Ατομική Διπλωματική Εργασία

SOMETHING ANCIENT IS BAKED

Αντρέας Παντελή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάρτιος 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Something Ancient is Backed

Αντρέας Παντελή

Επιβλέπων Καθηγητής
Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάρτιος 2019

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου για όλα αυτά που μάθανε και την υποστήριξη που μου έχουν δώσει. Επίσης, θα ήθελα να ευχαριστήσω τους φίλους μου Κωνσταντίνα Ανδρέου, Αλίκη Γεωργίου και Παύλος Γαβριηλίδης για την βοήθεια που μου δώσαν για την τελειοποίηση του project. Και τον Ανδρέα Αριστείδου που με βοήθησε για την εισαγωγή κινήσεων στο χαρακτήρα μου.

Περίληψη

Το Something Ancient is Backed είναι ένα παιχνίδι μαγειρικής που στόχος του είναι ο χρήστης να μάθει διάφορες συνταγές, από ποια υλικά αποτελείται η κάθε μια και μέσω του παιχνιδιού να αναπτύξει τις μαγειρικές του ικανότητες και στην πραγματική ζωή.

Το παιχνίδι αυτό ανήκει στην κατηγορία των «Παιχνιδιών Εκμάθησης» και διασκέδασης.

Είναι χτισμένο σε τρισδιάστατο περιβάλλον (3D Environment) του προγράμματος ανάπτυξης παιχνιδιών Unity 3D, είναι φτιαγμένο για windows form χρησιμοποιώντας τη γλώσσα προγραμματισμού C# μέσω του προγράμματος Visual Studio 2017.

Στο Something Ancient is Backed ο χρήστης θα πρέπει να καθοδηγεί τον εικονικό μάγειρα που υπάρχει στο παιχνίδι και με τη χρήση του ποντικιού θα επιλέγει τα υλικά που χρειάζεται για να εκτελέσει τη συνταγή που έχει επιλεχθεί. Κατά τη διάρκεια του παιχνιδιού όλα τα αντικείμενα που θα έχει επιλέξει ο χρήσης θα μπαίνουν στο inventory και από εκεί θα επιλέγονται ξανά για να εκτελεστεί η συνταγή.

Περιεχόμενα

Κεφάλαιο 1 Εισαγωγή..... **1**

1.1 Τι είναι τα εκπαιδευτικά παιχνίδια; 1

1.2 Εκπαιδευτικά παιχνίδια και Ψηφιακή Τεχνολογία 2

1.3 Χαρακτηριστικά των ψηφιακών και εκπαιδευτικών παιχνιδιών 2

1.4 Εκπαιδευτικά παιχνίδια μαγειρικής 3

1.4.1 Κίνητρο για τη δημιουργία ενός μαγειρικού παιχνιδιού 4

Κεφάλαιο 2 Περιγραφή Παιχνιδιού.....**5**

2.1 Περιγραφή του παιχνιδιού 5

2.2 Σκοπός του παιχνιδιού 6

2.3 Τρόποι Αξιολόγησης 6

2.4 Δυσκολίες παιχνιδιού 7

2.5 Γιατί επέλεξα το “Something Ancient is Baked”; 7

2.6 Ηλικιακή Κλίμακα Χρηστών 8

Κεφάλαιο 3 Σφαιρική εικόνα του παιχνιδιού..... **9**

3.1 Προγραμματιστικό περιβάλλον του παιχνιδιού 9

3.2 Συστατικά παιχνιδιού 10

3.2.1 Assets 10

3.2.1.1 Ο χαρακτήρας (Chef) 10

3.2.1.2 Η κουζίνα 11

3.2.1.3 Τα υλικά 12

3.2.2 Sound Effect και Animation 13

3.2.3 Scores 13

3.3 Πως λειτουργά το παιχνίδι; 14

3.3.1 Τα δύο στάδια 16

3.4 User Interface (UI) 17

Κεφάλαιο 4 Τεχνικό κομμάτι..... **20**

4.1 Συνθήκες 20

4.2 Αλληλεπίδραση 21

4.3 Αντίδραση 21
4.3.1 Ηχητική αντίδραση 21
4.3.2 Κινητική αντίδραση 21
4.3.3 Προσθήκη στο Inventory 22
4.3.4 Αφαίρεση από Inventory 22
4.3.5 Game Object Reaction 23
4.3.6 Αλλαγή συνθήκης 23
4.4 Επιλογή υλικών 23
4.5 Βοηθητικά κουμπιά του χαρακτήρα 24
4.6 Inventory buttons 25
4.7 Αντικείμενα 25
4.8 Διάγραμμα του Animation 26
4.9 Τα υπόλοιπα κουμπιά 27
4.10 Game manager Scripts 28

Κεφάλαιο 5 Εγχειρίδιο χρήστη..... **29**
5.1 Μενού 29
5.2 Πρώτο Στάδιο 29
5.3 Δεύτερο Στάδιο 31

Κεφάλαιο 6 Συμπεράσματα..... **33**
6.1 Μεταστατικά Συμπεράσματα 33
6.2 Αποτελέσματα από τη δοκιμή του παιχνιδιού 33
6.3 Μελλοντική Εργασία 34

Βιβλιογραφία **35**

Παράστημα Α..... **Α-1**
Παράστημα Β..... **Β-1**
Παράστημα Γ..... **Γ-1**
Παράστημα Δ..... **Δ-1**

Κεφάλαιο 1

Εισαγωγή

1.1 Τι είναι τα εκπαιδευτικά παιχνίδια; 1

1.2 Εκπαιδευτικά παιχνίδια και Ψηφιακή Τεχνολογία 2

1.3 Χαρακτηριστικά των ψηφιακών και εκπαιδευτικών παιχνιδιών 2

1.4 Εκπαιδευτικά παιχνίδια μαγειρικής 3

1.4.1 Κίνητρο για δημιουργία ενός μαγειρικού παιχνιδιού 4

1.1 Τι είναι τα εκπαιδευτικά παιχνίδια;

Ένα εκπαιδευτικό παιχνίδι θεωρείται το οποιοδήποτε παιχνίδι που έχει σχεδιαστεί και έχει τροποποιηθεί ειδικά για να διδάξει τους ανθρώπους ένα συγκεκριμένο θέμα. Μέσα από ένα εκπαιδευτικό παιχνίδι, οι χρήστες μπορούν να αναπτύξουν και να εξασκήσουν συγκεκριμένες δεξιότητες, με τρόπο ευχάριστο και δημιουργικό. Τα εκπαιδευτικά παιχνίδια απευθύνονται σε όλες τις ηλικίες και το περιεχόμενό τους μπορεί να καλύπτει μεγάλο φάσμα θεμάτων τα οποία ανταποκρίνονται στις προσδοκίες του κάθε χρήστη, ανάλογα με το μαθησιακό θέμα το οποίο αναπτύσσεται μέσα από αυτό.

Εικόνα 1.1: Κάποια εκπαιδευτικά παιχνίδια

1.2 Εκπαιδευτικά παιχνίδια και Ψηφιακή Τεχνολογία

Τα εκπαιδευτικά παιχνίδια έχουν εξελιχθεί στο πέρασμα του χρόνου και η τεχνολογία έχει προκαλέσει σοβαρές αλλαγές στο τρόπο εκμάθησης .
Τα εκπαιδευτικά παιχνίδια, πλέον ακολουθούν τη δομή ενός τυπικού ψηφιακού παιχνιδιού, το οποίο συνήθως είναι τύπου περιπέτειας, τροποποιώντας το σενάριο και την εξέλιξη της ιστορίας με τις πληροφορίες που απαιτούνται για τη ανάκτηση της γνώσης ή των δεξιοτήτων που ο δημιουργός θέλει ο χρήστης να αποκτήσει.

1.3 Χαρακτηριστικά των ψηφιακών και εκπαιδευτικών παιχνιδιών

- Ευχαριστούν και διασκεδάζουν τον χρήστη
- Παρέχουν ένταση και ενεργή απασχόληση
- Διαθέτουν δομημένο περιβάλλον
- Παρέχουν κίνητρα στον χρήστη
- Οι παίκτες έχουν τη δυνατότητα να αλληλοεπιδρούν
- Προσαρμόζονται, ώστε να ικανοποιούν τον παίκτη
- Δημιουργούν καταστάσεις νίκης και ενισχύουν την αυτοπεποίθηση του παίκτη
- Μέσα από καταστάσεις αντιθέσεων, διλημμάτων και προκλήσεων ανεβάζουν την αδρεναλίνη
- Ενισχύουν τη δημιουργικότητα προσφέροντας προβλήματα προς λύση
- Ενθαρρύνουν την επικοινωνία μεταξύ των παικτών και τη σύσταση κοινωνικών ομάδων
- Προκαλούν συναισθήματα μέσω του σεναρίου και του περιβάλλοντος εργασίας
- Παράγουν αποτελέσματα και παρέχουν ενημέρωση, προκαλώντας μάθηση

1.4 Εκπαιδευτικά παιχνίδια με βάση τη μαγειρική

Η μαγειρική είναι μια καθημερινή ασχολία του ανθρώπου και γι' αυτό υπάρχει έντονα η ανάγκη για ανάπτυξη της σε όλους τους τομείς. Τα παιχνίδια μαγειρικής είναι ένας πολύ καλός και εύκολος τρόπος εκμάθησης μαγειρικών δεξιοτήτων και συνταγών και γι' αυτό είναι πολύ διαδεδομένα ανά τη παγκόσμια αγορά.

Εικόνα 1.2: Παιχνίδι μαγειρικής
Εικόνα 1.3: Παιχνίδι μαγειρικής 2

1.4.1 Κίνητρο για τη δημιουργία ενός μαγειρικού παιχνιδιού

Μετά από έρευνα καταλήγουμε στο συμπέρασμα τα περισσότερα παιχνίδια μαγειρικής που υπάρχουν σήμερα στην αγορά και ασχολούνται με την εκμάθηση μαγειρικών συνταγών είναι δισδιάστατα. Για τον λόγο αυτό δημιουργείται η ανάγκη για μελέτη και υλοποίηση ενός παρόμοιου παιχνιδιού στις τρις διαστάσεις. Με την εξέλιξη της τεχνολογίας και με βάση τα δεδομένα που υπάρχουν στα σημερινά εκπαιδευτικά παιχνίδια, οι προσδοκίες του χρήστη όσο αφορά το περιβάλλονται την ποιότητα του παιχνιδιού έχουν γίνει πόλη πιο απαιτητικές. Για το λόγω αυτό, η μετατροπή ενός ήδη ύπαρχων δισδιάστατου παιχνιδιού σε τρισδιάστατο παρέχει στο χρήστη μια πιο καινοτόμα και ρεαλιστική εμπειρία με το παιχνίδι.

Εικόνα 1.4: Από 2D σε 3D

ΚΕΦΑΛΑΙΟ 2

Περιγραφή Παιχνιδιού

2.1 Περιγραφή του παιχνιδιού 5
2.2 Σκοπός του παιχνιδιού 6
2.3 Τρόποι Αξιολόγησης 6
2.4 Δυσκολίες παιχνιδιού 6
2.5 Γιατί επέλεξα το “Something Ancient is Baked”; 7
2.6 Ηλικιακή Κλίμακα Χρηστών 8

2.1 Περιγραφή του παιχνιδιού

Στο παιχνίδι όπου έχω δημιουργήσει, ο χρήστης θα περιηγείται αρχικά στην αποθήκη όπου θα πρέπει να επιλέξει μέσα από τα διάφορα υλικά που βρίσκονται εκεί για να ακολουθήσει την συνταγή του φαγητού που θα του αναφέρεται στην αρχή κάθε σταδίου. Όταν θα έχει επιλέξει όλα τα υλικά της συνταγής τότε θα μπορεί να δει τη συνταγή του φαγητού όπου θα εκτελέσει πατώντας το «W». Επίσης, θα εμφανιστούν στα δεξιά και στα αριστερά διάφορα εικονίδια τα οποία θα τον βοηθήσουν στην εκτέλεση μερικών κινήσεων του μάγειρα για να μπορέσει να δημιουργήσει τα τελικά υλικά και να φτάσει στο ζητούμενο αποτέλεσμα. Καθώς θα εκτελεί ο χρήστης τα στάδια, θα μπορεί να παρακολουθήσει τη συνταγή και τα στάδια τα οποία θα έχει εκτελέσει θα αλλάζουν χρώμα για να γνωρίζει ότι έχει τελειώσει τα συγκεκριμένα στάδια. Για την εκτέλεση των ιδικών κινήσεων του μάγειρα θα πρέπει να ικανοποιηθούν μερικές προδιαγραφές και θα πρέπει να επιλέξει στο χώρο της κουζίνας σε πιο αντικείμενο θα θέλει να εκτελεστούν. Με αυτό τον τρόπο ο χρήστης θα μάθει τη σωστή διαδικασία για τη κάθε διεργασία.

2.2 Σκοπός του παιχνιδιού

Ο χρήστης μπορεί να νικήσει εύκολα το παιχνίδι γιατί αν κάνει τέσσερα λάθη στην επιλογή, τα υλικά που του λυπόνται για την εκτέλεση της συνταγής θα του εμφανιστούν στη καταγραφή των εμπορευμάτων. Επίσης, οι εκτελέσεις που πρέπει να κάνει ο χρήστης για να βοηθήσουν τον μάγειρα να κάνει τις απαραίτητες κινήσεις είναι απλές.

Ο χρήστης θα θέλει να ξαναπαίξει το παιχνίδι για να πετύχει καλύτερη την βαθμολογία του όπου θα εμφανίζεται στο τέλος κάθε σταδίου. Με αυτό τον τρόπο ο χρήστης θα μάθει καλύτερα τα υλικά που χρειάζονται για τη κάθε συνταγή.

Ο σκοπός του παιχνιδιού είναι να μάθει ο παίχτης συνταγές της κυπριακής κουζίνας η οποίες είναι ξεχασμένες. Επίσης, τις συνταγές θα μπορεί ο παίχτης, αφού πρώτα μάθει την εκτέλεση και τα υλικά μιας συνταγής, να τις εκτελέσει και μόνος του στο σπίτι του κάνοντας την εκτέλεση που κάνει ο μάγειρας του παιχνιδιού. Η εκτέλεση που κάνει ο μάγειρας είναι πιο άμεση από ότι στη πραγματικότητα. Αυτό όμως δεν είναι πρόβλημα γιατί στο παιχνίδι φαίνονται οι αλλαγές που γίνονται στα υλικά.

2.3 Τρόποι Αξιολόγησης

Ο χρήστης βαθμολογείται με βάση το 90%. Το 90 το παίρνει αυτόματα γιατί είναι η εκτέλεση της συνταγής όπου θα ακολουθήσει τα βήμα που του έχουν δοθεί. Το άλλο 10% θα μπορεί να το αποκτήσει βρίσκοντας τα σωστά υλικά που θα χριστεί η συνταγή. Με αυτό τον τρόπο ο χρήστης θα μαθαίνει τα υλικά της κάθε συνταγής για να μπορέσει να πάρει το 100% στο τέλος του σταδίου.

Το 10% κυμαίνεται σύμφωνα με τις σωστές επιλογές του χρήστη πριν κάνει τέσσερα λάθη. Αν βρει για παράδειγμα 3 σωστά υλικά από τα 4 τότε θα πάρει 7

από τα 10, ενώ αν βρει και τα 4 υλικά και θα έκανε 3 λάθη τότε θα πάρει 10 από τα 10. Με αυτό τον τρόπο θα μπορέσει να βελτιώσει την μνήμη και τη βαθμολογία του στο τέλος.

2.4 Δυσκολίες παιχνιδιού

Η δυσκολία που έχει το παιχνίδι είναι ότι παίχτης θα πρέπει να γνωρίζει τις συνταγές προηγούμενων σταδίων για να εκτελέσει τις επόμενες οι οποίες θα χρειάζεται κάποια άλλα κύρια κομμάτια προηγούμενων συνταγών, π.χ. ο κιμάς. Με αυτό τον τρόπο ο παίχτης θα μαθαίνει τις συνταγές που θα του δίνονται και θα του είναι πιο ενδιαφέρον τραβώντας του την προσοχή κάνοντας να θέλει να συνεχίζει να παίζει. Επίσης, ο χρήστης θα καλυτερεύει τη μνήμη του με αυτό τον τρόπο, και για την συνέχιση τις συνταγής θα πρέπει πρώτα να εκτελέσει τα προηγούμενα βασικά συστατικά, π.χ. ο κιμάς, για να μπορέσει να κατασκευάσει κάποια άλλα συστατικά τις συνταγής.

2.5 Γιατί επέλεξες το “Something Ancient is baked”;

Ο λόγος που επέλεξα ένα παιχνίδι μαγειρικής είναι ότι στον ελεύθερο μου χρόνο μου αρέσει να μαγειρεύω διάφορες συνταγές τις οποίες τις βρίσκω στο διαδίκτυο ή στα διάφορα βιβλία μαγειρικής που έχω στο σπίτι μου. Επίσης, δεν υπάρχουν πολλές ιστοσελίδες που να περιέχουν συνταγές της κυπριακής κουζίνας ή παιχνίδια που να έχουν συνταγές από τη κυπριακή κουζίνα και για αυτό τον λόγο μου είχε έρθει η ιδία να δημιουργήσω ένα παιχνίδι που να είναι εύκολο στην εκτέλεση του για παρακινήσω και άλλους ανθρώπους να μάθουν τη κυπριακή κουζίνα και να αρχίσουν να μαγειρεύουν πιο συχνά κάποια πιο ξεχασμένα άλλα ωραία φαγητά τις κυπριακής κουζίνας.

Η ιδέα του παιχνιδιού βασίζεται στην εκτέλεση μίας κυπριακής συνταγής όπου ο παίχτης θα πρέπει να βρίσκει τα υλικά που χρειάζεται για μια συγκεκριμένη συνταγή. Το πρώτο κομμάτι θα επιλέγει υλικά τα οποία θα είναι σκορπισμένα στο χώρο. Τα αντικείμενα τα οποία είναι σωστά θα μπαίνουν στη καταγραφή εμπορευμάτων ενώ τα λανθασμένα θα εμφανίζουν ένα κόκκινο χι κάτω

αριστερά. Ο παίχτης θα έχει τέσσερις ευκαιρίες να βρει τα σωστά υλικά. Αν χάσει και τις τέσσερις ευκαιρίες τότε θα συμπληρώνονται τα υλικά που του λείπουν στη καταγραφή εμπορευμάτων έτσι ώστε να μπορεί να εκτελέσει την συνταγή. Ακολούθως, ο παίχτης θα προχωρά στη κουζίνα όπου θα εκτελέσει τη συνταγή.

2.6 Ηλικιακή Κλίμακα Χρηστών

Το παιχνίδι το οποίο έχω δημιουργήσει ανταποκρίνεται σε ένα μεγάλο φάσμα ηλικιών ξεκινώντας από 10 χρονών. Ο οποιοδήποτε θα μπορεί να παίξει αυτό το παιχνίδι γιατί είναι πολύ ευχάριστο και θα μπορούν να μάθουν διάφορες και ξεχασμένες συνταγές της κυπριακής κουζίνας. Τα παιδιά θα μπορούν να μάθουν συνταγές και να βοηθούν τις μητέρες τους στην κουζίνα και να περνούν με πιο ευχάριστο τρόπο το χρόνο τους με τους γονείς.

Κεφάλαιο 3

Σφαιρική εικόνα του παιχνιδιού

3.1 Προγραμματιστικό περιβάλλον του παιχνιδιού 9
3.2 Συστατικά του παιχνιδιού 10
3.2.1 Assets 10
3.2.1.1 Ο χαρακτήρας (Chef) 10
3.2.1.2 Η κουζίνα 11
3.2.1.3 Τα υλικά 12
3.2.2 Sound Effect και Animation 13
3.2.3 Scores 13
3.3 Πώς λειτουργά το παιχνίδι; 14
3.3.1 Τα δύο στάδια 16
3.4 User Interface (UI) 17

3.1 Προγραμματιστικό περιβάλλον του παιχνιδιού

Το περιβάλλον που χρησιμοποίησα για τη δημιουργία του παιχνιδιού είναι το Unity. Ο λόγος της χρήσεις αυτού του περιβάλλον είναι η εύκολη χρήση του για την παραγωγή 3D παιχνιδιών και οι δυνατότητες που σου παρέχει για διάφορων μετατροπών και χρησιμότητας των εργαλείων του είναι πραγματικά θαυμάσια. Επίσης τα scripts τα οποία έχω γράψει είναι σε C# γλώσσα.

Επέλεξα το παιχνίδι μου να παίζει σε πλατφόρμα τον Windows γιατί είναι το λογισμικό που χρησιμοποιούν οι περισσότεροι και θα ήθελα να μπορεί ο καθένας να μπορεί να παίξει το παιχνίδι μου. Ταυτοχρόνως, θα μπορεί να το παίξουν το παιχνίδι όπου έχω δημιουργήσει και στα σχολεία αφού το λογισμικό που έχουν στους ηλεκτρονικούς υπολογιστές είναι Windows.

3.2 Συστατικά του παιχνιδιού

Τα συστατικά του παιχνιδιού είναι τα Assets, το Sound Effect, το Animation και τα Scores. Όλα μαζί λειτουργούν από μόνα τους και επιβλέπονται από το Game Manager. Τα περισσότερα από τα συστατικά του παιχνιδιού τα έχω βρει στο Unity Asset Store[2,3,4,5] .

3.2.1 Assets

Τα Assets του παιχνιδιού, τα περισσότερα τα βρήκα από το Asset Store του Unity. Αρχικά, είναι ο Chef μας ο οποίος είχε τις βασικές κινήσεις ενός Chef. Για παράδειγμα να κόβει, να ψιλοκόβει, να ανακατώνει, να ψήνει, να αναποδογυρίζει τηγανίτες, να ελέγχει αν το φαγητό είναι καλό ή αν του λείπει κάποιο συστατικό, να αλατίζει, να ρίχνει υγρά στο τηγάνι ή να πίνει κρασί και να μας χαιρετά. Δεν είχε μερικές κινήσεις που θα χρειαζόμουν όπως να περπατά και να παίρνει αντικείμενα. Πρόσθεσα αυτές τις κινήσεις χρησιμοποιώντας το Motion Builder. Ακολούθως, βρήκα μια απλή κουζίνα και την τροποποίησα για να είναι πιο προσιτή στο παιχνίδι. Τέλος, βρήκα δύο διαφορετικά assets που είχαν υλικά για μαγειρικής. Κτίζοντας το παιχνίδι χρειάστηκε να δημιουργήσω μερικά prefabs τα οποία θα βοηθούσας τον παίχτη να βλέπει μια πιο ξεκάθαρη εικόνα για το τι γίνεται στο παιχνίδι και στα αντικείμενα του.

3.2.1.1 Ο χαρακτήρας (Chef)

Ο χαρακτήρας που διάλεξα είχε δύο βασικά προβλήματα. Αρχικά, είναι οι κινήσεις walk και pick up δεν ήταν στο χαρακτήρα. Αυτό το πρόβλημα με την βοήθεια από τον υπεύθυνο που με παράπεμψε ο κύριος Χρυσάνθου μπόρεσα να ενσωματώσω αυτές τις δύο κινήσεις. Το δεύτερο πρόβλημα που βρήκα είναι ότι ο χαρακτήρας δεν μπορεί να κινηθεί, από το σημείο που βρισκόταν ή έκανε

τη κίνηση του ξανά και ξανά. Τη λύση που βρήκα είναι να μη κινώ τον χαρακτήρα με τον Agent αλλά με το Navigation Mess

Εικόνα 3.1: Ο Μάγειρας

3.2.1.2 Η κουζίνα

Η κουζίνα που βρήκα ήταν τρις κουζίνες η μια δίπλα στην άλλη εγώ επέλεξα την πρώτη κουζίνα και της αφαίρεσα τον εσωτερικό πάγκο που είχε. Όλα τα άλλα τα αντικείμενα τα αφαίρεσα για να βάλω μόνο αυτά που χεριάζομε.

Εικόνα 3.2: Η κουζίνα

3.2.1.3 Τα υλικά

Τα υλικά που βρήκα είναι αυτά τα δύο που θα δείτε πιο κάτω. Το πρώτο το πακέτο με τα υλικά ήταν πολύ χρήσιμο και ήταν αυτό που μου έδωσε την ιδέα να έχω σκορπισμένα στο χώρο τα υλικά της συνταγής. Το δεύτερο το πακέτο είχε υλικά τα οποία χρειαζόμουν για τις συνταγές και δεν με πειράζει που είναι σε πιο κάτω ποιότητα σε σχέση με το πρώτο.

Εικόνα 3.3: Υλικά Α

Εικόνα 3.4: Υλικά Β

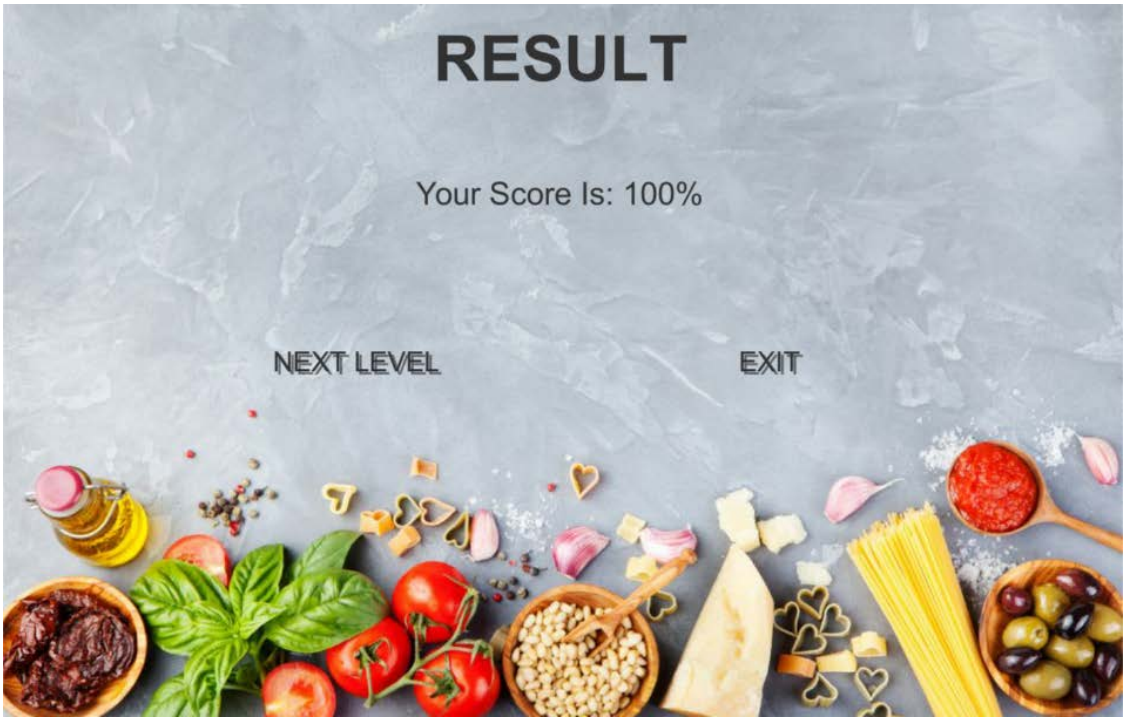
3.2.2 Sound Effect και Animation

Τα Sound Effect τα οποία βρήκα και χρησιμοποιώ στο παιχνίδι είναι οι ήχοι για το κόψιμο, την ροή ενός υγρού και το τσιγάρισμα. Αυτοί, οι ήχοι παίζουν κατά την διάρκεια του παιχνιδιού σε συγκεκριμένα σημεία. Όσο για τον ήχο στο περιθώριο είναι μια απαλή μελωδία έτσι ώστε το παιχνίδι να μην είναι μουντό και βαρετό.

Το Animation που υπάρχει στο παιχνίδι είναι μόνο του χαρακτήρα. Ο χαρακτήρας μας στην αρχή μας χαιρετά μέχρι να πατήσουμε σε κάποιο αντικείμενο ή στο δάπεδο για να μπορέσει να κινηθεί. Αν πατήσουμε σε αντικείμενο τότε ο χαρακτήρας μας πλησιάζει το αντικείμενο σε ένα συγκεκριμένο σημείο και εκτελεί την κίνηση που παίρνει ένα αντικείμενο. Μόλις πάει στη κουζίνα μπορεί να κάνει διάφορα Animation αλλά εξαρτάτε σε πιο αντικείμενο θα επιλέξει ο παίχτης και αν θα χρειάζεται αυτή η κίνηση για τη συνταγή.

3.2.3 Scores

Το Score του κάθε σταδίου εξαρτάται από δύο μέρη. Το πρώτο μέρος είναι η επιλογή των υλικών. Αυτό το μέρος παίρνει μόνο 10% του τελικού βαθμού. Με βάση τα υλικά που βρήκε και τα υλικά που χρειάζεται θα υπολογίζεται αυτό το 10% του βαθμού με αυτή την πράξη: βαθμός = (10 * υλικά που βρήκε / υλικά που χιάζονται). Το δεύτερο μέρος είναι προκαθορισμένο ότι θα πάρει 90% γιατί απλός ακολουθεί τη συνταγή.



Εικόνα 3.5: Αποτελέσματα σταδίου

3.3 Πως λειτουργεί το παιχνίδι;

Το παιχνίδι αρχίζει με το Menu μας το οποίο ο παίχτης θα μπορεί να επιλέξει να αρχίσει το παιχνίδι, να αλλάξει την ένταση της μουσικής, να επιλέξει κάποιο άλλο στάδιο ή να βγει από το παιχνίδι. Κάθε εντολή που θα κάνει ο παίχτης θα το στείλει στο ανάλογο κομμάτι του παιχνιδιού.

Εικόνα 3.6: Menu

Εικόνα 3.7: Settings

Εικόνα 3.8: Levels

3.3.1 Τα δύο στάδια

Τα δύο στάδια είναι φτιαγμένα με τον ίδιο τρόπο. Στην αρχή του κάθε σταδίου θα εμφανίζονται οι κανόνες του σταδίου όπου θα του λέει πια συνταγή θα εκτελέσει σε αυτό το στάδιο. Ακολούθως, θα εμφανίζεται το Inventory του σταδίου και ο παίχτης θα πρέπει να συμπληρώσει τα κενά με τα αντικείμενα. Κάθε κενή θέση είναι και ένα υλικό που πρέπει να συλλέξει. Μόλις πάρει όλα τα υλικά που χρειάζεται θα μπορεί να δει την συνταγή και να πάει να την εκτελέσει. Τα steps της συνταγής μπορούν να γίνονται ανάλογα και όχι με τη σειρά που εμφανίζονται. Κάθε step που θα εκτελείται θα αλλάζει και το χρώμα του για να ξέρει ο παίχτης τι έχει κάνει μέχρι στιγμής. Για να τελειώσει το στάδιο θα πρέπει να έχει το τελικό αντικείμενο στο Inventory.

Εικόνα 3.9: Στάδιο 1

Εικόνα 3.10: Στάδιο 2

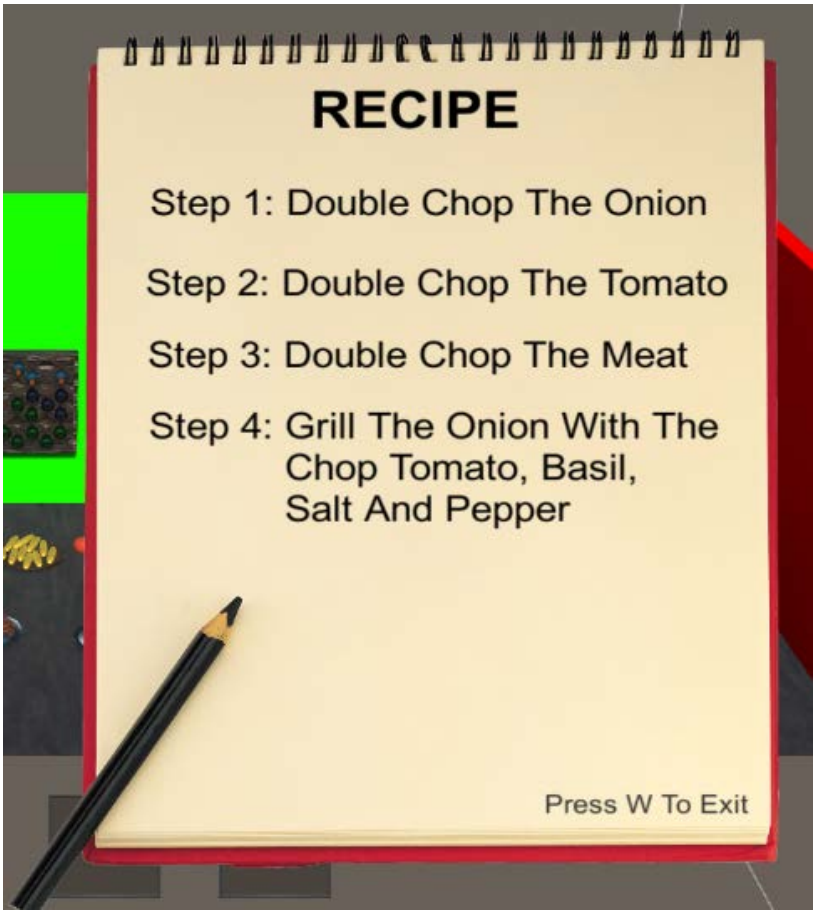
3.4 User Interface (UI)

Το User Interface που εμφανίζεται είναι στο Menu και για την παρουσίαση των κανόνων, το Inventory, οι διάφορες κινήσει που μπορεί να κάνει ο χαρακτήρας και τα αποτελέσματα του κάθε σταδίου.

Εικόνα 3.11: Rules


Εικόνα 3.12: False Choose And Inventory

Εικόνα 3.13: Inventory and Characters Moves



Εικόνα 3.14:Recipe

Κεφάλαιο 4

Τεχνικό κομμάτι

4.1 Συνθήκες 20
4.2 Αλληλεπίδραση 21
4.3 Αντίδραση 21
4.3.1 Ηχητική αντίδραση 21
4.3.2 Κινητική αντίδραση 21
4.3.3 Προσθήκη στο Inventory 22
4.3.4 Αφαίρεση από Inventory 22
4.3.5 Game Object Reaction 23
4.3.6 Αλλαγή συνθήκης 23
4.4 Επιλογή υλικών 23
4.5 Βοηθητικά κουμπιά του χαρακτήρα 24
4.6 Inventory buttons 25
4.7 Αντικείμενα 25
4.8 Διάγραμμα του Animation 26
4.9 Τα υπόλοιπα κουμπιά 27
4.10 Game manager Scripts 28

4.1 Συνθήκες

Οι συνθήκες είναι το πιο βασικό κομμάτι στο παιχνίδι. Όλα τα αντικείμενα που είναι στο παιχνίδι έχουν και μια συνθήκη. Οι συνθήκες μας βοηθάνε να κάνουμε πολλές διαφορετικές αλληλεπιδράσεις, αναλόγως των συνθηκών που βάλαμε για προϋπόθεση αυτής της αντίδρασης. Μια συνθήκη έχει ένα όνομα και μία Boolean μεταβλητή όπου αν είναι αληθής τότε μπορούμε να εκτελέσουμε τη συγκεκριμένη αλληλεπίδραση.

4.2 Αλληλεπίδραση

Μια αλληλεπίδραση έχει διάφορες αντιδράσεις της οποίες μπορεί να τις εκτελέσει ταυτόχρονα ή να τις εκτελέσει την μια μετά την άλλη με κάποια χρονικά διαστήματα. Μια αλληλεπίδραση για εκτελεστεί μπορεί να έχει κάποιες συνθήκες ή μπορεί και να μην έχει και καθόλου.

4.3 Αντίδραση

Η αντίδραση που μπορεί να γίνει στο παιχνίδι είναι ηχητική, κινητική, προσθήκη, αφαίρεση, αλλαγής συνθήκης και η εμφάνιση ή εξαφάνιση κάποιου αντικειμένου μέσα στο παιχνίδι. Κάθε μια από αυτές τις αντιδράσεις έχει τα δικά της πεδία και μας αλλάζουν την προοπτική του παιχνιδιού.

4.3.1 Ηχητική αντίδραση

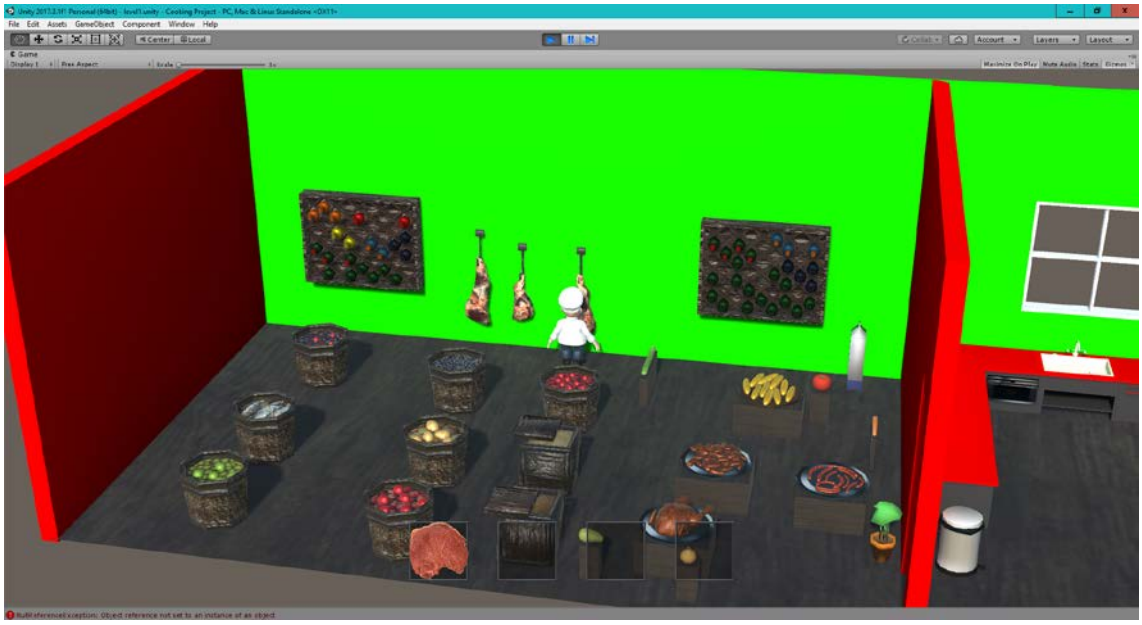
Στην ηχητική αντίδραση μπορούμε να προσθέσουμε κάποιο ηχητικό κομμάτι την ώρα που θα κάνει μια διαδικασία και να μας φέρνει την διαίσθηση ότι γίνεται στη πραγματικότητα εκείνη τη χρονική στιγμή.

4.3.2 Κινητική αντίδραση

Στη κινητική αντίδραση μπορούμε να βάλουμε τον χαρακτήρα μας να κάνει μια από τις κινήσεις του χρησιμοποιώντας το κατάλληλο trigger. Με αυτό τον τρόπο μπορούμε να κάνουμε διάφορες κινήσεις χωρίς να χρειάζεται να επαναλαμβάνουμε τον κώδικα μας για διαφορετικές περιπτώσεις

4.3.3 Προσθήκη στο Inventory

Η αντίδραση προσθήκη στο inventory την χρησιμοποιούμε συχνότερα από όποια άλλη αντίδραση. Με αυτή την αντίδραση παίρνουμε ένα αντικείμενο και το προσθέτουμε στο inventory. Αυτό μας βοηθά στην προσθήκη διαφόρων αντικειμένων στο inventory χωρίς να χρειάζεται να τον συνεχές έλεγχο για να δούμε πιο αντικείμενο έχει αφαιρεθεί και σε πια θέση θα πρέπει να μπει.



Εικόνα 4.1: Προσθήκη στο Inventory

4.3.4 Αφαίρεση από Inventory

Η αντίδραση αφαίρεση από το inventory είναι εξίσου σημαντική και χρησιμοποιείτε όπως και η προσθήκη. Στην αφαίρεση απλός παίρνουμε το αντικείμενο που δεν χρειαζόμαστε πια από το inventory και το αφαιρούμε και αφήνουμε στην θέση του μια κενή και έτοιμη θέση να πάρει ένα άλλο αντικείμενο.

Εικόνα 4.2: Αφαίρεση από Inventory

4.3.5 Game Object Reaction

Η αντίδραση του game object είναι πιο απλή αλλά μας δίνει μια πιο ολοκληρωμένη αίσθηση καθώς παίζουμε το παιχνίδι. Απλός μπορεί να μας εμφανίσει κάποια αντικείμενα που είναι στο παιχνίδι ή να τα εξαφανίσει.

4.3.6 Αλλαγή συνθήκης

Η αντίδραση αλλαγή συνθήκης μας βοηθά να κάνουμε μερικές αλληλεπιδράσεις ξανά και ξανά ή να πάμε σε μια εντελώς διαφορετική και καινούργια αλληλεπίδραση.

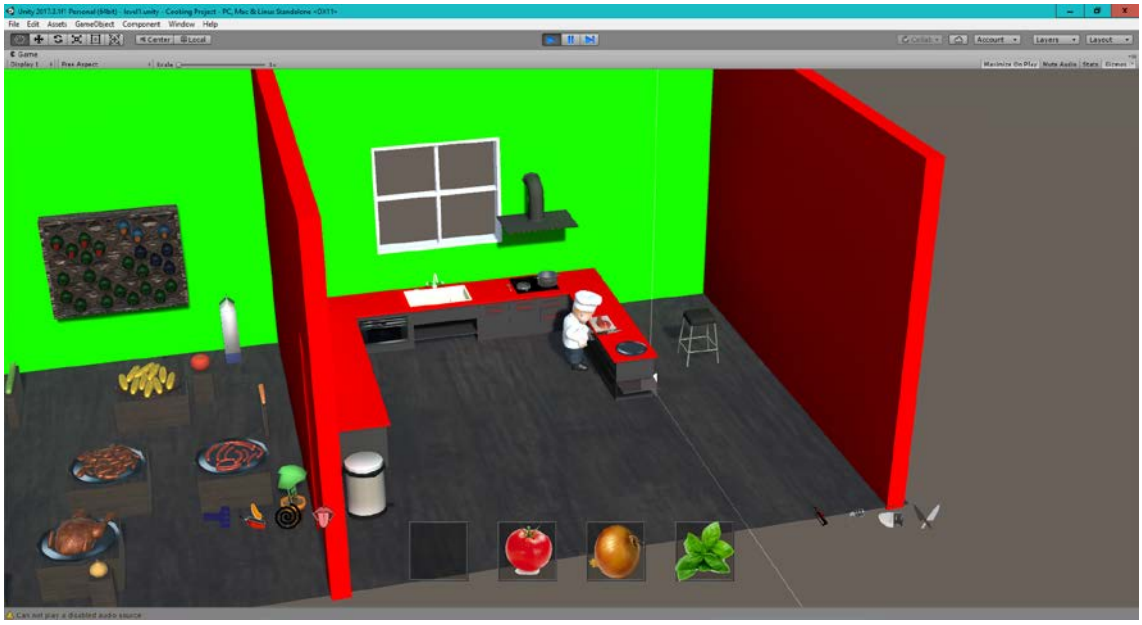
4.4 Επιλογή υλικών

Για την επιλογή των υλικών προσθέσαμε δύο αντιδράσεις, την κινητική και την προσθήκη. Με βάση των game manager μας ελέγχουμε αν το υλικό που

πήραμε μας κάνει. Αν όμως δεν μας κάνει τότε εμφανίζουμε εν Χ στο κάτω μέρος αριστερά. Αν ο παίχτης επιλέξει τρις φορές λάθος αντικείμενο τότε θα του προστεθούν στο inventory τα υπόλοιπα υλικά και θα μπορεί να πάει στην κουζίνα και να εκτελέσει τη συνταγή. Η επιλογή των υλικών είναι σημαντική και θέλουμε να γίνεται σωστά στον κώδικα και να φαίνεται σωστά κατά την εκτέλεση. Ένα μικρό πρόβλημα βρήκα σε αυτή τη διαδικασία που ήταν στον κώδικα που είχα γράψει. Το πρόβλημα ήταν ότι μου αφαιρούσε συνέχεια το πρώτο το αντικείμενο από το inventory και δεν μπορούσε να κρατήσει σωστά τα δεδομένα. Με λίγη υπομονή και σκέψη βρήκα ότι είχα κάποιες διαδικασίες σε ένα for loop και δεν μπορούσε να μου κρατήσει τις νέες πληροφορίες που του πρόσθετα.

4.5 Βοηθητικά κουμπιά του χαρακτήρα

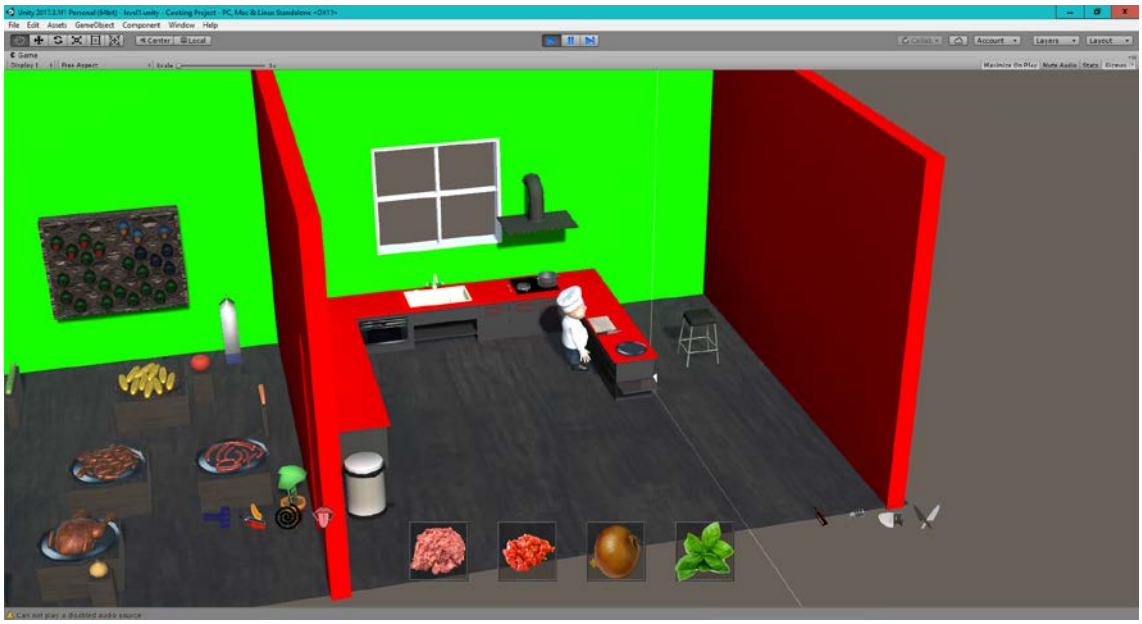
Τα βοηθητικά κουμπιά του χαρακτήρα είναι οκτώ. Το κάθε κουμπί κάνει και μια διαφορετική συνθήκη αληθές. Αρχίζοντας από τα αριστερά στα δεξιά έχουμε το κουμπί για τηγάνισμα, μετά είναι το κουμπί για αναποδογύρισμα τηγανίτας, το κουμπί του ανακατέματος, της δοκιμής, της προσθήκης υγρού, του αλατίσματος, της κανονικής κοπής και της ψιλής κοπής.



Εικόνα 4.3: Βοηθητικά κουμπιά

4.6 Inventory buttons

Τα inventory κουμπιά είναι η θέση του κάθε αντικειμένου. Όταν πατηθεί κάποιο από τα κουμπιά, θα θέσει τη συνθήκη αυτού του αντικειμένου σε αληθές και μόλις επιλέξει ο παίχτης μια ενεργή θέση τότε θα θέση μια σειρά από αντιδράσεις να λάβουν μέρος. Στο game manager θα υπάρχουν όλες οι πιθανές συνθήκες που θα λάβουν μέρος για την κάθε συνταγή.



Εικόνα 4.4: Inventory buttons, το κρεμμύδι επιλέγεται

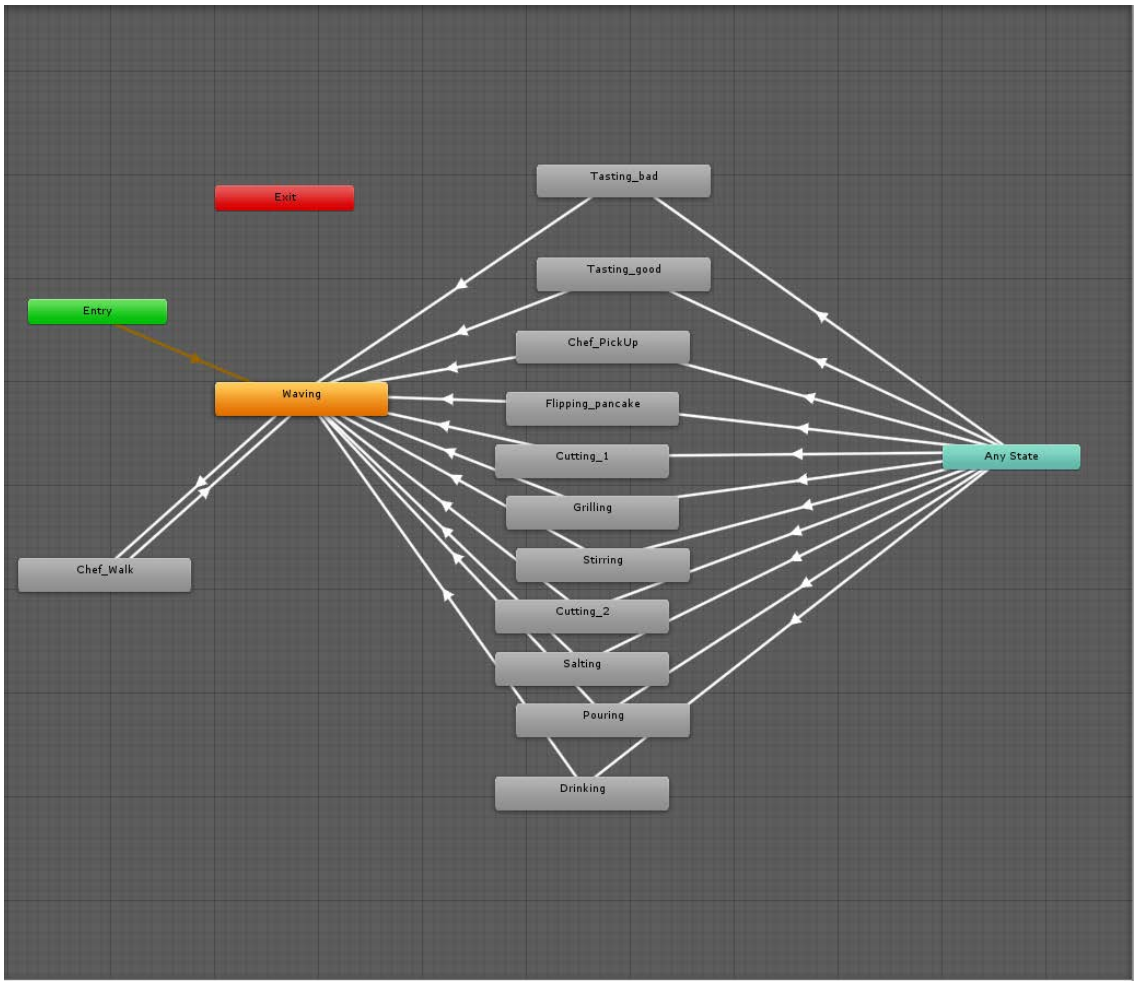
4.7 Αντικείμενα

Τα αντικείμενα έχουν μόνο μία διαδικασία. Μεταφέρουν τα sprites σαν μια πληροφορία και μπορούμε να τα διαχειριστούμε και να τα εμφανίσουμε στο inventory μας. Επίσης τα χρησιμοποιούμε για στις αντιδράσεις της προσθήκης και της αφαίρεσης. Κάθε υλικό και κάθε διαφορετική παραλλαγή του υλικού για να μπορούμε πιο εύκολα να τα διαχειριζόμαστε, μέσον τον διαφόρων διαδικασιών και την επίβλεψη από τον game manager.

Εικόνα 4.5: Αντικείμενα στη καταγραφή αποθεμάτων

4.8 Διάγραμμα του Animation

Το διάγραμμα του animation που εμφανίζεται πιο κάτω παρουσιάζει την ροή των διαφόρων κινήσεων του χαρακτήρα μας. Αρχικά, όταν αρχίζει το παιχνίδι ο χαρακτήρας θα μας χαιρετά. Μόλις επιλέξουμε μια περιοχή στο χώρο μας ο χαρακτήρας θα αρχίσει να κινείτε μέχρι να φτάσει στο επιθυμητό σημείο. Όταν επιλέξουμε ένα αντικείμενο τότε ο χαρακτήρας μας πηγαίνει στο αντικείμενο και ελέγχοντας την αλληλεπίδραση, ενεργοποιείτε το trigger Take και κάνει την κίνηση του Pick Up. Καθώς θα βρίσκεται στην κουζίνα θα εμφανιστούν τα βοηθητικά κουμπιά του Chef. Επιλέγοντας ένα από αυτά τα κουμπιά κάνουμε αληθές την συνθήκη τους και όταν πάμε στο επιθυμητό σημείο, το trigger τις κάθε μιας από αυτές τις συνθήκες θα ενεργοποιηθεί και θα εκτελεστή η κίνηση του ανάλογου trigger.



Εικόνα 4.6: Animator

4.9 Τα υπόλοιπα κουμπιά

Τα κουμπιά τα οποία είναι στο παιχνίδι κάνουν τα ακόλουθα function. Στο Menu υπάρχουν 4 κουμπιά, το play, το settings, το levels και το exit. Το play μας παραπέμπει στη πρώτη σκηνή του παιχνιδιού. Το settings μας εμφανίζει ένα παράθυρο το οποίο έχει μια ράβδο όπου ελέγχει την ένταση της μουσικής και το back κουμπί που μας παίρνει πίσω στο menu. Το levels μας εμφανίζει ένα άλλο παράθυρο το οποίο έχει τρία κουμπιά, το level 1, level 2 που μας περνούν στο συγκεκριμένο στάδιο και το back κουμπί που μας παίρνει πίσω στο menu. Στο τέλος του κάθε σταδίου όπου είναι τα αποτελέσματα του σταδίου υπάρχουν δύο κουμπιά, το next level και το exit. Το next level κουμπί μας παίρνει στο επόμενο

στάδιο. Αν δεν υπάρχει τότε μας παίρνει στο menu. Το exit κουμπί μας κλείνει το πρόγραμμα.

Εικόνα 4.7: Κουμπιά Menu

4.10 Game manager Scripts

Το αντικείμενο game manager έχει δύο scripts. Το πρώτο script ελέγχει αν επέλεξε σωστά τα υλικά για να μπουν στο inventory και το δεύτερο script ελέγχει και τροποποιεί το χρώμα των κειμένων της συνταγής και αναθέτει τα conditions για τη κάθε θέση του inventory και τα conditions για τα βοηθητικά του chef.

Κεφάλαιο 5

Εγχειρίδιο χρήστη

5.1 Μενού 29

5.2 Πρώτο Στάδιο 29

5.3 Δεύτερο Στάδιο 31

5.1 Μενού

Βήμα 1: Όταν τρέξουμε το παιχνίδι αρχικά θα μας εμφανίσει το μενού του παιχνιδιού. Εκεί θα έχει τέσσερα κουμπιά: Start, Levels, Settings και Quit.

Βήμα 2: Για να αρχίσει το παιχνίδι θα πατήσεις το Start κουμπί για να μεταβείς στο πρώτο στάδιο. Αν πατήσεις το Level κουμπί θα μεταβείς στη λίστα με όλα τα στάδια, εκεί μπορείς να επιλέξεις όποιο από τα δύο στάδια για να μεταβείς απευθείας σε αυτό. Αν πατήσεις το Settings κουμπί θα μεταβείς στη ρύθμιση του ήχου, όπου θα μπορείς να μειώσεις την ένταση της μουσικής και έπειτα να πατήσεις το Back κουμπί για να επιστρέψεις στο μενού. Αν πατήσεις στο Quit κουμπί θα τερματίσεις το παιχνίδι.

5.2 Πρώτο Στάδιο

Βήμα 1: Όταν πάμε στο πρώτο στάδιο θα δούμε αρχικά κάποιους κανόνες για το πρώτο στάδιο. Για να αρχίσουμε να παίξουμε το στάδιο θα πρέπει να πατήσεις το Q κουμπί για να φύγουν οι κανόνες.

Βήμα 2: Θα πρέπει να πάρουμε τέσσερα υλικά για να μπορέσουμε να μας εμφανίσει τα βοηθητικά κουμπιά του μάγειρα.

Βήμα 3: Τα σωστά υλικά είναι το τρίτο κρέας που κρέμεται στο τοίχος, η ντομάτα που είναι κοντά στο κουτί του γαλατούς, το κρεμμύδι που είναι στο κάτω μέρος και το βασιλικό που είναι το πράσινο δεντράκι κοντά στη πόρτα. Μόλις πάρεις και τα τέσσερα υλικά θα εμφανιστούν τα βοηθητικά κουμπιά.

Βήμα 4: Τώρα μπορείς να δεις τη συνταγή πατώντας το W κουμπί για να δεις τα βήματα που χρειάζεται να γίνουν για να περάσεις το στάδιο. Για να συνεχίσεις με την εκτέλεση στης συνταγής θα πρέπει να ξαναπατήσεις το ίδιο κουμπί.

Βήμα 5: Για να ψιλοκόψεις τα υλικά θα πρέπει πρώτα να επιλέξεις το υλικό που θέλεις και μετά να πατήσεις στο σανιδάκι κοπής που είναι στο μπάγκο της κουζίνας. Μετά θα πρέπει να πατήσεις το βοηθητικό κουμπί με τα δύο μαχαίρια και να ξαναπατήσεις το σανιδάκι κοπής. Με αυτό τον τρόπο θα ψιλοκόψεις τα υλικά.

Βήμα 6: Για να τηγανίσεις τα υλικά σου θα πρέπει πρώτα να επιλέξεις ένα-ένα τα υλικά και επιλέγεις το τηγάνι μετά για να βάλεις ένα-ένα τα υλικά στο τηγάνι. Μόλις βάλεις όλα τα υλικά θα πρέπει να πατήσεις το βοηθητικό κουμπί που είναι μια μπλε σπάτουλα και να πατήσεις μετά το τηγάνι για να τηγανίσει τα υλικά.

Βήμα 7: Μόλις τηγανιστούν τα υλικά θα σου εμφανίσει τα αποτελέσματα του πρώτου σταδίου. Εκεί θα μπορείς να πας στο επόμενο στάδιο ή να βγεις από το παιχνίδι.

Επιπρόσθετο βήμα: Σε οποιαδήποτε στιγμή στο παιχνίδι μπορείς να πατήσεις το ESC κουμπί και να παγώσεις το παιχνίδι. Επίσης, μπορείς να συνεχίσεις το παιχνίδι, τα πας στο κεντρικό μενού ή να βγεις από το παιχνίδι.

5.3 Δεύτερο Στάδιο

Βήμα 1: Μόλις μπεις στο δεύτερο στάδιο θα δεις τους κανόνες του δεύτερου σταδίου. Για να αρχίσεις να επιλέγεις υλικά θα πρέπει να πατήσεις το Q κουμπί όπως και στο πρώτο στάδιο.

Βήμα 2: Τα τέσσερα από τα έξι υλικά είναι τα ίδια με το πρώτο στάδιο, τα άλλα δύο είναι η πατάτα που βρίσκεται στο δεύτερο κασόνι στη δεύτερη σειρά και το λάδι που είναι στη δεύτερη κάβα με μπουκάλες κοντά στη ντομάτα.

Βήμα 3: Μόλις πάρουμε τα υλικά μπορείς να δεις και πάλει τη συνταγή πατώντας το W κουμπί. Για να συνεχίσεις με την εκτελέσει θα πρέπει να ξαναπατήσεις το ίδιο κουμπί.

Βήμα 4: Για να κόψεις τη πατάτα θα πρέπει να κάνεις την ίδια διαδικασία με το να ψιλοκόψεις τα άλλα υλικά αλλά αντί να πατήσεις στα βοηθητικά κουμπιά τα δύο μαχαίρια θα πατήσεις αυτό με το ένα μαχαίρι και έπειτα το σανιδάκι κοπής.

Βήμα 5: Θα πρέπει να τηγανίσεις πρώτα τα υλικά για να κάνεις τον κιμά και μετά να τηγανίσεις τη κομμένη πατάτα.

Βήμα 6: Για να τηγανίσεις τη πατάτα θα πρέπει να βάλεις το λάδι και τη πατάτα στο τηγάνι και μετά να πατήσεις τη μπλε σπάτουλα που βρίσκεται στα βοηθητικά κουμπιά.

Βήμα 7: Μόλις τηγανίσεις τη πατάτα θα μπορείς να στήσεις το πιάτο. Πρώτα θα πρέπει να επιλέξεις τη πατάτα από τη καταγραφή αποθεμάτων και να επιλέξεις μετά το πιάτο που είναι στο πάγκο στης κουζίνας. Ακολούθως, θα πρέπει να επιλέξεις το κιμά και μετά το πιάτο πάλι. Τέλος θα πρέπει να επιλέξεις τη πατάτα ξανά και το πιάτο ακολούθως για να τελειώσεις το στάδιο.

Βήμα 8: Μόλις στήσεις το μουσακά πατάτας θα εμφανιστεί ένα τελευταίο στάδιο όπου όταν κάνεις τη συνταγή στο σπίτι θα πρέπει να το ακολουθήσεις. Για να μεταβείς στα αποτελέσματα του σταδίου θα πρέπει να πατήσεις τη λέξει Results.

Βήμα 9: Εκεί θα μπορείς να πας στο επόμενο στάδιο αν υπάρχει ή να βγεις από το παιχνίδι.

Επιπρόσθετο βήμα: Σε οποιαδήποτε στιγμή στο παιχνίδι μπορείς να πατήσεις το ESC κουμπί και να παγώσεις το παιχνίδι. Επίσης, μπορείς να συνεχίσεις το παιχνίδι, τα πας στο κεντρικό μενού ή να βγεις από το παιχνίδι.

Κεφάλαιο 6

Συμπεράσματα

6.1 Μεταστατικά Συμπεράσματα 28
6.2 Αποτελέσματα από τη δοκιμή του παιχνιδιού 28
6.3 Μελλοντική Εργασία 30

6.1 Μεταστατικά Συμπεράσματα

Το παιχνίδι που έχω υλοποιήσει είναι εκπαιδευτικού περιεχομένου, γιατί ο παίχτης θα μπορεί να μάθει μέσω αυτού διάφορες συνταγές και να τις εφαρμόσει μετέπειτα στη καθημερινή του ζωή. Το «Something ancient is baked» έχει την δυσκολία που θα κάνει τον παίχτη να θέλει να μάθει και να αναπτύξει τις γνώσεις του στη κυπριακή κουζίνα. Αυτό πετυχαίνετε μέσω του βαθμολογικού συστήματος του παιχνιδιού το οποίο δίνει το απαραίτητο κίνητρο στο χρήστη για να ξαναπαίξει το παιχνίδι πετυχαίνοντας καλύτερη βαθμολογία και έτσι μαθαίνει και να αναγνωρίζει τα διάφορα υλικά που βρίσκονται σε ένα φαγητό.

6.2 Αποτελέσματα από τη δοκιμή του παιχνιδιού

Τα γενικά αποτελέσματα και εντυπώσεις από τη δοκιμή του παιχνιδιού ήταν πολύ θετικά με μερικές παρατηρήσεις όπως:

1) Θα ήταν καλό να υπήρχε ένα κουμπί «οδηγίες» το οποίο θα περιγράφει όλα τα διαδραστικά εργαλεία που υπάρχουν στο παιχνίδι και να επεξηγεί τη χρήση τους.
2) Να μην είναι τόσο περιοριστική η επιφάνεια αφής του χρήστη.
3) Τελειώνοντας το παιχνίδι θα ήταν καλό να εμφανιζόταν μια εικόνα με το πιάτο στην τελική του μορφή.

6.3 Μελλοντική Εργασία

Μελλοντικά θα έκανα μερικές αλλαγές στις κινήσεις του μάγειρα. Για παράδειγμα, να κάνει τη κίνηση να παίρνει τα αντικείμενα που βρίσκονται στην αποθήκη (pick up). Επίσης, θα έβαζα περισσότερες πίστες με πιο περίπλοκες συνταγές για να μπορεί ο χρήστης να εξοικειωθεί με τα φαγητά της κυπριακής κουζίνας. Θα έβαζα την επιλογή να μαγειρεύει μαζί με το χρήστη ακόμα ένας επιπλέων χαρακτήρας (π.χ. η Unity-chan).

Επίσης, θα πρόσθετα και άλλα υλικά στην αποθήκη για έχουμε περισσότερη ποικιλία συνταγών.

Επιπρόσθετα, θα έκανα πιο ρεαλιστική τη μαγειρική διαδικασία, τη παρουσίαση των φαγητών και το χρόνο που θα χρειάζεται ένα φαγητό να εκτελεστεί, ούτος ώστε να γνωρίζει ο χρήστης τον πραγματικό χρόνο που θα χρειάζεται η συγκεκριμένη συνταγή να εκτελεστεί. Θα πρόσθετα τα sound effects για τη κάθε εκτέλεση των κινήσεων του μάγειρα (π.χ. ήχος κοπής, ήχος τηγανίσματος).

Ακόμα, θα δημιουργούσα μια βάση δεδομένων για να μπορούν να φυλάγονται τα αποτέλεσμα του χρήστη για το κάθε στάδιο τα οποία θα φαίνονται με το όνομα που θα δίνει ο χρήστης στην αρχή του παιχνιδιού.

Ακολούθως, θα εφάρμοζα την τεχνική της βοηθητικής ένδειξης εφόσον ο χρήστης κάνει τρείς λανθασμένες επιλογές, κάνοντας λίγο πιο ευδιάκριτη τη σωστή επιλογή των υλικών.

Τέλος, θα έκανα το παιχνίδι να δουλεύει και σε πλατφόρμα για έξυπνες συσκευές και διαφορετικά λογισμικά συστήματα.

Βιβλιογραφία

[1] https://unity3d.com/learn/tutorials/projects/adventure-game-tutorial

[2] https://assetstore.unity.com/packages/3d/props/food/food-grocery-items-low-poly-75494

[3] https://assetstore.unity.com/packages/3d/props/food/medieval-food-pack-72561

[4] https://assetstore.unity.com/packages/3d/environments/kitchen-creation-kit-2854

[5] https://assetstore.unity.com/packages/3d/characters/chef-caricature-106799

[6] https://www.techradar.com/news/phone-and-communications/mobile-phones/will-samsung-3d-video-conversion-software-pop-up-in-galaxy-s4-1136772

Παράρτημα Α

GameCotroller.cs

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

public class GameCotroller : MonoBehaviour {

    public GameObject errorUI;
    public Condition fail;
    public GameObject helpUI;
    public Item[] items = new Item[Inventory.numItemSlots];
    public Item finalItem;
    public Inventory inventory;
    private bool[] bitems = new bool[Inventory.numItemSlots];
    private bool[] binv = new bool[Inventory.numItemSlots];
    public Condition[] invcon = new Condition[Inventory.numItemSlots];
    public int itemsLength;
    public string result;
    public Text resultText;
    public GameObject resultUI;
```

```
public Condition error1;
public Condition error2;
public Condition error3;
private float score;
private bool inbool;
private int correct;
private bool invCor;

private void Start()
{
    for (int i = 0; i < Inventory.numItemSlots; i++)
    {
        bitems[i] = false;
        binv[i] = false;
        invcon[i].satisfied = false;
    }
    fail.satisfied = false;
    itemsLength = Inventory.numItemSlots;
    result = "";
    score = 0;
    inbool = true;
    correct = 0;
    invCor = true;
    error1.satisfied = false;
```

```
        error2.satisfied = false;
        error3.satisfied = false;
    }

    private void Update()
    {
        if (inbool)
        {
            cheakInventory();
        }
        else
        {
            errorUI.SetActive(false);
            helpUI.SetActive(true);
            score = ((correct * 10) / itemsLength);
        }
        for (int i = 0; i < itemsLength; i++)
            if (inventory.items[i] == finalItem)
            {
                score += 90;
                result = "Your Score Is: " + score.ToString() + "%";
                resultText.text = result;
                resultUI.SetActive(true);
            }
```

```
}

void cheakInventory() {
    invCor = true;
    for (int i = 0; i < itemsLength; i++)
    {
        for (int j = 0; j < itemsLength; j++)
        {
            if (inventory.items[i] != null)
            {
                if (inventory.items[i].Equals(items[j]))
                {
                    binv[i] = true;
                    bitems[j] = true;
                }
            }
        }
    }
    if (fail.satisfied)
    {
        for (int w = 0; w < itemsLength; w++)
            if (bitems[w])
                correct++;
```

```
        for (int w = 0; w < itemsLength; w++)
            for (int z = 0; z < itemsLength; z++)
                if (!binv[w] && !bitems[z])
                {
                    inventory.items[w] = items[z];
                    inventory.itemImages[w].sprite = items[z].sprite;
                    inventory.itemImages[w].enabled = true;
                    bitems[z] = true;
                    binv[w] = true;
                }
        inbool = false;
    }
    invCorrect();
    return;
}

private void invCorrect()
{
    for (int i = 0; i < itemsLength; i++)
    {
        if (bitems[i] == false)
            invCor = false;
    }
```

```
        if (fail.satisfied == false && invCor)
        {
            correct = itemsLength;

            inbool = false;
        }

        return;
    }
}
```

RecipeCotroller.cs

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

public class RecipeCotroller : MonoBehaviour {

    public const int maxItems = 7;
    public Item[] items = new Item[maxItems];
    public Condition[] conditions = new Condition[maxItems];
    private Condition[] inConditions = new Condition[Inventory.numItemSlots];
    public Condition[] helpConditions = new Condition[8];
```

```
public Text[] recipeText = new Text[3];
public Inventory invetory;
private int maxInv = Inventory.numItemSlots;
public Color color;


    // Use this for initialization
    void Start () {
  for (int i = 0; i < maxInv; i++) {
     if (invetory.items[i] == null)
         inConditions[i] = null;
     conditions[i].satisfied = false;
  }
    }


    // Update is called once per frame
    void Update () {
  for (int i = 0; i < maxInv; i++)
  {
     if (invetory.items[i] != null)
     {
         for (int j = 0; j < maxItems; j++)
         {
             if (invetory.items[i] == items[j])
             {
```

```
                inConditions[i] = conditions[j];
                if (j == 4)
                {
                    recipeText[2].color = color;
                }
                else if (j == 5)
                {
                    recipeText[1].color = color;
                }
                else if (j == 6)
                {
                    recipeText[0].color = color;
                }
            }
        }
    }
  }

public void helpButtons(int i)
{
    helpConditions[i].satisfied = true;
}
```

```
    public void invetoryButton(int i)
    {
        inConditions[i].satisfied = true;
    }
}
```

Inventory.cs

```
using UnityEngine;
using UnityEngine.UI;

public class Inventory : MonoBehaviour
{
    public Image[] itemImages = new Image[numItemSlots];
    public Item[] items = new Item[numItemSlots];



    public const int numItemSlots = 4;



    public void AddItem(Item itemToAdd)
    {
        for (int i = 0; i < items.Length; i++)
        {
            if (items[i] == null)
```

```
            {
                items[i] = itemToAdd;
                itemImages[i].sprite = itemToAdd.sprite;
                itemImages[i].enabled = true;
                return;
            }
        }
    }


    public void RemoveItem (Item itemToRemove)
    {
        for (int i = 0; i < items.Length; i++)
        {
            if (items[i] == itemToRemove)
            {
                items[i] = null;
                itemImages[i].sprite = null;
                itemImages[i].enabled = false;
                return;
            }
        }
    }
}
```

Παράρτημα Β

GameCotoller1.cs

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

public class GameCotroller1 : MonoBehaviour {

    public GameObject errorUI;
    public Condition fail;
    public GameObject helpUI;
    public Item[] items = new Item[Inventory2.numItemSlots];
    public Item finalItem;
    public Inventory2 inventory;
    private bool[] bitems = new bool[Inventory2.numItemSlots];
    private bool[] binv = new bool[Inventory2.numItemSlots];
    public Condition[] invcon = new Condition[Inventory2.numItemSlots];
    public int itemsLength;
    public string result;
    public Text resultText;
    public GameObject finalStep;
```

```
public GameObject resultUI;
private float score;
private bool inbool;
private int correct;
public Condition error1;
public Condition error2;
public Condition error3;

private void Start()
{
    for (int i = 0; i < Inventory2.numItemSlots; i++)
    {
        bitems[i] = false;
        binv[i] = false;
        invcon[i].satisfied = false;
    }
    fail.satisfied = false;
    itemsLength = Inventory2.numItemSlots;
    result = "";
    score = 0;
    inbool = false;
    correct = 0;
    error1.satisfied = false;
    error2.satisfied = false;
```

```
    error3.satisfied = false;
}

private void Update()
{
    if (!inbool)
    {
        cheakInventory();
    }
    else
    {
        errorUI.SetActive(false);
        helpUI.SetActive(true);
        score = ((correct * 10) / itemsLength);
    }
    for (int i = 0; i < itemsLength; i++)
            if (inventory.items[i] == finalItem)
            {
                score += 90;
                result = "Your Score Is: " + score.ToString() + "%";
                resultText.text = result;
                finalStep.SetActive(true);
            }
}
```

```
void cheakInventory()
{
    for (int i = 0; i < itemsLength; i++)
    {
        for (int j = 0; j < itemsLength; j++)
        {
            if (inventory.items[i] != null)
            {
                if (inventory.items[i].Equals(items[j]))
                {
                    binv[i] = true;
                    bitems[j] = true;
                }
            }
        }
    }

    if (fail.satisfied)
    {
        for (int w = 0; w < itemsLength; w++)
            if (bitems[w])
                correct++;
```

```
        for (int w = 0; w < itemsLength; w++)
            for (int z = 0; z < itemsLength; z++)
                if (!binv[w] && !bitems[z])
                {
                    inventory.items[w] = items[z];
                    inventory.itemImages[w].sprite = items[z].sprite;
                    inventory.itemImages[w].enabled = true;
                    bitems[z] = true;
                    binv[w] = true;
                }
    }
    invCorrect();
    return;
}

private void invCorrect()
{
    for (int i = 0; i < itemsLength; i++)
    {
        if (!bitems[i])
            return;
    }

    if (!fail.satisfied)
```

```
            correct = itemsLength;


        inbool = true;
    }


    public void results()
    {
        finalStep.active = false;
        resultUI.active = true;
    }
}
```

RecipeCotroller1.cs

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


public class RecipeCotroller1 : MonoBehaviour {


    public const int maxItems = 12;
    public Item[] items = new Item[maxItems];
    public Condition[] conditions = new Condition[maxItems];
    private Condition[] inConditions = new Condition[Inventory2.numItemSlots];
```

```
public Condition[] helpConditions = new Condition[8];
public Text[] recipeText = new Text[5];
public Inventory2 invetory;
private int maxInv = Inventory2.numItemSlots;
public Color color;
private bool con5 = false;
private int in5;
private bool con8 = false;
private int in8;

    // Use this for initialization
    void Start () {
  for (int i = 0; i < maxInv; i++) {
     if (invetory.items[i] == null)
         inConditions[i] = null;
     conditions[i].satisfied = false;
  }
    }

    // Update is called once per frame
    void Update () {
  for (int i = 0; i < maxInv; i++)
  {
     if (invetory.items[i] != null)
```

```
{
    for (int j = 0; j < maxItems; j++)
    {
        if (invetory.items[i] == items[j])
        {
            inConditions[i] = conditions[j];
            if (j == 4)
            {
                con5 = true;
                in5 = i;
            }
            if (j == 10)
            {
                recipeText[0].color = color;
                con8 = true;
                in8 = i;
            }
            else if (j == 9)
            {
                recipeText[1].color = color;
            }
            else if (j == 11)
            {
                recipeText[3].color = color;
```

```
                }
            }
            if (con5 && invetory.items[in5] == null)
                recipeText[2].color = color;
            if (con8 && invetory.items[in8] == null)
                recipeText[4].color = color;
        }
    }
  }
   }

public void helpButtons(int i)
{
    helpConditions[i].satisfied = true;
}

public void invetoryButton(int i)
{
    inConditions[i].satisfied = true;
}
}
```

Inventory2.cs

```
using UnityEngine;
using UnityEngine.UI;

public class Inventory2 : MonoBehaviour
{
    public Image[] itemImages = new Image[numItemSlots];
    public Item[] items = new Item[numItemSlots];



    public const int numItemSlots = 6;



    public void AddItem(Item itemToAdd)
    {
        for (int i = 0; i < items.Length; i++)
        {
            if (items[i] == null)
            {
                items[i] = itemToAdd;
                itemImages[i].sprite = itemToAdd.sprite;
                itemImages[i].enabled = true;
                return;
```

```
            }
        }
    }



    public void RemoveItem(Item itemToRemove)
    {
        for (int i = 0; i < items.Length; i++)
        {
            if (items[i] == itemToRemove)
            {
                items[i] = null;
                itemImages[i].sprite = null;
                itemImages[i].enabled = false;
                return;
            }
        }
    }
}
```

Παράρτημα Γ

PlayerMovement.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.AI;
using UnityEngine.EventSystems;

public class PlayerMovement : MonoBehaviour
{
    public Animator animator;
    public NavMeshAgent agent;
    public float turnSmoothing = 15f;
    public float speedDampTime = 0.1f;
    public float slowingSpeed = 0.175f;
    public float turnSpeedThreshold = 0.5f;
    public float inputHoldDelay = 0.5f;

    private Interactable currentInteractable;
    private Vector3 destinationPosition;
    private bool handleInput = true;
    private WaitForSeconds inputHoldWait;

    private readonly int hashSpeedPara = Animator.StringToHash("Speed");
```

```
    private readonly int hashLocomotionTag = Animator.StringToHash("Locomotion");

    public const string startingPositionKey = "starting position";

    private const float stopDistanceProportion = 0.1f;
    private const float navMeshSampleDistance = 4f;

    private void Start()
    {
        agent.updateRotation = false;
        inputHoldWait = new WaitForSeconds (inputHoldDelay);
        destinationPosition = transform.position;
    }

    private void Update()
    {
        if (agent.pathPending)
            return;
        float speed = agent.desiredVelocity.magnitude;

        if (agent.remainingDistance <= agent.stoppingDistance * stopDistanceProportion)
            Stopping(out speed);
```

```
        else if (agent.remainingDistance <= agent.stoppingDistance)
            Slowing(out speed, agent.remainingDistance);
        else if (speed > turnSpeedThreshold)
            Moving();
        animator.SetFloat(hashSpeedPara, speed, speedDampTime,
Time.deltaTime);
    }

    private void Stopping(out float speed)
    {
#pragma warning disable CS0618 // Type or member is obsolete
        agent.Stop();
#pragma warning restore CS0618 // Type or member is obsolete
        transform.position = destinationPosition;
        speed = 0f;
        if (currentInteractable)
        {
            transform.rotation = currentInteractable.interactionLocation.rotation;
            currentInteractable.Interact();
            currentInteractable = null;
            StartCoroutine(WaitForInteraction());
        }
    }
```

```
    private void Slowing(out float speed, float distanceToDestination)
    {
#pragma warning disable CS0618 // Type or member is obsolete
        agent.Stop();
#pragma warning restore CS0618 // Type or member is obsolete
        float proportionalDistance = 1f - distanceToDestination / agent.stoppingDistance;
        Quaternion targetRotation = currentInteractable ? currentInteractable.interactionLocation.rotation : transform.rotation;
        transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, proportionalDistance);
        transform.position = Vector3.MoveTowards(transform.position, destinationPosition, slowingSpeed * Time.deltaTime);
        speed = Mathf.Lerp(slowingSpeed, 0f, proportionalDistance);
    }

    private void Moving()
    {
        Quaternion targetRotation = Quaternion.LookRotation(agent.desiredVelocity);
        transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, turnSmoothing * Time.deltaTime);
    }

    public void OnGroundClick(BaseEventData data)
    {
```

```
        if (!handleInput)
            return;

        currentInteractable = null;
        PointerEventData pData = (PointerEventData)data;
        NavMeshHit hit;
        if    (NavMesh.SamplePosition(pData.pointerCurrentRaycast.worldPosition,
out hit, navMeshSampleDistance, NavMesh.AllAreas))
            destinationPosition = hit.position;
        else
            destinationPosition = pData.pointerCurrentRaycast.worldPosition;
        agent.SetDestination(destinationPosition);
#pragma warning disable CS0618 // Type or member is obsolete
        agent.Resume();
#pragma warning restore CS0618 // Type or member is obsolete
    }

    public void OnInteractableClick(Interactable interactable)
    {
        if (!handleInput)
            return;
        currentInteractable = interactable;
        destinationPosition = currentInteractable.interactionLocation.position;
        agent.SetDestination(destinationPosition);
```

```
#pragma warning disable CS0618 // Type or member is obsolete
        agent.Resume();
#pragma warning restore CS0618 // Type or member is obsolete
    }


    private IEnumerator WaitForInteraction()
    {
        handleInput = false;
        yield return inputHoldWait;
        while (animator.GetCurrentAnimatorStateInfo(0).tagHash != hashLocomotionTag)
        {
            yield return null;
        }
        handleInput = true;
    }
}
```

Παράρτημα Δ

PauseMenu.cs

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

public class PauseMenu : MonoBehaviour {

    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;

    // Update is called once per frame
    void Update() {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }
    public void Resume()
```

```
        {
            pauseMenuUI.SetActive(false);
            Time.timeScale = 1f;
            GameIsPaused = false;
        }
    public void Pause ()
        {
            pauseMenuUI.SetActive(true);
            Time.timeScale = 0f;
            GameIsPaused = true;
        }
    public void LoadMenu()
    {
        SceneManager.LoadScene("main");
    }

    public void QuitGame()
    {
        Application.Quit();
    }
}
```

MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.Audio;
using UnityEngine.UI;
```

```
public class MainMenu : MonoBehaviour
{
public AudioMixer audioMixer;

    public Slider sl;

    public void Level1()
    {
        SceneManager.LoadScene("level1");
    }

    public void Level2()
    {
        SceneManager.LoadScene("level2");
    }


    public void SetVolume()
    {
        float volume = sl.value;
        audioMixer.SetFloat("Volume", volume);
    }

    public void QuitGame()
    {
        Application.Quit();
    }
}
```

CanvasScript.cs

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

public class CanvasScript : MonoBehaviour {

    public GameObject recipeUI;
    public GameObject errorUI;
    public GameObject ruleUI;
    public GameObject inventoryUI;
    public static bool GameIsPaused = true;

    void Update()
    {
#pragma warning disable CS0618 // Type or member is obsolete
        if (ruleUI.active && Input.GetKeyDown(KeyCode.Q))
#pragma warning restore CS0618 // Type or member is obsolete
        {
            GameIsPaused = false;
            ruleUI.SetActive(false);
            Time.timeScale = 1f;
            inventoryUI.SetActive(true);
        }

#pragma warning disable CS0618 // Type or member is obsolete
        if (!errorUI.active && Input.GetKeyDown(KeyCode.W))
#pragma warning restore CS0618 // Type or member is obsolete
        {
```

```
            if (GameIsPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }
    public void Resume()
    {
        recipeUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }
    void Pause()
    {
        recipeUI.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
    }


    public void Next()
    {
#pragma warning disable CS0618 // Type or member is obsolete
        if (SceneManager.Equals(SceneManager.GetActiveScene(),
SceneManager.GetSceneByBuildIndex(1)))
#pragma warning restore CS0618 // Type or member is obsolete
        {
            SceneManager.LoadScene("level2");
        }
```

```
        if (SceneManager.Equals(SceneManager.GetActiveScene(),
SceneManager.GetSceneByBuildIndex(2)))
        {


            SceneManager.LoadScene(0);
        }
    }
}
```