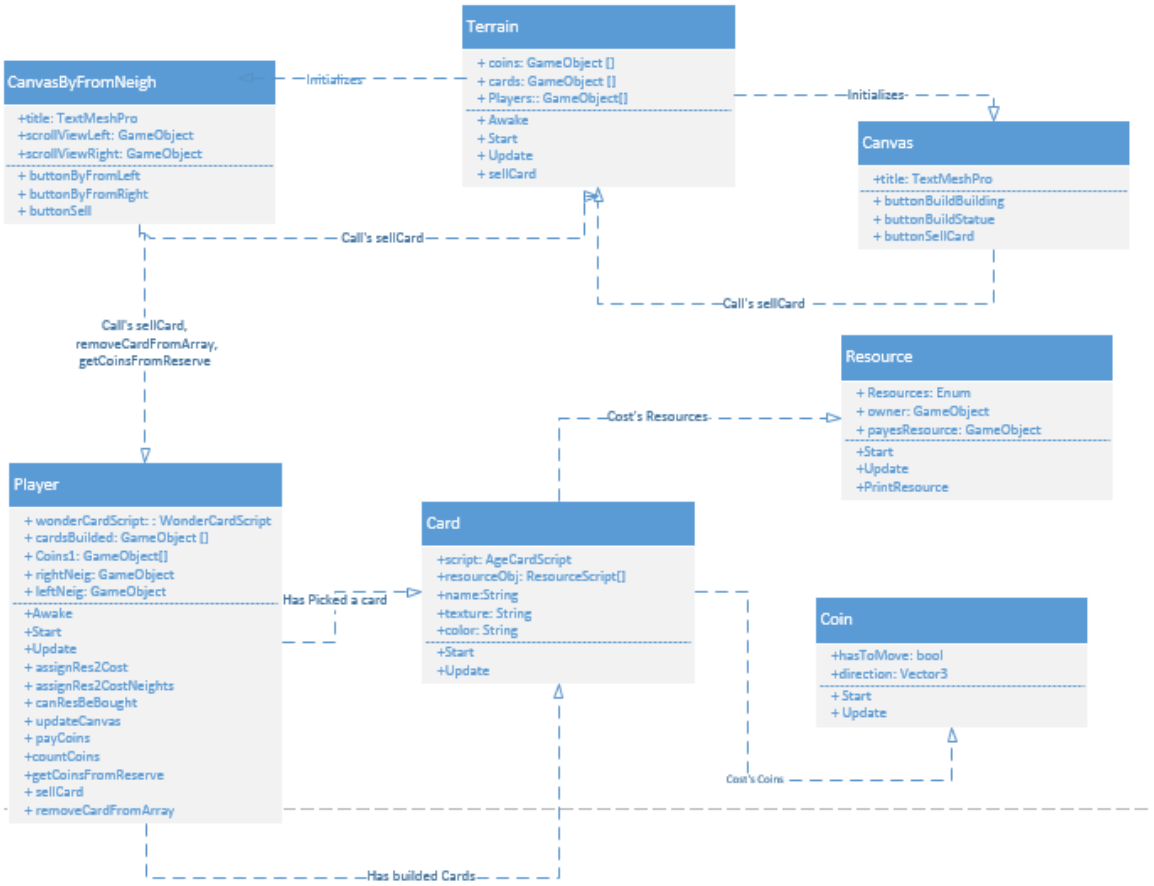
Εικόνα 43: Διάγραμμα κλάσεων αναζήτησης πόρων και αγοράς από τους γείτονες. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 44: Διάγραμμα καταστάσεων κτίσης κάρτας από γειτονικούς πόρους. Πηγή: Αλέξανδρος Βεκίλογλου

4.2.2.2 Πώληση κάρτας

Στη περίπτωση που ο παίχτης δεν μπορεί να κατασκευάσει την κάρτα μπορεί να προβεί στην πώληση της. Η πώληση μπορεί να πραγματοποιηθεί είτε από το GUI επιλογής ενέργειας είτε από το GUI επιλογής πόρων από τους γείτονες. Η προσέγγιση αυτή επιλέχθηκε καθώς ο παίχτης δεν γνωρίζει εξ αρχής αν μπορεί να κατασκευάσει την κάρτα, καθώς δεν έχει πάντα πλήρη εικόνα για τους πόρους που διαθέτουν οι γείτονες, συνεπώς πρώτα ελέγχει αν οι γείτονες διαθέτουν τους πόρους και αν όχι τότε μπορεί να τη πουλήσει.



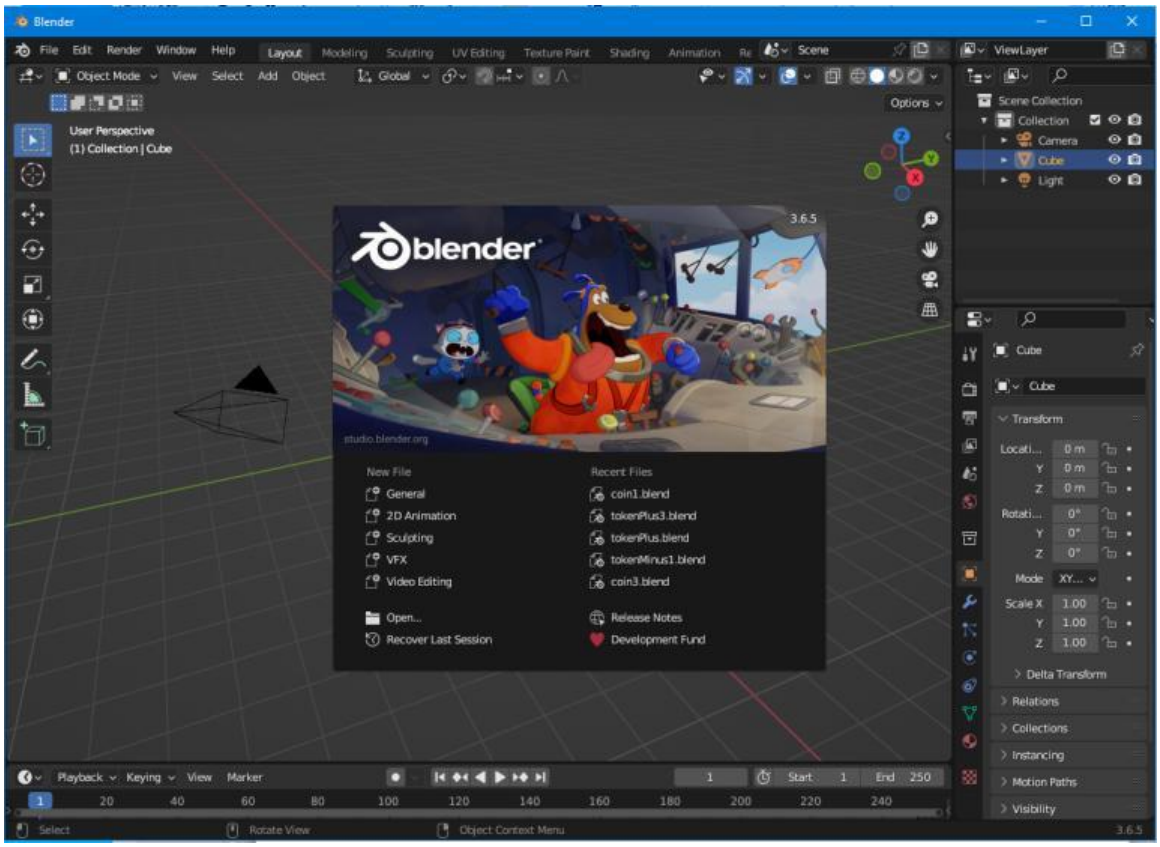
Εικόνα 45: Διάγραμμα κλάσεων πώλησης κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

5 ΚΕΦΑΛΑΙΟ 5ο : Ανάπτυξη εφαρμογής

Στο κεφάλαιο αυτό θα παρουσιαστεί η ανάπτυξη της εφαρμογής. Πιο συγκεκριμένα θα παρουσιαστούν αναλυτικά τα βήματα για τη δημιουργία μοντέλων με την εφαρμογή Blender, την εισαγωγή τους στη Unity, την δημιουργία των υπολοίπων μοντέλων και τη συγγραφή του απαιτούμενου κώδικά. Στόχος της παρούσας παρουσίασης αποτελεί η εξοικείωση του αναγνώστη με τις τεχνολογίες που απαιτούνται για την κατασκευή της εφαρμογής. Η παρουσίαση θα ξεκινήσει με την δημιουργία των μοντέλων στο Blender, ακολούθως η εισαγωγή τους στη Unity και ακολούθως η πλήρη παρουσίαση της ανάπτυξης της εφαρμογής στη Unity.

5.1 Κατασκευή Μοντέλων

Η Unity μπορεί να δημιουργήσει βασικά μοντέλα (κύβους, σφαίρες, κλπ.) όμως δεν διαθέτει τη δυνατότητα να κατασκευάσει τα σύνθετα μοντέλα που απαιτούνται στη παρούσα εφαρμογή όπως οι κάρτες, τα νομίσματα καθώς και τα Military Tokens. Για το σκοπό αυτό τα μοντέλα αυτά θα κατασκευαστούν στην εφαρμογή Blender. Πρόκειται για μία ιδιαίτερα δημοφιλή εφαρμογή δημιουργίας τρισδιάστατων μοντέλων, η οποία έχει καταφέρει να διεκδικήσει μια αξιόλογη θέση στο χώρο, έναντι άλλων κολοσσών όπως το 3D Studio Max της Autodesk. Η εφαρμογή διατίθεται από τον επίσημο ισότοπο της εταιρίας https://www.blender.org/ ενώ παράλληλα είναι δωρεάν και ανήκει στα λογισμικά ανοιχτού κώδικα.

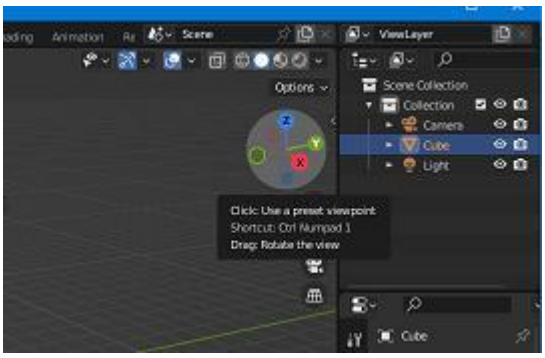


Εικόνα 46: Η εφαρμογή δημιουργίας 3D μοντέλων Blender. Πηγή: Αλέξανδρος Βεκίλογλου

5.1.1 Κατασκευή Καρτών

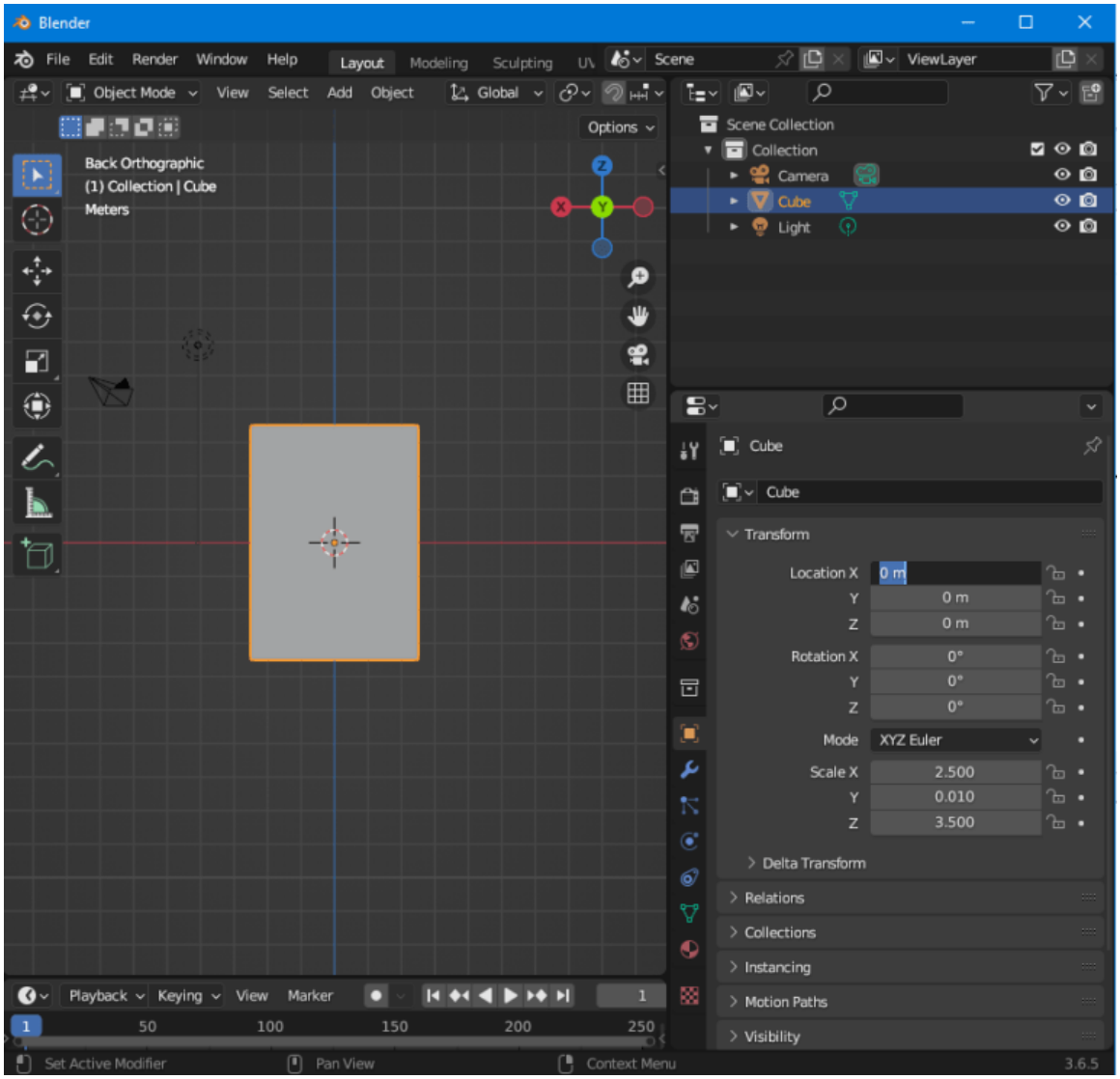
Δημιουργία μοντέλου

Η παρουσίαση θα ξεκινήσει από την κατασκευή των καρτών καθώς αποτελούν από τις πιο απλές κατασκευές και συνεπώς είναι κατάλληλες για τον χρήστη που δεν διαθέτει εξοικείωση με το αντικείμενο. Πρώτο βήμα αποτελεί η εκκίνηση της εφαρμογής. Η εφαρμογή έχει ήδη κατασκευάσει ένα κύβο στο κέντρο της οθόνης και παρουσιάζει ένα παράθυρο που δείχνει στο χρήστη βασικές λειτουργίες όπως άνοιγμα νέου αρχείου ή πρόσφατου. Κάνοντας κλικ οπουδήποτε εκτός του παραθύρου αυτού το παράθυρο αυτό κλείνει. Ακολούθως ο χρήστης θα πρέπει να προσαρμόσει τη γωνία θέασης του αντικείμενου ώστε να το κοιτάει ακριβώς από μπροστά πατώντας στον άξονα ‘Y’.



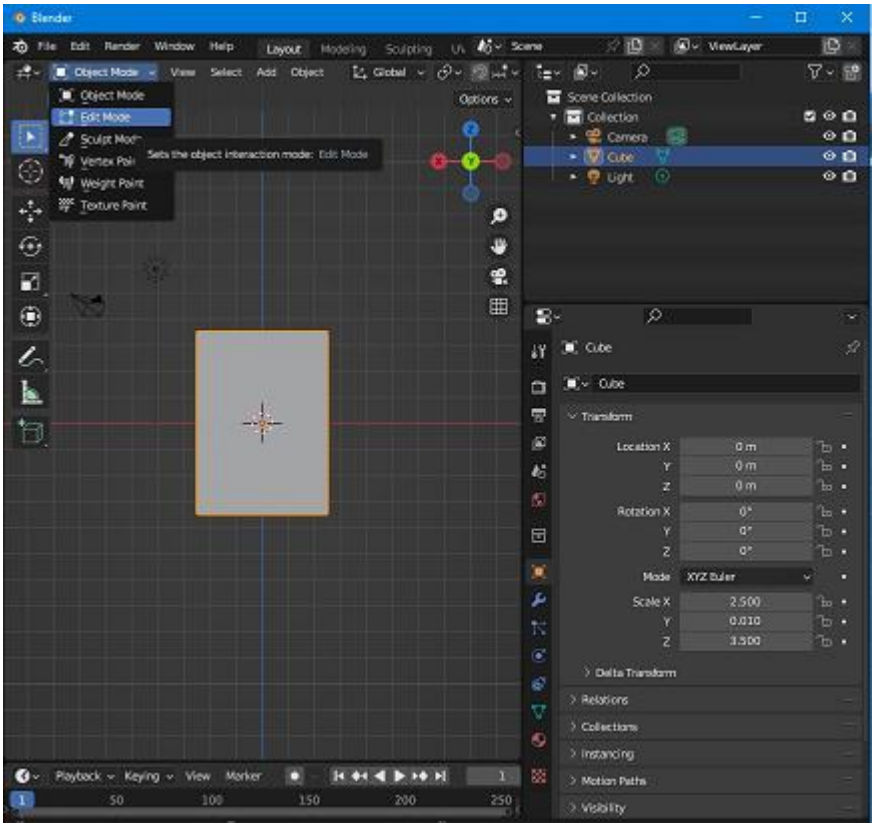
Εικόνα 47: Προσαρμογή γωνίας θέασης αντικειμένων. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως, έχοντας ο χρήστης τον κύβο επιλεγμένο, θα προσαρμόσει τις διαστάσεις του ώστε να πάρει το κατάλληλο σχήμα κάρτας. Οι διαστάσεις εισάγονται από την καρτέλα Object και ακολούθως Transform και τέλος στην ενότητα Scale.



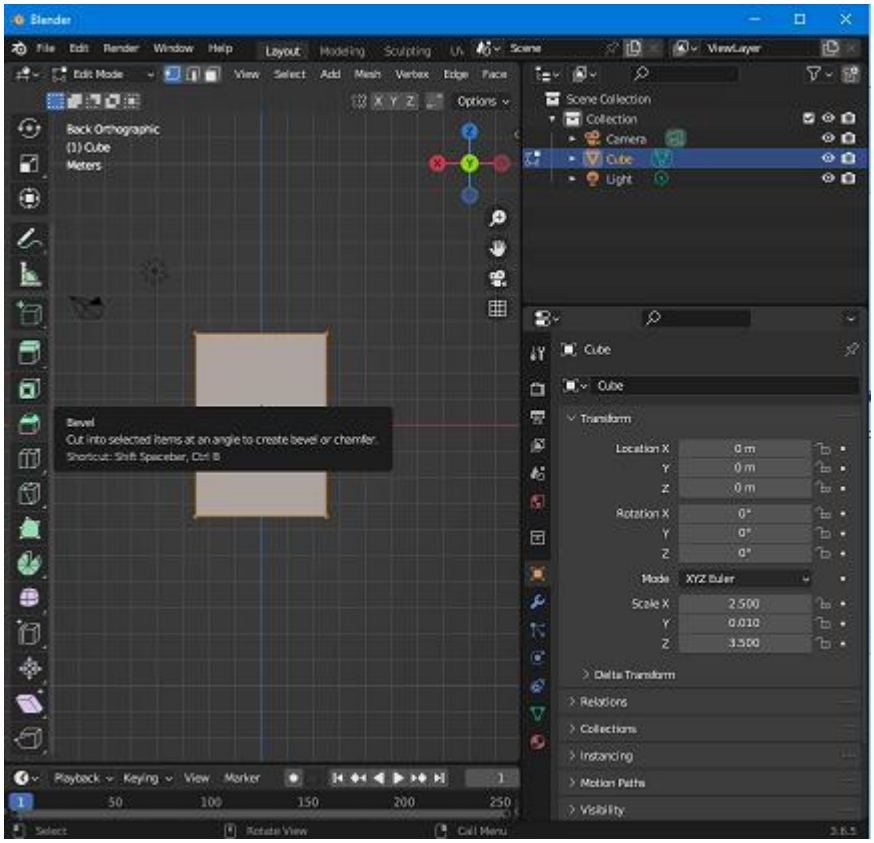
Εικόνα 48: Δημιουργία μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

Η κάρτα έχει στρογγυλεμένες άκρες και όχι γωνίες. Συνεπώς κρίνεται σκόπιμο να αυξηθεί ο αριθμός των ακμών στο σημείο αυτό ώστε να γίνει πιο στρογγυλή. Προκειμένου να επιτευχθεί αυτό θα πρέπει να ενεργοποιηθεί η λειτουργία επεξεργασία (Edit).

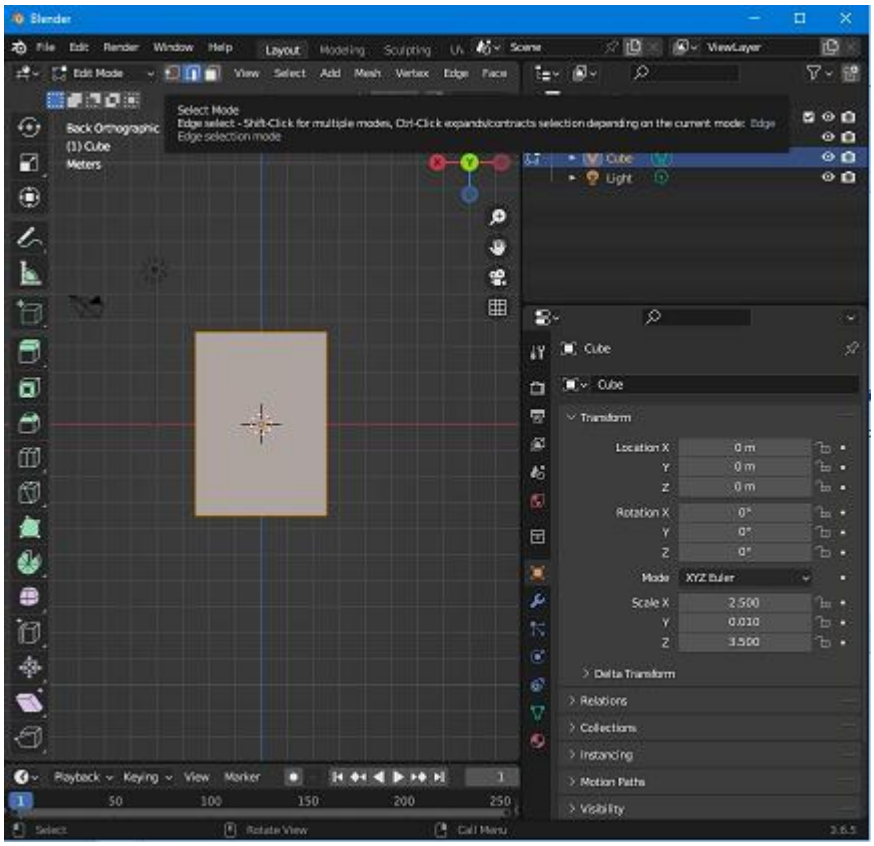


Εικόνα 49: Λειτουργία επεξεργασίας. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις ενεργοποιηθεί η λειτουργία αυτή η εφαρμογή εμφανίζει περισσότερες λειτουργίες στη μπάρα αριστερά με τα διαθέσιμα εργαλεία. Ο χρήστης θα πρέπει να επιλέξει τη λειτουργία Bevel και ακολούθως να επιλέξει ότι η λειτουργία θα εφαρμοστεί στις ακμές του αντικειμένου επιλέγοντας το μεσαίο κουμπί στην πάνω μπάρα.

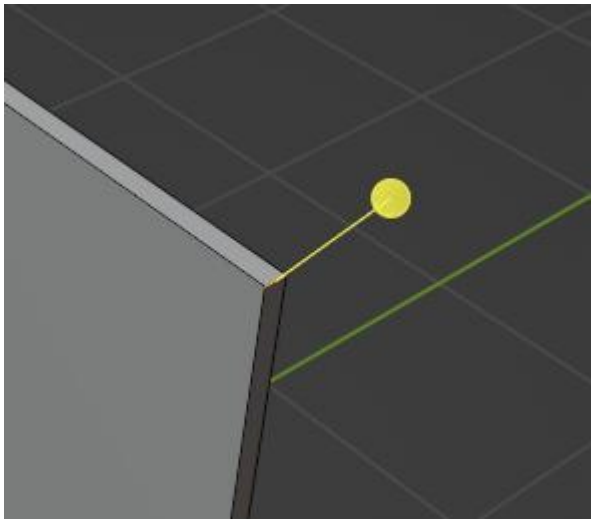


Εικόνα 50: Επιλογή του εργαλείου Bevel. Πηγή: Αλέξανδρος Βεκίλογλου



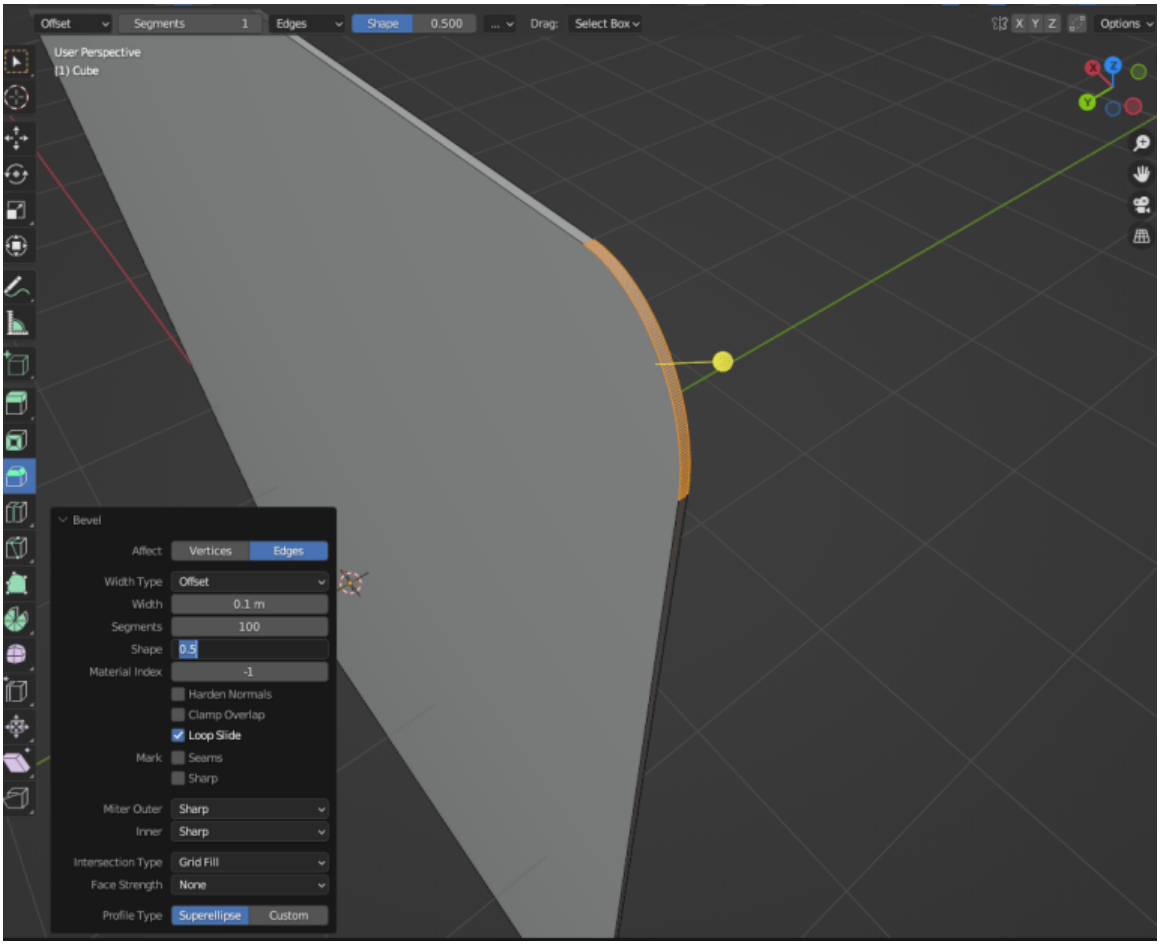
Εικόνα 51: Εφαρμογή του Bevel στις ακμές του μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

Πλέον ο χρήστης μπορεί να επιλέξει την ακμή και να την τροποποιήσει. Αυτό μπορεί να επιτευχθεί περιστρέφοντας την γωνία θέασης κρατώντας το μεσαίο κουμπί του ποντικιού πατημένο και ακολούθως μετακινώντας το. Στη συνέχεια θα επιλέξει την ακμή που επιθυμεί να επεξεργαστεί και η εφαρμογή θα εμφανίζει ένα κίτρινο μπαλόνι με μια ακμή που να συνδέεται στην επιλεγμένη γωνία.

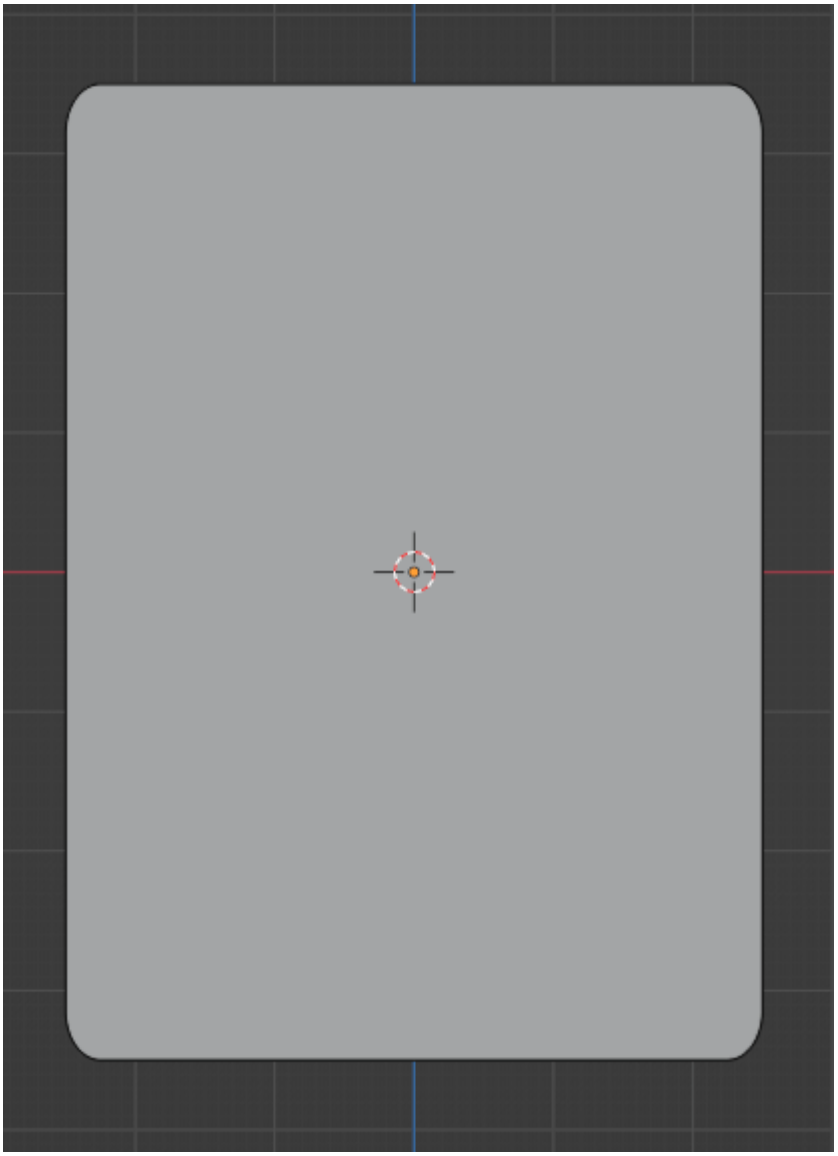


Εικόνα 52: Η ακμή που θα επεξεργαστεί. Πηγή: Αλέξανδρος Βεκίλογλου

Μετακινώντας ελάχιστα το μπαλόνι στο κάτω αριστερό μέρος εμφανίζεται ένα κουμπί με το όνομα Bevel. Ουσιαστικά αποτελεί ένα μενού επιλογών που θα το ανοίξει ο χρήστης και θα εισάγει τιμές στη τιμή πλάτος (απόσταση μεταξύ των νέων ακμών) και θα προσθέσει επιπλέον τμήματα (Segments) ώστε η γωνία να εμφανίζεται στρογγυλεμένη. Η ίδια διαδικασία θα γίνει και για τις άλλες τρείς γωνίες.



Εικόνα 53: Επεξεργασία γωνίας. Πηγή: Αλέξανδρος Βεκίλογλου

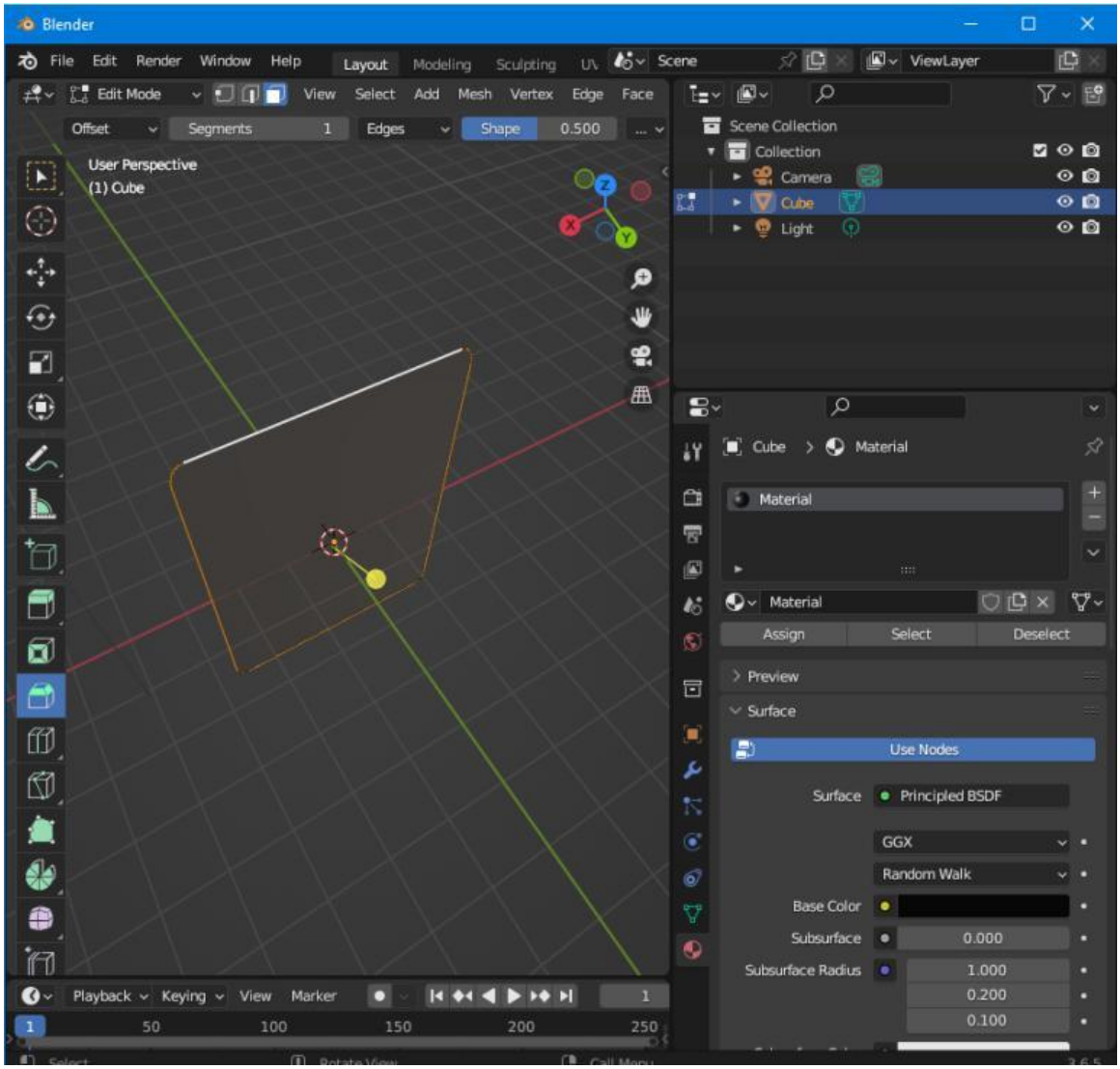


Εικόνα 54: Τελικό σχήμα. Πηγή: Αλέξανδρος Βεκίλογλου

Δημιουργία Υλικού

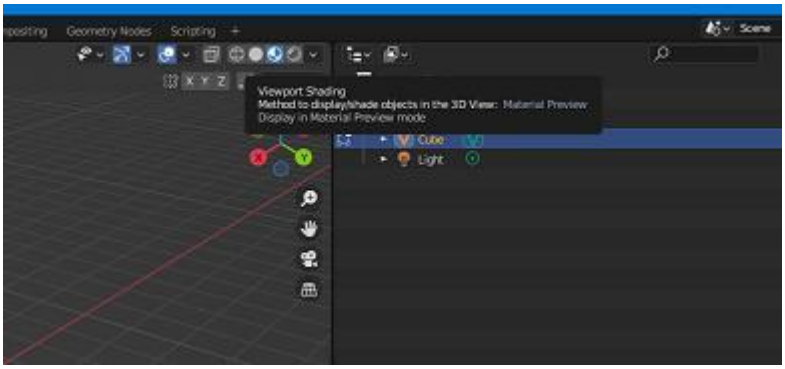
Στη συνέχεια η κάρτα θα ‘ντυθεί’ με τα κατάλληλα υλικά. Η κάθε πλευρά θα περιέχει διαφορετικά υλικά, η μπροστινή όψη και η πίσω συγκεκριμένα. Η όψεις της περιφέρειας θα έχουν απλά ένα χρώμα καθώς δεν θα είναι εμφανής. Για την επιλογή των όψεων της περιφέρειας επιλέγεται αρχικά το κουμπί των όψεων από την πάνω μπάρα αντί του κουμπιού ακμής που είναι τώρα επιλεγμένο, ακολούθως επιλέγονται όλες οι περιφέρειες πατώντας το κουμπί Shift και ακολούθως επιλέγοντας την επόμενη όψη.

Για το χρωματισμό των μοντέλων ο χρήστης θα πρέπει να επιλέξει την καρτέλα με την σφαίρα η οποία περιέχει τις απαιτούμενες λειτουργίες. Εκεί θα εντοπίσει τη λίστα με τα διαθέσιμα υλικά. Αρχικά η εφαρμογή περιέχει ένα μόνο υλικό το οποίο θα χρησιμοποιηθεί για το χρωματισμό των ακμών της περιφέρειας. Θα επιλεγεί σαν χρώμα (Base Color) το μαύρο και ακολούθως θα πατηθεί το κουμπί Assign το οποίο θα εφαρμόσει το επιλεγμένο υλικό στις επιλεγμένες ακμές.

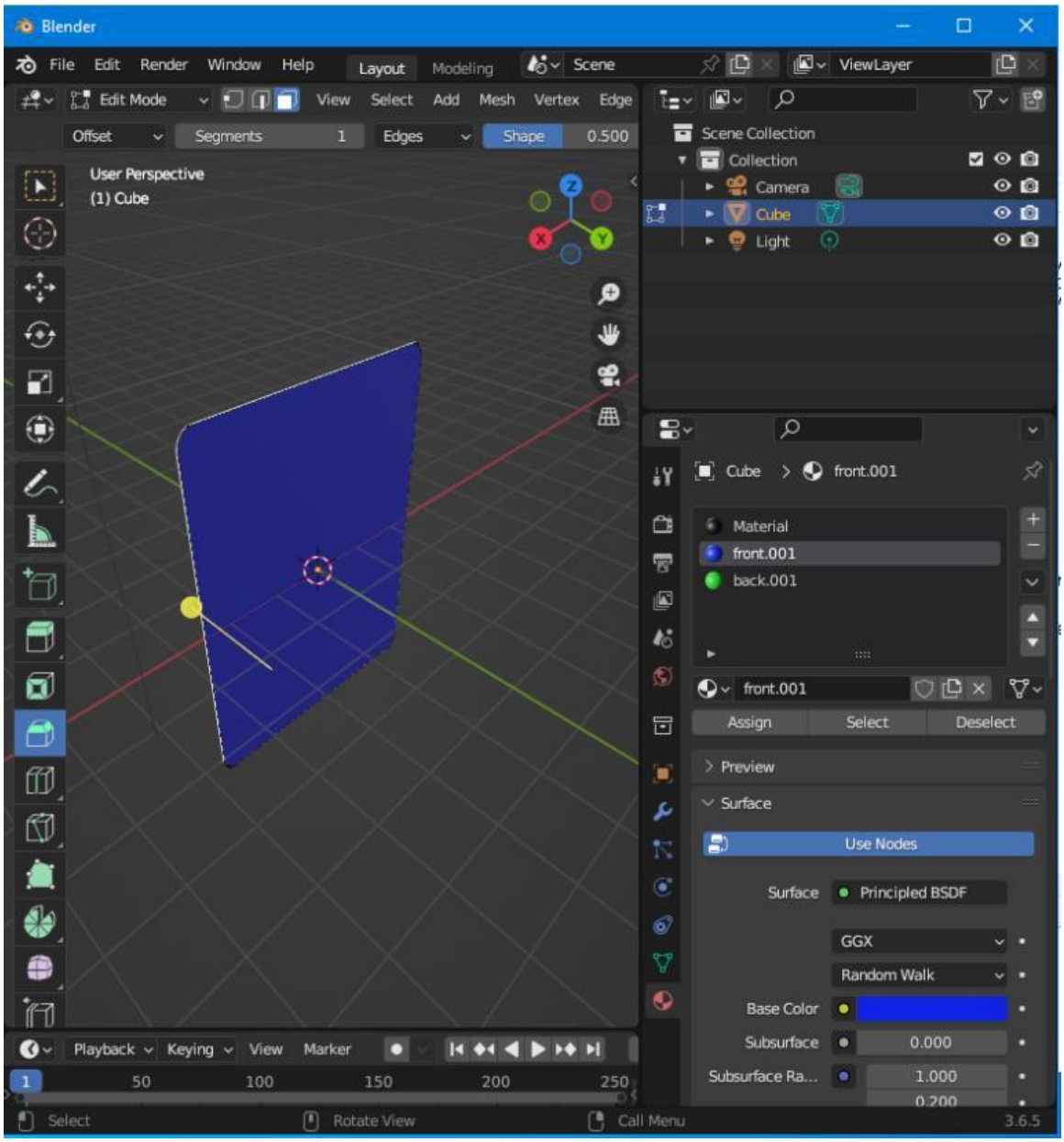


Εικόνα 55: Χρωματισμός ακμών με βασικό χρώμα. Πηγή: Αλέξανδρος Βεκίλογλου

Για την μπροστά όψη και την πίσω θα δημιουργηθούν νέα υλικά ώστε η κάθε πλευρά να έχει το δικό της υλικό. Συνεπώς ο χρήστης θα πρέπει να επιλέξει το κουμπί '+' στα δεξιά της λίστας με τα υλικά ώστε να δημιουργήσει νέα. Θα δημιουργηθούν δύο νέα υλικά με ονόματα front και back για τις δύο όψεις αντίστοιχα όπου θα ανατεθούν διαφορετικά χρώματα ώστε να ξεχωρίζουν και ακολούθως θα εφαρμοστούν στις ακμές. Για να μπορέσει να δει ο χρήστης τις αλλαγές θα πρέπει να επιλεγεί λειτουργία Material Preview.



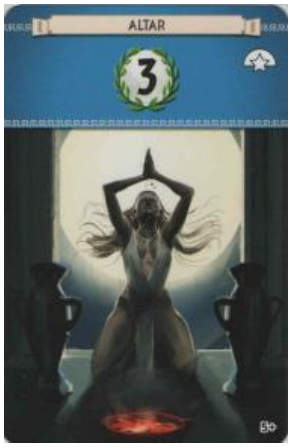
Εικόνα 56: Λειτουργία προβολής υλικών. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 57: Το μοντέλο χρωματισμένο με τα υλικά. Πηγή: Αλέξανδρος Βεκίλογλου

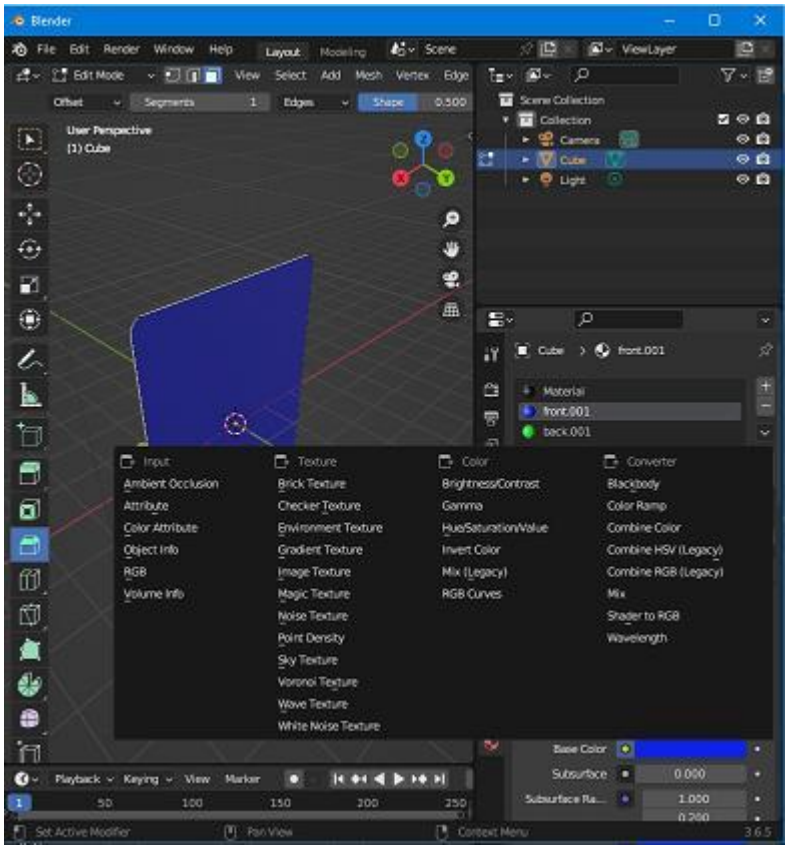
Η κάρτα όμως δεν θα ντυθεί με ένα απλό χρώμα αλλά με εικόνα. Πιο συγκεκριμένα η πίσω όψη της κάρτας θα έχει την εικόνα της πίσω όψης της κάρτας ενώ η μπροστινή όψη (αυτή που θα είναι ορατή μόνο στο παίχτη) θα περιέχει την μπροστά όψη. Επειδή οι κάρτες θα περιέχουν πολλές και διαφορετικές μπροστά όψεις η Unity θα δημιουργεί τις κάρτες με τη πίσω όψη σταθερή και για την μπροστά θα επιλέγει την κατάλληλη για κάθε κάρτα. Την εικόνα δηλαδή που θα αποτελέσει το υλικό την μπροστά όψης θα την καθορίσει η Unity.

Εικόνα 58: Η πίσω όψη της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου



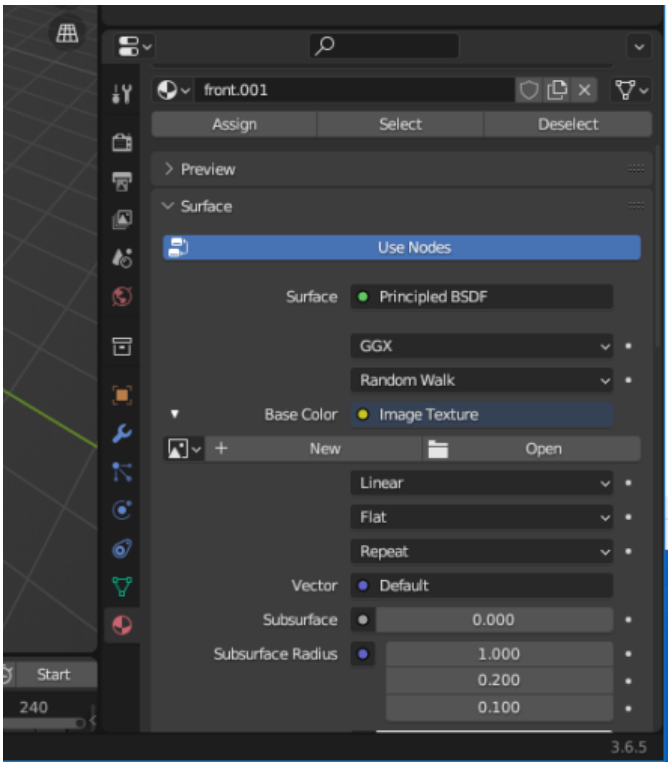
Εικόνα 59: Μία από τις μπροστά όψεις των καρτών. Πηγή: Αλέξανδρος Βεκίλογλου

Για την επιλογή εικόνας αντί χρώματος ο χρήστης θα επιλέξει το κουμπί που βρίσκεται στο Base Color. Ακολούθως θα εμφανιστεί ένα αναδυόμενο μενού από το οποίο θα επιλέξει Image Texture.



Εικόνα 60: Επιλογής εικόνας αντί χρώματος. Πηγή: Αλέξανδρος Βεκίλογλου

Η εφαρμογή θα εμφανίσει διαφορετικές επιλογές όπου ο χρήστης θα επιλέξει το κουμπί Open για να ανοίξει το αρχείο της εικόνας που επιθυμεί να περιέχει το υλικό.

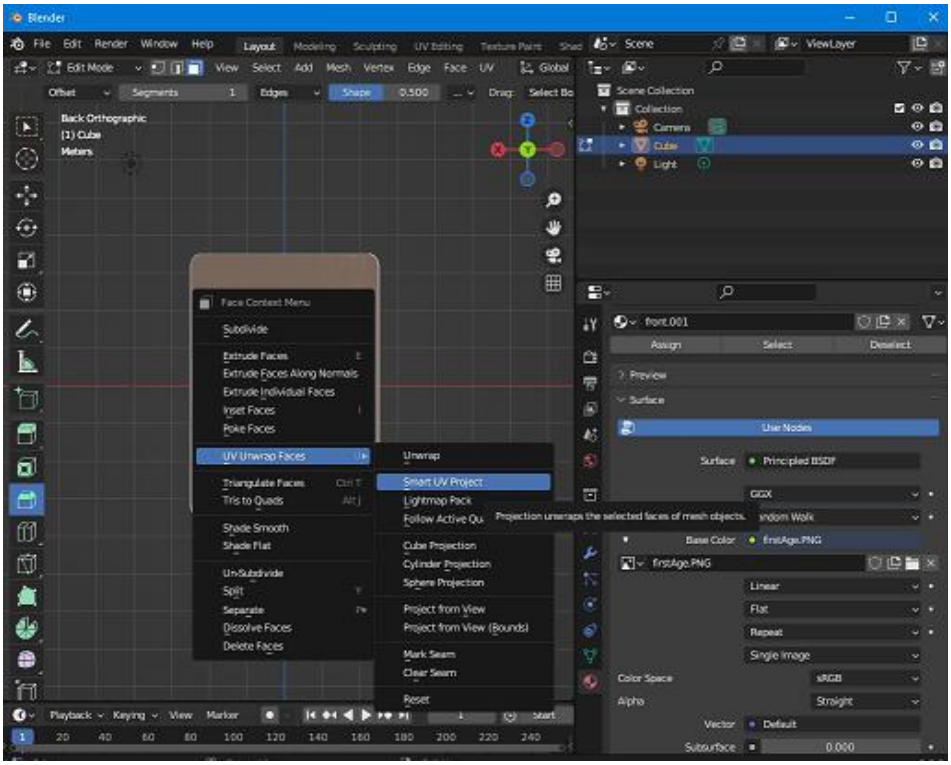


Εικόνα 61: Επιλογή εικόνας. Πηγή: Αλέξανδρος Βεκίλογλου

Επιλέγοντας Assign η εικόνα ανατίθεται στην επιλεγμένη πλευρά. Το αποτέλεσμα δεν είναι το επιθυμητό καθώς η εικόνα δεν εμφανίζεται σωστά πάνω στο αντικείμενο και για το λόγο αυτό θα πρέπει να γίνουν επιπλέον ενέργειες. Η διαδικασία αυτή επιτυγχάνεται μέσα από τη χρήση του UV Map.

Εικόνα 62: Εφαρμογή εικόνας πάνω στο αντικείμενο. Πηγή: Αλέξανδρος Βεκίλογλου

Αρχικά θα εφαρμοστεί η αυτόματη διαδικασία παραμετροποίησης του UV χάρτη. Για το σκοπό αυτό ο χρήστης θα πρέπει να επιλέξει την όψη και ακολούθως να πατήσει το δεξί κουμπί του ποντικιού. Θα εμφανιστεί ένα αναδυόμενο μενού από το οποίο ο χρήστης θα επιλέξει UV Unwrap Faces / Smart UV Project.



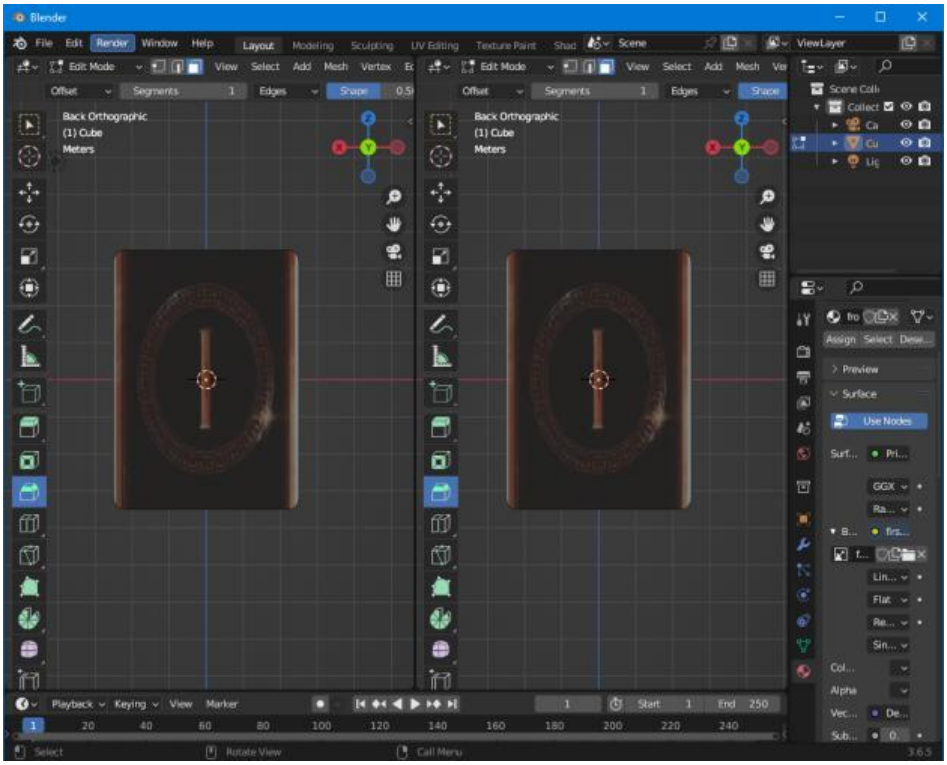
Εικόνα 63: Ενεργοποίηση λειτουργίας επεξεργασίας χάρτη. Πηγή: Αλέξανδρος Βεκίλογλου

Στο παράθυρο που εμφανίζεται θα επιλέξει οκ και το πρόγραμμα θα προσπαθήσει να προσαρμόσει την εικόνα στο αντικείμενο.

Εικόνα 64: Αποτέλεσμα της διαδικασίας. Πηγή: Αλέξανδρος Βεκίλογλου

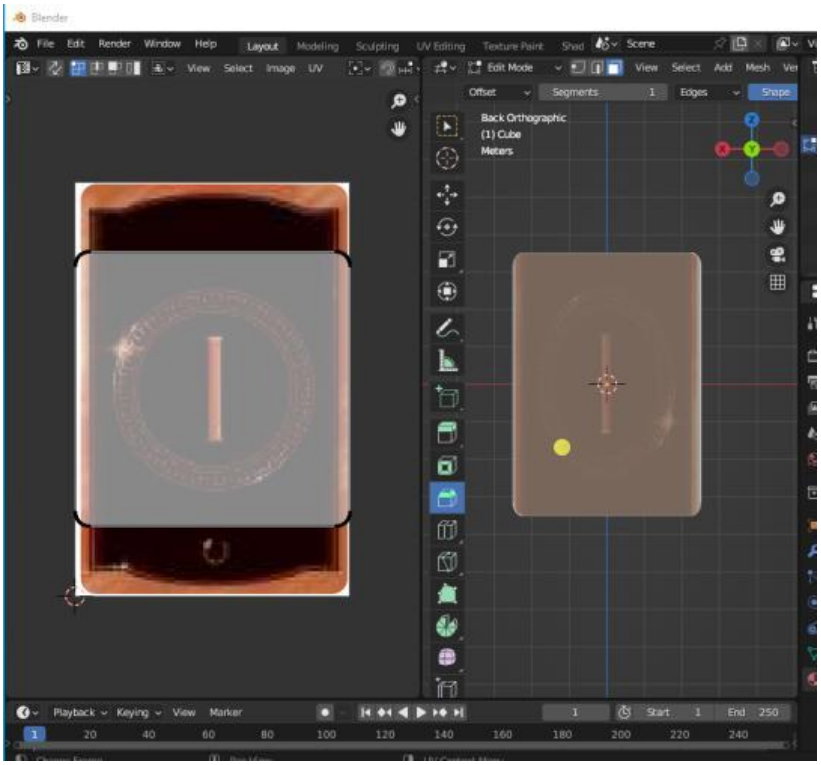
Το αποτέλεσμα είναι αρκετά βελτιωμένο όμως απαιτούνται επιπλέον ενέργειες καθώς η εικόνα φαίνεται να έχει τεντωθεί προς τα πάνω και κάτω ώστε να μπορεί να καλύψει όλη την κάρτα.

Προκειμένου να διορθωθεί αυτό θα ανοιχτεί ένα νέο παράθυρο ώστε ο χρήστης να έχει ένα παράθυρο για επεξεργασία και άλλο ένα για τη προβολή του αποτελέσματος. Για το σκοπό αυτό τοποθετεί τον κέρσορα στην πάνω αριστερή γωνία, πατάει το αριστερό κουμπί και ακολούθως σέρνει το παράθυρο δεξιά με αποτέλεσμα ένα νέο παράθυρο να εμφανίζεται από πίσω.



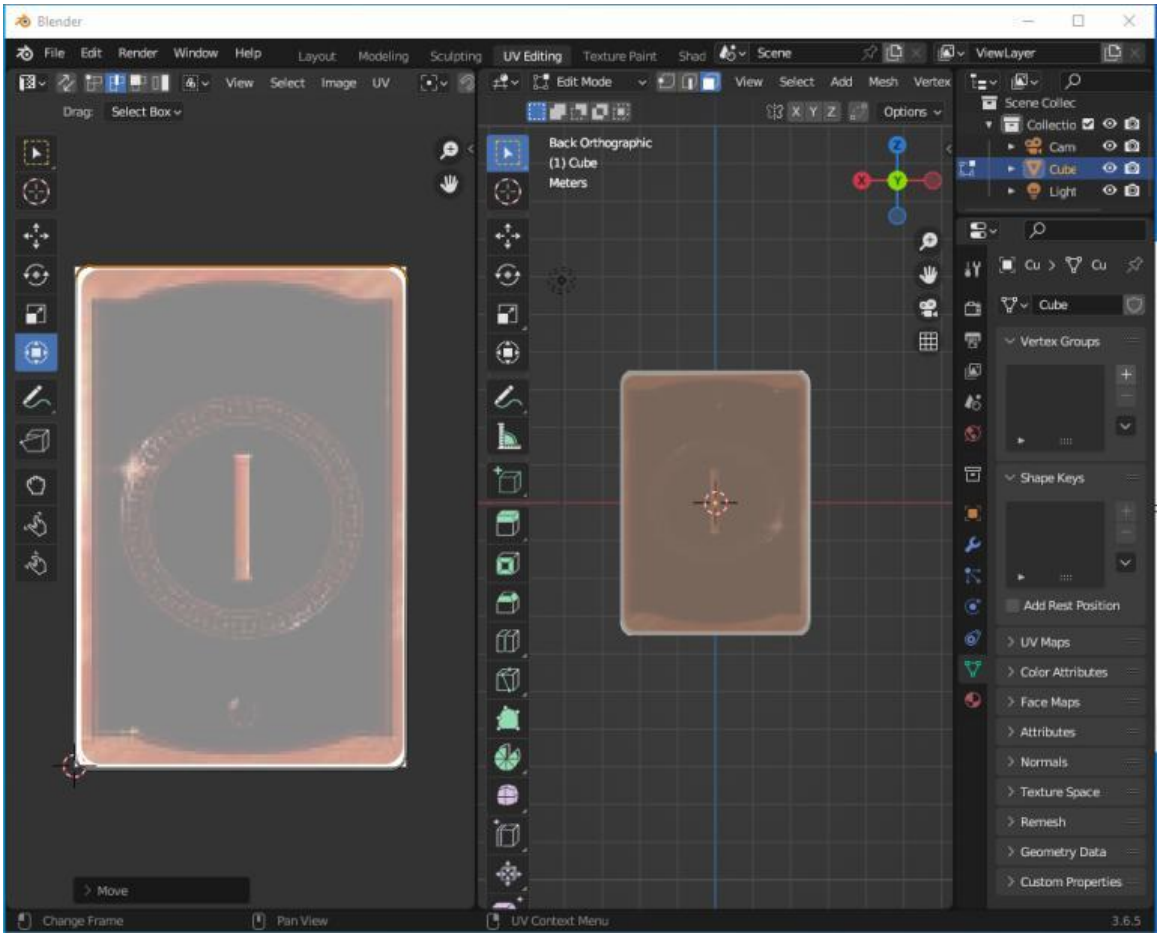
Εικόνα 65: Εμφάνιση διπλού παραθύρου. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως, στο αριστερό παράθυρο θα ανοίξει τον UV Editor από το εικονίδιο Editor Type στην πάνω αριστερή γωνία. Επιλέγοντας την όψη που επιθυμεί να επεξεργαστεί από το δεξί παράθυρο θα εμφανιστεί ο χάρτης στο αριστερό.



Εικόνα 66: UV editor. Πηγή: Αλέξανδρος Βεκίλογλου

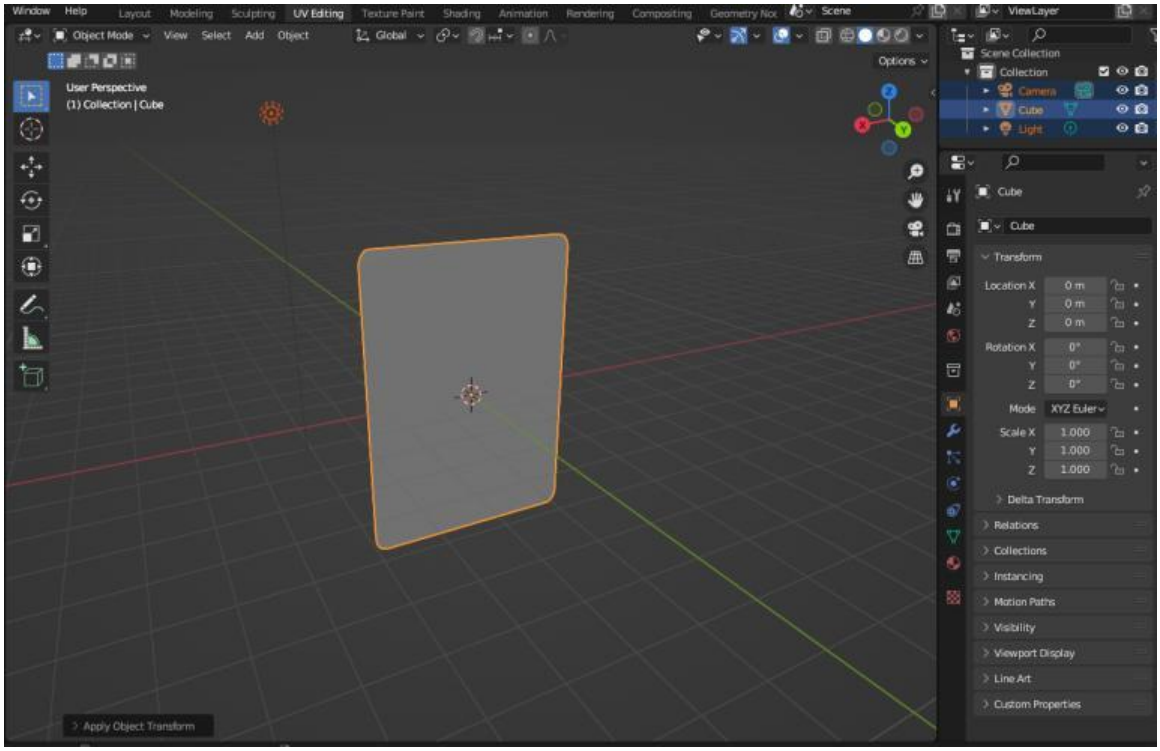
Από εκεί είναι εμφανές ότι ο χάρτης δεν καλύπτει όλη την επιφάνεια της πλευράς και αυτός είναι ο λόγος για τον οποίο η εικόνα ‘τεντώνεται’ για να καλύψει όλη την επιφάνεια. Θα πρέπει να μεγαλώσει ώστε να καλύπτει όλη την επιφάνεια και αυτό επιτυγχάνεται επιλέγοντας UV Editing από την πάνω μπάρα και ακολούθως το κουμπί Transform. Με τη λειτουργία αυτή μπορεί να επιλέξει είτε κορυφή (σημείο), είτε ακμή (γραμμή) είτε όψη, συνεπώς θα πρέπει να επιλέξει ακμή και να μετακινήσει κατάλληλα τις ακμές ώστε ο χάρτης να καταλαμβάνει όλη την όψη. Το αποτέλεσμα είναι τελειοποιημένο.



Η διαδικασία του ορισμού του μεγέθους του χάρτη θα πραγματοποιηθεί και για την άλλη όψη ώστε όταν η Unity προσθέτει εικόνα στο υλικό αυτή να εμφανίζεται με τις σωστές διαστάσεις όπως και στην όψη που μόλις κατασκευαστικέ. Παράλληλα, μέσα από τη διαδικασία αυτή ο χρήστης μπορεί να περιστρέψει την εικόνα ώστε να εμφανίζεται με τη σωστή όψη πάνω, κλπ.

Εφαρμογή αλλαγής μεγέθους

Τελευταίο βήμα προτού να εξαχθεί το μοντέλο στη Unity είναι η εφαρμογή της αλλαγής του μεγέθους του. Πιο συγκεκριμένα το μοντέλο έχει αλλάξει κλίμακα και θα πρέπει η αλλαγή αυτή να εφαρμοστεί ώστε να είναι κατάλληλο για τη Unity. Θα πρέπει να επιλεγεί όλο το αντικείμενο από τη λειτουργία Object Mode κάνοντας κλικ οπουδήποτε στην οθόνη και κατόπιν πατώντας το A. Στη συνέχεια από το μενού Object επιλέγεται το Apply και κατόπιν το Scale. Πλέον στις διαστάσεις του αντικειμένου εμφανίζεται η τιμή 1.

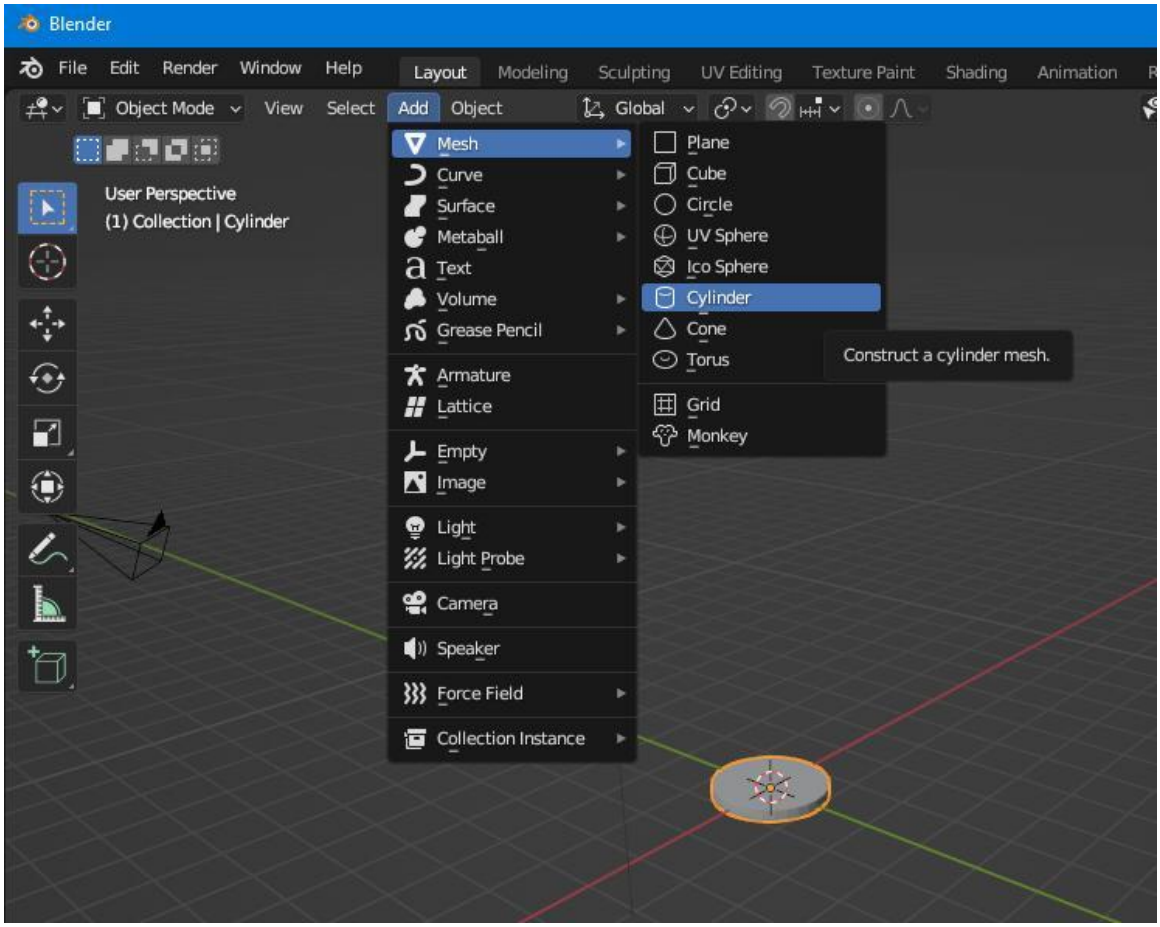


Εικόνα 67: Το αντικείμενο έτοιμο για εξαγωγή. Πηγή: Αλέξανδρος Βεκίλογλου

5.1.2 Κατασκευή Νομισμάτων

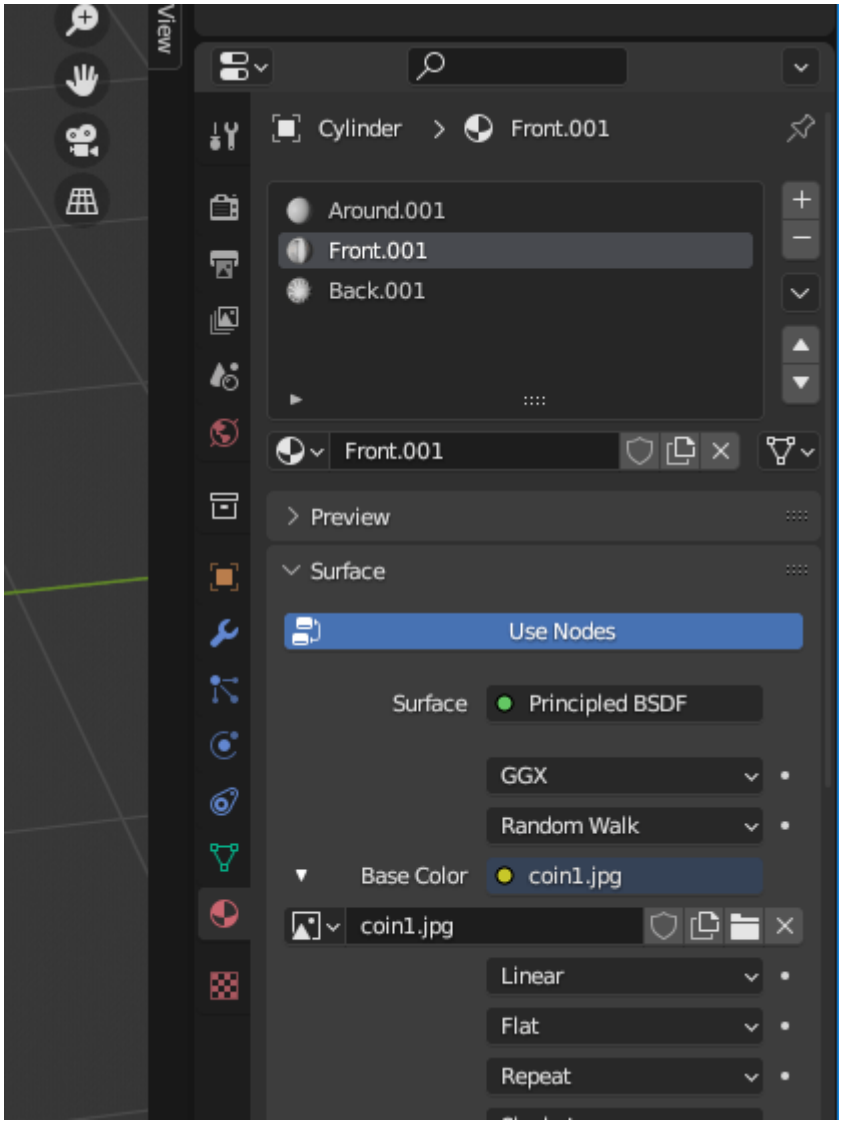
Για τα νομίσματα θα κατασκευαστούν δύο, ένα το οποίο θα χρησιμοποιηθεί για τα νομίσματα του ενός και άλλο ένα για τα νομίσματα των 3. Το νόμισμα του ενός είναι λίγο μικρότερο από αυτό των τριών. Για τη κατασκευή του μοντέλου θα χρησιμοποιηθεί ένας κύλινδρος ο οποίος θα έχει μήκος 0.1. Ο κύλινδρος, όπως και πολλά άλλα βασικά αντικείμενα δημιουργούνται στο Blender έχοντας επιλέξει τη λειτουργία αντικειμένων (Object Mode) και ακολούθως από το μενού Add.

Εικόνα 68: Τα νομίσματα του 7 Wonders. Πηγή: 7 wonders



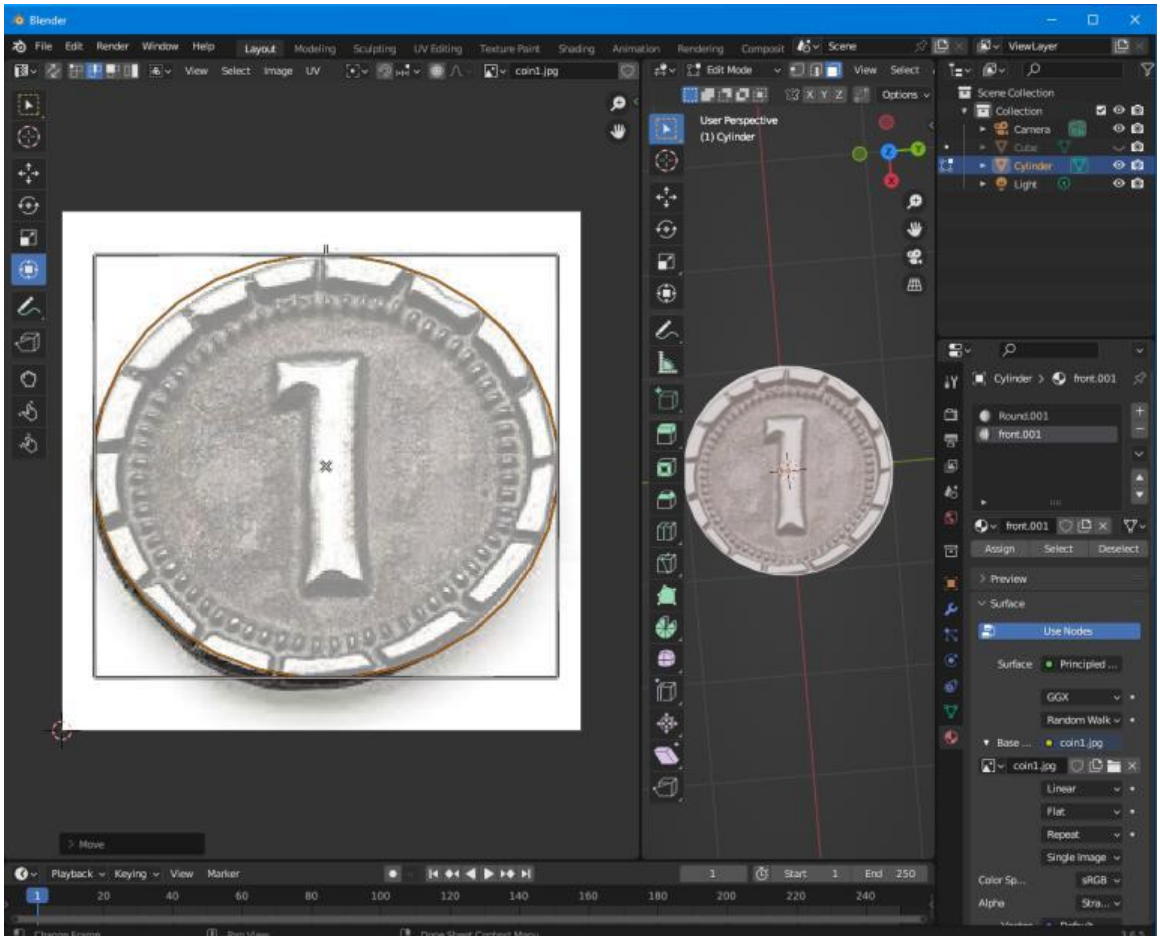
Εικόνα 69: Κατασκευή κυλίνδρου για τα κέρματα. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις δημιουργηθεί το μοντέλο και δοθούν οι κατάλληλες διαστάσεις θα δημιουργηθούν τα υλικά που θα το ντύσουν όπως ακριβώς παρουσιάστηκε στη προηγούμενη παράγραφο. Πιο συγκεκριμένα θα δημιουργηθούν τρία υλικά, ένα για τη μπροστά όψη, ένα για τη πίσω και τέλος ένα για τη περιφέρεια του κάθε νομίσματος.



Εικόνα 70: Τα υλικά του νομίσματος του ενός. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συνέχεια θα τροποποιηθεί ο χάρτης ώστε να ταιριάζει απόλυτα στο μοντέλο. Η διαδικασία επιτυγχάνεται αρχικά με την αυτόματη διαδικασία και στη συνέχεια, μέσω του UV editor, με μετακίνηση αλλά και αλλαγή των διαστάσεων του χάρτη επιτυγχάνεται το βέλτιστο αποτέλεσμα.

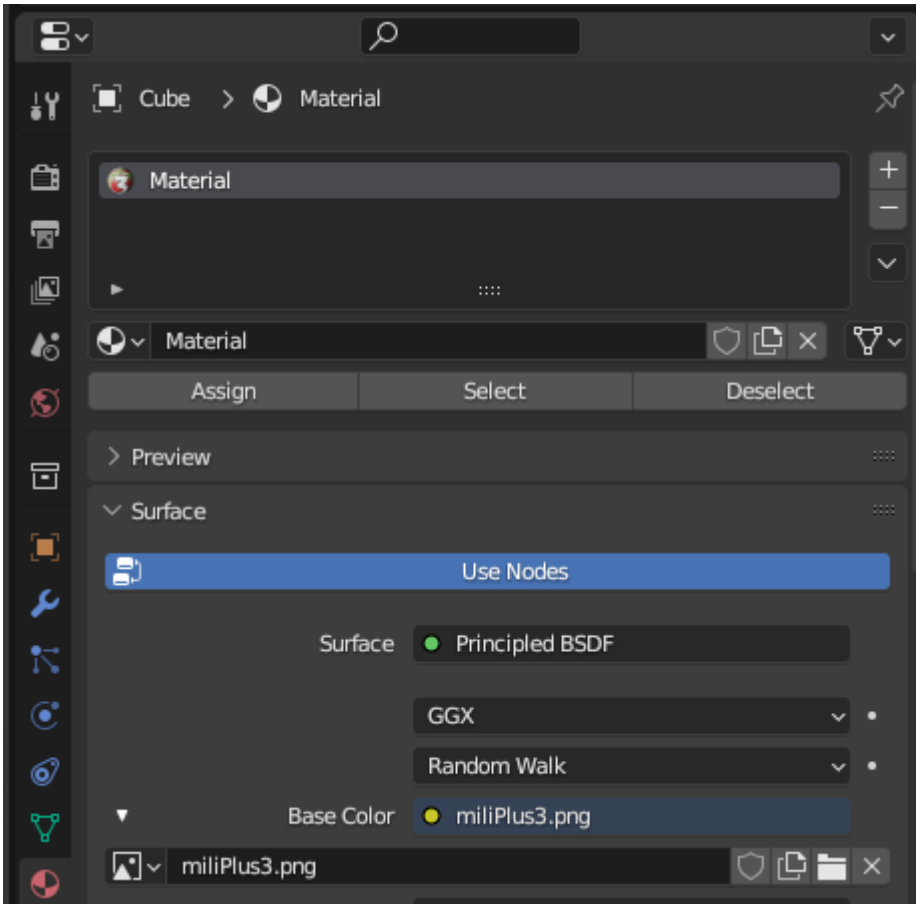


Εικόνα 71: Μετακίνηση και προσαρμογή του UV χάρτη. Πηγή: Αλέξανδρος Βεκίλογλου

Αντίστοιχα κατασκευάζεται και το άλλο νόμισμα και ντύνεται με το κατάλληλο υλικό. Τέλος προσαρμόζονται οι διαστάσεις του και πλέον είναι έτοιμο να εξαχθεί στη Unity.

5.1.3 Κατασκευή Military Tokens

Η κατασκευή ξεκινά από ένα αντικείμενο που θα φέρει την εικόνα του token η οποία θα χρησιμοποιηθεί σαν patron πάνω στο οποίο θα δημιουργηθεί το μοντέλο. Συνεπώς αρχικά δημιουργείται ένα αντικείμενο τετράγωνο, επιλέγεται η πλευρά στην οποία θα ανατεθεί η εικόνα μέσω της επιλογής ‘Edit Mode’ και ‘Face Select’. Ακολούθως ο χρήστης θα επιλέξει τη καρτέλα ‘Material/New’ και ακολούθως θα πατήσει το κουμπί με το χρώμα δίπλα στην επιλογή ‘Base Color’. Από το αναδυόμενο μενού θα επιλέξει ‘Image Texture’ και η εφαρμογή θα εμφανίσει το κουμπί ‘Open’ για να επιλέξει την εικόνα που επιθυμεί. Ακολούθως θα επιλέξει την πλευρά του αντικειμένου στην οποία θέλει να ανατεθεί και τέλος το κουμπί ‘Assign’.



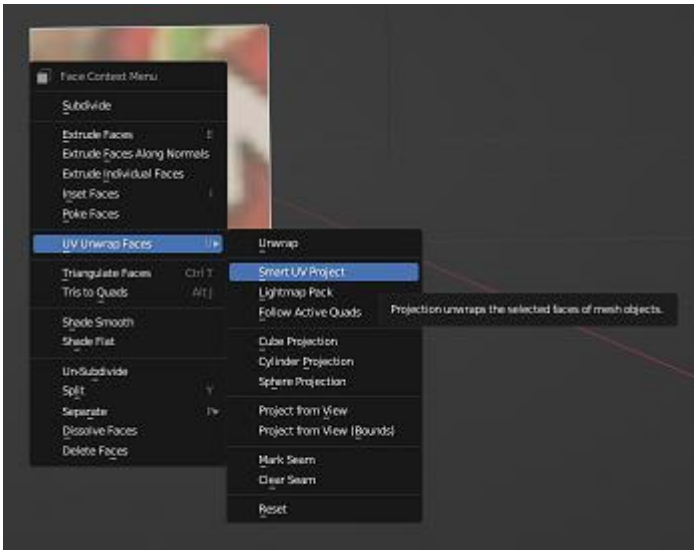
Εικόνα 72: Ανάθεση εικόνας σε πλευρά του αντικειμένου. Πηγή: Αλέξανδρος Βεκίλογλου

Για να μπορέσει να δει το τελικό αποτέλεσμα θα πρέπει να επιλέξει ‘ViewPort Shading’.

Εικόνα 73: Αλλαγή προβολής του μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

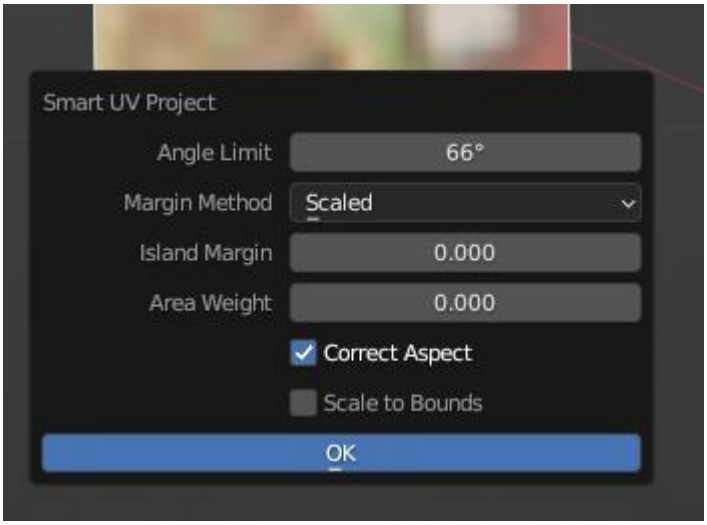
Εικόνα 74: Το μοντέλο μετά την ανάθεση της εικόνας. Πηγή: Αλέξανδρος Βεκίλογλου

Είναι προφανές ότι ο χρήστης θα πρέπει να καθοδηγήσει την εφαρμογή ώστε να τοποθετήσει σωστά την εικόνα στο αντικείμενο. Αρχικά εκτελεί δεξί κλικ στο αντικείμενο και από το αναδυόμενο μενού επιλέγει ‘UV Unwrap Faces/ Smart UV Project’.



Εικόνα 75: Επιλογή UV Unwrap Faces. Πηγή: Αλέξανδρος Βεκίλογλου

Η εφαρμογή εμφανίζει το παράθυρο με τις επιλογές όπου ο χρήστης θα επιλέξει ‘OK’.

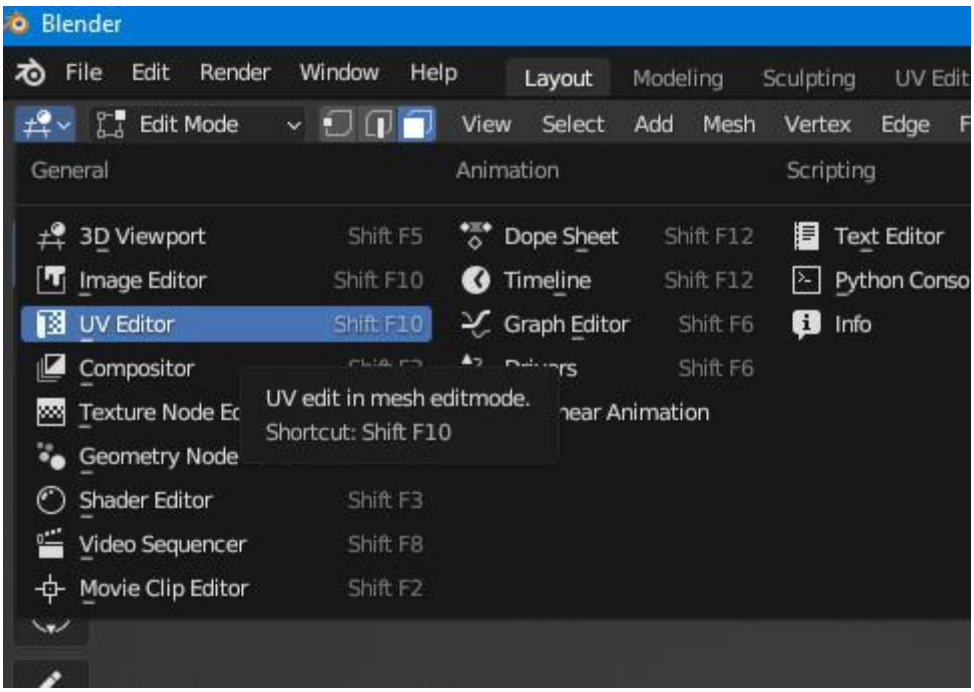


Εικόνα 76: Παράθυρο 'Smart UV Project'. Πηγή: Αλέξανδρος Βεκίλογλου

Μετά το ‘OK’ η εικόνα εμφανίζεται στο μοντέλο σαφώς καλύτερα.

Εικόνα 77: Το αποτέλεσμα του Smart UV Project. Πηγή: Αλέξανδρος Βεκίλογλου

Η αυτοματοποιημένη διαδικασία όμως δεν αρκεί για το επιθυμητό αποτέλεσμα καθώς η εικόνα εμφανίζεται ανάποδα αλλά και δεν εμφανίζεται ολόκληρη. Για το σκοπό αυτό ο χρήστης θα τοποθετήσει τον κέρσορα στην πάνω αριστερή γωνία ώσπου να μετατραπεί σε σταυρό και ακολούθως θα κάνει δεξί κλικ και θα σύρει προς τα δεξιά. Με τον τρόπο αυτό η οθόνη θα διαιρεθεί σε δύο τμήματα όπου το ένα θα χρησιμοποιηθεί για την επεξεργασία του χάρτη και το 2ο για την προβολή των αποτελεσμάτων, όπως ακριβώς και στη προηγούμενη περίπτωση. Στη συνέχεια θα ενεργοποιήσει τον 'UV Editor' επιλέγοντας το κατάλληλο εικονίδιο από το μενού.



Εικόνα 78: Ενεργοποίηση του UV Editor. Πηγή: Αλέξανδρος Βεκίλογλου

Ο χρήστης θα πρέπει να επιλέξει από το αριστερό παράθυρο την επιλογή του μενού ‘UV Editing’ και ακολούθως θα επιλέξει από το δεξί παράθυρο την πλευρά που του αντικειμένου που η εικόνα έχει ανατεθεί.

Εικόνα 79: UV Editing. Πηγή: Αλέξανδρος Βεκίλογλου

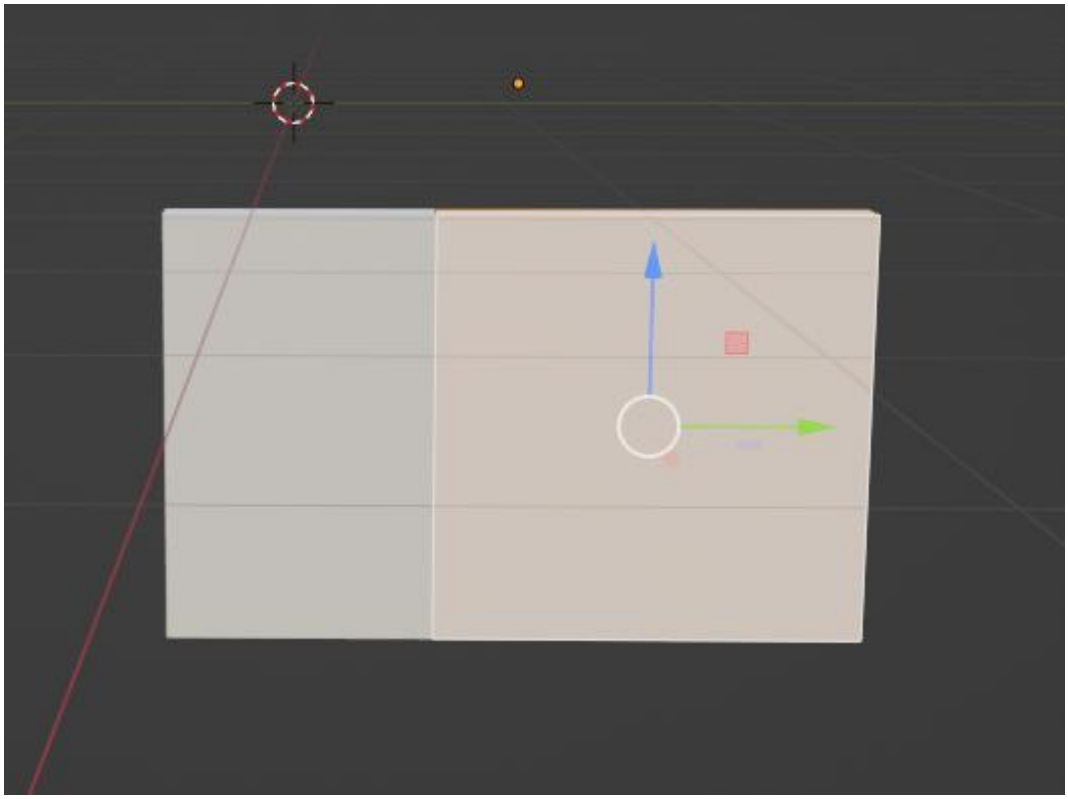
Πλέον γίνεται προφανές στον αναγνώστη για πιο λόγο η εικόνα εμφανίζεται κομμένη στο μοντέλο. Ο χρήστης μπορεί να επιλέξει ‘Rotate’ και να περιστρέψει το χάρτη αλλά και ‘Scale’ ώστε να τον μεγαλώσει αρκετά ώστε να περιλαμβάνει ολόκληρη την εικόνα.

Εικόνα 80: Το τελικό αποτέλεσμα. Πηγή: Αλέξανδρος Βεκίλογλου

Για να κλείσει το νέο παράθυρο θα τοποθετήσει τον κέρσορα στην πάνω αριστερή γωνία του παραθύρου που θέλει να διατηρήσει, θα πατήσει το αριστερό κλικ και θα σύρει το παράθυρο πάνω σε εκείνο που επιθυμεί να κλείσει. Ο κέρσορας θα μετατραπεί σε βέλος και μόλις αφεθεί το κουμπί του ποντικιού το παράθυρο θα κλείσει.

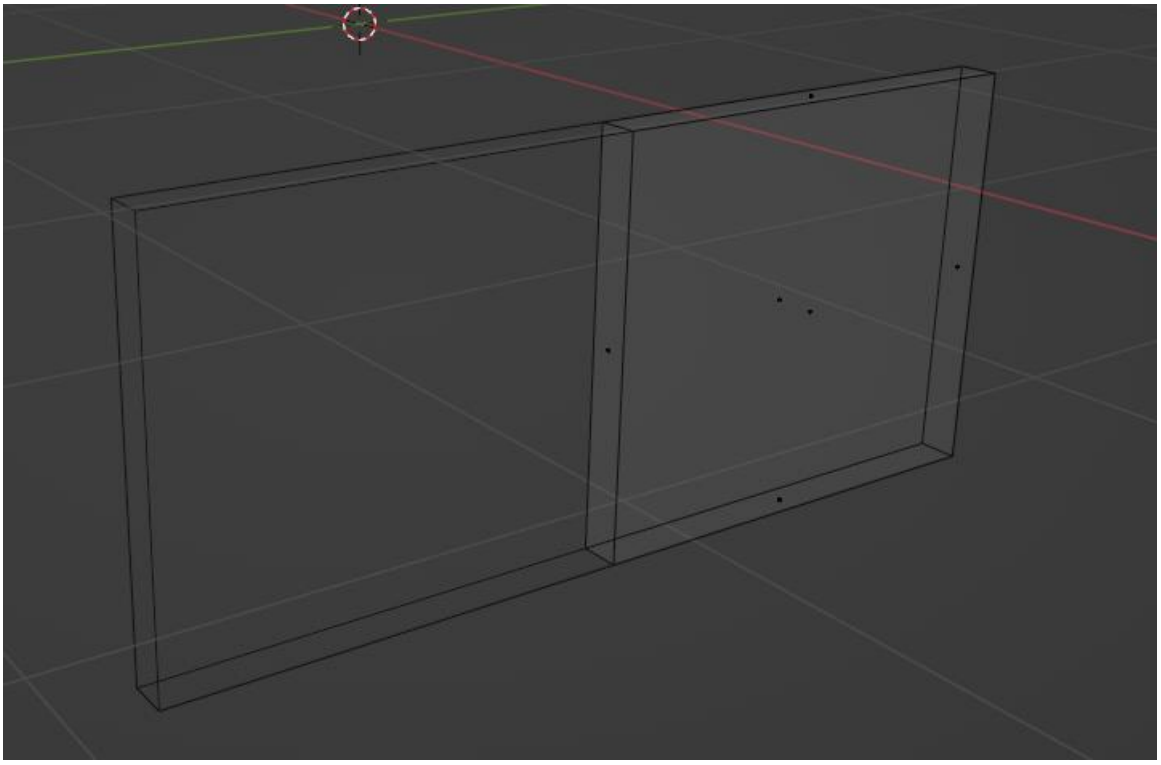
Έχοντας ένα μοντέλο με την εικόνα σαν οδηγό μπορεί πλέον να ξεκινήσει η διαδικασία της κατασκευής του πραγματικού μοντέλου το οποίο θα δημιουργηθεί σε ένα τετράγωνο ίδιων διαστάσεων. Για το λόγο αυτό θα αντιγραφεί το μοντέλο επιλέγοντας 'Object Mode/ Object/ Duplicate Objects' και από το 2^{ο} θα διαγραφεί η εικόνα.

Για τη δημιουργία του μοντέλου θα εφαρμοστεί ο 'Mirror Modifier' στο μοντέλο ώστε ο χρήστης να κατασκευάσει την μια πλευρά και η άλλη να δημιουργηθεί από την εφαρμογή σαν το είδωλο της πρώτης. Για το σκοπό αυτό θα επιλεγεί το αντικείμενο και ακολούθως από την καρτέλα 'Modifiers' θα επιλεγεί ο 'Mirror'. Σαν άξονας θα επιλεγεί ο Y και ακολούθως ο χρήστης θα επιλέξει 'G' από το πληκτρολόγιο και ακολούθως Y ώστε να δημιουργηθεί το αντίγραφο. Ο χρήστης θα επιλέξει την επιλογή 'Clipping' ώστε τα αντικείμενα να κολλήσουν και ακολούθως θα τα κολλήσει. Πλέον ότι επεξεργασία γίνεται στη μία πλευρά θα γίνεται αυτόματα και στην άλλη.



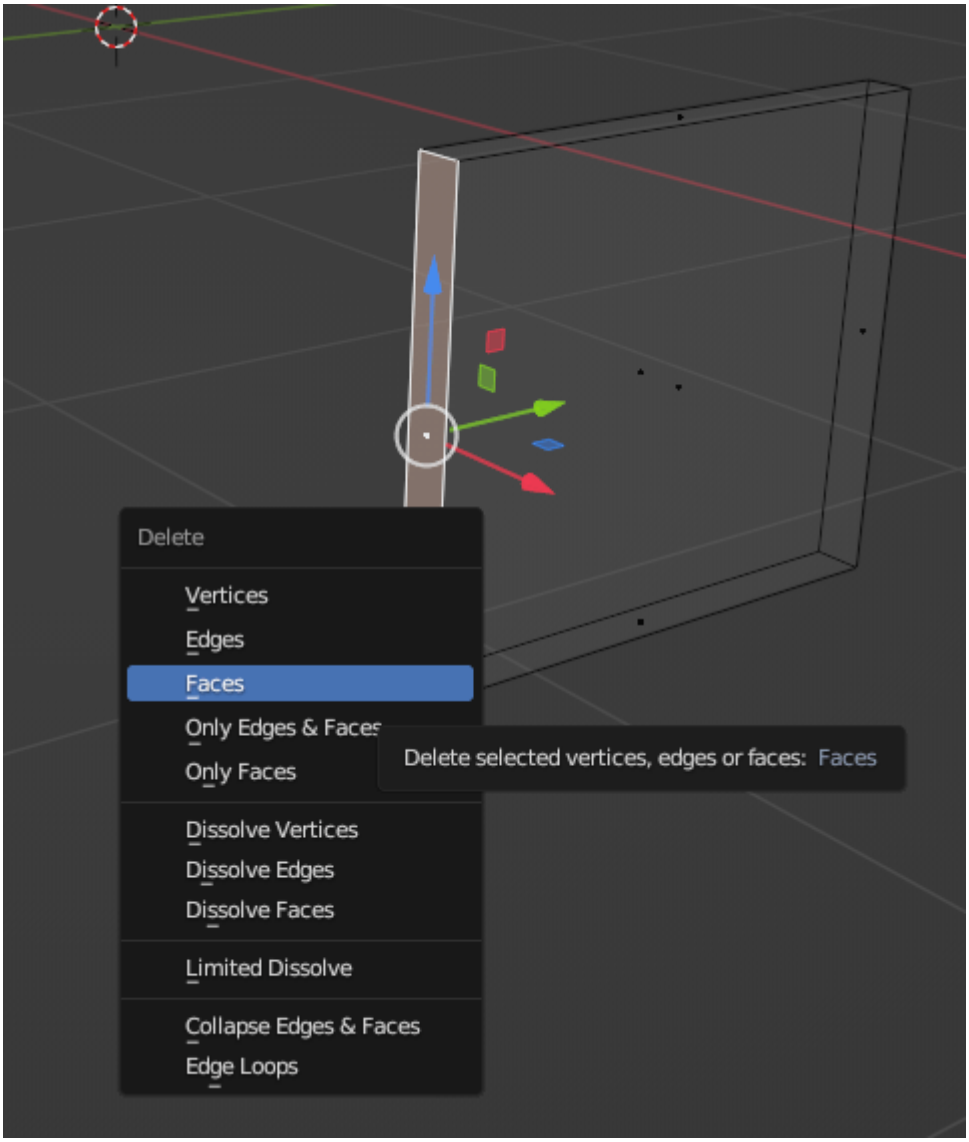
Εικόνα 81: Το μοντέλο σε mirror και clipping. Πηγή: Αλέξανδρος Βεκίλογλου

Για να μπορεί ο χρήστης να δει μέσα από το μοντέλο θα πατήσει το πλήκτρο Z και από το αναδυόμενο μενού θα επιλέξει 'Wireframe'.



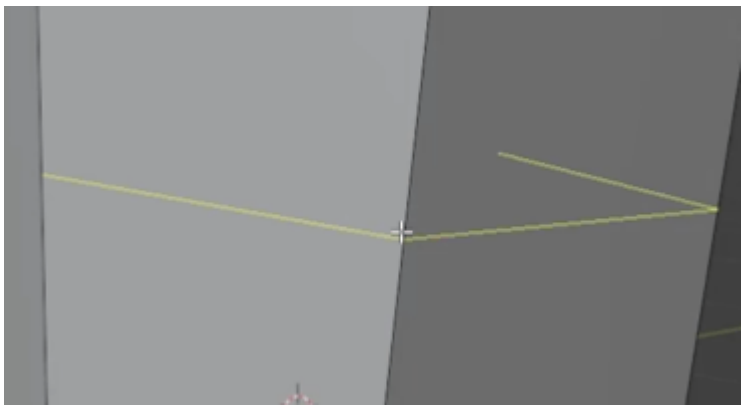
Εικόνα 82: Το Wireframe του μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

Η όψη που υπάρχει ανάμεσα στα δύο μοντέλα θα πρέπει να διαγραφεί για να απλοποιηθεί η διαδικασία της μοντελοποίησης. Για το σκοπό αυτό αφαιρείται προσωρινά η λειτουργία του mirror, ώστε να κρυφτεί το 2ο μοντέλο από το χώρο, απενεργοποιώντας την επιλογή 'Realtime' από τις παραμέτρους του Mirror Modifier. Ακολούθως επιλέγεται η πλευρά, και με το πλήκτρο X εμφανίζεται το μενού διαγραφής από το οποίο επιλέγεται το 'Faces'.



Εικόνα 83: Διαγραφή της πλευράς. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως ενεργοποιείται εκ νέου η λειτουργία του καθρέπτη, ενεργοποιώντας το 'RealTime από τις ιδιότητες του Modifier και πλέον μπορεί να ξεκινήσει η δημιουργία του μοντέλου. Για το σκοπό αυτό θα πρέπει να προστεθούν επιπλέον ακμές στο αντικείμενο. Η διαδικασία επιτυγχάνεται επιλέγοντας 'Edit Mode/Edge Select' και ακολούθως ο χρήστης θα πρέπει να τοποθετήσει το κέρσορα του στην ένωση των δύο πλευρών στις οποίες επιθυμεί να δημιουργηθούν οι επιπλέον άκρες.



Εικόνα 84: Προσθήκη άκρων. Πηγή: Αλέξανδρος Βεκίλογλου

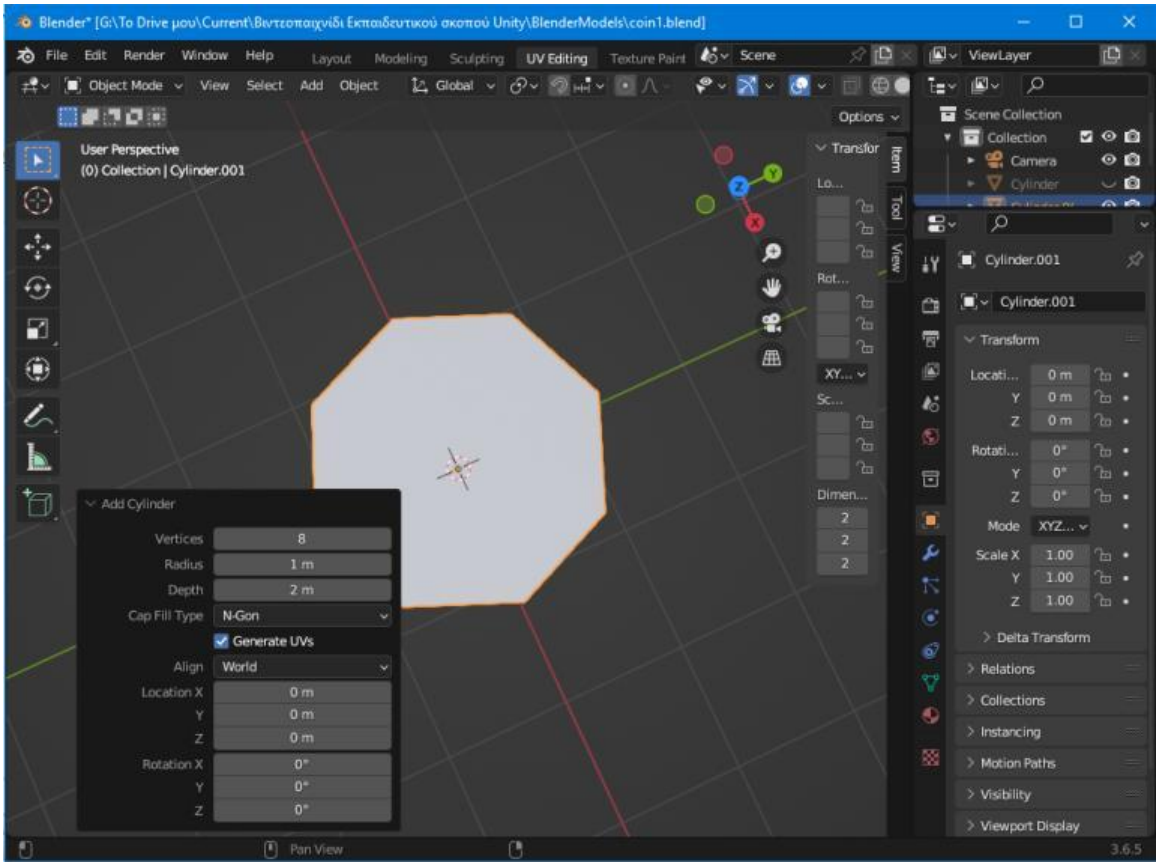
Μόλις τοποθετηθεί ο κέρσορας στη σωστή θέση επιλέγεται ‘Ctrl + R’ και γυρνώντας τη ροδέλα του ποντικιού προς τα επάνω αυξάνεται ο αριθμός των ακμών που τοποθετούνται, ενώ γυρνώντας τη ροδέλα προς τα κάτω μειώνεται. Πατώντας το αριστερό κουμπί οι ακμές δημιουργούνται και τέλος η κίνηση του ποντικιού πάνω ή κάτω μεταφέρει τις νέες ακμές στις τελικές του θέσεις. Πατώντας τέλος το κουμπί εκ νέου οι ακμές τοποθετούνται στις καθορισμένες θέσεις. Στη περίπτωση που η ακμή θα πρέπει να μετακινηθεί εκ νέου ο χρήστης επιλέγει την ακμή και πατάει δύο φορές το πλήκτρο G. Πλέον ο χρήστης μπορεί να μετακινήσει τις ακμές ώστε να επιτύχει το κατάλληλο σχήμα.

Μετά την ολοκλήρωση του μοντέλου ακολουθεί το ντύσιμο του με τα κατάλληλα υλικά όπως παρουσιάστηκε στις προηγούμενες παραγράφους.

Εικόνα 85: Κατασκευή του μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

5.1.4 Military token -1

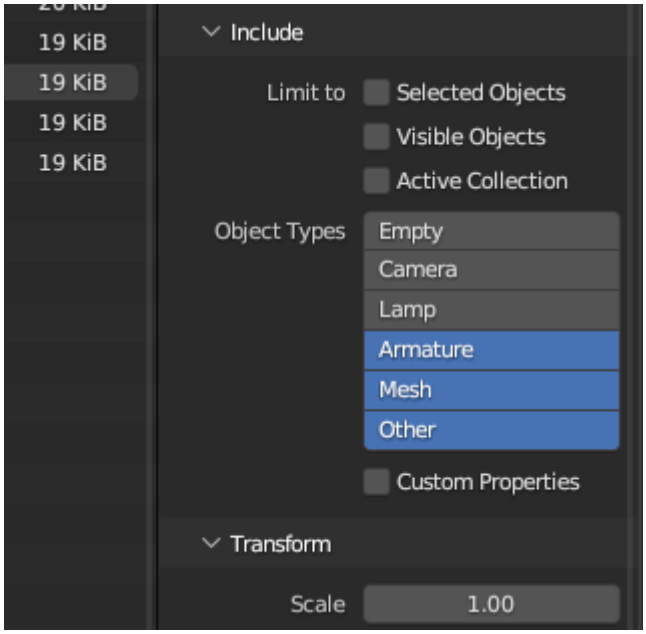
Για τα αντικείμενα αυτά θα δημιουργηθεί όμοια ένα αντικείμενο κυλίνδρου και από το αναδυόμενο μενού κάτω δεξιά Add Cylinder θα οριστεί ο αριθμός των πλευρών σε 8. Ακολούθως θα ντυθεί όπως και τα υπόλοιπα αντικείμενα που παρουσιάστηκαν στις προηγούμενες παραγράφους.



Εικόνα 86: Κατασκευή μοντέλου. Πηγή: Αλέξανδρος Βεκίλογλου

5.1.5 Εξαγωγή των μοντέλων

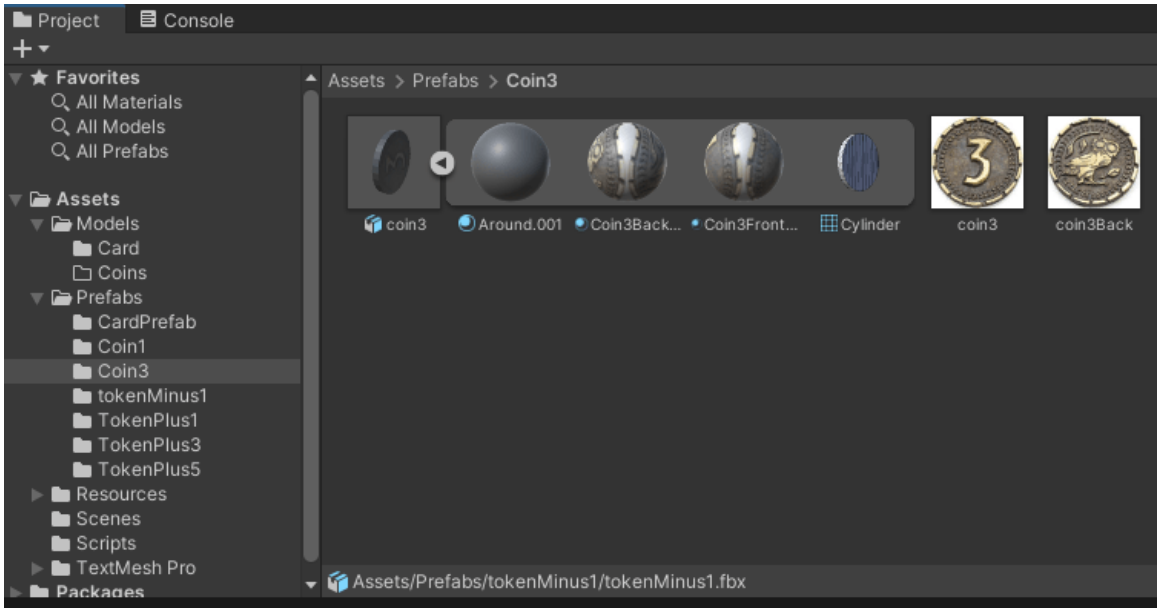
Μόλις ολοκληρωθεί η κατασκευή των μοντέλων ο χρήστης θα πρέπει να εξάγει το αρχείο από το Blender σε κατάλληλη μορφή για τη Unity. Η διαδικασία επιτυγχάνεται από την καρτέλα File και ακολούθως η επιλογή Export FBX. Από το παράθυρο που θα εμφανιστεί ο χρήστης πέρα από το όνομα του αρχείου και τη διαδρομή του θα πρέπει να επιλέξει τα μέρη του αντικειμένου που θα εξαχθούν. Από τα διαθέσιμα θα επιλέξει Armature, Mesh και Other. Τα υπόλοιπα, κάμερα κλπ., δεν θα πρέπει να εξαχθούν καθώς η Unity διαθέτει δική της κάμερα, φωτισμό, κλπ.



Εικόνα 87: Επιλογή των δεδομένων του μοντέλου που θα εισαχθούν στο αρχείο εξαγωγής. Πηγή: Αλέξανδρος Βεκίλογλου

5.2 Εισαγωγή μοντέλων στη Unity

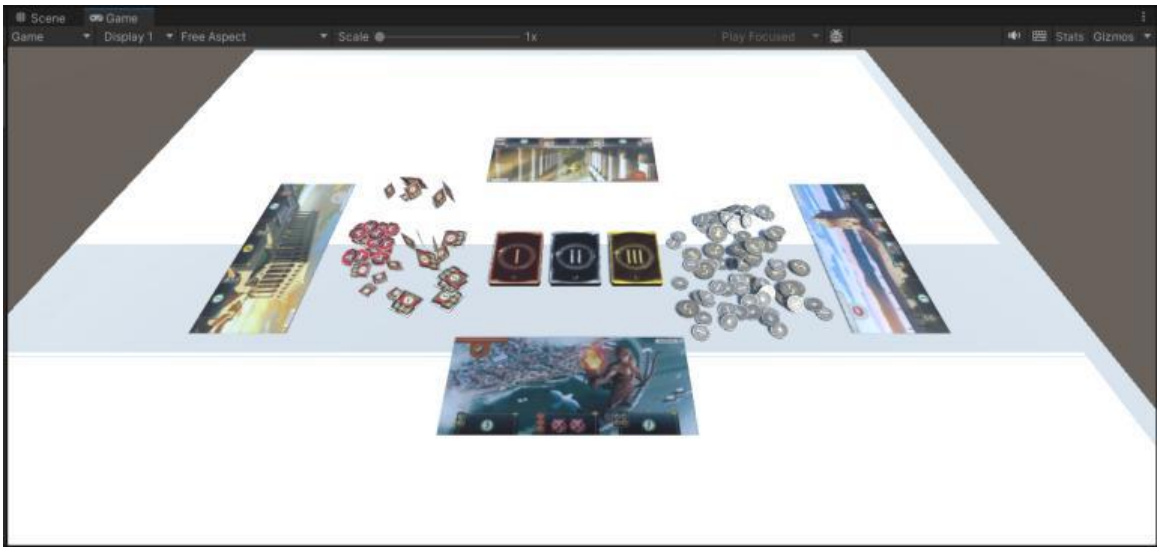
Τα διάφορα μοντέλα θα πρέπει να εισαχθούν στη Unity. Η διαδικασία είναι αρκετά εύκολη και επιτυγχάνεται επιλέγοντας το αρχείο του μοντέλου αλλά και των διάφορων εικόνων που ενδεχομένως να περιέχει και ακολούθως με drag and drop τοποθετούνται στον κατάλληλο φάκελο στη καρτέλα Projects.



Εικόνα 88: Εισαγωγή μοντέλου νομίσματος στη Unity. Πηγή: Αλέξανδρος Βεκίλογλου

5.3 Στήσιμο του παιχνιδιού

5.3.1 Το τραπέζι



Εικόνα 89: Η αρχική κατάσταση της σκηνής. Πηγή: Αλέξανδρος Βεκίλογλου

Το παιχνίδι ξεκινά παρουσιάζοντας ένα κύβο ο οποίος θα αναπαριστά το τραπέζι πάνω στο οποίο οι παίχτες θα παίζουν το παιχνίδι. Ο κύβος διαθέτει Rigidbody ώστε τα αντικείμενα να μην διέρχονται από μέσα του αλλά να σταματάν στην επιφάνεια του. Το αντικείμενο περιέχει και ένα πρόγραμμα, ένα Script το οποίο δημιουργεί τα αντικείμενα μέσα στη σκηνή και τα τοποθετεί στις συγκεκριμένες θέσεις. Το Script αυτό αποτελεί και το βασικό Script της εφαρμογής καθώς πέρα από τη δημιουργία των αντικειμένων αναλαμβάνει να τα συντονίζει, να ελέγχει τους κύκλους του παιχνιδιού καθώς και άλλα.

Πιο συγκεκριμένα το Script αρχικά εκτελεί τη συνάρτηση Awake η οποία εκτελείται μόλις το αντικείμενο δημιουργηθεί από τη Unity. Μέσα στη συνάρτηση αυτή δημιουργούνται οι πίνακες που θα φιλοξενήσουν τα διάφορα αντικείμενα. Οι πίνακες χρησιμοποιούνται ώστε όλα τα αντικείμενα του ίδιου είδους να είναι συγκεντρωμένα στον ίδιο πίνακα ώστε να είναι ευκολότερη η διαχείριση τους.

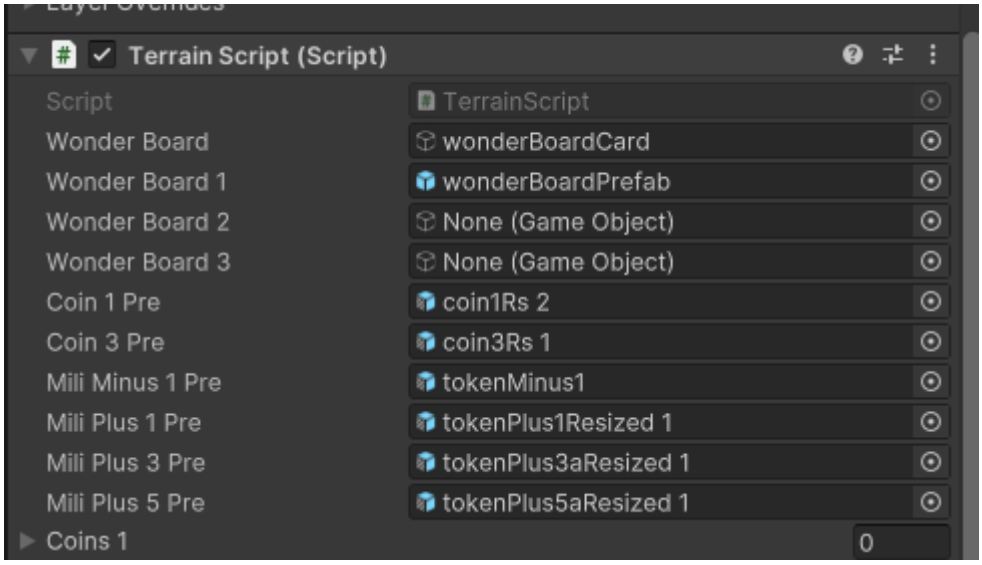
```
void Awake()
{
    ageCards = new GameObject[49];
    secAgeCards = new GameObject[49];
    thirdAgeCards = new GameObject[50];
    secAgeMat = new Material[4];
    coins1 = new GameObject[54];
    coins3 = new GameObject[24];
    miliPlus5 = new GameObject[8];
    miliPlus3 = new GameObject[8];
    miliPlus1 = new GameObject[8];
    miliMinus = new GameObject[24];
}
```

Εικόνα 90: Συνάρτηση Awake. Πηγή: Αλέξανδρος Βεκίλογλου

Τα αντικείμενα αντιστοιχούν σε πραγματικά 3D μοντέλα που έχουν δημιουργηθεί στο παιχνίδι. Για να επιτευχθεί αυτό τα αντικείμενα δημιουργούνται ως κοινόχρηστα (SerializeField) στο Script ώστε η Unity να έχει πρόσβαση σε αυτά και ακολούθως ανατίθεται με drag and drop στις θέσεις – μεταβλητές που εμφανίζονται στο Script. Με τον τρόπο αυτό ένα αντικείμενο (το Terrain για παράδειγμα) αποκτά πρόσβαση σε κάποιο άλλο και μπορεί να το χειριστεί.

```
public class TerrainScript : MonoBehaviour
{
    [SerializeField] GameObject wonderBoard;
    [SerializeField] GameObject wonderBoard1;
    [SerializeField] GameObject wonderBoard2;
    [SerializeField] GameObject wonderBoard3;

    [SerializeField] GameObject coin1Pre, coin3Pre, miliMinus1Pre, miliPlus1Pre, miliPlus3Pre, miliPlus5Pre, camvas, msgCanvas;
   public GameObject[] coins1, coins3, miliMinus, miliPlus1, miliPlus3, miliPlus5;
    [SerializeField] GameObject ageCardPref, ageCard, playerPrefab;
    [SerializeField] Texture2D[] ageTextures;
    [SerializeField] Texture2D[] ageTexturesBack;
    [SerializeField] GameObject[] ageCards, secAgeCards, thirdAgeCards, testCards;
    [SerializeField] ageCardScript cardScr;
    [SerializeField] PlayerScript playerScr0, playerScr1, playerScr2, playerScr3;
    [SerializeField] GameObject[] players;
    public TMP_Text msgPlay1, msgPlay2, msgPlay3;
    public GameObject resourcePrefab;
    public TMP_Text canvasMsgTitle, canvasMsgText, RoundInfoLeft, RoundInfoCenter, RoundInfoRight, RoundInfoTitle;
    public GameObject alertCanvas, ByFromNeigCanvas, roundInfo;
    public int gameTurn;
    public bool turnIsCompleted = false;
    public bool isStackMoved = false;
    public bool millisHas2Move = false;
```

Εικόνα 91: Αρχικοποίηση αντικειμένων. Πηγή: Αλέξανδρος Βεκίλογλου

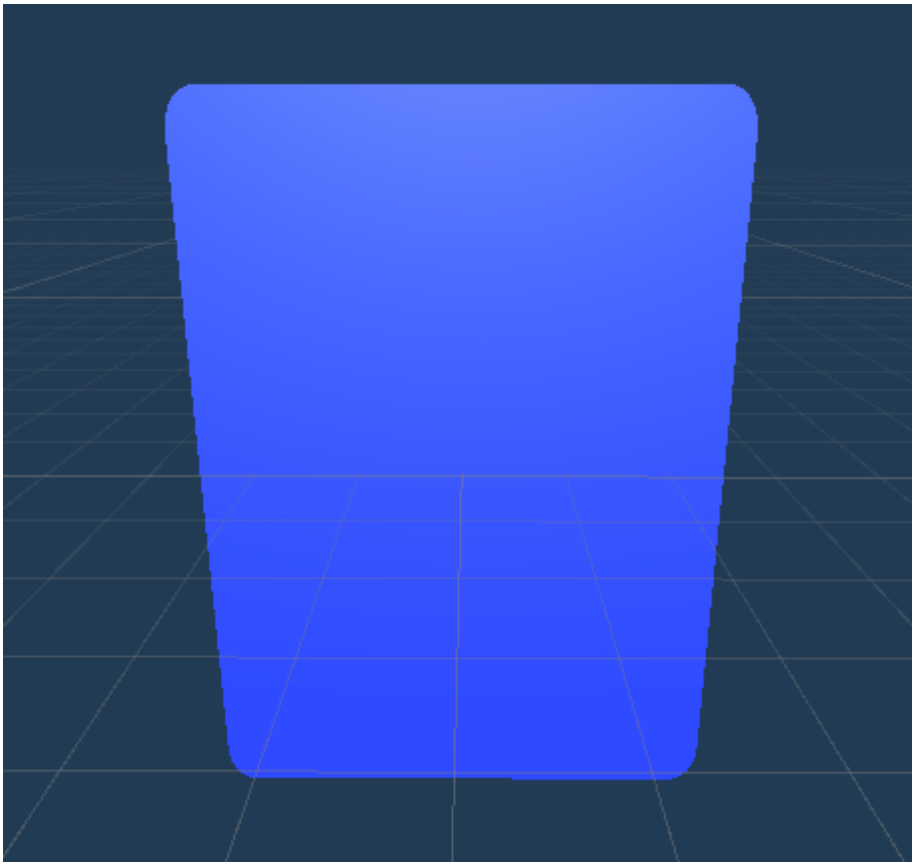


Εικόνα 92: Τα μοντέλα ανατεθειμένα στις μεταβλητές του Script. Πηγή: Αλέξανδρος Βεκίλογλου

Τα μοντέλα αποτελούν Prefabs, δηλαδή μοντέλα τα οποία έχουν δημιουργηθεί και δεν έχουν εισαχθεί στη σκηνή. Πιο συγκεκριμένα, τα Prefabs αποτελούν οντότητες όπως οι κλάσεις. Δηλαδή ο χρήστης μπορεί να δημιουργήσει ένα αντικείμενο και ακολούθως να γράψει τον κατάλληλο κώδικα ώστε η Unity να δημιουργήσει ίδια ή παραλλαγές αυτών σε διαφορετικές θέσεις ή στάσεις. Έτσι ο προγραμματιστής θα δημιουργήσει για παράδειγμα μια κάρτα και η Unity θα δημιουργήσει 24 από αυτή χρησιμοποιώντας κάθε φορά διαφορετική εικόνα για την μπροστά όψη. Όμοια ο χρήστης θα δημιουργήσει από ένα Prefab για κάθε ένα νόμισμα και Military Token και η Unity θα αναλάβει να δημιουργήσει το σύνολο των αντικειμένων αλλά και να τα τοποθετήσει σε τυχαίες θέσεις και με τυχαίο προσανατολισμό εντός της ορισμένης περιοχής. Συνεπώς τα αντικείμενα αυτά δημιουργούνται από το Script του τραπεζιού μόλις ξεκινά το παιχνίδι και τοποθετούνται στις κατάλληλες θέσεις. Παράλληλα, σαν Prefabs, δίνεται η δυνατότητα στο χρήστη να τα τροποποιήσει όλα μαζί απλά τροποποιώντας τις ιδιότητες του Prefab.

5.3.2 Εισαγωγή μοντέλων

Οι κάρτες, λόγο του μεγάλου αριθμού τους έχουν δημιουργηθεί σαν μοντέλα χωρίς εικόνα. Η κάθε κάρτα αποκτά τις όψεις της προγραμματιστικά μέσω του Script. Πιο συγκεκριμένα οι εικόνες των όψεων έχουν ανατεθεί σε πίνακες και η κάθε κάρτα αποκτά την απαραίτητη όψη μαζί με τα υπόλοιπα χαρακτηριστικά της τη στιγμή που δημιουργείται.



Εικόνα 93: Το μοντέλο της κάρτας που θα χρησιμοποιηθεί στην εφαρμογή. Πηγή: Αλέξανδρος Βεκίλογλου

Οι όψεις των καρτών έχουν τοποθετηθεί σε κατάλληλους φακέλους μέσα στο φάκελο Resources της εφαρμογής.

Εικόνα 94: Οι εικόνες των καρτών στα αρχεία της εφαρμογής. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως το Script μέσα στη Start συνάρτηση, η οποία εκτελείται με το που δημιουργηθεί το αντικείμενο, διαβάζονται τα περιεχόμενα του φακέλου και αποθηκεύονται στον κατάλληλο πίνακα.

```
//back texture
ageTexturesBack = Resources.LoadAll<Texture2D>("Materials/ageCards/back/");
```

Εικόνα 95: Αποθήκευση εικόνων στο κατάλληλο πίνακα. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συνέχεια το Script αναλαμβάνει να δημιουργήσει το σύνολο των καρτών που υπάρχουν στο παιχνίδι. Πιο συγκεκριμένα το Script, σε μία συνάρτηση η οποία εκτελείται από την Start ξεκινά αρχικοποιώντας την κάρτα μέσω της Instantiate και στη συνέχεια αποκτά μια αναφορά στο Script με όνομα ageCardScript το οποίο έχει ανατεθεί στο Prefab της κάρτας. Την αναφορά την επιστρέφει η GetComponent η οποία αποτελεί συνάρτηση του κάθε GameObject. Έχοντας την αναφορά αυτή έχει τη δυνατότητα να αρχικοποιήσει τις ιδιότητες του Script της συγκεκριμένης κάρτας, όπως το όνομα της, το χρώμα, τον αριθμό των ελάχιστων παιχτών όπου θα χρησιμοποιείται, κλπ. Στη συνέχεια αποκτά μια αναφορά στα υλικά της κάρτας (στα Materials) και αναθέτει την πίσω όψη εικόνα ανάλογα την περίοδο. Ειδικότερα για την μπροστά αναζητά στον πίνακα με τις εικόνες αυτή που το όνομα της ταυτίζεται με το όνομα της κάρτας.

```
ageCards[i] = Instantiate(ageCardPref);
cardScr = ageCards[i].GetComponent<ageCardScript>();
cardScr.name = "altar";
cardScr.textureName = "altar";
cardScr.color = "Blue";
cardScr.minOfplayers = 5;
cardScr.chainSymb[0] = "Star";
cardScr.victPoints = 3;
ageCardMaterials = ageCards[i].GetComponent<Renderer>().materials;
ageCardMaterials[0].SetTexture("_MainTex", ageTexturesBack[0]);
ageCardMaterials[1].SetTexture("_MainTex", findText(cardScr.textureName));
```

Εικόνα 96: Δημιουργία καρτών. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις δημιουργηθούν οι κάρτες τοποθετούνται στο κέντρο του τραπεζιού η μία πάνω στην άλλη. Για το σκοπό αυτό μέσα στο ίδιο Script δημιουργείται μια συνάρτηση η οποία θα περάσει από όλες τις κάρτες που βρίσκονται στους πίνακες καρτών και θα ενημερώσει τις μεταβλητές θέσης και περιστροφής. Πιο συγκεκριμένα, για τις κάρτες τις πρώτης περιόδου δημιουργείται ένα νέο διάνυσμα όπου περιέχει τις νέες συντεταγμένες τις κάθε κάρτας, τις συντεταγμένες που θα έχει πάνω στη στοίβα των καρτών δηλαδή. Οι τιμές x και y είναι σταθερές η τιμή του άξονα z όμως κάθε φορά αυξάνεται κατά 0.08 ώστε η νέα κάρτα να προστίθεται λίγο πιο ψηλά από τη προηγούμενη έτσι ώστε όλες οι κάρτες να είναι σε μια στοίβα και να πέφτουν πάνω στο τραπέζι από χαμηλό ύψος. Παράλληλα οι κάρτες θα πρέπει να περιστραφούν κατά 90 μοίρες στον άξονα X και στο δικό τους σύστημα συντεταγμένων ώστε έχουν την μπροστά όψη προς τα κάτω. Οι κάρτες τοποθετούνται λίγο πιο ψηλά από την επιφάνεια του τραπεζιού και επειδή διαθέτουν Rigidbody η βαρύτητα τις κάνει να πέφτουν κατά μήκος του άξονα Y. Για να επιτευχθεί αυτό έχει εισαχθεί περιορισμός τους άξονες που κινείται το Prefab ώστε να κινούνται μόνο κατά μήκος του άξονα Y.

```
void stackCards()
{
    float pos = -0.12F;
   for(int i=0;i<49;i++)
    {
        if (ageCards[i] != null)
        {
            ageCards[i].transform.position = new Vector3(47, pos, 25);
            ageCards[i].transform.Rotate(90.0f, 00.0f, 0.0f, Space.Self);
            pos += 0.08F;
        }
```

Εικόνα 97: Τοποθέτηση καρτών στο τραπέζι. Πηγή: Αλέξανδρος Βεκίλογλου

Αντίστοιχη διαδικασία ακολουθείται και για τις κάρτες των θαυμάτων που αρχικοποιούνται από το Prefab wonderBoard, τοποθετούνται σε συγκεκριμένες θέσεις και τέλος περιστρέφονται κατάλληλα. Τέλος, μέσω της GetComponent αποκτά ο προγραμματιστής πρόσβαση στα υλικά που χρησιμοποιούν και τους αντιστοιχίζεται η εικόνα κατάλληλη εικόνα από τον πίνακα των εικόνων.

```
wonderBoard1 = Instantiate(wonderBoard, wonderBoard.transform.position + new Vector3(-30,0,15), Quaternion.Euler(new Vector3(0, -90, 0)));
wonderBoard1.GetComponent<Renderer>().material = wonderMat[0];
wonderBoard2 = Instantiate(wonderBoard, wonderBoard.transform.position + new Vector3(0, 0, 30), Quaternion.Euler(new Vector3(0, 0, 0)));
wonderBoard2.GetComponent<Renderer>().material = wonderMat[2];
wonderBoard3 = Instantiate(wonderBoard, wonderBoard.transform.position + new Vector3(30, 0, 15), Quaternion.Euler(new Vector3(0, 90, 0)));
wonderBoard3.GetComponent<Renderer>().material = wonderMat[1];
```

Εικόνα 98: Οι κάρτες των θαυμάτων. Πηγή: Αλέξανδρος Βεκίλογλου

Τα κέρματα και τα military tokens, προκειμένου να δίνεται η ψευδαίσθηση της τυχαιότητας τοποθετούνται σε τυχαίες θέσεις σε μία περιοχή, η πάνω όψη καθορίζεται τυχαία όπως και η περιστροφή κατά τον άξονα Y. Για το σκοπό αυτό χρησιμοποιείται η συνάρτηση Random.Range η οποία επιστρέφει τυχαία ένα αριθμό από το εύρος που λαμβάνει ως όρισμα. Όμοια, τα μοντέλα ανατίθενται σε πίνακες ώστε να είναι εύκολη η διαχείριση τους.

```
int[] degrees = { 90, -90 };

for (int i = 0; i < 54; i++)
{
    int degRes = degrees[Random.Range(0, 2)];
    int posX = Random.Range(64, 75);
    int posZ = Random.Range(18, 33);
    coins1[i] =  Instantiate(coin1Pre, new Vector3(posX, 1, posZ), Quaternion.Euler(new Vector3(degrees[Random.Range(0, 2)], Random.Range(0, 360), 0)));
}
```

Εικόνα 99: Δημιουργία και τοποθέτηση κερμάτων του 1. Πηγή: Αλέξανδρος Βεκίλογλου

5.3.3 Ανακάτεμα των χαρτιών

Μόλις ολοκληρωθεί η εισαγωγή των μοντέλων στη πίστα στις καθορισμένες θέσεις θα πρέπει τα χαρτιά να ανακατευτούν ώστε κάθε φορά ο κάθε παίχτης να παίρνει διαφορετικά χαρτιά. Η διαδικασία επιτυγχάνεται μέσω της συνάρτησης ShuffleCards η οποία καλείται από τη Start μόλις ολοκληρωθεί η αρχικοποίηση των αντικειμένων. Η συνάρτηση λαμβάνει ως όρισμα τον πίνακα των χαρτιών που θα ανακατέψει και αντιμεταθέτει το κάθε στοιχείο του πίνακα σε μία νέα τυχαία θέση. Στο σημείο αυτό αξίζει να επισημανθεί ότι η φυσική θέση των καρτών (αλλά και των αντικειμένων

γενικότερα) δεν σχετίζεται με τις θέσεις τους στους πίνακες που έχουν αποθηκευτεί. Πιο συγκεκριμένα οι κάρτες είναι αποθηκευμένες σε ένα πίνακα ώστε να μπορούν να ελεγχτούν και η κάθε κάρτα έχει σαν ιδιότητα τις συντεταγμένες θέσης που ορίζουν που βρίσκεται στο χώρο (μεταβλητή pickedCardPos τύπου Vector3). Συνεπώς για το ανακάτεμα των καρτών οι κάρτες αλλάζουν θέσεις στο πίνακα που βρίσκονται αλλά η πραγματική τους θέση στη στοίβα δεν διαφοροποιείται. Για το λόγο αυτό όταν η κάρτες μοιράζονται στη συνέχεια, μοιράζονται με τη σειρά που είναι αποθηκευμένες στον πίνακα οπότε η κάρτα που φεύγει για να πάει στον παίχτη από τη στοίβα δεν είναι απαραίτητα αυτή που είναι στη πάνω θέση της στοίβας.

```
void ShuffleCards(GameObject[] cardArray)
{
    for (int i = 0; i < cardArray.Length;i++)
    {
        int roundIndex = Random.Range(0, cardArray.Length+1);
        GameObject temp = cardArray[i];
        cardArray[i] = cardArray[roundIndex];
        cardArray[roundIndex] = temp;

    }
}
```

Εικόνα 100: Ανακάτεμα χαρτιών. Πηγή: Αλέξανδρος Βεκίλογλου

```
public class ageCardScript : MonoBehaviour
{
    [SerializeField] GameObject cardPrefab;
    string[] colors = { "Brown", "Grey", "Blue", "Yellow", "Red", "Green", "Purple" };
    string[] comResources = { "Clay", "Stone", "Ore", "Wood" };
    string[] rareResources = { "Glass", "Cloth", "Papyrus" };
    string[] scienceDomains = { "Writer","Maths", "Machinery" };
    string[] chainSymbols = { "Tend","Drip","Petal","Mortar","Star", "Scale","Book","Mask","Hammer","Target","oilLamp" };
    public GameObject ResourcePref;
    public string name;
    public string textureName;
    public string color;
    public string[] cost;
    public GameObject[] costObj;
    public string[] resources;
    public GameObject[] resourcesObj;
    public string[] resources2by;
    public GameObject[] resources2byObj;
    public string[] chainSymb;
    public string[] resources2byFrom;
    public string sience;
    public int coins;
    public int victPoints;
    public int miliStrength;
    public int minOfplayers;
    public bool hasToMove = false;
    public Vector3 direction;
    public int rotDegrees;
    public int rotDegrIncre;
    public int rotDegrEnd;
    public Vector3 play0Show;
    public bool moveToCenter;
    public bool isCardPicked;
    public Vector3 pickedCardPos;
    public Vector3 pickedCardPosRotate;
    public bool isUsed;
    public bool isPayed;
    public bool moneyPayed;
    public GameObject resourceCard;
    // check if card is used by any neighbour
    public bool play0HasUse;
    //public isNewCard;
    [SerializeField] Texture2D[] ageTextures;
    [SerializeField] Texture2D[] ageTexturesBack;
    [SerializeField] Texture2D[] ageTexturesFront;
```

Εικόνα 101: Ιδιότητες του αντικειμένου κάρτα. Πηγή: Αλέξανδρος Βεκίλογλου

5.3.4 Μοίρασμα των καρτών

Ο κάθε παίχτης θα πρέπει να λάβει κυκλικά από 7 κάρτες. Προκειμένου η εφαρμογή να έχει όσο το δυνατόν πιο αυξημένη την αίσθηση της αληθοφάνειας επιλέχθηκε οι κάρτες να μοιράζονται στους παίχτες όπως ακριβώς σε ένα πραγματικό παιχνίδι. Για το σκοπό αυτό έχει δημιουργηθεί αρχικά μία συνάρτηση η οποία μοιράζει τις κάρτες στους πίνακες των παιχτών.

```
void dealCards()
{

    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    Vector3 wondPos = play0.wonderCard.transform.position;
    PlayerScript play1 = players[1].GetComponent<PlayerScript>();
    Vector3 wondPos1 = play1.wonderCard.transform.position;
    PlayerScript play2 = players[2].GetComponent<PlayerScript>();
    Vector3 wondPos2 = play2.wonderCard.transform.position;
    PlayerScript play3 = players[3].GetComponent<PlayerScript>();
    Vector3 wondPos3 = play3.wonderCard.transform.position;
```

Εικόνα 102:Συνάρτηση dealCards. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση αρχικά αρχικοποιεί τις μεταβλητές που θα χρησιμοποιήσει. Πιο συγκεκριμένα ανακτά από τον πίνακα με τους παίχτες το Script του κάθε παίχτη ώστε να αποκτήσει πρόσβαση στης μεταβλητές του Script.

```
int playIndex = 0;
for (int i = 0; i < 28; i += 4)
{
    play0.ageCards[playIndex] = ageCards[i];
    ageCardScript ageScript0 = play0.ageCards[playIndex].GetComponent<ageCardScript>();
    ageScript0.hasToMove = true;
    ageScript0.direction = wondPos + new Vector3(-13, 0.02F, -1);
    ageScript0.rotDegrees = 0;
    ageScript0.rotDegrEnd = 0;

    play1.ageCards[playIndex] = ageCards[i+1];
    ageCardScript ageScript1 = play1.ageCards[playIndex].GetComponent<ageCardScript>();
    ageScript1.hasToMove = true;
    ageScript1.direction = wondPos1 + new Vector3(-1, 0.02F, 13);
    ageScript1.rotDegrees = 15;
    ageScript1.rotDegrEnd = 90;
```

Εικόνα 103: Μοίρασμα των καρτών. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως για κάθε παίχτη παίρνει μια κάρτα από τον πίνακα των καρτών και την μεταφέρει στον πίνακα καρτών του παίχτη. Αναθέτει αληθές στη μεταβλητή που δηλώνει ότι η κάρτα πρέπει να κινηθεί και εισάγει σε μεταβλητές τη θέση που θα πρέπει να καταλάβει η κάρτα, το βήμα της περιστροφής και τέλος τη γωνία την οποία θα πρέπει να έχει όταν ολοκληρωθεί η περιστροφή της. Για παράδειγμα η πρώτη κάρτα δεν χρειάζεται να περιστραφεί ενώ η 2^{η} θα πρέπει να περιστραφεί κατά 90 μοίρες και το βήμα περιστροφής θα είναι 15 μοίρες ανά κίνηση. Η θέση η οποία θα πρέπει να καταλάβει η κάθε κάρτα είναι αριστερά ακριβώς από την κάρτα του θαύματος. Μόλις ο βρόχος περάσει από όλους τους παίχτες αυξάνεται ο αύξων αριθμός των καρτών ώστε οι κάρτες να μοιράζονται κυκλικά και ανατίθεται στη μεταβλητή isDealing η τιμή true.

```
            play3.ageCards[playIndex] = ageCards[i+3];
            ageCardScript ageScript3 = play3.ageCards[playIndex].GetComponent<ageCardScript>();
            ageScript3.hasToMove = true;
            ageScript3.direction = wondPos3 + new Vector3(1, 0, -13);
            ageScript3.rotDegrees = 90;
            ageScript3.rotDegrEnd = 270;

            playIndex++;
        }

        isDealing = true;

    }
```

Εικόνα 104: Ολοκλήρωση της διανομής. Πηγή: Αλέξανδρος Βεκίλογλου

Στο σημείο αυτό ο κάθε παίχτης φέρει στον δικό του πίνακα τις 7 κάρτες που του αντιστοιχούν. Οι κάρτες όμως παραμένουν στην αρχική στοίβα καθώς δεν έχουν ακόμα μετακινηθεί χωρικά. Η μετακίνηση του πραγματοποιείται στη συνάρτηση moveCards η οποία εκτελείται από την Update.

```
void Update()
{
    if (isDealing)
        moveCards();
}
```

Εικόνα 105: Η συνάρτηση Update. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση αυτή, σε αντίθεση με την Start η οποία εκτελείται αφού το αντικείμενο έχει δημιουργηθεί, εκτελείται κάθε φορά που η εφαρμογή εμφανίζει ένα νέο καρέ στην οθόνη. Για το λόγο αυτό η συνάρτηση moveCards έχει εισαχθεί μέσα στο βρόχο ελέγχου ώστε να εκτελείται μόνο όταν η μεταβλητή isDealing έχει τεθεί σε αληθές, δηλαδή υπάρχουν αντικείμενα για μετακίνηση.

```
void moveCards()
{
    var step = 15.0F * Time.deltaTime;

    PlayerScript play0 = players[playersCount].GetComponent<PlayerScript>();
    ageCardScript ageScript0 = play0.ageCards[runThrowCards].GetComponent<ageCardScript>();
    if (ageScript0.hasToMove)
    {
        play0.ageCards[runThrowCards].transform.position = Vector3.MoveTowards(play0.ageCards[runThrowCards].transform.position, ageScript0.direction, step);
        ageScript0.rotDegrIncre += ageScript0.rotDegrees;
        if (ageScript0.rotDegrIncre <= ageScript0.rotDegrEnd)
        {
            play0.ageCards[runThrowCards].transform.Rotate(new Vector3(0, ageScript0.rotDegrees, 0));
        }
    }
```

Εικόνα 106: Συνάρτηση moveCards. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση αρχικά υπολογίζει το βήμα της μετακίνησης. Η τιμή προκύπτει πολλαπλασιάζοντας το βήμα με τη τιμή που επιστρέφει η deltaTime του αντικειμένου Time. Η τιμή αυτή αντιστοιχεί στο χρονικό διάστημα σε δευτερόλεπτα που έχει παρέλθει από την προβολή του προηγούμενου καρέ. Για παράδειγμα αν από το προηγούμενο καρέ έχουν περάσει 1ms το αντικείμενο θα μετακινηθεί κατά 15 μέτρα ενώ αν έχουν παρέλθει 3ms το αντικείμενο θα μετακινηθεί κατά 45. Με τον τρόπο αυτό επιτυγχάνεται μια ομοιόμορφη κίνηση του αντικειμένου ανεξαρτήτως του ρυθμού προβολής καρέ ανά δευτερόλεπτο, ο οποίος μπορεί να ποικίλει. Στη συνέχεια ανακτάται ο πίνακας των καρτών του χρήστη και ελέγχεται αν η συγκεκριμένη κάρτα θα πρέπει να μετακινηθεί, μεταβλητή η οποία έχει

οριστεί σε αληθές από την προηγούμενη συνάρτηση dealCards. Για το πέρασμα από τον πίνακα των καρτών έχει δημιουργηθεί και αρχικοποιηθεί σε 0 η playersCount η οποία αυξάνεται μόλις η συνάρτηση moveCards ολοκληρώσει. Με τον τρόπο αυτό κάθε φορά που εκτελείται η moveCards θα ελέγχεται και θα μετακινείται μία κάρτα μέχρι να ελεγχθούν όλες οι κάρτες. Η μέθοδος προτιμάται έναντι ενός βρόχου for μέσα στην moveCards καθώς ο βρόχος θα έπρεπε να ολοκληρωθεί ανά καρέ αυξάνοντας σημαντικά το χρόνο εκτέλεσης του καρέ, πράγμα που θα έκανε το παιχνίδι να ΄ κολλάει ΄ στο συγκεκριμένο σημείο.

Η μετακίνηση της κάρτας γίνεται μέσω της συνάρτησης MoveTowards του αντικειμένου Vector3 η οποία επιστρέφει ένα νέο διάνυσμα το οποίο περιγράφει τη θέση που θα λάβει ένα αντικείμενο αν μετακινηθεί από την αρχική του θέση προς την τελική σε απόσταση ίση με το βήμα που της δίνεται ως το τελευταίο όρισμα. Η συνάρτηση εκτελείται ανά καρέ συνεπώς το αντικείμενο μετακινείται σταδιακά από την τρέχουσα θέση στη τελική. Ταυτόχρονα το αντικείμενο περιστρέφεται με συγκεκριμένο βήμα έως ώσπου να φτάσει στο τελικό του προσανατολισμό.

```
if (play0.ageCards[runThrowCards].transform.position == ageScript0.direction)
{
    ageScript0.hasToMove = false;
    playersCount++;

}
if (playersCount == 4)
{
    runThrowCards++;
    playersCount = 0;
}
 if(runThrowCards==7)
    isDealing = false;
```

Εικόνα 107: Ολοκλήρωση της συνάρτησης. Πηγή: Αλέξανδρος Βεκίλογλου

Ο έλεγχος του αν η κάρτα έφτασε στο τελικό της προορισμό πραγματοποιείται μετά την εκτέλεση της MoveTowards, συγκρίνοντας τη τρέχουσα θέση του αντικειμένου με αυτή που έχει ανατεθεί στη μεταβλητή direction του αντικειμένου από τη συνάρτηση dealCards. Αν η συνθήκη ισχύει, η μεταβλητή της κάρτας hasToMove αλλάζει σε false ώστε στον επόμενο έλεγχο που θα πραγματοποιηθεί στο επόμενο καρέ η κάρτα του συγκεκριμένου παίχτη να μην ελεχθεί για μετακίνηση. Η μεταβλητή playersCount αυξάνεται ώστε στην επόμενη εκτέλεση να εξετασθεί η κάρτα του επόμενου παίχτη. Αν ο έλεγχος έχει περάσει και από τους 4 παίχτες θα πρέπει να αρχίσει από την αρχή. Τέλος το μοίρασμα ολοκληρώνεται όταν η runThrowCards φτάσει στην τιμή 7 που σημαίνει ότι όλοι οι παίχτες έχουν πάρει από 7 κάρτες. Τότε στη μεταβλητή isDealing ανατίθεται false ώστε η συνάρτηση να πάψει να εκτελείται.

5.3.5 Διανομή των κερμάτων

Όταν το παιχνίδι αρχικά ξεκινά ο κάθε παίχτης θα πρέπει να έχει στην κάρτα του πάνω 3 κέρματα αξίας 1. Τα κέρματα αρχικά βρίσκονται πάνω στο τραπέζι και μόλις το παιχνίδι ξεκινά τρία κέρματα μεταφέρονται στο δεξί τμήμα της κάρτας του κάθε παίχτη. Η διαδικασία επιτυγχάνεται ακολουθώντας παρόμοια προσέγγιση με τη διανομή των καρτών. Αρχικά δημιουργούνται πίνακες

ικανοί να χωρέσουν το σύνολο των κερμάτων που υπάρχουν στο παιχνίδι καθώς είναι πιθανό ένας παίχτης να κερδίσει όλα τα χρήματα. Ακολούθως για κάθε παίχτη καθορίζεται η περιοχή της κάρτας του στην οποία θα αποθέτονται τα κέρματα. Πιο συγκεκριμένα, για να βελτιώνεται η αληθοφάνεια του παιχνιδιού τα κέρματα δεν τοποθετούνται σε συγκεκριμένες θέσεις αλλά σε τυχαίες θέσεις εντός μιας συγκεκριμένης περιοχής. Το κέντρο της περιοχής αυτής καθορίζεται με τη δημιουργία του παίχτη στη συνάρτηση initPlayers.

```
void initPlayers()
{

    players[0] = Instantiate(playerPrefab);
    playerScr0 = players[0].GetComponent<PlayerScript>();
    playerScr0.wonderCard = wonderBoard;
    playerScr0.setCoinsArea(new Vector3(8, 0.08F, 0));

    players[1] = Instantiate(playerPrefab);
    playerScr1 = players[1].GetComponent<PlayerScript>();
    playerScr1.wonderCard = wonderBoard1;
    playerScr1.setCoinsArea(new Vector3(0, 0.08F, -8));

    players[2] = Instantiate(playerPrefab);
```

Εικόνα 108: Καθορισμός της περιοχής των κερμάτων. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση setCoinsArea λαμβάνει ως όρισμα ένα διάνυσμα που περιέχει τις συντεταγμένες της περιοχής σε σχέση με το κέντρο την κάρτας του θαύματος. Στο σώμα της η συνάρτηση υπολογίζει το κέντρο της περιοχής των κερμάτων προσθέτοντας στο διάνυσμα της θέσης της κάρτας το διάνυσμα που λαμβάνει ως όρισμα.

```
public void setCoinsArea(Vector3 pos)
{
    coinsArea = wonderCard.transform.position + pos ;
}
```

Εικόνα 109: Συνάρτηση setCoinsArea του αντικειμένου PlayerScript. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις το παιχνίδι ξεκινά και αφού έχουν αρχικοποιηθεί τα αντικείμενα των παιχτών και τους έχει ανατεθεί κάρτα θαύματος εκτελείται η συνάρτηση dealCoins η οποία μοιράζει τα κέρματα στους παίχτες. Πιο συγκεκριμένα η συνάρτηση διατρέχει τον πίνακα των κερμάτων και μεταφέρει τρία κέρματα για κάθε παίχτη από τον πίνακα αυτό στο πίνακα κερμάτων του 1 του κάθε παίχτη. Σε κάθε κέρμα αναθέτει τη μεταβλητή hasToMove σε αληθές ώστε το κέρμα να μετακινείται στη συνέχεια από τη συνάρτηση onUpdate, όπως ακριβώς και με τις κάρτες των παιχτών. Για κάθε κέρμα καθορίζεται και η νέα θέση που θα λάβει στην κάρτα θαύματος του κάθε παίχτη λαμβάνοντας το κέντρο της περιοχής από τη μεταβλητή coinsArea που διαθέτει ο κάθε παίχτης και στη συνέχεια αλλάζει τυχαία κατά ένα μέτρο τις αντίστοιχες συντεταγμένες. Οι συντεταγμένες που αλλάζουν είναι

η x και z (αριστερά – δεξιά, μπρος – πίσω) και όχι η y η οποία αντιστοιχεί στο ύψος του κέρματος καθώς αυτό παραμένει σταθερά πάνω στο τραπέζι.

```
void dealCoins()
{
    int[] coords = { -1, 0, 1 };
    int coinCount = 0;
    for (int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 3; j++) {
                Coin1Script coin1S = coins1[coinCount].GetComponent<Coin1Script>();
                coin1S.hasToMove = true;
                PlayerScript play0 = players[i].GetComponent<PlayerScript>();
                float xCord  = play0.coinsArea.x;
                float yCord = play0.coinsArea.y;
                float zCord = play0.coinsArea.z;
                coin1S.direction =   new Vector3(xCord + coords[Random.Range(0,3)], yCord, zCord + coords[Random.Range(0, 3)]);
                play0.coins1[j] = coins1[coinCount];
                coinCount++;
        }
    }
}
```

Εικόνα 110: Συνάρτηση dealCoins. Πηγή: Αλέξανδρος Βεκίλογλου

Τέλος η χωρική μετακίνηση των κερμάτων γίνεται στη συνάρτηση moveCoins η οποία εκτελείται από την Update ανά καρέ. Σε κάθε εκτέλεση ελέγχονται 3 κέρματα από τον πίνακα των κερμάτων και αν πρέπει να μετακινηθούν μετακινούνται στη νέα τους θέση με τη χρήση της συνάρτησης MoveTowards. Για τα κέρματα προτιμήθηκε να κινούνται ανά 3 ώστε να φτάνουν πιο γρήγορα στον κάθε παίχτη. Όταν φτάσουν στη τελική τους θέση η μεταβλητή hastToMove ανατίθεται σε ψευδές οπότε και παύει η μετακίνηση τους.

```
void moveCoins()
{
    var step = 15.0F * Time.deltaTime;

    Coin1Script coin1S = coins1[coinsCount].GetComponent<Coin1Script>();
    if (coin1S.hasToMove)
    {
      coins1[coinsCount].transform.position = Vector3.MoveTowards(coins1[coinsCount].transform.position, coin1S.direction, step);
    }
    if (coins1[coinsCount].transform.position == coin1S.direction)
    {
      coin1S.hasToMove = false;
    }

    Coin1Script coin1aS = coins1[coinsCount+1].GetComponent<Coin1Script>();
    if (coin1aS.hasToMove)
    {
      coins1[coinsCount + 1].transform.position = Vector3.MoveTowards(coins1[coinsCount + 1].transform.position, coin1aS.direction, step);
    }
    if (coins1[coinsCount + 1].transform.position == coin1aS.direction)
    {
       coin1aS.hasToMove = false;
    }
```

Εικόνα 111: Τμήμα συνάρτησης moveCoins. Πηγή: Αλέξανδρος Βεκίλογλου

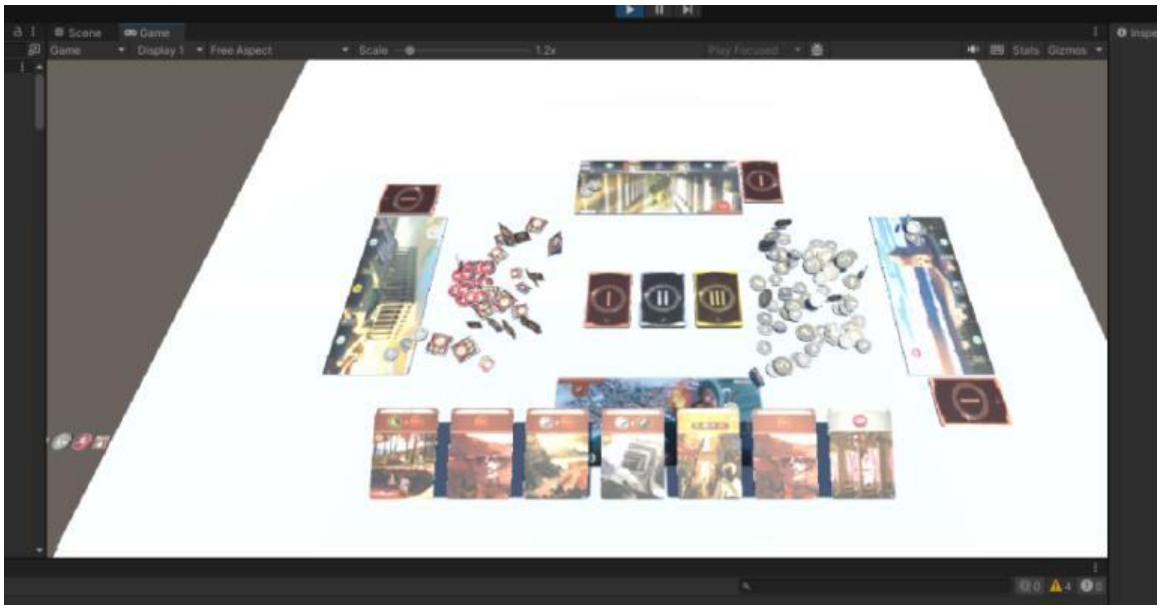
5.3.6 Μετακίνηση Military Tokens

Η μετακίνηση των military Tokens γίνεται με τον ίδιο τρόπο που μετακινούνται τα νομίσματα. Έτσι έχουν οριστεί οι περιοχές πάνω στη κάρτα θαύματος του κάθε παίχτη και κάθε φορά που θα πρέπει να μετακινηθούν tokens ενημερώνεται η αντίστοιχη μεταβλητή που δηλώνει ότι το token θα πρέπει να μετακινηθεί ενώ παράλληλα αντιστοιχίζεται στις μεταβλητές που περιέχουν τις συντεταγμένες προορισμού η θέση η οποία θα πρέπει να καταλάβουν. Η θέση αυτή είναι η περιοχή θαύματος του παίχτη που θα αποκτήσει το token. Όμοια η τελική θέση μέσα στη περιοχή, όπως και ο προσανατολισμός του token καθορίζεται τυχαία ώστε να ενισχύεται η αληθοφάνεια του παιχνιδιού. Τέλος, μέσα στην Update ελέγχεται κάθε φορά μια επόμενη θέση του πίνακα που περιέχει τα tokens αν υπάρχουν τέτοια για μετακίνηση και αν ναι τότε αυτά μετακινούνται κατά μία θέση. Ακολούθως

στη νέα κλήση της Update μετακινούνται άλλη μια θέση ώσπου να φτάσουν στο τελικό προορισμό τους.

5.4 Εκτέλεση παιχνιδιού

5.4.1 Παρουσίαση καρτών στον παίχτη



Εικόνα 112: Παρουσίαση των καρτών του παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

5.4.1.1 Άπλωμα καρτών

Μόλις ο κάθε παίχτης λάβει τις κάρτες που του αντιστοιχούν το παιχνίδι ξεκινά. Ο παίχτης θα πρέπει να δει τις κάρτες που έχει πάρει και στη συνέχεια να αποφασίσει ποια εξ αυτών θα παίξει. Αντίστοιχα θα πρέπει να πράξουν και οι υπόλοιποι παίχτες με αυτοματοποιημένο τρόπο όμως καθώς οι αποφάσεις για τους παίχτες αυτούς λαμβάνονται σύμφωνα με τον κώδικα που έχει γραφεί. Πρώτο βήμα για να αποφασίσει ο παίχτης ποια κάρτα θα παίξει αποτελεί η προβολή των καρτών στο παίχτη. Πιο συγκεκριμένα οι κάρτες βρίσκονται σε μια στοίβα στην αριστερή πλευρά της κάρτας θαύματος. Η εφαρμογή αναλαμβάνει να τις περιστρέψει και να τις απλώσει μπροστά του ώστε εκείνος να μπορεί να τις δει. Η διαδικασία εκκινείται από το Script του terrain μόλις ολοκληρωθεί το μοίρασμα των καρτών.

Η διαδικασία μοιράσματος των καρτών διαρκεί μερικά δευτερόλεπτα συνεπώς η προβολή των καρτών στον παίχτη θα πρέπει να ξεκινήσει αφού η μοιρασιά ολοκληρωθεί. Για το σκοπό αυτό η κλήση της συνάρτησης showCards2Player0, η οποία διεκπεραιώνει τη διαδικασία, δεν μπορεί να γίνει μετά την συνάρτηση που μοιράζει τις κάρτες καθώς αυτό θα προκαλούσε την κλήση της προτού η προηγούμενη ολοκληρώσει. Η Unity διαθέτει ένα μηχανισμό για να αντιμετωπίζει τα θέματα αυτά και αυτό επιτυγχάνεται μέσω της συνάρτησης Invoke. Η συνάρτηση αυτή λαμβάνει ως όρισμα το όνομα της συνάρτησης που θα καλέσει και ακολούθως το χρονικό διάστημα που θα περιμένει πριν τη κλήση της συνάρτησης αρχίζοντας από την έναρξη της εκτέλεσης της εφαρμογής.

```
    createFirstAgeCards();
    assignResourc2Cards();
    ShuffleCards(ageCards);
    stackCards();
    createSecAgeCards();
    createThirdAgeCards();
    coinInit();

    initPlayers();
    dealCards();

    dealCoins();

    Invoke("showCards2Player0", 22);
    Invoke("othersPick", 23);


    gameTurn = 0;

}
```

Εικόνα 113: Συναρτήσεις μέσα στην Start του Script του Terrain. Πηγή: Αλέξανδρος Βεκίλογλου

Με τον τρόπο αυτό ορίζεται ότι η συνάρτηση showCards2Player0 που θα δείξει τις κάρτες στον παίχτη θα εκτελεστεί 22 δευτερόλεπτα μετά από την έναρξη του παιχνιδιού. Η συνάρτηση που θα διαλέξει τις κάρτες που θα διαλέξουν οι υπόλοιποι παίκτες θα κληθεί 23 δευτερόλεπτα μετά την έναρξη της εφαρμογής, ώστε όταν ο βασικός παίχτης παίξει οι άλλοι να έχουν παίξει, ώστε να παίξουν σχετικά όλοι μαζί.

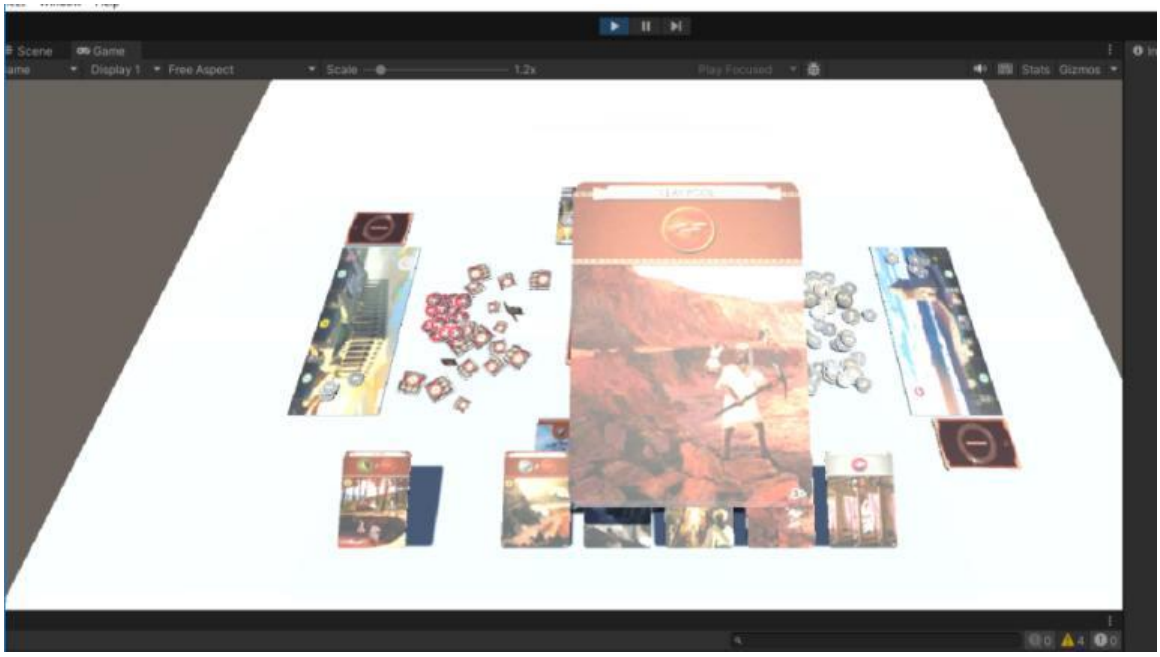
```
void showCards2Player0()
{
    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    int pos = 16;
    for (int i = 0; i < 7; i++)
    {
        if (play0.ageCards[i] != null)
        {
            // prevent from falling
            Rigidbody rigidbody = play0.ageCards[i].GetComponent<Rigidbody>();
            rigidbody.useGravity = false;
            // print("Card size: " + play0.ageCards[i].GetComponent<Renderer>().bounds.size);

            play0.ageCards[i].transform.position = play0.wonderCard.transform.position + new Vector3(pos, 8, -5);
            // store this position to the card show it can return back.
            ageCardScript ageCardsc = play0.ageCards[i].GetComponent<ageCardScript>();
            ageCardsc.play0Show = play0.wonderCard.transform.position + new Vector3(pos, 8, -5);
            pos -= 6;
            play0.ageCards[i].transform.Rotate(new Vector3(-200, 0, 0));
        }
        //else
        //    print("play0.ageCards[i]: " + i + "is null");
    }
    showCards2Player0Flag = true;
}
```

Εικόνα 114: Συνάρτηση ShowCards2Player0. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση αρχικά λαμβάνει μια αναφορά στο Script του πρώτου παίχτη, από τον πίνακα των παιχτών, όπου ο πρώτος παίχτης είναι ο βασικός παίχτης του παιχνιδιού. Αυτός είναι ο πάγιος τρόπος στη Unity για να αποκτήσει ο προγραμματιστής πρόσβαση στις μεταβλητές που έχουν ανατεθεί στο συγκεκριμένο αντικείμενο. Πιο συγκεκριμένα όταν ο προγραμματιστής επιθυμεί να αναθέσει μεταβλητές σε ένα αντικείμενο, για παράδειγμα το όνομα στη κάρτα, θα δημιουργήσει ένα Script και θα το αναθέσει σαν ιδιότητα στο αντικείμενο μέσω του κουμπιού Add Component του συγκεκριμένου GameObject. Στη συνέχεια θα δημιουργήσει σε αυτό τις μεταβλητές που χρειάζεται. Ακολούθως μπορεί να αποκτά μια αναφορά στο Script αυτό και από εκεί να αποκτά πρόσβαση στις μεταβλητές αυτές. Η συνάρτηση, έχοντας πλέον πρόσβαση στις μεταβλητές του Script του χρήστη, έχει πρόσβαση και στις κάρτες που ο χρήστης έχει καθώς αυτές (αναφορές αυτών) βρίσκονται αποθηκευμένες σε ένα πίνακα στο Script του παίχτη. Στη συνέχεια αρχικοποιείται η μεταβλητή pos η οποία αντιστοιχεί στην απόσταση που θα πρέπει να έχει η μία κάρτα από την άλλη καθώς αυτές τοποθετούνται η μία δίπλα στην άλλη. Ακολούθως ξεκινά ένας βρόχος επανάληψης ο οποίος θα εκτελεστεί 7 φορές όσος και ο μέγιστος αριθμός καρτών που ο παίχτης μπορεί να έχει στη κατοχή του. Οι κάρτες καθώς απλώνονται μπροστά στο παίχτη δεν βρίσκονται πλέον πάνω στο τραπέζι αλλά ελαφρώς ψηλότερα ώστε να είναι πιο ευδιάκριτες στο παίχτη. Λόγο της βαρύτητας όμως που ασκείται πάνω τους οι κάρτες πέφτουν πάνω στο τραπέζι. Για το σκοπό αυτό και για όσο αυτές βρίσκονται στη κατάσταση αυτή απενεργοποιείται η βαρύτητα ώστε να μένουν στη θέση τους. Για να γίνει αυτό θα πρέπει να αποκτηθεί μια αναφορά στο αντικείμενο – ιδιότητα που υλοποιεί τη βαρύτητα. Η ιδιότητα αυτή είναι η Rigidbody και η αναφορά στη ιδιότητα λαμβάνεται μέσω της κλήσης GetComponent. Όλα τα αντικείμενα της Unity διαθέτουν τη συνάρτηση αυτή η οποία σαν τύπο επιστροφής λαμβάνει το όνομα της ιδιότητας του αντικειμένου που θα επιστρέψει. Θέτοντας την ιδιότητα του Rigidbody με όνομα useGravity σε ψευδές η βαρύτητα παύει να εφαρμόζεται στη κάρτα.

Στη συνέχεια ενημερώνεται η τιμή της position της ιδιότητας transform της κάρτας με ένα διάνυσμα που περιέχει τις συντεταγμένες της θέσης στην οποία η κάρτα θα πρέπει να μεταφερθεί. Αυτός είναι ένας τρόπος για να μεταφερθεί ένα αντικείμενο στη Unity. Διαφέρει από τον τρόπου που παρουσιάστηκε νωρίτερα (MoveTowards) η οποία συνάρτηση αναλαμβάνει να μετακινήσει ένα αντικείμενο σταδιακά προς μια κατεύθυνση. Τροποποιώντας τη μεταβλητή position του αντικειμένου αυτό έχει σαν αποτέλεσμα την ακαριαία μετακίνηση του στη νέα θέση. Η νέα θέση προκύπτει από τις συντεταγμένες κέντρου της κάρτας θαύματος, τροποποιημένες κατάλληλα. Πιο συγκεκριμένα η πρώτη κάρτα θα τοποθετηθεί στις συντεταγμένες 16, 8, -5. Η πρώτη συντεταγμένη αντιστοιχεί στον άξονα X ο οποίος αντιστοιχεί στο αριστερά – δεξιά, η επόμενη στον Y ο οποίος αντιστοιχεί στο πάνω – κάτω και τέλος ο Z ο οποίος αντιστοιχεί στο μακριά – κοντά από την κάμερα. Οι τιμές Y και Z είναι σταθερές σε όλες τις κάρτες, όλες θα τοποθετηθούν λίγο πιο ψηλά από την αρχική τους θέση και λίγο πιο κοντά στη κάμερα. Η μεταβλητή X αυξάνεται κατά 16 μονάδες ώστε η κάθε κάρτα να τοποθετείται 16 μέτρα πιο δεξιά από την προηγούμενη.

Ο παίχτης θα επιλέξει από τις κάρτες αυτές μία για να παίξει. Η προσέγγιση που επιλέχθηκε για τη διαδικασία αυτή είναι όταν ο χρήστης τοποθετεί τον κέρσορα του ποντικιού πάνω από μία κάρτα αυτή να μεταφέρεται σε νέα θέση ώστε να ξεχωρίζει από τις άλλες και να είναι πιο ευδιάκριτες οι λεπτομέρειες της. Αν ο χρήστης επιθυμεί να την παίξει κάνει κλικ πάνω της. Αν όμως αποφασίσει ότι αυτή δεν του κάνει να μεταφέρει τον κέρσορα σε νέα θέση και η κάρτα αυτή να επιστρέφει στη θέση που είχε πριν ο χρήστης τοποθετήσει τον κέρσορα πάνω της και η διαδικασία επαναλαμβάνεται εκ νέου.



Εικόνα 115: Προβολή της κάρτας στην οποία ο χρήστης τοποθετεί τον κέρσορα. Πηγή: Αλέξανδρος Βεκίλογλου

Η αρχική θέση της κάρτας, όταν αυτή μεταφέρεται, παραμένει κενή ώστε κατά την επαναφορά της να επιστρέφει στην αρχική της θέση. Για να επιτευχθεί αυτό η αρχική αυτή θέση αποθηκεύεται σε μια μεταβλητή με όνομα play0Show στο Script της κάρτας. Στη συνέχεια η συνάρτηση ανανεώνει τη μεταβλητή pos η οποία αντιστοιχεί στη θέση στον άξονα X που θα πάρει η επόμενη κάρτα στη σειρά και περιστρέφει την κάρτα ώστε να εμφανίζεται με την άλλη όψη πάνω και να έχει και μια ελαφριά κλήση. Η μεταβλητή showCards2Player0Flag λαμβάνει την τιμή αληθές ώστε να ελέγχει η εφαρμογή αν ο χρήστης τοποθέτησε τον κέρσορα πάνω σε κάρτα ώστε να εκκινείται η διαδικασία μεταφοράς της στο κέντρο της οθόνης.

5.4.1.2 Μετακίνηση της κάρτας στο κέντρο

Για να μεταφερθεί η κάρτα στο κέντρο θα πρέπει ο χρήστης να τοποθετήσει τον κέρσορα πάνω στη κάρτα αυτή. Συνεπώς, στο Script της κάρτας, που ονομάζεται AgeCardScript εισάγεται η συνάρτηση onMouseOver η οποία θα εκτελείται από τη Unity κάθε φορά που ο χρήστης τοποθετεί τον κέρσορα πάνω από το συγκεκριμένο αντικείμενο. Εντός της συνάρτησης στη μεταβλητή moveToCenter ανατίθεται η τιμή αληθές, ενώ αρχικά η τιμή της ήταν ψευδές.

```
void OnMouseOver()
{
    moveToCenter = true;
}
```

Εικόνα 116: Συνάρτηση OnMouseOver του αντικειμένου AgeCardScript. Πηγή: Αλέξανδρος Βεκίλογλου

Η μεταφορά της κάρτας στο κέντρο γίνεται από την Update του TerrainScript. Εκεί, σε ένα βρόχο που θα εκτελεστεί 7 φορές, όσες και οι κάρτες που μπορεί να έχει ο παίχτης, διατρέχεται ο πίνακας που περιέχει τις κάρτες του παίχτη. Σε μια προσπάθεια εξοικονόμησης υπολογιστικών πόρων η αναζήτηση πραγματοποιείται μόνο όταν η μεταβλητή showCards2Player0Flag έχει την τιμή true πράγμα που σημαίνει ότι οι κάρτες έχουν απλωθεί μπροστά στο παίχτη και αυτός μπορεί να τοποθετήσει τον κέρσορα πάνω σε αυτές. Για το λόγο αυτό ο κώδικας μετακίνησης της κάρτας εκτελείται μετά τον έλεγχο της συγκεκριμένης μεταβλητής. Ο βρόχος εκτελείται 7 φορές καθώς

τόσες είναι οι θέσεις του πίνακα που περιέχει τις αναφορές για τις κάρτες του παίχτη. Στη πραγματικότητα όμως ο πίνακας θα περιέχει 7 κάρτες μόνο στο πρώτο γύρο. Σε κάθε επόμενο θα περιέχει μια κάρτα λιγότερη καθώς ο παίχτης θα έχει παίξει μια κάρτα. Συνεπώς κρίνεται σκόπιμο, προτού ληφθεί η αναφορά της κάρτας στη συγκεκριμένη θέση του πίνακα να ελεχθεί αν στη θέση αυτή υπάρχει κάρτα ή υπάρχει η τιμή null. Όταν ο παίχτης παίζει μια κάρτα αυτή αφαιρείται από τον πίνακα και στη θέση της ανατίθεται η τιμή null συνεπώς είναι εφικτός ο έλεγχος. Η προσέγγιση αυτή κρίνεται απαραίτητη καθώς σε διαφορετική περίπτωση η εφαρμογή δεν θα μπορούσε να λειτουργήσει αν ο προγραμματιστής προσπαθούσε να ανακτήσει μια κάρτα από μια κενή θέση.

Αν η θέση αυτή δεν περιέχει την τιμή null τότε ανακτάται το Script της συγκεκριμένης κάρτας με όνομα AgeCardScript και από αυτό ελέγχεται αν ο χρήστης έχει τοποθετήσει τον κέρσορα πάνω της, δηλαδή αν έχει κληθεί η OnMouseOver και η μεταβλητή moveToCenter έχει την τιμή αληθές. Αν η συνθήκη ισχύει τότε η κάρτα θα πρέπει να μεταφερθεί στο μέσο της οθόνης. Η τιμή moveToCenter αλλάζει σε ψευδές, ώστε η διαδικασία να μπορεί να εκτελεστεί εκ νέου και στην ιδιότητα position του αντικειμένου transform ανατίθενται οι συντεταγμένες της νέας θέσης που θα πρέπει να λάβει η κάρτα.

Στη συνέχεια ελέγχεται αν στο κέντρο υπάρχει ήδη κάρτα από προηγούμενη προβολή. Ο έλεγχος πραγματοποιείται μέσω της μεταβλητής ageCardsc2Center στην οποία κάθε φορά που μεταφέρεται κάρτα στο κέντρο της οθόνης ανατίθεται η κάρτα στη τιμή αυτή. Με τον τρόπο αυτό είναι εφικτός ο έλεγχος του αν υπάρχει κάρτα στο κέντρο της οθόνης από προηγούμενη εκτέλεση καθώς και πια κάρτα υπάρχει, ώστε να μην μεταφέρεται η ίδια. Συνεπώς αν δεν υπάρχει κάρτα στο κέντρο τότε στη μεταβλητή αυτή ανατίθεται η συγκεκριμένη κάρτα. Σε αντίθετη περίπτωση η κάρτα που βρίσκεται στο κέντρο επιστρέφει στη αρχική της θέση η οποία είναι αποθηκευμένη στη μεταβλητή play0Show και στη μεταβλητή ageCardsc2Center ανατίθεται η συγκεκριμένη κάρτα.

```
if (showCards2Player0Flag)
{
    // move card to center
    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    for (int i = 0; i < 7; i++)
    {
        if (play0.ageCards[i] != null) {
            ageCardScript ageCardsc = play0.ageCards[i].GetComponent<ageCardScript>();
            if (ageCardsc.moveToCenter)
            {
                ageCardsc.moveToCenter = false;
                play0.ageCards[i].transform.position = new Vector3(53.5F, 38, 3);
                if (ageCardsc2Center == null)
                {
                    ageCardsc2Center = ageCardsc;
                }
                else if (ageCardsc2Center != ageCardsc)
                {
                    ageCardsc2Center.transform.position = ageCardsc2Center.play0Show;
                    ageCardsc2Center = ageCardsc;
                }
            }

        }

    }
```

Εικόνα 117: Μετακίνηση της κάρτας στο κέντρο της οθόνης. Πηγή: Αλέξανδρος Βεκίλογλου

5.4.1.3 Κλικ πάνω στη κάρτα

Αντίστοιχα με την μετακίνηση στο κέντρο εκτελείται και η διαδικασία της επιλογής της κάρτας. Μέσα στο Script της κάρτας υπάρχει η συνάρτηση OnMouseDown η οποία επίσης εκτελείται από τη Unity όταν ο χρήστης κάνει κλικ πάνω στη κάρτα.

```
private void OnMouseDown()
{
    isCardPicked = true;
}
```

Εικόνα 118: Συνάρτηση OnMouseDown του Script ageCardScript. Πηγή: Αλέξανδρος Βεκίλογλου

Τη μεταβλητή αυτή την παρακολουθεί το Script που έχει ανατεθεί στο terrain. Πιο συγκεκριμένα, σε συνέχεια του κώδικα της προηγούμενης παραγράφου ελέγχεται αν υπάρχει κάρτα στο κέντρο της οθόνης μέσω της μεταβλητής ageCardsc2Center. Αν υπάρχει κάρτα ελέγχεται η μεταβλητή isCardPicked που ενεργοποιεί η OnMouseDown και αν ναι τότε ξεκινά η διαδικασία επαναφοράς των υπολοίπων απλωμένων καρτών πίσω στη στοίβα και η κάρτα που έχει γίνει κλικ ανατίθεται στον παίχτη για μελλοντική χρήση. Πιο συγκεκριμένα για να μεταφερθούν οι υπόλοιπες κάρτες στη στοίβα διατρέχεται ο πίνακας με τις κάρτες του παίχτη από ένα βρόχο επανάληψης. Η κάθε θέση ελέγχεται για το εάν περιέχει κάρτα ή της έχει ανατεθεί η τιμή null και εφόσον υπάρχει κάρτα ανακτάται μια αναφορά στο Script της. Θα πρέπει να μετακινηθούν όλες οι κάρτες του πίνακα εκτός από αυτή που βρίσκεται στη μέση της οθόνης και στη μεταβλητή ageCardsc2Center. Συνεπώς ελέγχεται το κάθε Script να είναι διαφορετικό από αυτό της κεντρικής κάρτας και αν είναι ενημερώνεται η μεταβλητή position με τις συντεταγμένες της στοίβας του παίχτη, η κάρτα περιστρέφεται ώστε να είναι με την

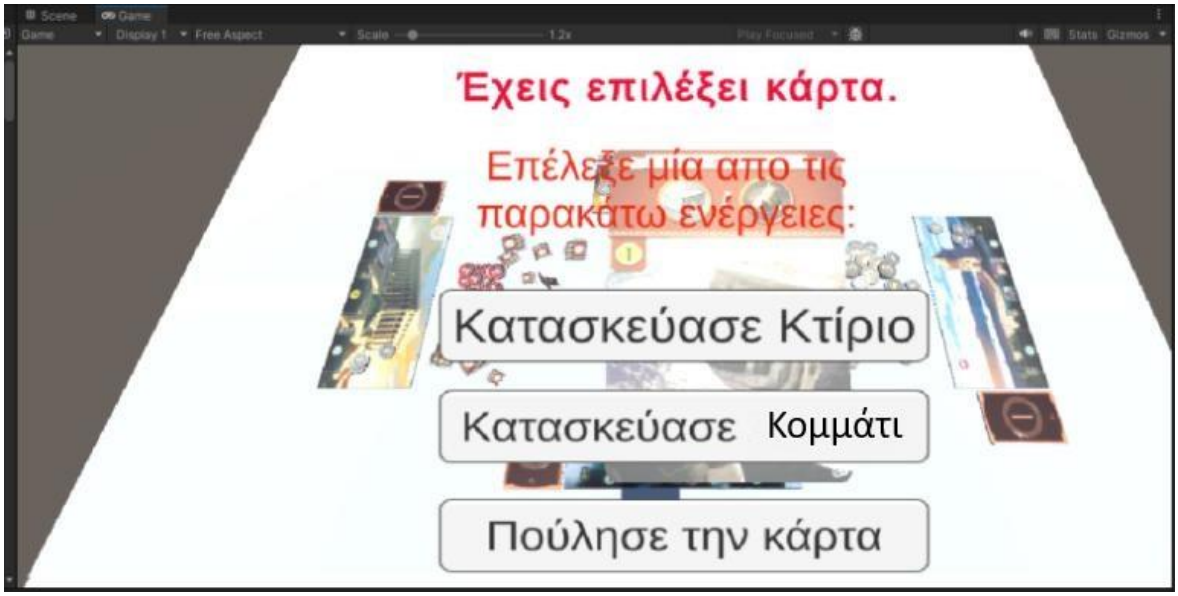
πίσω όψη κάτω και τέλος ενεργοποιείται και η βαρύτητα για την κάρτα αυτή. Η μεταβλητή showCards2Player0Flag τίθεται σε ψευδές ώστε ο κώδικας αυτός να μην εκτελείται στις επόμενες κλήσεις της Update καθώς ο παίχτης έχει επιλέξει κάρτα για να παίξει και το Script της κάρτας ανατίθεται σε μια μεταβλητή στο Script του παίχτη για μελλοντική αναφορά. Η διαδικασία αυτή εκτελείται καλώντας τη συνάρτηση getThis() του συγκεκριμένου Script η οποία θα επιστρέψει το Script. Η διαδικασία ολοκληρώνεται με την κλήση της συνάρτησης showActionMessage η οποία θα εμφανίσει το παράθυρο επιλογών στο χρήστη.

```
// center card is clicked
if (ageCardsc2Center != null)
{
    if (ageCardsc2Center.isCardPicked)
    {
        // move all cards back to stack
        for (int i = 0; i < 7; i++)
        {
            if (play0.ageCards[i] != null)
            {
                ageCardScript ageCardsc = play0.ageCards[i].GetComponent<ageCardScript>();
                if (ageCardsc2Center != ageCardsc)
                {
                    play0.ageCards[i].transform.position = play0.direction;
                    play0.ageCards[i].transform.Rotate(new Vector3(200, 0, 0));
                    Rigidbody rigidbody = play0.ageCards[i].GetComponent<Rigidbody>();
                    rigidbody.useGravity = true;
                }
            }
        }

        showCards2Player0Flag = false;
        // place pickedCard to the table and store it at players variable
        play0.pickedCard = ageCardsc2Center.getThis();
       // print("Player that picked this card name: " + play0.name);
       showActionMessage();
    }
}
```

Εικόνα 119: Επιλογή κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

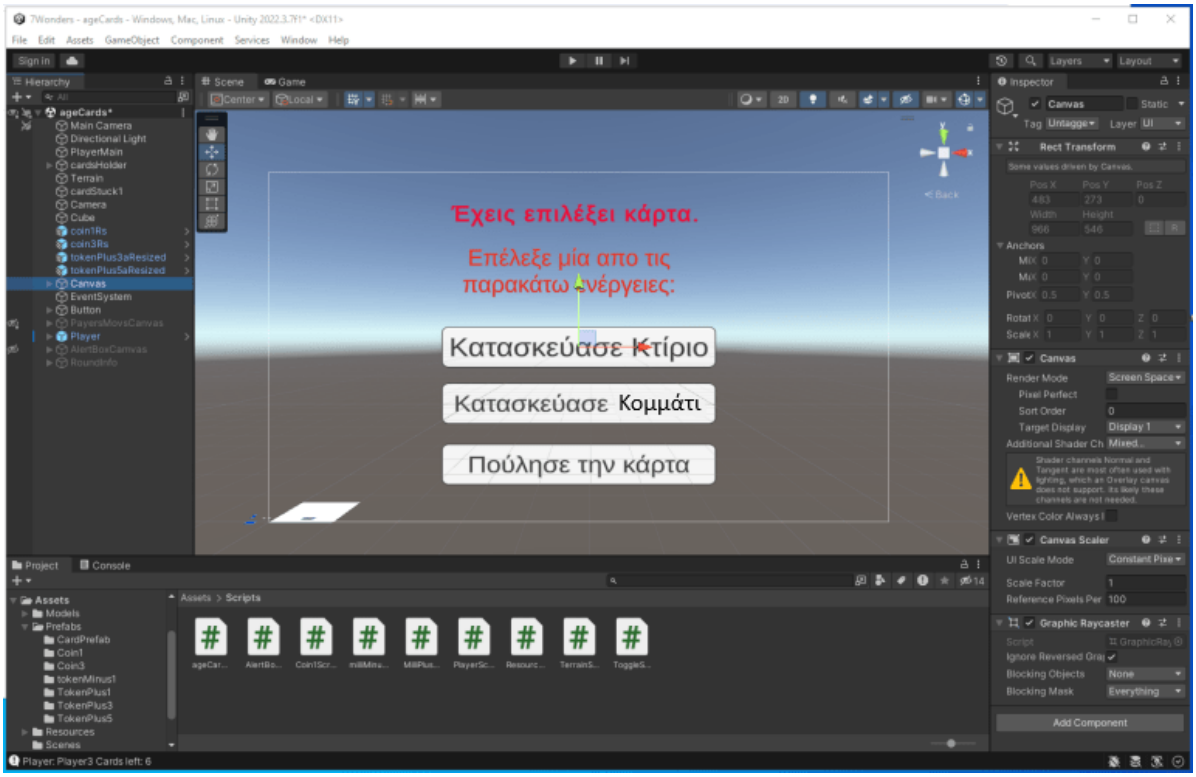
5.4.2 Το παράθυρο επιλογών

Ο παίχτης μόλις επιλέξει την κάρτα θα πρέπει να αποφασίσει τι θα κάνει με αυτή. Υπάρχουν τρεις διαθέσιμες επιλογές να επιλέξει και αυτές είναι να δημιουργήσει τη κατασκευή της κάρτας, να κατασκευάσει ένα μέρος του θαύματος ή να πουλήσει την κάρτα. Προκειμένου να επιλέξει μια από τις τρεις αποφασίσθηκε να δημιουργηθεί ένα παράθυρο επιλογών που θα εμφανίζει ένα κείμενο που θα εξηγεί στον παίχτη τι συμβαίνει και τι θα πρέπει να κάνει. Μετά το κείμενο θα υπάρχουν οι τρεις επιλογές ώστε ο παίχτης να πατήσει το κουμπί της επιλογής που επιθυμεί. Το παράθυρο είναι διάφανο ώστε να μπορεί από πίσω να βλέπει την κάρτα που έχει επιλέξει καθώς και τους υπόλοιπους συν παίχτες του.



Εικόνα 120: Το παράθυρο με τις διαθέσιμες επιλογές μετά την επιλογή κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

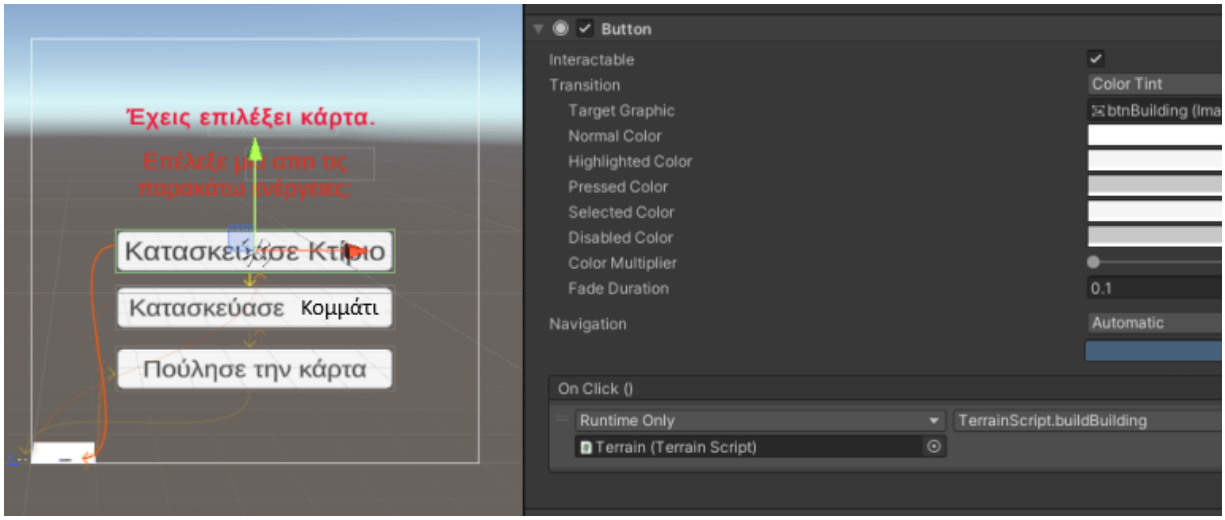
Δημιουργία του παραθύρου



Εικόνα 121: Δημιουργία παραθύρου. Πηγή: Αλέξανδρος Βεκίλογλου

Το παράθυρο αποτελεί και αυτό ένα αντικείμενο τύπου GameObject και πιο συγκεκριμένα ένα αντικείμενο Canvas. Το αντικείμενο αυτό θα το δημιουργήσει ο χρήστης από το μενού GameObject και ακολούθως UI και στη συνέχεια Canvas. Ο καμβάς αυτός αποτελεί ένα αντικείμενο που εμφανίζεται στο ψηφιακό κόσμο σε μια θέση που δεν σχετίζεται με τον τρόπο που η κάμερα βλέπει τον κόσμο και η Unity αναλαμβάνει να τον προβάλει μπροστά στη κάμερα. Συνεπώς δεν έχει νόημα να τον μετακινήσει ο χρήστης σε συγκεκριμένο σημείο. Φυσικά υπάρχουν και άλλες επιλογές οι οποίες καθορίζονται από την επιλογή Render Mode αλλά η συγκεκριμένη είναι η κατάλληλη για την περίπτωση αυτή.

Αφού ο χρήστης δημιουργήσει τον καμβά μπορεί να προσθέσει αντικείμενα σε αυτόν όπως background το οποίο αναπαρίσταται από ένα αντικείμενο image, τα πεδία κειμένου, τα οποία αποτελούν αντικείμενα τύπου Text, καθώς και κουμπιά. Τα αντικείμενα αυτά θα δημιουργηθούν επιλέγοντας τον καμβά και ακολούθως με δεξί κλικ πάνω του επιλέγεται από την επιλογή UI το Image, Text ή το Button. Με τον τρόπο αυτό τα αντικείμενα δημιουργούνται εντός του αντικειμένου καμβά με αποτέλεσμα να υπάρχει μια καλύτερη οργάνωση των αντικειμένων αφού το αντικείμενο λειτουργεί ως φάκελος που ανοίγει και κλείνει και περικλείει στο εσωτερικό του τα δικά του αντικείμενα. Ο συγκεκριμένος καμβάς θα πρέπει να είναι διάφανος και συνεπώς το image το οποίο θα αποτελεί το background δεν απαιτείται. Αφού δημιουργηθούν τα αντικείμενα μπορούν να μεταφερθούν σαν αναφορές και να αντιστοιχηθούν σε μεταβλητές άλλων Scripts ώστε να μπορούν να αλλαχθούν οι ιδιότητες τους όπως το κείμενο των πεδίων, προγραμματιστικά μέσω του Script. Τα κουμπιά για να εκτελούν κάποια συνάρτηση κατά το πάτημα τους θα πρέπει να τους ανατεθεί το Script που περιέχει τη συνάρτηση και ακολούθως από το drop down να επιλεγεί η κατάλληλη συνάρτηση που περιλαμβάνεται στο Script. Πιο συγκεκριμένα το κουμπί κατασκευής κτιρίου θα εκτελεί τη συνάρτηση buildBuilding του Script που έχει ανατεθεί στο terrain.



Εικόνα 122: Κλήση συνάρτησης από το κουμπί. Πηγή: Αλέξανδρος Βεκίλογλου

5.4.3 Χτίσιμο κατασκευής

Η συνάρτηση που εκτελείται με το πάτημα του κουμπιού ‘Κατασκεύασε Κτίριο ‘ αποτελεί μια από τις πιο σύνθετες της εφαρμογής και ο κώδικας της αγγίζει τις 100 γραμμές. Ο λόγος για τον οποίο συμβαίνει αυτό είναι διότι ο παίχτης για να κατασκευάσει την κάρτα θα πρέπει να διευκρινιστεί το κατά πόσο διαθέτει τους απαιτούμενους πόρους.

```
298     public void buildBuilding()
299     {
300
301         camvas.SetActive(false);
302         PlayerScript play0 = players[0].GetComponent<PlayerScript>();
303
304         print("Player: " + play0.name + "picked the card: " + play0.pickedCard.GetComponent<ageCardScript>().name);
305         ageCardScript play0AgeCardScript = play0.pickedCard.GetComponent<ageCardScript>();
306         bool hasCost = false;
307         for (int i = 0; i < 10; i++)
308         {
309             if ((play0AgeCardScript.costObj[i] != null) || (play0AgeCardScript.cost[i] != null))
310             {
311                 hasCost = true;
312                 break;
313             }
314         }
315         if (hasCost)
```

Εικόνα 123: Τμήμα της συνάρτησης buildBuilding. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση με το που ξεκινά αρχικά κρύβει το παράθυρο διαλόγου που εμφανιζόταν στην οθόνη από όπου ο παίχτης επέλεξε να εκτελέσει τη συγκεκριμένη ενέργεια. Η διαδικασία επιτυγχάνεται με τη κλήση της συνάρτησης SetActive του συγκεκριμένου καμβά και σαν όρισμα του δίνεται η τιμή ψευδές ώστε ο καμβάς να απενεργοποιηθεί και να μην φαίνεται πλέον. Ακολούθως η συνάρτηση θα πρέπει να αποκτήσει πρόσβαση στις μεταβλητές του παίχτη που είναι αποθηκευμένες στο Script του και αυτό το πετυχαίνει καλώντας και αυτό την GetComponent του πρώτου παίχτη που βρίσκεται στον πίνακα παιχτών, ζητώντας της να επιστρέψει το συστατικό PlayerScript. Το PlayerScript περιέχει, μεταξύ άλλων, μία μεταβλητή με όνομα pickedCard στην οποία έχει αποθηκευτεί η κάρτα η οποία επέλεξε ο παίχτης. Από τη μεταβλητή αυτή ανακτάται μια αναφορά στο Script της κάρτας ώστε να αποκτηθεί η πρόσβαση στις τιμές της κάρτας.

Η κάθε κάρτα μπορεί να έχει κάποιο κόστος. Το κόστος αυτό μπορεί να είναι ένα η περισσότερα νομίσματα ή και ένας ή περισσότεροι πόροι ή και τίποτα. Συνεπώς στο Script των καρτών έχει δημιουργηθεί ένας πίνακας με όνομα costObj στον οποίο ανατίθεται το κόστος της κάρτας. Ο πίνακας αυτός δέχεται αντικείμενα τύπου GameObject καθώς το κόστος της κάρτας αναπαρίσταται ως ένα αντικείμενο του συγκεκριμένου τύπου. Η αρχική προσέγγιση ήταν να αποτελεί μια συμβολοσειρά (ένα String) και για το σκοπό αυτό αρχικά είχε δημιουργηθεί ένας πίνακας cost ο οποίος θα φιλοξενούσε τα κόστη της κάρτας. Η προσέγγιση όμως αυτή δεν λειτούργησε καθώς το κόστος έχει πολλές ιδιότητες που θα πρέπει να αποθηκευτούν οπότε στη συνέχεια προστέθηκε και ο πίνακας costObj ο οποίος φιλοξενεί αντικείμενα τύπου κόστος.

Η κάθε κάρτα, όταν δημιουργείται στο terrainScript της δίνονται όλες οι ιδιότητες της, όπως το όνομα, το όνομα της εικόνας που θα εμφανίζει, το χρώμα της, τα κόστη της κ.α. Στη συνέχεια, στη συνάρτηση assignResource2Cards ελέγχεται αν έχει κάποιο κόστος και αν ναι τότε δημιουργείται το αντικείμενο αυτό του κόστους και αποθηκεύεται στον πίνακα costObject της κάρτας που αναφέρθηκε νωρίτερα. Με τον τρόπο αυτό όταν ο παίχτης αποφασίζει να δημιουργήσει την κατασκευή της συγκεκριμένης κάρτας ανακτά από το Script της τον πίνακα με τα κόστη και ελέγχει τι είδους κόστος αυτή απαιτεί. Ακολούθως ελέγχει αν τα διαθέτει ή αν μπορεί να τα αγοράσει από τους συμπαίχτες τους και ακολούθως το παιχνίδι προχωρά.

```
ageCards[i] = Instantiate(ageCardPref);
cardScr = ageCards[i].GetComponent<ageCardScript>();
cardScr.name = "guardTower";
cardScr.textureName = "guardTower1";
cardScr.color = "Red";
cardScr.miliStrength = 1;
cardScr.minOfplayers = 4;
cardScr.cost[0] = "Clay";
ageCardMaterials = ageCards[i].GetComponent<Renderer>().materials;
ageCardMaterials[0].SetTexture("_MainTex", ageTexturesBack[0]);
ageCardMaterials[1].SetTexture("_MainTex", findText(cardScr.textureName));
i++;
```

Εικόνα 124: Δημιουργία κάρτας. Ο πίνακας cost στη θέση 0 έχει το πρώτο κόστος της κάρτας, κ.ο.κ. Πηγή: Αλέξανδρος Βεκίλογλου

```
void assignResourc2Cards()
{
    for (int i = 0; i < 49; i++)
    {
        ageCardScript cardScr = ageCards[i].GetComponent<ageCardScript>();
        for (int j = 0; j < 10; j++)
        {
            if (cardScr.cost[j] != null)
            {
                GameObject resObj = Instantiate(resourcePrefab);
                ResourceScript resScr = resObj.GetComponent<ResourceScript>();
                if (cardScr.cost[j] == "Clay")
                {
                    resScr.resource = ResourceScript.Resources.Clay;
                    cardScr.costObj[j] = resObj;
                }

                if (cardScr.cost[j] == "Stone")
                {
                    resScr.resource = ResourceScript.Resources.Stone;
                    cardScr.costObj[j] = resObj;
                }
                if (cardScr.cost[j] == "Ore")
                {
```

Εικόνα 125: Συνάρτηση καθορισμού του κόστους κάθε κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Η δημιουργία του αντικειμένου κόστος πραγματοποιείται από τη συνάρτηση assignResourc2Cards η οποία εκτελείται από το terrainScript μόλις ολοκληρωθεί η δημιουργία των καρτών. Η συνάρτηση ανατρέχει όλο τον πίνακα των καρτών και ακολούθως για κάθε κάρτα αποκτά μια αναφορά στο Script της και από εκεί αποκτά πρόσβαση στον πίνακα με τα κόστη της κάθε κάρτας. Ο πίνακας είναι 10 θέσεων και ελέγχει σε όλες τις θέσεις να εντοπίσει αν υπάρχει το όνομα κάποιου πόρου. Εφόσον το εντοπίσει δημιουργεί αντικείμενο πόρου και το αποθηκεύει στην αντίστοιχη θέση στον πίνακα με τα κόστη που αποτελούν πλέον αντικείμενα.

Συνεπώς η συνάρτηση buildBuilding, προκειμένου να διευκρινίσει αν ο παίχτης μπορεί να δημιουργήσει την κατασκευή διατρέχει τον πίνακα με τα κόστη της επιλεγμένης κάρτας και ελέγχει αν υπάρχει σε κάποια θέση τιμή διαφορετική από τη null. Αν υπάρχει τότε η κάρτα έχει κάποιο κόστος και συνεπώς αντιστοιχεί στη μεταβλητή hasCost την τιμή αληθές και τερματίζεται ο βρόχος. Στη συνέχεια ελέγχεται η μεταβλητή hasCost και αν δεν είναι αληθές ο παίχτης μπορεί να δημιουργήσει την κατασκευή αφού δεν έχει κόστος. Αν όμως είναι αληθές ελέγχεται αν ο πόρος είναι

χρήματα ή όχι. Αν ο πόρος είναι χρήματα μετρώνται τα κέρματα που έχει στη διάθεση του ο παίχτης και αν είναι περισσότερα του ενός τότε καταβάλλεται το κόστος της κάρτας, μέσω της συνάρτησης payCoin2Reserve και αντιστοιχίζεται στη μεταβλητή του παίχτη moneyPayed η τιμή αληθές ώστε να είναι γνωστό ότι ο παίχτης έχει πληρώσει το τίμημα της κάρτας. Σε διαφορετική περίπτωση, δηλαδή αν ο πόρος δεν είναι χρήματα εκτελείται η assignRes2Cost η οποία θα διευκρινίσει αν ο παίχτης διαθέτει τους απαιτούμενους πόρους στις κάρτες του.

```
if (hasCost)
{
  if (play0AgeCardScript.cost[0] == "Coin")
    {
        print("This players card has a cost of 1 coin, player has: " + play0.count1coins());
       if(play0.count1coins() >= 1)
       {
           play0.payCoin2Reserve();
           play0AgeCardScript.moneyPayed = true;
       }

    }
    else
        play0.assignRes2Cost();
}
else
    print("Card is for free");
```

Εικόνα 126: Έλεγχος του είδους του κόστους της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Για τον καθορισμό του αριθμού κερμάτων του παίχτη εκτελείται η count1coins η οποία διατρέχει τον πίνακα κερμάτων του ενός του συγκεκριμένου παίχτη και μετρά σε πόσες θέσεις δεν υπάρχει η τιμή null.

```
public int count1coins()
{
    int money = 0;
    for (int i = 0; i < 54; i++)
    {
        if (coins1[i] != null)
            money++;
        else
            break;

    }
    return money;
}
```

Εικόνα 127: Συνάρτηση καθορισμού του αριθμού των διαθέσιμων κερμάτων του ενός του συγκεκριμένου παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

Τέλος για την πληρωμή εκτελείται η payCoin2Reserve η οποία μεταφέρει το τελευταίο νόμισμα από τον πίνακα νομισμάτων του παίχτη στη τελευταία θέση του πίνακα νομισμάτων του σωρού. Στη συνέχεια υπολογίζει τις τυχαίες συντεταγμένες θέσης που θα λάβει στην περιοχή των νομισμάτων στο τραπέζι και τις αναθέτει στο αντικείμενο position του συγκεκριμένου νομίσματος ώστε αυτό να μεταφερθεί στο σωρό. Τέλος αναθέτει στη θέση που βρισκόταν το νόμισμα στον πίνακα coins του παίχτη τη τιμή null ώστε να γνωστό ότι στη θέση αυτή δεν υπάρχει νόμισμα.

```
public void payCoin2Reserve()
{
    print("Player: " + name + " pays one coin to the reserve for card: " + pickedCard.GetComponent<ageCardScript>().name);
    int max1coins = count1coins();
    TerrainScript terScript = Terrain.GetComponent<TerrainScript>();
    GameObject coin = coins1[max1coins - 1];
    terScript.coins1[terScript.count1Coins() - 1] = coin;
    int posX = Random.Range(64, 75);
    int posZ = Random.Range(18, 33);
    coins1[max1coins - 1].transform.position = new Vector3(posX, 1, posZ);
    coins1[max1coins - 1] = null;
}
```

Εικόνα 128: Συνάρτηση πληρωμής νομίσματος. Πηγή: Αλέξανδρος Βεκίλογλου

Η assignRes2Cost θα αναζητήσει τον συγκεκριμένο πόρο στους πόρους που διαθέτει ο παίχτης. Οι πόροι μπορούν να βρίσκονται στις κατασκευές που έχει δημιουργήσει ο παίχτης ή στην κάρτα θαύματος η οποία και αυτή παρέχει ένα πόρο. Για το σκοπό αυτό όταν η κάρτα θαύματος ανατίθεται στον παίχτη της ανατίθεται και ο πόρος τον οποίο παρέχει αφού πρώτα δημιουργηθεί και ο πόρος.

```
void initPlayers()
{

    players[0] = Instantiate(playerPrefab);
    playerScr0 = players[0].GetComponent<PlayerScript>();
    playerScr0.name = "Player0";
    playerScr0.wonderCard = wonderBoard;
    wonderBoard.GetComponent<WonderCardScript>().resources = "Ore";
    GameObject resObj = Instantiate(resourcePrefab);
    ResourceScript resScr = resObj.GetComponent<ResourceScript>();
    resScr.resource = ResourceScript.Resources.Ore;
    resScr.owner = players[0];
    wonderBoard.GetComponent<WonderCardScript>().resourceObj = resObj;
    wonderBoard.GetComponent<WonderCardScript>().name = "Rhodos";
    // rotation of his stack cards
    playerScr0.rotDegrEnd = 0;
    playerScr0.resourcesArr[0] = "Ore";
    playerScr0.setCoinsArea(new Vector3(8, 0.08F, 0));
    playerScr0.setMilisArea(new Vector3(-8, 0.08F, 0));
    playerScr0.discardRotation = new Vector3(180, 90, 0);
```

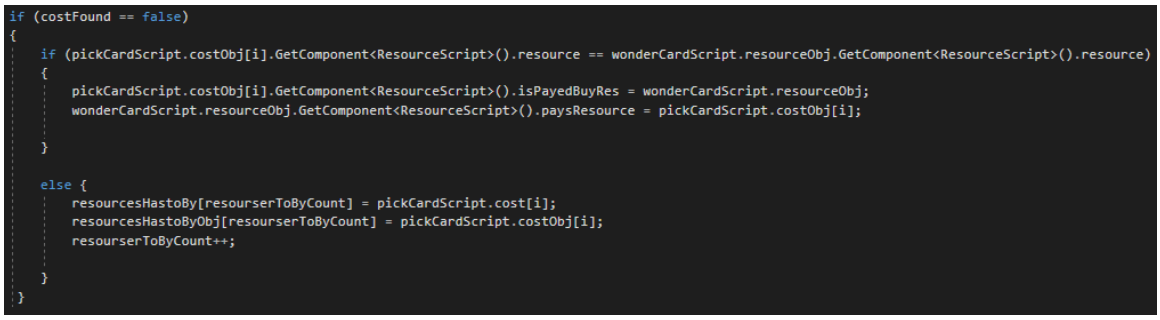
Εικόνα 129: Αρχικοποίηση παίχτη και ανάθεση του κάρτας θαύματος. Πηγή: Αλέξανδρος Βεκίλογλου

Συνεπώς η συνάρτηση ξεκινά και αποκτά πρόσβαση στο Script της επιλεγμένης κάρτας ώστε να ξέρει τι πόρους απαιτεί και στη συνέχεια αποκτά πρόσβαση στο Script της κάρτας θαύματος ώστε να ξέρει τι πόρους παρέχει εκείνη. Στη συνέχεια διατρέχει τον πίνακα που είναι αποθηκευμένα τα κόστη στην επιλεγμένη κάρτα και για κάθε κόστος που εντοπίζει διατρέχει τον πίνακα με τις κάρτες που έχει χτίσει ο παίχτης. Για κάθε κάρτα ανακτά το Script της και ελέγχει τον πίνακα των πόρων που παρέχει η κάρτα. Αν στους διαθέσιμου πόρους της κάρτας υπάρχει κάποιος που ταυτίζεται με τον πόρο που απαιτεί η επιλεγμένη κάρτα για να χτιστεί τότε ο πόρος αυτός αποθηκεύεται στη μεταβλητή isPayedByRes του πόρου της επιλεγμένης κάρτας. Στη συνέχεια ο πόρος που απαιτεί η επιλεγμένη κάρτα ανατίθεται στη μεταβλητή paysResource της χτισμένης κάρτας που διέθεσε τον πόρο. Ο λόγος για τον οποίο γίνεται η αντιστοίχιση αυτή είναι επειδή δεν μπορεί ο ίδιος πόρος να χρησιμοποιηθεί για την κάλυψη περισσότερων του ενός κόστους. Έτσι, στη συνέχεια υπάρχει η δυνατότητα να ελέγχεται όχι μόνο αν ο πόρος υπάρχει αλλά και αν έχει χρησιμοποιηθεί για να καλύψει άλλο κόστος.

```
public void assignRes2Cost()
{
    ageCardScript pickCardScript =  pickedCard.GetComponent<ageCardScript>();
    WonderCardScript wonderCardScript = wonderCard.GetComponent<WonderCardScript>();
    int costsCount = 0;
    int resourserToByCount = 0;
    bool costFound = false;
    for (int i = 0; i < 10; i++)
    { // for every cost of picked card
        if (pickCardScript.costObj[i] != null)
        {
            for (int j = 0; j < 24; j++)
            { // for every card player may have build
                if (cardsBuilded[j] != null)
                {
                    ageCardScript buildedCardScript = cardsBuilded[j].GetComponent<ageCardScript>();

                        for (int k = 0; k < 10; k++)
                        { // for every resource a builded card may produce
                            if (buildedCardScript.resourcesObj[k] != null)
                            {
                                if (pickCardScript.costObj[i].GetComponent<ResourceScript>().resource == buildedCardScript.resourcesObj[k].GetComponent<ResourceScript>().resource)
                                {
                                pickCardScript.costObj[i].GetComponent<ResourceScript>().isPayedBuyRes = buildedCardScript.resourcesObj[k];
                                buildedCardScript.resourcesObj[k].GetComponent<ResourceScript>().paysResource = pickCardScript.costObj[i];
                                costFound = true;
                                }
                            }
                        }

                }
            }
        }
    }
```

Εικόνα 130: Αναζήτηση απαιτούμενων πόρων στις κάρτες που έχει χτίσει ο παίχτης. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συνέχεια ελέγχεται η μεταβλητή costFound και αν είναι ψευδές, δηλαδή ο απαιτούμενος πόρος δεν μπορεί να εντοπιστεί από τις χτισμένες κάρτες τότε αναζητιέται στην κάρτα θαύματος. Αν εντοπιστεί πραγματοποιείται η αντιστοίχιση του προηγούμενου βήματος. Διαφορετικά ο πόρος δεν μπορεί να καλυφθεί. Επόμενος θα πρέπει να αναζητηθεί στους γειτονικούς παίχτες. Πιθανόν όμως η συγκεκριμένη κάρτα να απαιτεί περισσότερους του ενός πόρους και άρα περισσότεροι του ενός να πρέπει να αναζητηθούν σε γείτονες. Για το σκοπό αυτό δημιουργείται στον παίχτη ένας πίνακας με τους πόρους που θα πρέπει να αναζητήσει στους γείτονες. Στο πίνακα αυτό ανατίθεται ο πόρος ή οι πόροι που θα πρέπει να αναζητηθούν και ο βρόχος επαναλαμβάνεται για όλους τους πόρους που απαιτεί η επιλεγμένη κάρτα.



```
if (costFound == false)
{
    if (pickCardScript.costObj[i].GetComponent<ResourceScript>().resource == wonderCardScript.resourceObj.GetComponent<ResourceScript>().resource)
    {
        pickCardScript.costObj[i].GetComponent<ResourceScript>().isPayedBuyRes = wonderCardScript.resourceObj;
        wonderCardScript.resourceObj.GetComponent<ResourceScript>().paysResource = pickCardScript.costObj[i];

    }

    else {
        resourcesHastoBy[resourserToByCount] = pickCardScript.cost[i];
        resourcesHastoByObj[resourserToByCount] = pickCardScript.costObj[i];
        resourserToByCount++;

    }
}
```

Εικόνα 131: Αναζήτηση του πόρου στη κάρτα θαύματος ή αποθήκευσης του. Πηγή: Αλέξανδρος Βεκίλογλου

Πλέον ο παίχτης έχει σε ένα πίνακα όλους τους πόρους που θα πρέπει να αναζητήσει στους γείτονες του και οι δικοί του πόροι έχουν ανατεθεί στους απαιτούμενους της επιλεγμένης κάρτας, αν υπάρχουν. Συνεπώς στη συνέχεια θα πρέπει, αν υπάρχουν πόροι που θα πρέπει να αναζητηθούν σε γείτονες αυτοί να αναζητηθούν.

Η εκτέλεση του προγράμματος, μετά την assighRes2Cost επιστρέφει πίσω στο terrainScript και στη buildBulding. Πλέον έχει καθοριστεί αν η επιλεγμένη κάρτα μπορεί να κτιστεί δωρεάν ή έχει κάποιο κόστος. Αν η κάρτα δεν έχει κάποιο κόστος στη τιμή hasCost έχει ανατεθεί η τιμή ψευδές. Αν το κόστος είναι χρηματικό και έχει καλυφθεί τότε στη μεταβλητή moneyPayed έχει ανατεθεί η τιμή true. Αν η κάρτα έχει κόστος σε πόρους και αυτοί έχουν καλυφθεί είτε από τις κτισμένες κάρτες είτε από την κάρτα θαύματος, πράγμα που ελέγχεται από την isCostCovered τότε ο παίχτης μπορεί να

κατασκευάσει την συγκεκριμένη κάρτα. Συνεπώς, στη buildBuilding ελέγχονται αυτές οι τρείς μεταβλητές και αν όλες ικανοποιούνται ο κώδικάς προχωρά στην κατασκευή της επιλεγμένης κάρτας.

```
if (play0.isCostCovered() || (play0AgeCardScript.moneyPayed==true) || (hasCost==false))
{

    print("Player0 can buid the building");
    play0.buildBuilding();
    play0.removeCardFromArr();
    print("Player: " + play0.name + " Cards left: " + play0.countCards());
    play0.disopsalLastCard();
    turnIsCompleted = true;

}
else
```

Εικόνα 132: Συνέχεια της buildBulding. Πηγή: Αλέξανδρος Βεκίλογλου

Η συνάρτηση isCostCovered απλά ελέγχει τον πίνακα με τους πόρους που πρέπει να πληρώσει ο παίχτης. Τον πίνακα αυτό τον συμπληρώνει η assignRes2Cost που παρουσιάστηκε στη προηγούμενη παράγραφο. Η συνάρτηση αν εντοπίσει πόρο που δεν μπορεί να καλυφτεί από τους διαθέσιμους αποθηκεύει τους πόρους αυτούς που δεν μπορούν να καλυφθούν στον πίνακα αυτό. Στόχος είναι να αναζητηθούν οι πόροι αυτοί στους γείτονες. Συνεπώς η isCostCovered απλά διατρέχει τον πίνακα και αν εντοπίσει κάποιο πόρο σε κάποια θέση επιστρέφει ψευδές καθώς υπάρχει πόρος που δεν έχει καλυφθεί.

```
public bool isCostCovered()
{
    ageCardScript pickCardScript = pickedCard.GetComponent<ageCardScript>();

    for (int i = 0; i < 10; i++)
    {
        if (resourcesHastoByObj[i] != null)
        {
            return false;
        }
    }
    return true;
}
```

Εικόνα 133: Έλεγχος για το αν έχουν καλυφθεί οι πόροι που απαιτεί η συγκεκριμένη επιλεγμένη κάρτα. Πηγή: Αλέξανδρος Βεκίλογλου

Εφόσον ο παίχτης μπορεί να δημιουργήσει την κατασκευή εκτελείται η συνάρτηση buildBuilding του παίχτη. Η συνάρτηση αφού αποκτήσει πρόσβαση στο Script της κάρτας αναθέτει στην ιδιότητα της hasToMove την τιμή αληθές ώστε αυτή να κινηθεί σταδιακά από την Update, όπως παρουσιάστηκε σε προηγούμενη παράγραφο. Η κάρτα θα πρέπει να μεταφερθεί κάτω από την πάνω αριστερή γωνία της κάρτας θαύματος του κάθε παίχτη συνεπώς ο κώδικας καθορίζει σε ποιο παίχτη ανήκει η κάρτα και ενημερώνει την μεταβλητή position με τις ξεχωριστές συντεταγμένες του σημείου που θα πρέπει να πάει. Ακολούθως την περιστρέφει κατάλληλα και αναθέτει στον πίνακα με τους διαθέσιμους πόρους του συγκεκριμένου παίχτη τους πόρους που προσφέρει η κάρτα που χτίζει. Στη συνέχεια, στον πίνακα με τις κάρτες που ο παίχτης έχει κατασκευάσει προστίθεται η συγκεκριμένη

κάρτα, ώστε να είναι και αυτή μια από αυτές και αυξάνεται κατά μια μονάδα μια μεταβλητή που αντιστοιχεί στον συνολικό αριθμό των καρτών που ο παίχτης έχει χτίσει.

```
public void buildBuilding()
{
    ageCardScript play0AgeCardScript = pickedCard.GetComponent<ageCardScript>();
    play0AgeCardScript.hasToMove = true;
    if (name == "Player0")
    {
        pickedCard.transform.position = pickedCardPos + new Vector3(0, cardsBuildedCounter * (-0.1F), cardsBuildedCounter * (0.3F));
    }
    else if (name == "Player1")
        pickedCard.transform.position = pickedCardPos + new Vector3(cardsBuildedCounter * (0.5F), cardsBuildedCounter * (-0.1F), 0);
    else if (name == "Player2")
        pickedCard.transform.position = pickedCardPos + new Vector3(0, cardsBuildedCounter * (-0.1F), cardsBuildedCounter * (-0.3F));
    else
        pickedCard.transform.position = pickedCardPos + new Vector3(cardsBuildedCounter * (-0.5F), cardsBuildedCounter * (-0.1F), 0);
    pickedCard.transform.Rotate(pickedCardPosRotate);
    play0AgeCardScript.assignOwner2Resource(this.gameObject);
    cardsBuilded[cardsBuildedCounter] = pickedCard;
    cardsBuildedCounter++;
    print("Player: " + name + " builds: " + play0AgeCardScript.name);

}
```

Εικόνα 134: Κατασκευή κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Μετά την εκτέλεση της buildBulding του παίχτη ο κώδικας επιστρέφει στη buildBulding του terrainScript και εκτελείται η removeCardFromArr. Στόχος της συνάρτησης είναι να αφαιρέσει την κάρτα που κτίστηκε από τις κάρτες που έχουν μοιραστεί στον παίχτη. Πιο συγκεκριμένα, στην αρχή του κάθε γύρου ο κάθε παίχτης λαμβάνει 7 κάρτες, τις αποθηκεύει στον πίνακα ageCards που διαθέτει, παίζει τη μία και ακολούθως περνά τις υπόλοιπες στον γείτονα του. Συνεπώς η κάρτα που χτίζει θα πρέπει να αφαιρεθεί από τον πίνακα ageCards. Για να το πετύχει αυτό η συνάρτηση λαμβάνει το Script της επιλεγμένης κάρτας του παίχτη από την μεταβλητή pickedCard και το συγκρίνει με όλα τα Script των καρτών που υπάρχουν στον πίνακα του, καθώς δεν γνωρίζει το πρόγραμμα σε ποια θέση του πίνακα βρισκόταν η κάρτα που επέλεξε να παίξει. Συγκρίνει συνεπώς τα δύο Script και στη περίπτωση που ταυτίζονται αναθέτει στη θέση αυτή του πίνακα την τιμή null ώστε να διαγράψει τη συγκεκριμένη κάρτα από τον πίνακα. Ακολούθως τερματίζει το βρόχο και επιστρέφει.

```
public void removeCardFromArr()
{
    for (int i = 0; i < 7; i++)
    {
        if (ageCards[i] != null)
        {
            ageCardScript tablecardScr = ageCards[i].GetComponent<ageCardScript>();
            ageCardScript pickedCardScr = pickedCard.GetComponent<ageCardScript>();

            if (tablecardScr.textureName == pickedCardScr.textureName)
            {
                ageCards[i] = null;
                break;
            }

        }
    }
}
```

Εικόνα 135: Συνάρτηση διαγραφής χτισμένης κάρτας απο τον πίνακα με τις διαθέσιμες κάρτες. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως ελέγχεται αν η κάρτα αυτή που παίχτηκε ήταν η τελευταία του γύρου. Πιο συγκεκριμένα αν ο παίχτης έχει μόνο μία κάρτα στην κατοχή του και έχει παίξει τις υπόλοιπες έξι αυτή η τελευταία θα πρέπει να απορριφτεί στη στοίβα με τις κάρτες στο κέντρο του τραπεζιού και ο γύρος να ολοκληρωθεί. Συνεπώς εκτελείται η συνάρτηση disposalLastCard η οποία ελέγχει τον αριθμό των καρτών που απομένουν στον πίνακα καρτών του παίχτη. Το μέτρημα των καρτών εκτελείται με την countCards η οποία μετρά τις θέσεις του πίνακα οι οποίες περιέχουν τιμή διαφορετική του null και επιστρέφει τον αριθμό αυτό.

```
public int countCards()
{
    int counter = 0;
    for (int i = 0; i < 7; i++)
    {
        if (ageCards[i] != null)
        {
            counter++;
        }
    }
    return counter;
}
```

Εικόνα 136: Συνάρτηση countCards. Πηγή: Αλέξανδρος Βεκίλογλου

Αν ο αριθμός που η συνάρτηση επιστρέψει ισούται με τη μονάδα άρα η κάρτα αυτή που απομένει στον πίνακα θα πρέπει να απορριφτεί στη στοίβα με τις κάρτες. Συνεπώς η συνάρτηση εντοπίζει την κάρτα εντός του πίνακα με τις κάρτες η οποία είναι διαφορετική από αυτή που έχει επιλεγεί από τον παίχτη για κατασκευή και την μεταφέρει στον χώρο πάνω στο τραπέζι που είναι αφιερωμένος για αυτό το σκοπό. Ακολούθως την περιστρέφει ώστε να είναι με τη σωστή όψη και το σωστό προσανατολισμό και τέλος αναθέτει στη θέση αυτή του πίνακα με τις κάρτες του παίχτη την τιμή null ώστε να διαγραφεί η κάρτα από τον πίνακα με τις κάρτες.

```
public void disopsalLastCard()
{
    if (countCards() == 1)
    {
        print("Player: " + name + " has only 2 cards, disposing the one that is not picked");
        for (int i = 0; i < 7; i++)
        {
            if (ageCards[i] != null)
            {
                // if table's card is not the picked one disopsal it
                ageCardScript tablecardScr = ageCards[i].GetComponent<ageCardScript>();
                ageCardScript pickedCardScr = pickedCard.GetComponent<ageCardScript>();

                if (tablecardScr.textureName != pickedCardScr.textureName)
                {
                    ageCards[i].transform.position = new Vector3(52, 1, 33);
                    ageCards[i].transform.Rotate(discardRotation);
                    print("Player: " + name + " disposes his card: " + tablecardScr.name);
                    ageCards[i] = null; // remove disposed card from player's array
                }
                else
                    print("Player: " + name + " could not find card to disposa");
            }
        }
    }
}
```

Εικόνα 137: Συνάρτηση disposalLastCard. Πηγή: Αλέξανδρος Βεκίλογλου

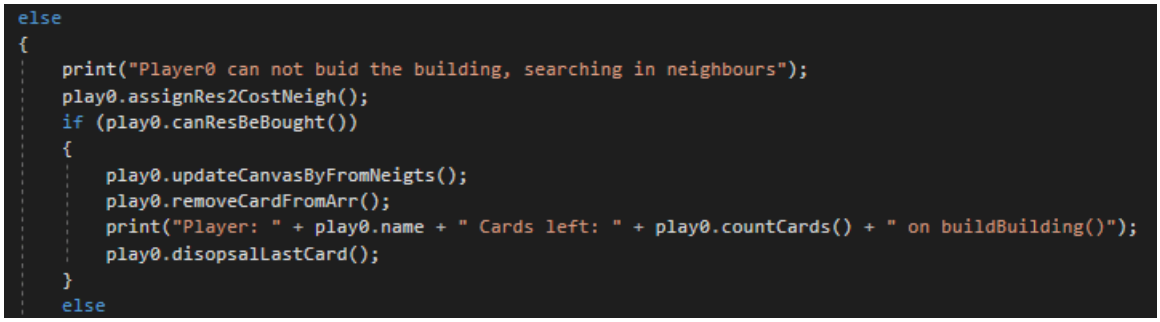
Μόλις ολοκληρωθεί η εκτέλεση και της συνάρτησης αυτής ο έλεγχος επιστρέφει πίσω στη buildBuilding του terrainScript και ανατίθεται στην μεταβλητή turnIsCompleted η τιμή αληθές ώστε να είναι γνωστό ότι η κάρτα έχει κτιστεί, ο γύρος έχει ολοκληρωθεί και το παιχνίδι μπορεί να συνεχίσει με τη μεταφορά των υπολοίπων καρτών στους γειτονικούς παίχτες.

Στη περίπτωση που καμία από τις συνθήκες (κάλυψη πόρου από ιδίους, πληρωμή χρημάτων, δωρεάν κάρτα) δεν ικανοποιείται τότε ο κώδικάς θα αναζητήσει στους γείτονες να εντοπίσει τους πόρους που λείπουν από τον παίχτη και έχουν αποθηκευτεί στον πίνακα resourcesHastoByObj. Η διαδικασία εκτελείται από τη συνάρτηση assighRes2CostNeigh η οποία ανατρέχει τον πίνακα με τους πόρους που πρέπει να αγοραστούν και αναζητά τον κάθε πόρο στους δύο γείτονες του παίχτη με τον ίδιο τρόπο που γίνεται και η αναζήτηση του απαιτούμενου πόρου στους διαθέσιμους πόρους του παίχτη. Πιο συγκεκριμένα αναζητά τον πόρο τόσο στις κάρτες που αυτοί έχουν χτίσει αλλά και στους πόρους που παρέχουν οι κάρτες θαύματα τους. Αν εντοπίσει τον πόρο τότε αντιστοιχεί στον πόρο που καλύπτεται τον πόρο που τον καλύπτει αλλά και το αντίστροφο, όπως ακριβώς έκανε η συνάρτηση που αναζητούσε τον πόρο στους δικούς του πόρους. Η αναζήτηση εδώ ελέγχει και αν ο πόρος έχει χρησιμοποιηθεί ξανά για να καλύψει την ίδια κατασκευή καθώς αυτό δεν επιτρέπεται.

Εικόνα 138: Αναζήτηση του απαιτούμενου πόρου στον αριστερό γείτονα. Πηγή: Αλέξανδρος Βεκίλογλου

Πλέον η εφαρμογή έχει εικόνα για όλους τους πόρους που υπάρχουν στον πίνακα με τους πόρους που πρέπει να καλυφθούν αν διατίθενται από τους γείτονες. Πιο συγκεκριμένα ο κάθε πόρος στον πίνακα με τους πόρους που θα πρέπει να καλυφθούν διαθέτει μεταβλητή isPayedByRes στην οποία έχει ανατεθεί ο πόρος που τον καλύπτει. Ακολούθως ελέγχεται αν όλοι οι πόροι του πίνακα μπορούν να παρασχεθούν από τους δύο γείτονες. Ο έλεγχος πραγματοποιείται από την canResBeBought που εκτελείται στη συνέχεια, η οποία διατρέχει τον πίνακα με τους απαιτούμενους πόρους και ελέγχει αν όλων η μεταβλητή isPayedByRes περιέχει κάποιο πόρο. Αν σε κάποια της έχει ανατεθεί η τιμή null αυτό σημαίνει ότι κάποιος πόρος δεν μπορεί να καλυφθεί από κάποιο γείτονα, η συνάρτηση επιστρέφει ψευδές και τερματίζεται ο βρόχος. Συνεπώς η κάρτα αυτή δεν μπορεί να κατασκευαστεί και ο παίχτης θα πρέπει να επιλέξει κάποια άλλη. Σε διαφορετική περίπτωση επιστρέφει αληθές ώστε ο παίχτης να προχωρήσει στην αγορά των συγκεκριμένων πόρων.

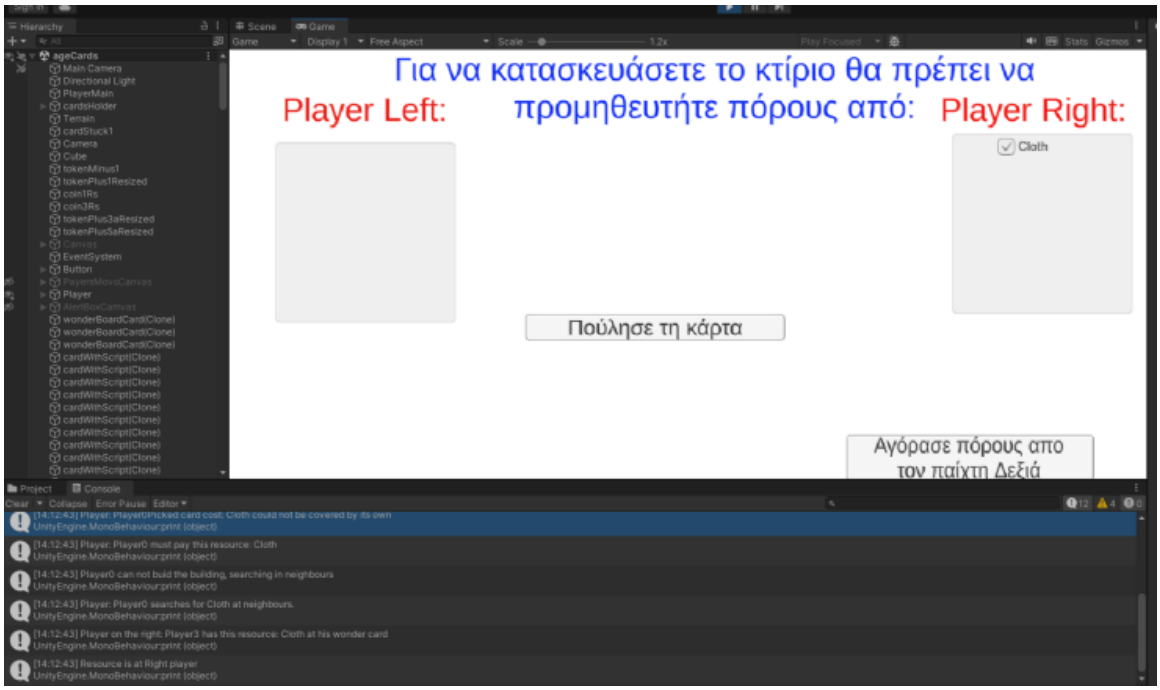
Εικόνα 139: Έλεγχος για το αν οι πόροι μπορούν να αγοραστούν. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 140: Κώδικας που εκτελείται αφού εντοπιστούν οι πόροι σε γείτονες. Πηγή: Αλέξανδρος Βεκίλογλου

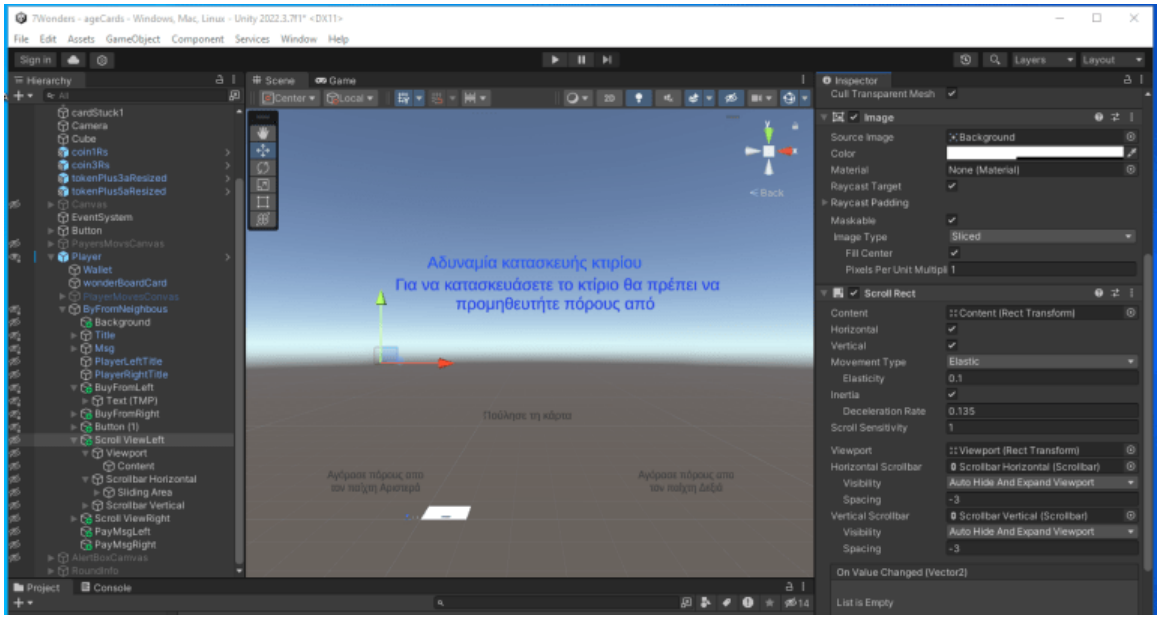
Εφόσον όλοι οι πόροι μπορούν να αγοραστούν από τους γείτονες θα πρέπει η εφαρμογή να παρουσιάσει στον παίχτη σε ποιο γείτονα εντόπισε ποιον πόρο και εκείνος να αποφασίσει από ποιον θα αγοράσει, αν τελικά επιθυμεί να αγοράσει. Το σκεπτικό είναι ότι ένας πόρος ή και περισσότεροι, μπορούν να παρέχονται από ένα ή και τους δύο γείτονες. Ο παίχτης με τους γείτονες αυτούς, στο τέλος του γύρου, θα αναμετρηθεί στρατιωτικά και η αναμέτρηση αυτή θα κρίνει το νικητή του γύρου. Συνεπώς ο παίχτης θα πρέπει να έχει την δυνατότητα επιλογής του παίχτη από τον οποίο θα αγοράσει. Μπορεί για παράδειγμα να επιλέξει να αγοράσει από τον πιο αδύναμο ώστε να αποφύγει να ενισχύσει τον ποιο δυνατό και να βελτιώσει τις πιθανότητες του αργότερα στην στρατιωτική αναμέτρηση. Ή ακόμα και να αποφασίσει ότι τελικά δεν επιθυμεί να αγοράσει από κάποιον και να επιλέξει κάποια άλλη κάρτα τελικά.

Για το σκοπό αυτό αποφασίστηκε να δημιουργηθεί ένα νέο παράθυρο διαλόγου το οποίο να έχει την παραπάνω λειτουργικότητα. Να εμφανίζει δηλαδή τους δύο γείτονες ξεχωριστά και από κάτω από τον κάθε γείτονα μια λίστα από check boxes τα οποία θα αποτελούν τους πόρους που ο κάθε γείτονας διαθέτει. Παράλληλα να υπάρχει και κουμπί πώλησης της κάρτας σε περίπτωση που δεν επιθυμεί να αγοράσει τελικά από κάποιο γείτονα.



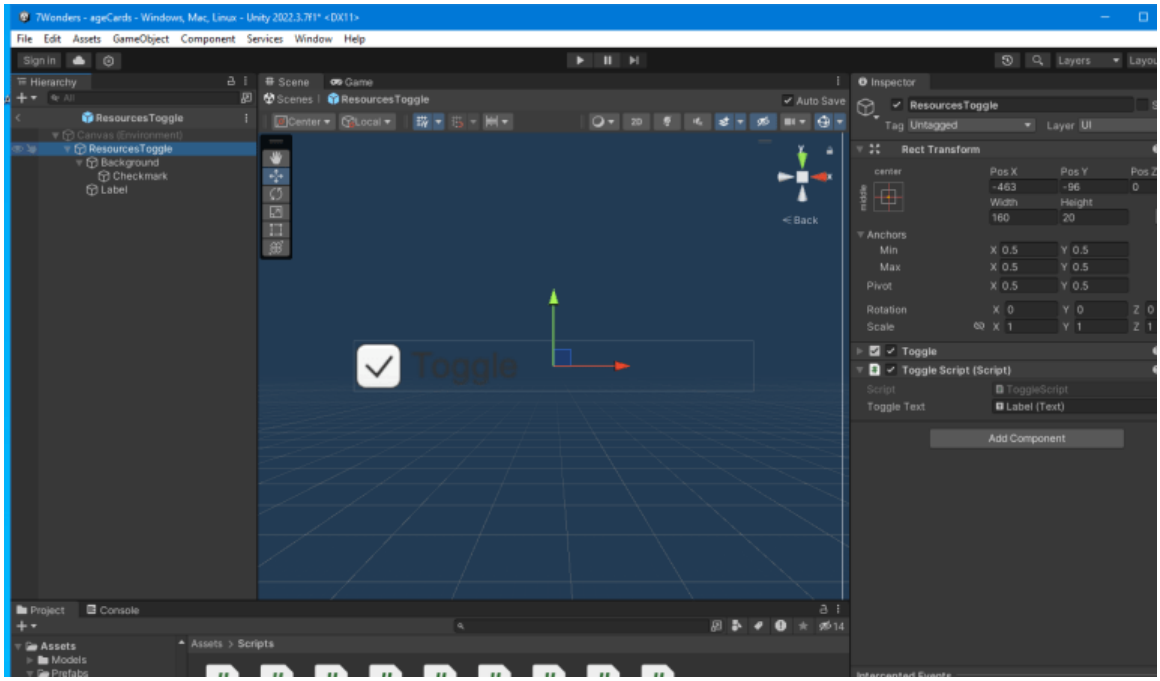
Εικόνα 141: Παράθυρο επιλογής πόρου από γείτονα. Πηγή: Αλέξανδρος Βεκίλογλου

Το παράθυρο κατασκευάζεται όπως παρουσιάστηκε στις προηγούμενες παραγράφουν με εξαίρεση το background και τις λίστες με τους πόρους. Οι λίστες αυτές όπου χρησιμοποιείται μία ανά γείτονα, αποτελούν αντικείμενα Scroll View. Τα αντικείμενα αυτά περιέχουν ένα αντικείμενο Scroll Rect το οποίο παρέχει τις απαραίτητες λειτουργίες προκειμένου να γίνονται scroll κάποια αντικείμενα, ένα ViewPort το οποίο ορίζει την ορατή περιοχή του scroll και τέλος το αντικείμενο content το οποίο ορίζει πως θα τοποθετούνται τα περιεχόμενα της scroll list. Το Scroll View μπορεί να περιέχει και μπάρες τόσο για οριζόντια κύλιση όσο και κάθετη.



Εικόνα 142: Αντικείμενο Scroll View. Πηγή: Αλέξανδρος Βεκίλογλου

Συνεπώς για να εισαχθούν οι πόροι στο Scroll View του κάθε παίχτη δημιουργούνται δύο πίνακες, ένας για κάθε παίχτη και σε κάθε πίνακα εισάγεται ένα αντικείμενο τύπου καμβάς το οποίο περιέχει το check box και ένα κείμενο. Ο καμβάς αυτός θα αντιστοιχεί στον πόρο που ο γείτονας παρέχει και ο παίχτης θα πρέπει να τον επιλέξει εφόσον το επιθυμεί. Παράλληλα περιέχει και ένα Script το οποίο διατηρεί μια μεταβλητή με το όνομα text η οποία θα αντιστοιχεί στο κείμενο που θα εμφανίζει.



Εικόνα 143: Το αντικείμενο που θα αναπαριστά τον πόρο, το Check Box στις λίστες. Πηγή: Αλέξανδρος Βεκίλογλου

Το αντικείμενο αυτό έχει ανατεθεί στον Player σαν Prefab συνεπώς αυτός μπορεί να δημιουργεί τέτοια αντικείμενα και να τα χειρίζεται. Επομένως ο κώδικας θα ανατρέξει τον πίνακα με τους πόρους που θα πρέπει να αγοράσει ο παίχτης και για κάθε πόρο που εντοπίζει ελέγχει την μεταβλητή isPayedByRes. Στη τιμή αυτή έχει ανατεθεί ο πόρος που τον καλύπτει, ο όποιος έχει κάποιο ιδιοκτήτη (owner). Ο παίχτης γνωρίζει ποιος παίχτης βρίσκεται αριστερά του και ποιος δεξιά συνεπώς για να

καθοριστεί σε ποιο παίχτη ανήκει ο πόρος αρκεί να συγκριθεί ο ιδιοκτήτης του πόρου με τους γείτονες. Αν ταυτίζεται τότε ο πόρος ανήκει σε αυτόν τον γείτονα.

```
if (resourcesHastoByObj[i] != null)
{
    if (resourcesHastoByObj[i].GetComponent<ResourceScript>().isPayedBuyRes == null)
    {
        print("This resourcesHastoByObj: " + i + " with name: " + resourcesHastoByObj[i].GetComponent<ResourceScript>().printResource() +
    }
    else if (resourcesHastoByObj[i].GetComponent<ResourceScript>().isPayedBuyRes.GetComponent<ResourceScript>().owner == playerLeft)
    {
        print("Resource is at Left player");
        GameObject toggle = Instantiate(togglePrefab, scrollLeft.transform);
        toggle.GetComponent<ToggleScript>().toggleText.text = resourcesHastoByObj[i].GetComponent<ResourceScript>().printResource();
        toggle.GetComponent<UnityEngine.UI.Toggle>().isOn = false;
        leftToggles[i] = toggle;
    }
    else
```

Εικόνα 144: Εντοπισμός ιδιοκτήτη του πόρου. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις εντοπισθεί ο ιδιοκτήτης του πόρου δημιουργείται ένα CheckBox στη θέση του ScrollView του αντίστοιχου γείτονα και στο τίτλο του αναγράφεται το όνομα του πόρου. Το γεγονός ότι το αντικείμενο δημιουργείται στο πατρικό GameObject έχει σαν συνέπεια να αποτελεί αντικείμενο τύπου content – συνεπώς προστίθεται στη λίστα των αντικειμένων Scroll View. Για να εισαχθεί το όνομα του πόρου στο αντικείμενο Text του έχει ανατεθεί στο Script του το αντικείμενο Label. Μέσω της GetComponent ανακτάται μια αναφορά στο αντικείμενο Label με όνομα toggleText και η ιδιότητα text αυτού παίρνει την τιμή του ονόματος του πόρου. Η τιμή isOn τίθεται σε ψευδές ώστε το CheckBox να μην είναι επιλεγμένο και το αντικείμενο εισάγεται στον πίνακα με τα CheckBoxes του συγκεκριμένου γείτονα.

Ο έλεγχος πλέον του παιχνιδιού περνά στον καμβά που μόλις παρουσιάστηκε, αφού αφαιρεθεί η κάρτα από τον πίνακα καρτών του παίχτη και ελεγχθεί αν είναι η τελευταία κάρτα του γύρου. Οι συναρτήσεις αυτές είναι οι ίδιες που εκτελούνται και όταν ο παίχτης ολοκληρώνει την κατασκευή με τους δικούς του πόρους. Σε διαφορετική περίπτωση, όταν δηλαδή οι γείτονες δεν παρέχουν τους πόρους που απαιτούνται η κάρτα επιστρέφει στην αρχική της θέση και η διαδικασία επιλογής κάρτας ξεκινά από την αρχή.

```
else
{
    alertCanvas.SetActive(true);
    canvasMsgTitle.text = "Αποτυχία";
    canvasMsgText.text = "Δεν μπορείτε να αγοράσετε το σύνολλο των πόρων της συγκεκριμένης κάρτας";
    ageCardsc2Center = null;
    // this card is not mouse ckliced
    play0AgeCardScript.isCardPicked = false;
    // send card back to the show cards position
    play0.pickedCard.transform.Rotate(new Vector3(200, 0, 0));
    // play0AgeCardScript.transform.position = play0AgeCardScript.play0Show;
    showCards2Player0();
    play0.initPlayer();
}
```

Εικόνα 145: Επιλογή νέας κάρτας καθώς η επιλεγμένη δεν είναι εφικτό να κατασκευαστεί. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις ο παίχτης επιλέξει κάποιο πόρο για αγορά και εφόσον διαθέτει τα χρήματα για να τον αγοράσει ενεργοποιείται το αντίστοιχο κουμπί ‘Αγόρασε πόρους από τον παίχτη’. Το κουμπί παραμένει αόρατο καθ’ όλη τη διάρκεια του παιχνιδιού και εμφανίζεται μόνο όταν ο παίχτης επιλέξει πόρους από συγκεκριμένο παίχτη αλλά και μπορεί να τους αγοράσει. Για να επιτευχθεί η διαδικασία αυτή το

Script του παίχτη ελέγχει συνεχώς τους πίνακες με τα CheckBoxes. Για κάθε checkbox που βρίσκει τσεκαρισμένο υπολογίζει το κόστος των επιλεγμένων πόρων αυξάνοντας κατά 2 μια global μεταβλητή. Συνεπώς ανά καρέ η εφαρμογή γνωρίζει πόσα checkboxes είναι επιλεγμένα και πόσα χρήματα πρέπει να καταβληθούν σε κάθε γείτονα.

```
public void estimateCost()
{
    int resCostLeft = 0;
    int resCostRIght = 0;
    for (int i = 0; i < 10; i++)
    {
        if (leftToggles[i] != null)
        {
            if (leftToggles[i].GetComponent<UnityEngine.UI.Toggle>().isOn)
            {
                resCostLeft += 2;

            }
        }
        if (rightToggles[i] != null)
        {
            if (rightToggles[i].GetComponent<UnityEngine.UI.Toggle>().isOn)
            {
                resCostRIght += 2;

            }
        }

    }
```

Εικόνα 146: Υπολογισμός κόστους πόρων. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συνέχεια η συνάρτηση ελέγχει αν ο χρήστης διαθέτει αρκετά νομίσματα. Στη περίπτωση που δεν διαθέτει ενημερώνει την αντίστοιχη τιμή κειμένου του καμβά ώστε ο χρήστης να γνωρίζει για ποιο λόγο δεν εμφανίζεται το κουμπί αγοράς πόρων αφού έχει επιλέξει πόρους. Στη περίπτωση που τα διαθέσιμα χρήματα είναι περισσότερα ή ίσα με το κόστος των πόρων που θα πρέπει να αγοραστούν και υπάρχουν πόροι για αγορά τότε εμφανίζεται το αντίστοιχο κουμπί εκτελώντας την συνάρτηση SetActive του αντικειμένου, παρέχοντας της ως όρισμα την τιμή αληθές.

```
if ((countCoins() >= resCostLeft) && (resCostLeft >= 2))
{
    // print("Player can pay for resources from left neigh");
    btnByFromLeft.SetActive(true);
}
else if (resCostLeft >= 2)
    moneyMesageLeft.text = "Δεν μπορείτε να αγοράσετε το σύνολλο των πόρων, συνολλο νομισμάτων του 1:" + countCoins();

if ((countCoins() >= resCostRIght) && (resCostRIght >= 2))
{
    // print("Player can pay for resources from right neigh");
    btnByFromRight.SetActive(true);
}
else if (resCostRIght >= 2)
    moneyMesageLeft.text = "Δεν μπορείτε να αγοράσετε το σύνολλο των πόρων, συνολλο νομισμάτων του 1:" + countCoins();
```

Εικόνα 147: Ενεργοποίηση ή όχι κουμπιού αγοράς πόρων. Πηγή: Αλέξανδρος Βεκίλογλου

Το κάθε κουμπί αγοράς του καμβά εκτελεί τη δική του συνάρτηση byFromNeighLeft για αγορά από τον γείτονα αριστερά και byFromNeighRight για αγορά από το γείτονα δεξιά. Η συνάρτηση μεταφέρει νομίσματα στον αντίστοιχο παίχτη εφόσον, ο βασικός παίχτης μπορεί να τα πληρώσει, απενεργοποιεί τον καμβά καθώς αυτός πλέον δεν χρειάζεται, κατασκευάζει την κάρτα μέσω της buildBuilding και θέτει τη μεταβλητή turnIsCompleted σε αληθές ώστε να γνωρίζει το terrainScript ότι ο γύρος έχει ολοκληρωθεί. Διαφορετικά, αν ο παίχτης δεν διαθέτει τα αντίστοιχα χρήματα ενημερώνει τον παίχτη κατάλληλα εμφανίζοντας έναν επιπλέον καμβά με τα κατάλληλα μηνύματα.

```
public void byFromNeighLeft()
{
    if (canIPay(leftToggles))
    {
        print("Pay coind to left");
        payCoins(2, playerLeft);
        ByFromNeigCanvas.SetActive(false);
        buildBuilding();
        TerrainScript terScript = Terrain.GetComponent<TerrainScript>();
        terScript.turnIsCompleted = true;
    }
    else
    {
        alertCanvas.SetActive(true);
        canvasMsgTitle.text = "Αποτυχία";
        canvasMsgText.text = "Δεν μπορείτε να αγοράσετε το σύνολλο των πόρων της συγκεκριμένης κάρτας";
    }
}
```

Εικόνα 148: Αγορά πόρων από τον γείτονα αριστερά. Πηγή: Αλέξανδρος Βεκίλογλου

5.4.4 Πώληση της κάρτας

Στη περίπτωση που ο παίχτης δεν μπορεί ή δεν επιθυμεί να αγοράσει πόρους από κάποιον γείτονα θα πατήσει το κουμπί πώλησης της κάρτας. Τότε η κάρτα θα μεταφερθεί στο σωρό με τις κάρτες, θα μεταφερθούν 3 νομίσματα του ενός στην κάρτα θαύματος του και ο καμβάς αγοράς πόρων από τους γείτονες θα κλείσει. Οι μεταβλητές θα αρχικοποιηθούν και θα ξεκινήσει ο νέος γύρος.

```
public void shellCard()
{
    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    ageCardsc2Center = null;
    play0.sellCard();
    camvas.SetActive(false);
    play0.closeCanvas();
    play0.removeCardFromArr();
    print("Player: " + play0.name + " Cards left: " + play0.countCards());
    play0.disopsalLastCard();
    turnIsCompleted = true;
}
```

Εικόνα 149: Πώληση κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι οι διαδικασίες που περιεγράφηκαν παραπάνω για το βασικό παίχτη εκτελούνται και από τους γείτονες του. Μοναδικές διαφορές αποτελούν το ότι οι υπόλοιποι παίχτες επιλέγουν τυχαία ποια κάρτα θα παίξουν και δεν επιλέγουν από ποιο γείτονα θα αγοράσουν πόρους στη περίπτωση που αυτό απαιτηθεί. Πιο συγκεκριμένα η εφαρμογή αγοράζει τον κάθε πόρο στο πρώτο γείτονα που τον βρίσκει ξεκινώντας από το γείτονα στα δεξιά. Στη περίπτωση

που ο πόρος δεν μπορεί να αγοραστεί η κάρτα πωλείται και ο παίχτης αποκτά τα χρήματα. Όλες οι διαδικασίες γίνονται μέσω των συναρτήσεων που παρουσιάστηκαν στα προηγούμενα βήματα.

5.4.5 Ολοκλήρωση γύρου

Η πρώτη περίοδος αποτελείται από 6 γύρους σε κάθε ένα από τους οποίους ο κάθε παίχτης επιλέγει μία κάρτα από αυτές που του έχουν μοιραστεί, την παίζει και στη συνέχεια περνά τις υπόλοιπες στο γείτονα του. Μόλις ο βασικός παίχτης ολοκληρώσει το παίξιμο ενεργοποιεί την μεταβλητή turnIsCompleted. Η μεταβλητή ελέγχεται στην Update του terrainScript και μόλις εντοπισθεί να είναι αληθές τότε αυξάνεται μια μεταβλητή κατά ένα η οποία μετρά πόσες κάρτες έχουν παιχθεί ώστε όταν φτάσει στο 6 να γνωρίζει ότι ο γύρος έχει ολοκληρωθεί. Ακολούθως αρχικοποιεί τις απαραίτητες μεταβλητές για τον επόμενο γύρο και μεταφέρει τις κάρτες του κάθε παίχτη στον επόμενο μέσω της cardsShouldMove.

```
if (turnIsCompleted)
{

    gameTurn++;
    print("Turn is completed, starting rount: "+ gameTurn);
    turnIsCompleted = false;
    ageCardsc2Center = null;
    playersCount = 0;
    players[0].GetComponent<PlayerScript>().initPlayer();
    cardsShouldMove();
    // change showcards firt card position
    pos = 15 - gameTurn;

}
```

Εικόνα 150: Ολοκλήρωση γύρου. Πηγή: Αλέξανδρος Βεκίλογλου

Για την μεταφορά των καρτών η συνάρτηση ενεργοποιεί την μεταβλητή isStackMoving και στη συνέχεια περνά από τους πίνακες καρτών όλων των παιχτών και ενεργοποιεί τη μεταβλητή hasToMove της κάθε κάρτας.

```
public void cardsShouldMove()
{
    isStackMoving = true;
    for (int i = 0; i < 4; i++)
    {
        PlayerScript play = players[i].GetComponent<PlayerScript>();
        for(int j = 0; j < 7; j++)
        {
            if (play.ageCards[j] != null)
                play.ageCards[j].GetComponent<ageCardScript>().hasToMove = true;
        }
    }
}
```

Εικόνα 151: Συνάρτηση cardsShouldMove. Πηγή: Αλέξανδρος Βεκίλογλου

Αυτό έχει σαν συνέπεια στην Update, εφόσον είναι αληθές η isStackMoving να εκτελείται η moveStack η οποία μεταφέρει σταδιακά όλες τις κάρτες στις νέες θέσεις τους.

```
void moveStack()
{
    var step = 45.0F * Time.deltaTime;
    PlayerScript play0 = players[playersCount].GetComponent<PlayerScript>();
    for (int i = 0; i < 7; i++)
    {
        if (play0.ageCards[i] != null)
        {
            ageCardScript ageScript = play0.ageCards[i].GetComponent<ageCardScript>();
            if (ageScript.hasToMove)
            {
                PlayerScript nextPlay;
                if ((playersCount + 1) > 3)
                {
                    nextPlay = players[0].GetComponent<PlayerScript>();
                    play0.ageCards[i].transform.position = Vector3.MoveTowards(play0.ageCards[i].transform.position, nextPlay.direction, step);
                }
                else
                {
                    nextPlay = players[(playersCount + 1)].GetComponent<PlayerScript>();
                    play0.ageCards[i].transform.position = Vector3.MoveTowards(play0.ageCards[i].transform.position, nextPlay.direction, step);
                }

                // if last player's last card is at final dest all cards are at final dest then  send cards to next players array.
                PlayerScript play3 = players[3].GetComponent<PlayerScript>();
                PlayerScript playA = players[0].GetComponent<PlayerScript>();

               // print("PlayA.direction: " + playA.direction);

                if (play3.ageCards[play3.lastCardIndex()].transform.position == playA.direction)
                {
                    print("Final card is at position");
                    ageScript.hasToMove = false;
                    isStackMoving = false;
                    passCards2Neght();

                  //  print("rotate each player's card");
                    for(int k=0;k<4;k++)
                        players[k].GetComponent<PlayerScript>().rotateStackCards();

                    // start new round
                    isStackMoved = true;
                    }


            }
        }
    }
    playersCount++;
    if (playersCount == 4)
        playersCount = 0;
```

Εικόνα 152: Συνάρτηση μεταφοράς στοίβας καρτών. Πηγή: Αλέξανδρος Βεκίλογλου.

Ακολούθως για η μεταφορά του πίνακα των καρτών του κάθε παίχτη στον επόμενο γίνεται μέσω της passCards2Neigh.

```
public void passCards2Neght()
{

    GameObject[] temp = new GameObject[7];
    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    PlayerScript play3 = players[3].GetComponent<PlayerScript>();
    PlayerScript play2 = players[2].GetComponent<PlayerScript>();
    PlayerScript play1 = players[1].GetComponent<PlayerScript>();
    temp = play0.ageCards;
    play0.ageCards = play3.ageCards;
    play3.ageCards = play2.ageCards;
    play2.ageCards = play1.ageCards;
    play1.ageCards = temp;


}
```

Εικόνα 153: Μεταφορά πίνακα καρτών στον επόμενο γείτονα. Πηγή: Αλέξανδρος Βεκίλογλου

Τέλος οι κάρτες πέρα από τη μετακίνηση τους θα πρέπει να περιστραφούν κατάλληλα ώστε η στοίβα του κάθε παίχτη να έχει τον κατάλληλο προσανατολισμό. Το πόσο θα πρέπει να περιστραφεί η κάθε

κάρτα είναι αποθηκευμένο σε κατάλληλη μεταβλητή στο Script του κάθε παίχτη και η περιστροφή γίνεται με τη συνάρτηση του κάθε παίχτη rotateStackCards.

```
public void rotateStackCards()
{
    for (int i = 0; i < 7; i++)
    {
        if (ageCards[i] != null)
        {
            ageCards[i].transform.eulerAngles = new Vector3(0, rotDegrEnd, 0);
        }
    }
}
```

Εικόνα 154: Περιστροφή της κάθε στοίβας καρτών του παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις περιστραφούν και οι κάρτες όλων των παιχτών η διαδικασία ολοκληρώνεται, συνεπώς θα πρέπει να ξεκινήσει ο νέος γύρος. Αυτό επιτυγχάνεται με την ανάθεση στη μεταβλητή isStackMoved σε αληθές πράγμα που ελέγχεται στην Update της terrainScript. Μετά τον έλεγχο της μεταβλητής αυτής ο κώδικας ελέγχει την τιμή της gameTurn η οποία κρατά πόσες κάρτες έχουν παιχτεί. Αν η τιμή της είναι μικρότερη του 6 η εφαρμογή απλώνει τις νέες κάρτες που έχει πάρει ο βασικός παίχτης ώστε να διαλέξει ποια θα παίξει και εκτελεί τη συνάρτηση που ολοκληρώνει τη διαδικασία παιξίματος των υπολοίπων παιχτών. Με τον τρόπο αυτό ένα νέος γύρος ξεκινά.

```
if(isStackMoved)
{
    isStackMoved = false;
    if (gameTurn < 6)
    {
        showCards2Player0();
        othersPick();
    }
}
```

Εικόνα 155: Ολοκλήρωση γύρου. Πηγή: Αλέξανδρος Βεκίλογλου

Διαφορετικά, αν ήδη έχουν παιχτεί 6 κάρτες τότε απομένει η στρατιωτική αναμέτρηση η οποία ολοκληρώνεται μέσω της findWinner.

```
if ((gameTurn == 6) && (millisHas2Move == false))
{
    for (int i = 0; i < 4; i++)
        players[i].GetComponent<PlayerScript>().printBuldedCards();
    print("Round is completed!!!");
    findWinner();
    gameTurn = 7;


}
```

Εικόνα 156: Ολοκλήρωση πίστας. Πηγή: Αλέξανδρος Βεκίλογλου

Για τον καθορισμό του νικητή η συνάρτηση αρχικά μετρά πόσους πόντους παρέχουν οι κάρτες που έχει κατασκευάσει ο κάθε παίχτης. Οι πόντοι αποθηκεύονται σε ένα πίνακα ώστε ο πρώτος παίχτης να είναι στη θέση 0, ο ένας γείτονας στη θέση 1 και ο άλλος στη θέση 2.

```
public void findWinner()
{
    int[] playersMilis = new int[3];
    int[] ties = new int[3];
    for (int i = 0; i < 3; i++)
        ties[i] = -1;

    int tieCount = 0;
    playersMilis[0] = countMiliTockens(players[0].GetComponent<PlayerScript>());
    playersMilis[1] = countMiliTockens(players[1].GetComponent<PlayerScript>());
    playersMilis[2] = countMiliTockens(players[3].GetComponent<PlayerScript>());
    PlayerScript play0 = players[0].GetComponent<PlayerScript>();
    PlayerScript play1 = players[1].GetComponent<PlayerScript>();
    PlayerScript play3 = players[3].GetComponent<PlayerScript>();


    int[] res0 = new int[2]; // 0 victory points, 1 defet points
    res0[0] = 0;
    res0[1] = 0;

    res0 = compearePlayers(play0, play1, res0);
    res0 = compearePlayers(play0, play3, res0);

    assigneMillis(play0, res0);
```

Εικόνα 157: Εντοπισμός νικητή ανάμεσα στους τρείς πρώτους παίχτης. Πηγή: Αλέξανδρος Βεκίλογλου

Η καταμέτρηση γίνεται μέσω της countMiliTockens.

```
public int countMiliTockens(PlayerScript play)
{
    int milisNum = 0;
    //foreach (GameObject card in play.cardsBuilded)
    for(int i = 0; i < 24; i++)
    {
        if (play.cardsBuilded[i] != null)
        {
            ageCardScript cardScr = play.cardsBuilded[i].GetComponent<ageCardScript>();
            milisNum += cardScr.miliStrength;

        }
    }
    return milisNum;
}
```

Εικόνα 158: Καταμέτρηση στρατιωτικών πόντων από τις κάρτες του παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

Ακολούθως η συνάρτηση ορίζει ένα πίνακα δύο ακεραίων όπου σε κάθε θέση θα αποθηκεύσει πόσες νίκες και πόσες ήττες έχει ο πρώτος παίχτης συγκρινόμενος με τους δύο γείτονες του. Η σύγκριση επιτυγχάνεται μέσω της compearePlayers. Σαν όρισμα λαμβάνει πρώτα τον πρώτο παίχτη που θα συγκρίνει, ακολούθως με ποιον θα τον συγκρίνει και τέλος σε ποιο πίνακα θα αποθηκεύσει το αποτέλεσμα.

```
public int[] compearePlayers(PlayerScript playA, PlayerScript playB, int[] res)
{
    if (countMiliTockens(playA) > countMiliTockens(playB))
    {
      res[0] += 1;
    }
    else if (countMiliTockens(playA) < countMiliTockens(playB))
        res[1] += 1;
    return res;
}
```

Εικόνα 159: Καθορισμός πόντων. Πηγή: Αλέξανδρος Βεκίλογλου

Πιο συγκεκριμένα η συνάρτηση συγκρίνει τους πόντους του παίχτη με τον ένα γείτονα. Αν είναι περισσότεροι αυξάνει κατά ένα τον αριθμό που βρίσκεται στη πρώτη θέση του πίνακα res. Αν είναι λιγότεροι αυξάνει κατά ένα τον αριθμό που βρίσκεται στη $2^{η}$ θέση του πίνακα. Στη συνέχεια μέσω της assigneMillis μεταφέρει τα tokens από τη στοίβα στον παίχτη. Πιο συγκεκριμένα λαμβάνει από τον πίνακα τον αριθμό των token που θα πρέπει να μεταφερθούν, τα μεταφέρει από τον αρχικό πίνακα που βρίσκονται στον πίνακα του παίχτη και στη συνέχεια τα μεταφέρει στην περιοχή της κάρτας θαύματος του κάθε παίχτη.

```
public void assigneMillis(PlayerScript player, int[] resultArr)
{
    int minusMillisCount = countMillis(miliMinus, 24);
    int[] coords = { -1, 0, 1 };
    // transfer miliMinus to player, remove it from reserve and make it move....
    for (int i = 0; i < resultArr[1]; i++)
    {
        player.miliMinus[i] = miliMinus[minusMillisCount - 1];
        miliMinus[minusMillisCount - 1] = null;
        minusMillisCount--;
        milliMinusScript miliScr = player.miliMinus[i].GetComponent<milliMinusScript>();
        miliScr.hasToMove = true;
        float xCord = player.milisArea.x;
        float yCord = player.milisArea.y;
        float zCord = player.milisArea.z;
        miliScr.direction = new Vector3(xCord + coords[Random.Range(0, 3)], yCord, zCord + coords[Random.Range(0, 3)]);
        print("One mili Minus is movig to Player: " + player.name);

    }
```

Εικόνα 160: Μεταφορά των token ήττας στον παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

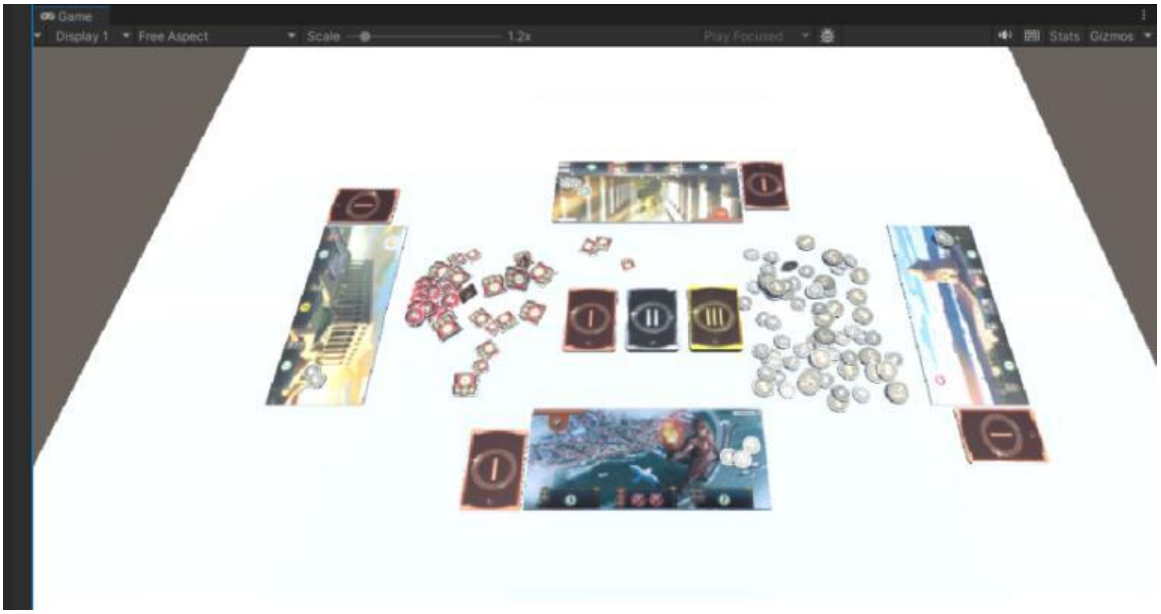
Η ίδια σύγκριση πραγματοποιείται για όλους τους παίχτες και πλέον ο συνολικός αριθμός των token νίκης και ήττας βρίσκεται, πέρα από την κάρτα θαύματος του κάθε παίχτη και στον πίνακα res του κάθε παίχτη. Έτσι η εφαρμογή μπορεί να γνωρίζει ποιος παίχτης επικράτησε ποιανού και να αποφασίσει για τον νικητή του γύρου.

6 ΚΕΦΑΛΑΙΟ 6ο : Παρουσίαση Εκτέλεσης

6.1 Πρώτη Περίοδος

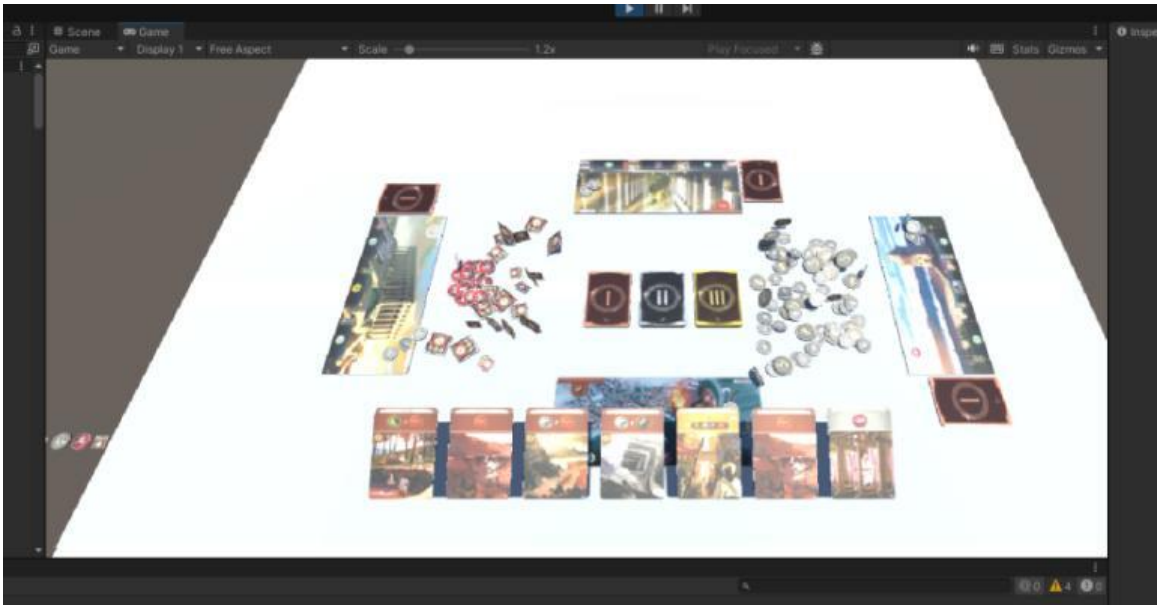
Η εφαρμογή με το που ξεκινά εμφανίζει το τραπέζι και πάνω του τις τέσσερεις κάρτες των θαυμάτων, μία για κάθε παίχτη μοιρασμένες αντίστοιχα στις τέσσερεις πλευρές του τραπεζιού. Ακολούθως, για να ενισχυθεί η αίσθηση της αληθοφάνειας πέφτουν πάνω στο τραπέζι τα κέρματα στη δεξιά πλευρά και στην αριστερή τα military tokens. Τόσο τα κέρματα αλλά και τα military tokens καταλαμβάνουν τυχαίες θέσεις εντός ορισμένης περιοχής και τυχαίο προσανατολισμό. Παράλληλα πέφτουν και οι κάρτες και στοιβάζονται στο μέσο του τραπεζιού και διανέμονται από τρία κέρματα του ενός σε κάθε παίχτη τα οποία αποθέτονται σε συγκεκριμένη περιοχή στην αντίστοιχη κάρτα θαύματος. Όμοια και τα κέρματα αυτά αποθέτονται σε τυχαίες θέσεις με τυχαίο προσανατολισμό.

Στη συνέχεια η εφαρμογή μοιράζει στους παίχτες τις κάρτες τους όπως θα γινόταν στο πραγματικό κόσμο. Ο κάθε παίχτης λαμβάνει κυκλικά από 7 κάρτες. Προκειμένου η εφαρμογή να έχει όσο το δυνατόν πιο αυξημένη την αίσθηση της αληθοφάνειας επιλέχθηκε οι κάρτες να μοιράζονται στους παίχτες όπως ακριβώς σε ένα πραγματικό παιχνίδι. Οι κάρτες που μοιράζονται στους παίχτες στοιβάζονται στην αριστερή πλευρά της κάθε κάρτας θαύματος.



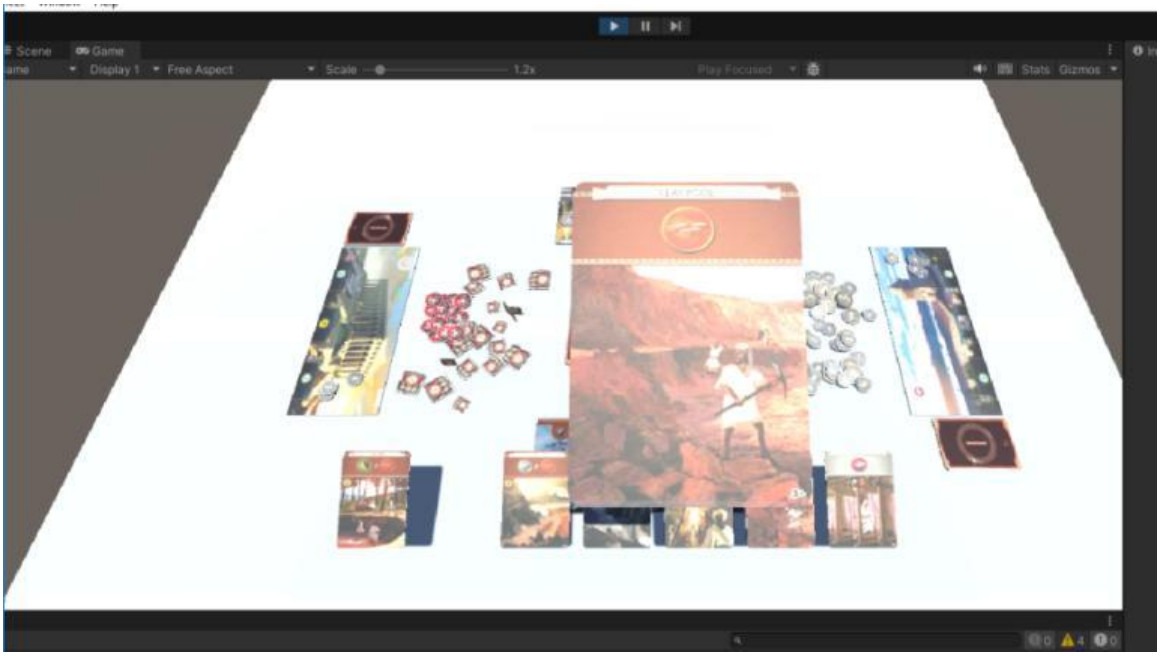
Εικόνα 161: Τοποθέτηση αντικειμένων στο τραπέζι. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις οι κάρτες μοιραστούν στους παίχτες παρουσιάζονται στο βασικό παίχτη ώστε να μπορεί να διαλέξει ποια κάρτα θα παίξει.



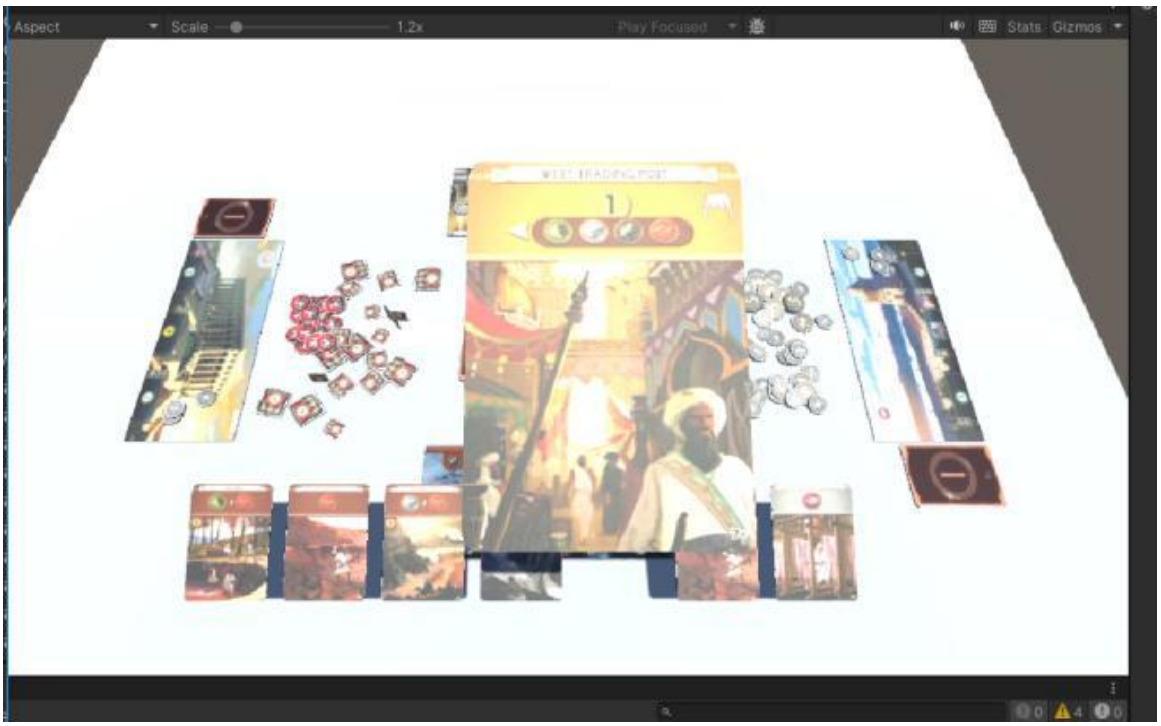
Εικόνα 162: Παρουσίαση των καρτών του βασικού παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

Στη συγκεκριμένη στιγμή οι κάρτες δεν είναι ιδιαίτερα ευκρινής και για το σκοπό αυτό όταν ο χρήστης τοποθετεί το ποντίκι πάνω από μία κάρτα αυτή εμφανίζεται μεγαλύτερη στο κέντρο της οθόνης.



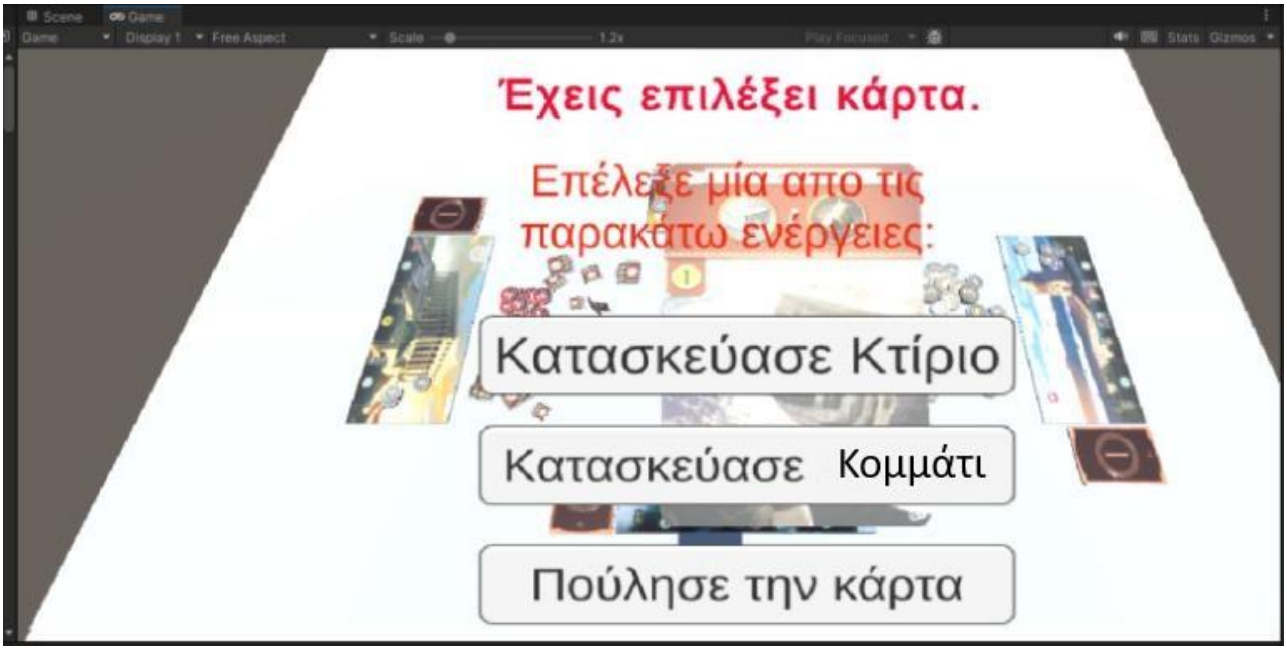
Εικόνα 163: Προβολή της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Αν ο χρήστης δεν είναι ικανοποιημένος με την κάρτα αυτή μπορεί να τοποθετήσει το ποντίκι σε μία άλλη και αυτή να πάρει τη θέση της.



Εικόνα 164: Εναλλαγή καρτών. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις αποφασίσει πια κάρτα θα χρησιμοποιήσει πατάει το αριστερό πλήκτρο που ποντικιού όπου η εφαρμογή του εμφανίζει ένα διάφανο παράθυρο. Μέσω αυτού τον ενημερώνει ότι διάλεξε κάρτα και ότι έχει πλέον τρείς επιλογές: Α) Να κατασκευάσει κτίριο, Β) Να κατασκευάσει κομμάτι του θαύματος και τέλος Γ) Να πουλήσει την κάρτα.

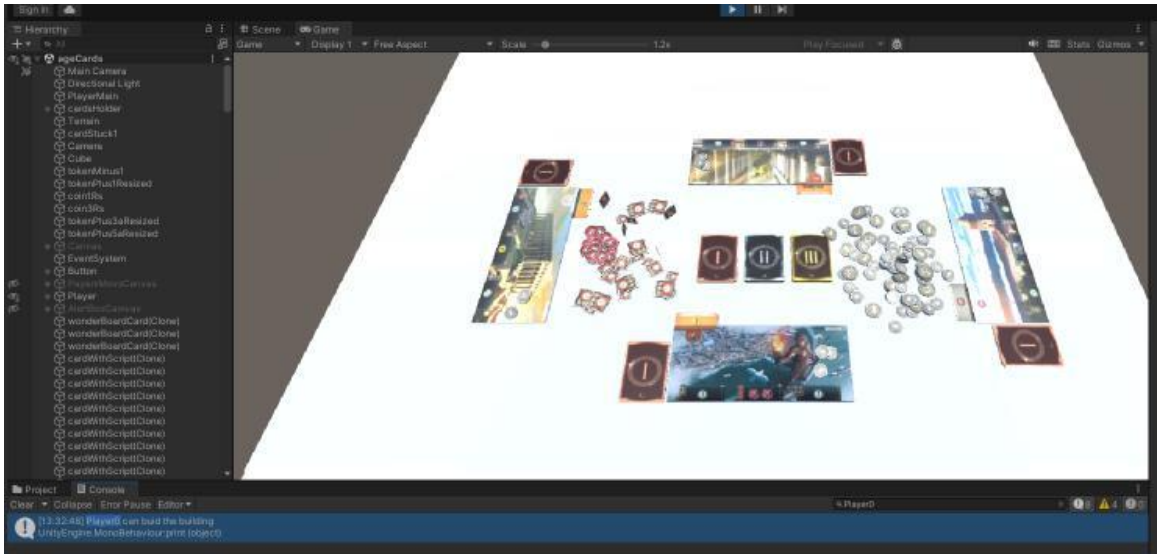


Εικόνα 165: Παρουσίαση του μενού των επιλογών. Πηγή: Αλέξανδρος Βεκίλογλου

6.1.1 Κατασκευή κτιρίου

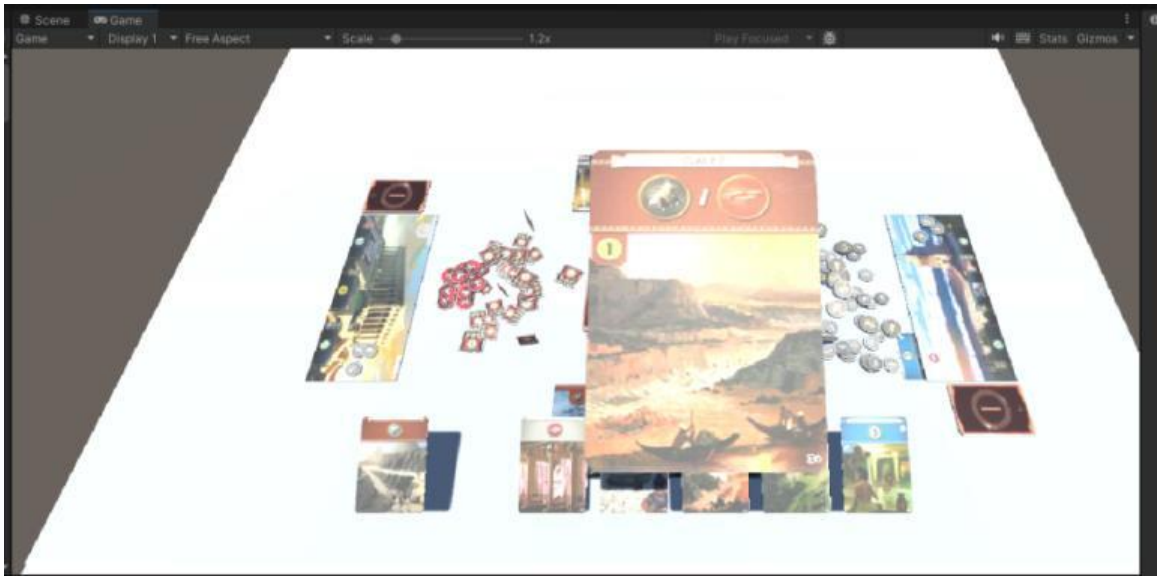
Η πρώτη επιλογή που έχει ο παίχτης είναι να κατασκευάσει το κτίριο. Για να πετύχει αυτό θα πρέπει να εξασφαλίσει ότι μπορεί να το κάνει. Προκειμένου να κτιστεί ένα κτίριο υπάρχουν τρείς διαφορετικές περιπτώσεις: Α) Το κτίριο δεν απαιτεί κάποιο πόρο για να χτιστεί Β) απαιτεί την καταβολή ενός κέρματος για να χτιστεί Γ) απαιτεί κάποιον ή κάποιους πόρους για να κτιστεί.

Στη περίπτωση που δεν απαιτεί κάποιο πόρο για να κτιστεί τότε πατώντας το κουμπί 'Κατασκεύασε Κτίριο' η εφαρμογή θα τοποθετήσει την κάρτα στη καθορισμένη θέση και θα στοιβάξει εκ νέου τις κάρτες στην αρχική τους θέση αριστερά της κάρτας θαύματος ώστε να είναι έτοιμες να περαστούν στον επόμενο παίχτη για τον επόμενο γύρο. Παράλληλα και οι υπόλοιποι παίχτες θα παίξουν τις κάρτες που τους έχουν ανατεθεί με τυχαίο όμως τρόπο. Η εφαρμογή ενημερώνει στις εγγραφές συμβάντων ότι η κάρτα του Player0 που είναι ο πρώτος παίχτης είναι δωρεάν και μπορεί να χτιστεί και προσθέτει τους πόρους που ενδεχομένως η κάρτα προσφέρει στους πόρους που ο χρήστης μπορεί να χρησιμοποιήσει.

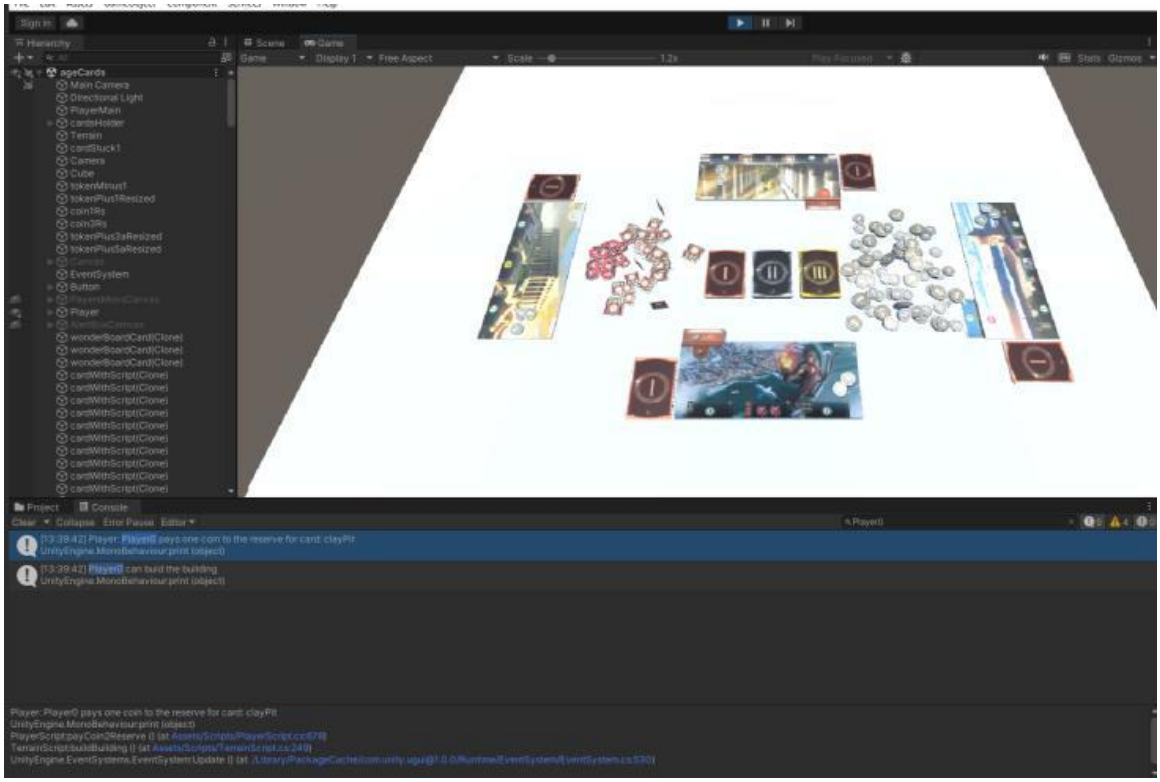


Εικόνα 166: Χτίσιμο δωρεάν κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Στη περίπτωση που η κάρτα απαιτεί νόμισμα για να κτιστεί τότε εφόσον ο παίχτης διαθέτει αρκετά χρήματα μετακινεί ένα νόμισμα από τη στοίβα του και το τοποθετεί στη στοίβα του τραπεζιού. Στη συνέχεια τοποθετεί την κάρτα στη κατάλληλη θέση και ενημερώνει τους πόρους του χρήστη.



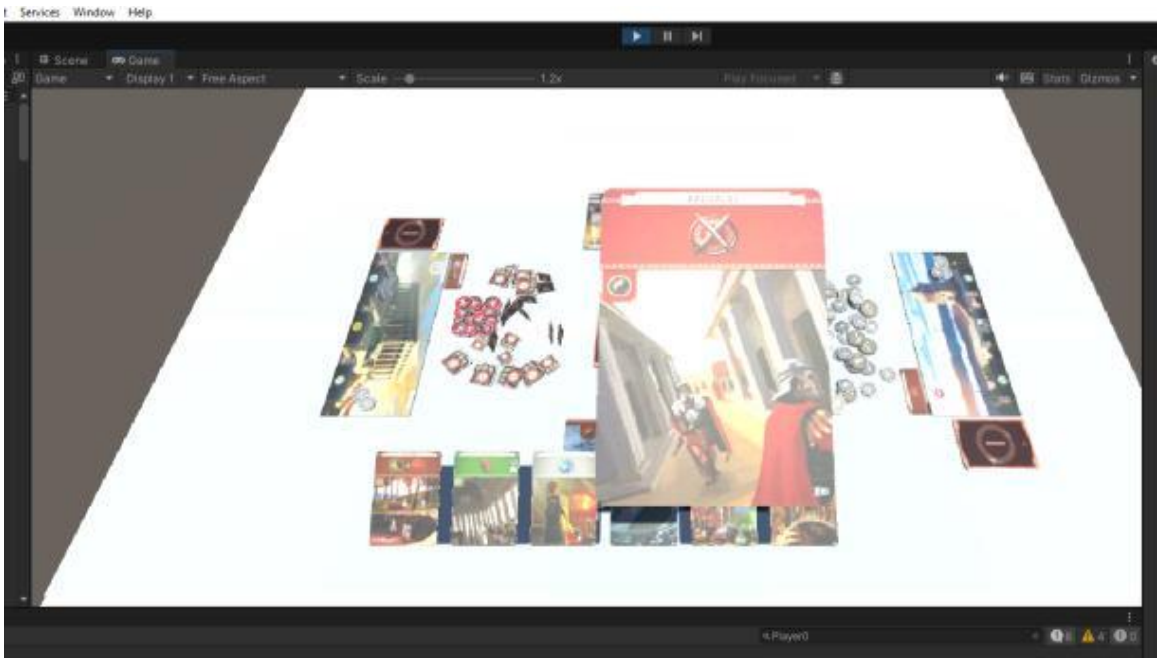
Εικόνα 167: Επιλογή κάρτας με κόστος ενός νομίσματος. Πηγή: Αλέξανδρος Βεκίλογλου



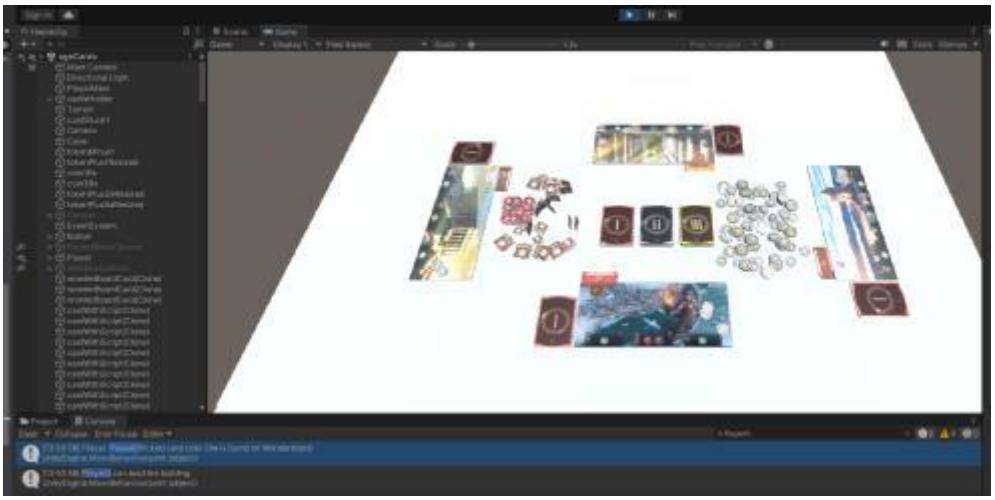
Εικόνα 168: Καταβολή χρηματικού κόστους για την χρήση της κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Η επόμενη περίπτωση που θα παρουσιαστεί είναι αυτή όπου η κάρτα απαιτεί κάποιο πόρο. Στη περίπτωση αυτή υπάρχουν δύο διαθέσιμες επιλογές: Α) Ο απαιτούμενος πόρος ή οι πόροι μπορούν να καλυφθούν από τους πόρους που διαθέτει ο παίχτης, είτε από τη κάρτα θαύματος είτε από τις κάρτες που έχει ήδη χτίσει, Β) είτε να διατίθενται από τους γείτονες του από τους οποίους μπορεί να τους αγοράσει εφόσον διαθέτει το απαραίτητο υπόλοιπο.

Για το σκοπό αυτό η εφαρμογή αρχικά συγκρίνει τους πόρους που απαιτεί η κάρτα με τους πόρους που διαθέτει ο ίδιος ο παίχτης. Αν οι πόροι μπορούν να καλυφθούν στο σύνολο τους τότε η κάρτα χτίζεται.

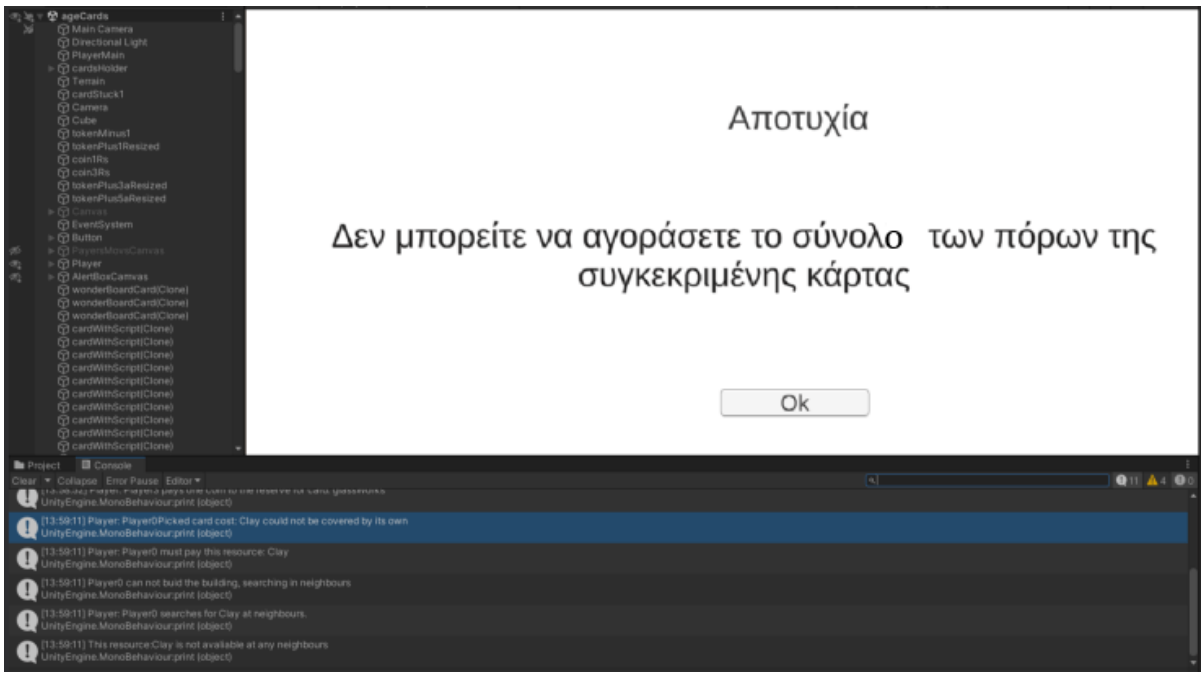


Εικόνα 169: Επιλογή κάρτας που απαιτεί πόρο που ήδη διαθέτει ο παίχτης από την κάρτα θαύματος του. Πηγή: Αλέξανδρος Βεκίλογλου



Εικόνα 170: Χτίσιμο κάρτας χρησιμοποιώντας τον πόρο που προσφέρει η κάρτα θαύματος. Πηγή: Αλέξανδρος Βεκίλογλου

Στη περίπτωση που ο πόρος δεν μπορεί να καλυφθεί από τους ιδίους τότε η εφαρμογή τους αναζητά στους πόρους που παρέχουν οι γείτονες. Συγκεκριμένα για τον παίχτη Player0 οι γείτονες του είναι ο παίχτης αριστερά του (Player1) και ο παίχτης δεξιά του (Player3). Συνεπώς στους παίχτες αυτού η εφαρμογή αναζητά τους πόρους. Αν ο πόρος δεν εντοπιστεί σε κανέναν από τους δύο ενημερώνει τον παίχτη.



Εικόνα 171: Αποτυχία εντοπισμού πόρου σε γείτονες. Πηγή: Αλέξανδρος Βεκίλογλου

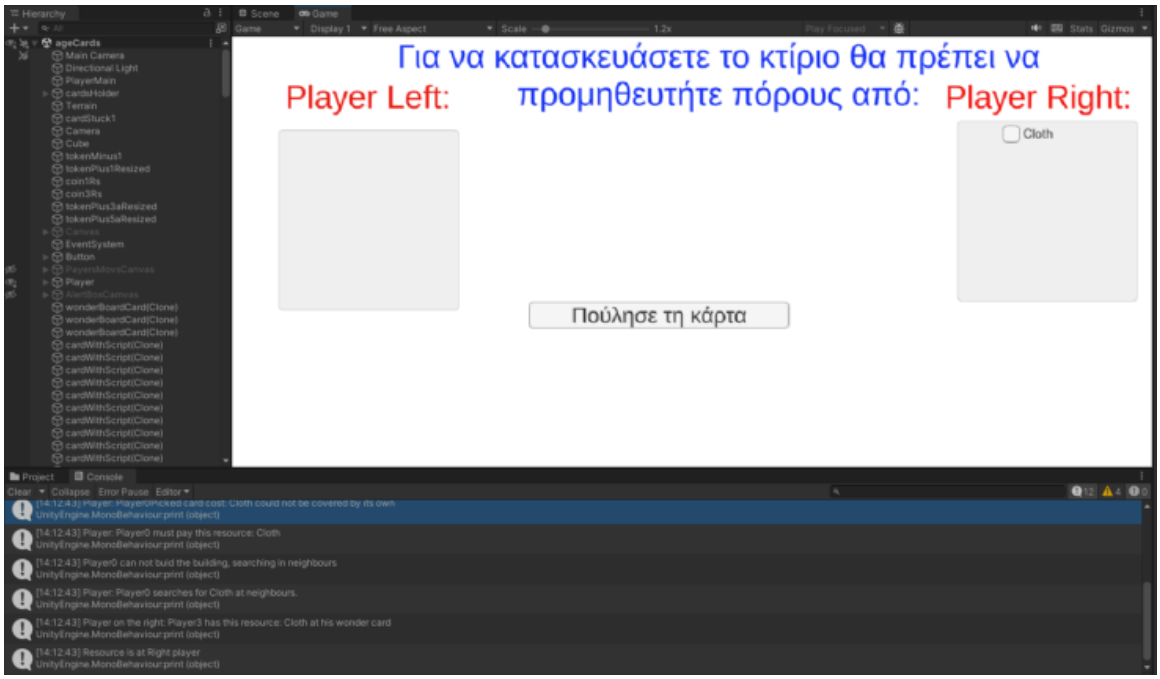
Μόλις ο χρήστης πατήσει το κουμπί 'Ok' το παράθυρο εξαφανίζεται και επανέρχεται στη προηγούμενη κατάσταση με την επιλεγμένη κάρτα στο κέντρο της οθόνης. Πλέον ο χρήστης μπορεί να επιλέξει μια άλλη κάρτα από τις διαθέσιμες. Φυσικά ο χρήστης μπορεί να επιλέξει και πάλι την ίδια κάρτα και μέσω του παραθύρου επιλογών αντί να επιλέξει το κτίσιμο της κάρτας να επιλέξει το κτίσιμο θαύματος ή την πώληση αυτής.

Εικόνα 172: Επιστροφή στη προηγούμενη κατάσταση επιλογής κάρτας. Πηγή: Αλέξανδρος Βεκίλογλου

Στην επόμενη εικόνα παρουσιάζεται ως επιλεγμένη μία κάρτα της οποίας η κατασκευή απαιτεί τον πόρο 'Πανί' (Cloth) ο οποίος διατίθεται από την κάρτα θαύματος του παίχτη στα αριστερά (Player3) του βασικού παίχτη.

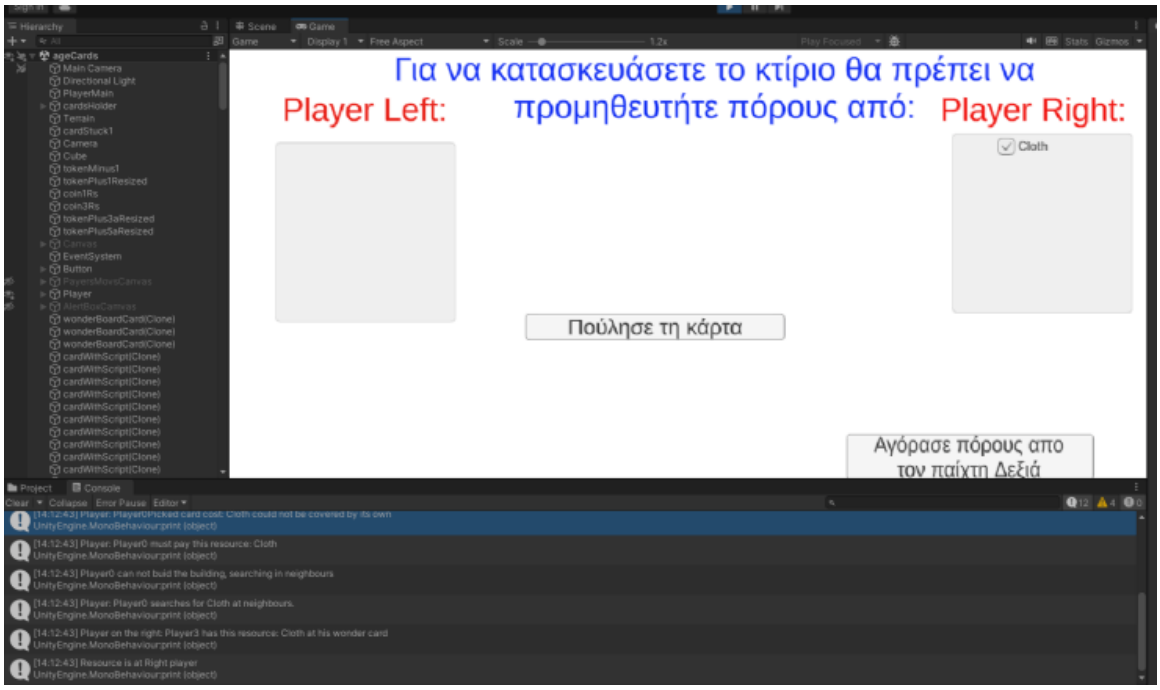
Εικόνα 173: Επιλογή κάρτας που απαιτεί πόρο ο οποίος διατίθεται από γειτονικό παίχτη. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις η εφαρμογή εντοπίσει τον ζητούμενο πόρο όχι στον ίδιο παίχτη αλλά στους γειτονικούς εμφανίζει ένα παράθυρο στο οποίο ενημερώνει τον παίχτη ποιους πόρους κατέχει ποιος γείτονας. Ο κάθε πόρος εμφανίζεται με τη μορφή του 'Check box' το οποία αρχικά δεν είναι επιλεγμένο και αν ο χρήστης επιθυμεί να αγοράσει τον πόρο απλά επιλέγει το κατάλληλο κουτί. Η προσέγγιση αυτή επιλέχθηκε ως καταλληλότερη διότι δίνει τη δυνατότητα στο παίχτη να επιλέξει ποιον πόρο (ή ποιούς πόρους) θα αγοράσει από ποιον γείτονα μιας και ο ίδιος πόρος μπορεί να διατίθεται σε περισσότερους γείτονες του ενός. Με τον τρόπο ο παίχτης έχει τη δυνατότητα, να παράδειγμα να αγοράσει τον πόρο από τον αδύναμο παίχτη και να μην ενισχύσει τον ισχυρότερο ο οποίος μπορεί στη συνέχεια να τον κερδίσει στη μάχη. Φυσικά του δίνεται και η δυνατότητα να πουλήσει την κάρτα αν δεν επιθυμεί να ενισχύσει τον κάτοχο του πόρου.



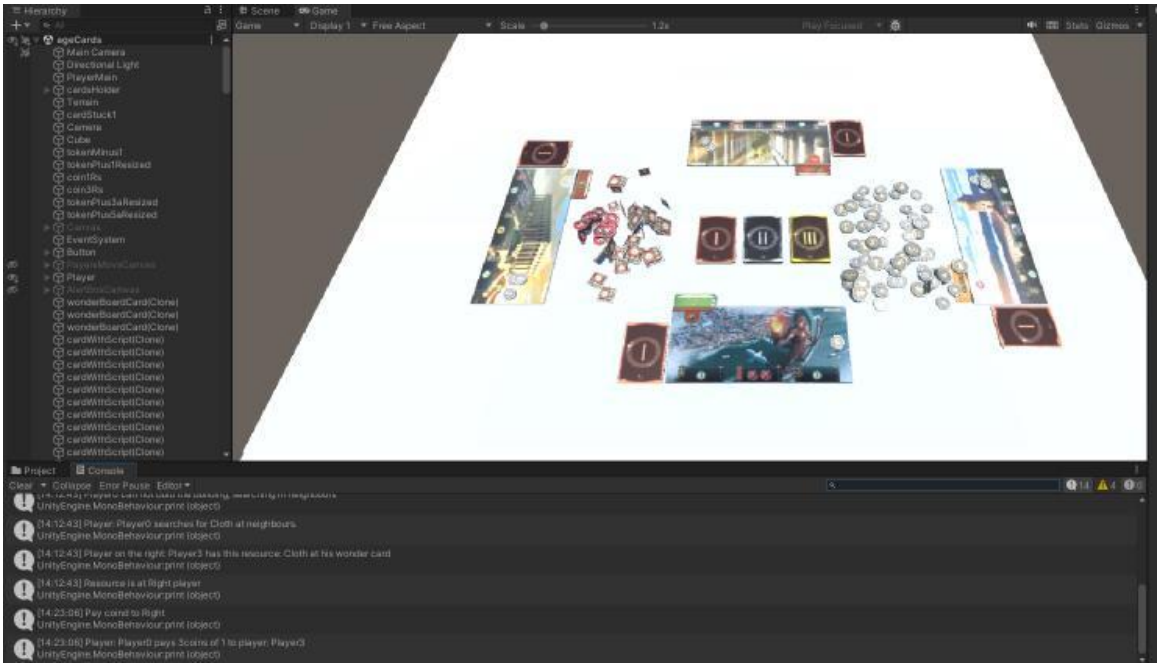
Εικόνα 174: Επιλογή πόρων από τους γείτονες. Πηγή: Αλέξανδρος Βεκίλογλου

Μόλις ο παίχτης επιλέξει πόρο ή πόρους κάποιου γείτονα εμφανίζεται το κουμπί ‘Αγόρασε πόρους από το παίχτη’. Στην επόμενη εικόνα εμφανίζεται ο πόρος Cloth που διαθέτει ο παίχτης στα δεξιά και το ενεργοποιημένο κουμπί αγοράς.



Εικόνα 175: Επιλογή πόρου προς αγορά. Πηγή: Αλέξανδρος Βεκίλογλου

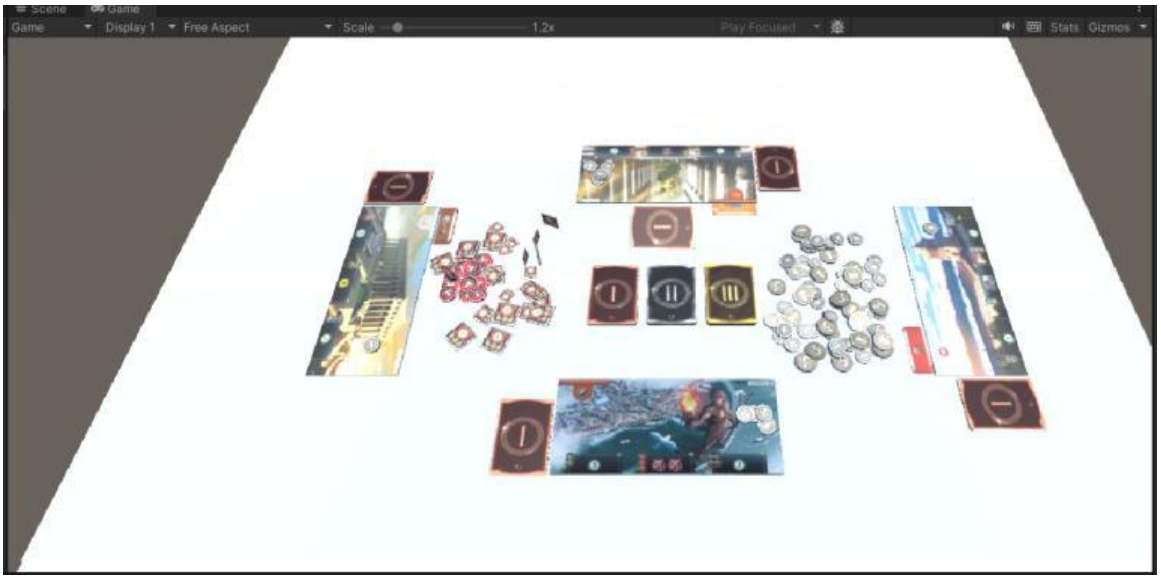
Μόλις πατηθεί το κουμπί ‘Αγόρασε πόρους...’ η εφαρμογή ελέγχει προκειμένου να διαπιστώσει αν ο παίχτης διαθέτει το απαιτούμενο χρηματικό ποσό. Αν το διαθέτει μεταφέρει τον απαιτούμενο αριθμό κερμάτων στον αντίστοιχο γείτονα και χτίζει την κάρτα.



Εικόνα 176: Αγορά των επιλεγμένων πόρων από τους γείτονες. Πηγή: Αλέξανδρος Βεκίλογλου

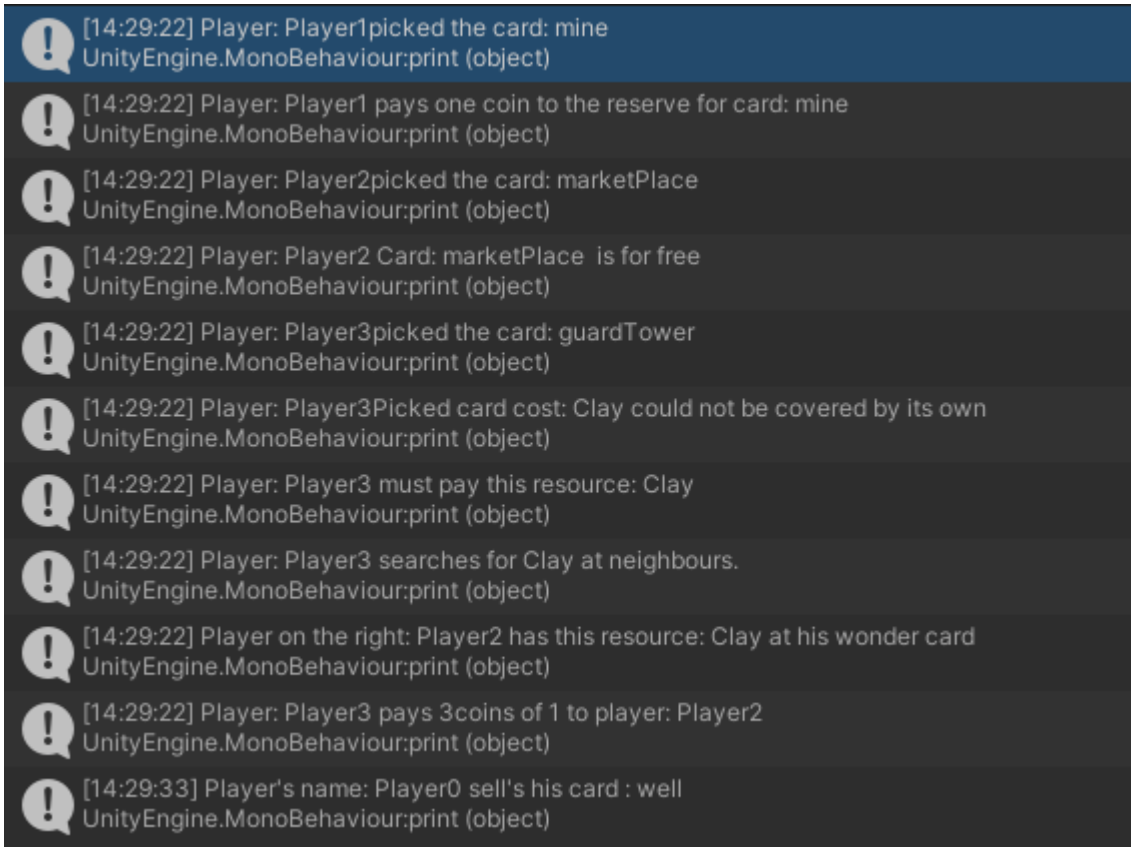
Η τελευταία επιλογή που έχει ο παίχτης είναι να πουλήσει την κάρτα. Πατώντας του κουμπί πώλησης είτε από το αρχικό παράθυρο επιλογών είτε από το παράθυρο επιλογής πόρων από τους γείτονες, η

εφαρμογή θα τοποθετήσει την κάρτα στο σημείο που τοποθετούνται οι κάρτες που πουλούν οι παίχτες με όψη προς τα κάτω και θα μεταφέρει δύο κέρματα από τη στοίβα στη περιοχή των κερμάτων του παίχτη.



Εικόνα 177: Πώληση κάρτας και αποταμίευση κερμάτων. Πηγή: Αλέξανδρος Βεκίλογλου

Οι υπόλοιποι παίχτες εκτελούν την ίδια διαδικασία αυτόματα. Πιο συγκεκριμένα αν η κάρτα που τους έχει ανατεθεί με τυχαίο τρόπο χτίζεται χωρίς κόστος τότε ο παίχτης τη χτίζει. Αν απαιτεί χρήματα τότε ο παίχτης καταβάλει το κέρμα αν το διαθέτει. Διαφορετικά προβαίνει στην πώληση της κάρτας. Αν η κάρτα απαιτεί πόρους τότε η εφαρμογή τους αναζητά στους διαθέσιμους. Αν δεν τους εντοπίσει τους αναζητά στους γείτονες. Αν τους εντοπίσει τους αγοράζει από τον πρώτο γείτονα στον οποίο εντοπίζει αρχίζοντας από τον γείτονα στα δεξιά. Αν ο πόρος δεν διατίθεται τότε προβαίνει στην πώληση της κάρτας.



Εικόνα 178: Παράδειγμα εκτέλεσης της εφαρμογής. Πηγή: Αλέξανδρος Βεκίλογλου

7 ΚΕΦΑΛΑΙΟ 7ο : Επίλογος

Στόχο της παρούσας διπλωματικής εργασίας ήταν η σχεδίαση και ανάπτυξη σε τρισδιάστατο περιβάλλον μίας ψηφιακής εκδοχής υπαρκτού επιτραπέζιου (μη ψηφιακού) παιχνιδιού, εκπαιδευτικού χαρακτήρα.

Για το σκοπό αυτό στο πρώτο κεφάλαιο επιχειρήθηκε μια εισαγωγή στο θέμα παρουσιάζοντας τις βασικές έννοιες που διαπραγματεύονται στην εργασία όπως το παιχνίδι, οι κατηγορίες του, τα οφέλη της εκπαιδευτικής διαδικασίας μέσα από το παιχνίδι καθώς και άλλες απαραίτητες έννοιες. Σαν μελέτη περίπτωσης αποφασίστηκε να μελετηθεί το δημοφιλές παιχνίδι στρατηγικής 7 Wonders το οποίο παρουσιάζεται εκτενώς στο 2ο κεφάλαιο της εργασίας.

Για την κατασκευή της εφαρμογής χρησιμοποιήθηκε μία από τις πλέον επικρατέστερες μηχανές παιχνιδιών στο χώρο, η Unity 3D, η οποία παρουσιάζεται στο 3ο κεφάλαιο. Η έκταση και οι δυνατότητες της μηχανής αυτής είναι τεράστιες, θα μπορούσε κανείς να εντοπίσει πληθώρα βιβλίων που περιγράφουν τους βασικούς μηχανισμούς της και για το λόγο αυτό στην παρούσα εργασία επιλέχθηκαν βασικά τμήματά της να παρουσιαστούν.

Στη συνέχεια παρουσιάστηκε η αρχιτεκτονική του κώδικα της εφαρμογής. Στόχος ήταν να αποκτήσει ο αναγνώστης μια σφαιρική εικόνα του κώδικα της εφαρμογής μέσα από τα διάφορα διαγράμματα ώστε να μπορεί να ακολουθήσει την πλήρη παρουσίαση που ακολουθεί στο επόμενο κεφάλαιο. Αυτό είναι και το μεγαλύτερο κεφάλαιο της εργασίας καθώς εδώ αναλυτικά παρουσιάζονται τα βήματα για την ανάπτυξη της εφαρμογής. Πιο συγκεκριμένα, παρουσιάζεται αναλυτικά, τόσο η δημιουργία των διαφόρων τρισδιάστατων αντικειμένων στην εφαρμογή Blender, η εισαγωγή τους στη Unity, καθώς και όλος ο απαραίτητος κώδικας για τη λειτουργία της εφαρμογής.

Τέλος, στο επόμενο 6ο κεφάλαιο πραγματοποιείται μια παρουσίαση της λειτουργίας της εφαρμογής.

Αρκετές ήταν οι προκλήσεις που αντιμετωπίσθηκαν κατά την ανάπτυξη της εφαρμογής. Η Unity, μπορεί να είναι μία από τις πλέον επικρατέστερες μηχανές παιχνιδιών στο χώρο, για τη δημιουργία της εφαρμογής όμως δεν αρκούν μόνο οι γνώσεις πάνω στο προγραμματισμό. Απαιτούνται επιπλέον λογισμικά όπως επεξεργασίας εικόνας (Adobe Photoshop) για την δημιουργία των εικόνων που χρησιμοποιήθηκαν στα μοντέλα αλλά και το Blender, όπως παρουσιάστηκε στη προηγούμενη παράγραφο, για την κατασκευή των διαφόρων μοντέλων (κάρτες, tokens, κλπ.). Αλλά και στη Unity αυτή καθ' εαυτή υπήρξαν εκτεταμένα τμήματα κώδικα τα οποία γράφτηκαν, αποσφαλματώθηκαν, ελέγχθηκαν ώστε να λειτουργούν σωστά και στη συνέχεια πετάχτηκαν και ξαναγράφτηκαν από την αρχή καθώς δεν οδηγούσαν στο επιθυμητό αποτέλεσμα. Από την άλλη πλευρά οι μηχανισμοί που διαθέτει η μηχανή έκαναν εφικτή την κατασκευή της εφαρμογής και χωρίς αυτούς τίποτα δεν θα ήταν εφικτό. Πιο συγκεκριμένα ο αντικειμενοστραφής προγραμματισμός που η C# διαθέτει και η Unity χρησιμοποιεί επέτρεψε την δημιουργία αντικειμένων για όλες τις οντότητες της εφαρμογής (κάρτες, κόστη, πόρους, κλπ.) πράγμα που απλοποίησε σημαντικά τη διαδικασία και κατέστησε εφικτή την κατασκευή της εφαρμογής.

Τέλος, στις προτάσεις για μελλοντικές επεκτάσεις της ανάπτυξης, προσθήκες, ή και έρευνες, αρκετές είναι οι ιδέες που θα μπορούσαν να αξιοποιηθούν.

Στον εκπαιδευτικό άξονα, θα άξιζε να οργανωθεί μία δοκιμή με κοινό (παίχτες) ώστε να διερευνηθεί στην πράξη η λειτουργικότητα, σαφήνεια και 'παικτικότητα' (playability) της ψηφιακής εκδοχής του παιχνιδιού, αλλά επίσης να αξιολογηθεί / μετρηθεί και ο εκπαιδευτικός χαρακτήρας του παιχνιδιού. Αν και αυτό δεν κατέστη εφικτό μέσα στο χρονικό πλαίσιο της παρούσας διπλωματικής, αποτελεί ωστόσο έναν εφικτό στόχο που θα δώσει στους μελλοντικούς developers πολύτιμη ανατροφοδότηση για την βελτίωση της εφαρμογής.

Στον τεχνικό άξονα, ενδεικτικές κατευθύνσεις περαιτέρω ανάπτυξης είναι

- η ανάπτυξη των υπολοίπων δύο φάσεων του παιχνιδιού (εδώ για πρακτικούς λόγους αναπτύχθηκε η $1^{η}$ φάση μόνο),
- η υλοποίηση της κατασκευής κομματιού του κάθε θαύματος (τα οποία επίσης δεν αναπτύχθηκαν στην παρούσα διπλωματική),
- η βελτίωση των γραφικών,
- η υποστήριξη επιπλέον παιχτών (το παιχνίδι παίζεται με 3-7 παίχτες) μέσω δικτύου,

καθώς και άλλα πολλά. Άλλωστε, όπως συνήθως συμβαίνει με τα έργα την πληροφορικής, η φαντασία αποτελεί το όριο.

8 Βιβλιογραφία

Aksakal, N. (2015, May 31). Theoretical View to The Approach of The Edutainment. *sciencedirect*, σσ. 1232-1239.

Esposito, N. (2005). A Short and Simple Definition of What a Videogame Is. *Digital Games Research Conference .* Vancouver, British Columbia, Canada: dblp.

Feldman, R. (2010). *DEVELOPMENT ACROSS THE LIFE SPAN.* Αγγλία: Pearson.

Freud, S. (1955). *Beyond the Pleasure Principle.* London: New York: Dover Publications.

Frost, J. L. (2009). *A History of Children's Play and Play Environments.* New York: Taylorfrancis.

Fullerton, T. (2008). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games.* San Francisco: Morgan Kaufmann.

Gambacorta C, N. M. (2018, Jul). An action video game for the treatment of amblyopia in children: A feasibility study. *Vision Res*, σσ. 1-14.

Kamisah Osman, A.-N. L. (2018). *Encyclopedia of Information Science and Technology.* Hershey: IGI Global.

Kiryakova, G. &. (2014). GAMIFICATION IN EDUCATION. *9th International Balkan Education and Science Conference.* Edirne.

Kuo-Ting Huang, C. B. (2019, Φεβρουαρίου 19). Augmented Versus Virtual Reality in Education: An Exploratory Study Examining Science Knowledge Retention When Using Augmented Reality/Virtual Reality Mobile Applications. *International Journal of Advanced Research in Computer Engineering*, σσ. 1947-1952.

Prensky, M. (2007). *Digital Game-Based Learning.* Αγγλία: Paragon House.

Renu Mary Daniel, E. B. (2016). Deriving Practical Applicability of Hierarchical Identity Based Encryption in Massively Multiplayer Online Role Playing Games. *Procedia Computer Science*, σσ. 839-846.

Richard N. Van Eck, V. J. (2017). Leveling Up: Game Design Research and Practice for Instructional Designers. *Pearson Education*, σσ. 227-285.

Sebastian Deterding, D. D. (2011). From Game Design Elements to Gamefulness: Defining Gamification. *Envisioning Future Media Environments* (σσ. 9-15). Tampere: International Academic MindTrek Conference.

Wikipedia. (2024, Ιανουάριος 23). Ανάκτηση από Παιχνίδι (δραστηριότητα): https://el.wikipedia.org/wiki/%CE%A0%CE%B1%CE%B9%CF%87%CE%BD%CE%AF%CE%B4%CE%B9_(%CE%B4%CF%81%CE%B1%CF%83%CF%84%CE%B7%CF%81%CE%B9%CF%8C%CF%84%CE%B7%CF%84%CE%B1)

Winnicott, D. (1971). *Playing and Reality.* London: Tavistock.

Αντωνιάδης, Α. (1994). *ΤΟ ΠΑΙΧΝΙΔΙ.* Αθήνα: University Studio Press.

Κάππας, Χ. (2005). *Ο ΡΟΛΟΣ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ ΣΤΗΝ ΠΑΙΔΙΚΗ ΗΛΙΚΙΑ.* Αθήνα: ΑΤΡΑΠΟΣ.