

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΤΕΧΝΩΝ ΗΧΟΥ ΚΑΙ ΕΙΚΟΝΑΣ

Σχεδιασμός, Ανάπτυξη και Έκδοση Δισδιάστατου Βιντεοπαιχνιδιού

Παπαδόπουλος Γεώργιος
Επιβλέπων Καθηγητής Παναγόπουλος Μιχαήλ

Κέρκυρα, Ιούνιος 2020

Συνοπτική Περίληψη

Η παρούσα εργασία αποτυπώνει όλα τα στάδια τα οποία χρειάζονται για την έκδοση ενός βιντεοπαιχνιδιού. Η αρχή γίνεται με τη παρουσίαση των κατηγοριών βιντεοπαιχνιδιών που υπάρχουν αυτή τη στιγμή στην αγορά και ακολουθεί το κομμάτι της υλοποίησης, από την αρχική ιδέα μέχρι την διάθεση του παιχνιδιού στο αγοραστικό κοινό. Αναφέρονται οι τεχνικές και τα μέσα που απαιτούνται, οι δυσκολίες που προκύπτουν και τρόποι αντιμετώπισης ενός τέτοιου είδους εγχειρήματος. Τέλος παρουσιάζονται οι αξιολογήσεις διαφόρων χρηστών που έπαιξαν το βιντεοπαιχνίδι, προτάσεις για περαιτέρω ανάπτυξη και παραρτήματα με το κώδικα και τα αρχεία που χρησιμοποιήθηκαν για την υλοποίηση του.

Abstract

This thesis captures all the steps needed to make a video game. The beginning is with the presentation of the video game categories that currently exist in the market and follows the part of the implementation, from the initial idea to the availability of the game to the buying public. The techniques and means required, the difficulties that arise and ways to deal with such an endeavor are mentioned. Finally, the evaluations of various users who played the video game, suggestions for further development and appendices with the code and the files used for its implementation are presented.

Περίληψη

Η πτυχιακή εργασία εκπονήθηκε στο τμήμα Τεχνών, Ήχου και Εικόνας του Ιονίου Πανεπιστημίου υπό την επίβλεψη του καθηγητή Παναγόπουλου Μιχαήλ. Το θέμα της εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η έκδοση ενός Δυσδιάστατου Βιντεοπαιχνιδιού το οποίο πέρα από το κομμάτι της διασκέδασης θα λειτουργεί και ως τουριστικός οδηγός για το νησί της Κέρκυρας. Το παιχνίδι, αν και προοριζόταν αρχικά για υπολογιστές, κατά την ολοκλήρωση του εκδόθηκε και για κινητά smartphone υπό την μορφή εφαρμογής. Το παιχνίδι συνοδεύεται από την επίσημη ιστοσελίδα του όπου μπορεί κανείς να το κατεβάσει είτε για υπολογιστή είτε για κινητό και tablet. Η σελίδα, η οποία συνδέεται άμεσα με το βιντεοπαιχνίδι, δημιουργήθηκε για την παροχή πληροφοριών και οπτικοακουστικού υλικού, όπως το διαφημιστικό βίντεο το οποίο προβάλλεται αυτόματα με την είσοδο του επισκέπτη. Τέλος δημιουργήθηκαν και οι αντίστοιχες σελίδες στα κοινωνικά δίκτυα Facebook και Instagram για την προώθηση και την αλληλεπίδραση των χρηστών του βιντεοπαιχνιδιού.

Το παιχνίδι διαδραματίζεται στη Κέρκυρα και ακολουθεί τον ήρωα με το όνομα Σπύρος στην προσπάθεια του να εξερευνήσει τοποθεσίες του νησιού. Για να φέρει εις πέρας την κάθε αποστολή καλείται να προσπεράσει διάφορα εμπόδια, να αποφύγει ή να εξουδετερώσει εχθρούς, παγίδες και κινδύνους, να συλλέξει τα χρυσά νομίσματα και όλα αυτά σε όσο το δυνατόν λιγότερο χρόνο. Με μια μπάρα να δείχνει συνέχεια το υπόλοιπο της ζωής που του απομένει, ξεκινάει τη περιπέτεια του, όπου στη πορεία θα ανακαλύψει και μπουκάλια από το παραδοσιακό λικέρ κουμκουάτ να του προσφέρουν την χαμένη ενέργεια. Η κάθε πίστα είναι αφιερωμένη και σε μια τοποθεσία του νησιού με την αρχή να γίνεται από τη παραλία της Γαρίτσας, να ακολουθεί το παλαιό Φρούριο και το τρίτο επίπεδο να φέρνει τον ήρωα σε μια παραλία όπου θα έχει τη δυνατότητα να ανακαλύψει και το θαλάσσιο κόσμο.

Για την υλοποίηση κάθε πτυχής του παιχνιδιού χρειάστηκε και το αντίστοιχο λογισμικό. Η κύρια πλατφόρμα όπου στήθηκε το παιχνίδι είναι η Unity όπου πραγματοποιήθηκε ο σχεδιασμός όλων των επιπέδων, τα animation και η σύνδεση όλων των στοιχείων μεταξύ τους. Ο κώδικας γράφτηκε στο Visual Studio σε γλώσσα C# και στη συνέχεια εντάσσεται πάνω στα γραφικά για να αποκτήσουν και την ανάλογη ιδιότητα. Τα γραφικά χωρίζονται σε δύο κατηγορίες, τα γραφικά

περιβάλλοντος και τα γραφικά χαρακτήρων. Η πρώτη κατηγορία αποτελείται από φωτογραφίες οι οποίες τραβήχτηκαν από τη πόλη της Κέρκυρας ενώ η δεύτερη κατηγορία κυρίως από διαδικτυακές πηγές. Όλα επεξεργάστηκαν στο λογισμικό του Photoshop μέχρι να φτάσουν στη τελική τους μορφή. Οι ήχοι του παιχνιδιού επεξεργάστηκαν στο Reaper και το Wavelab ενώ το βίντεο για το διαφημιστικό trailer στο Adobe After Effect. Η σελίδα του παιχνιδιού δημιουργήθηκε στην online πλατφόρμα Wix όπου ο χρήστης μπορεί να κατεβάσει το παιχνίδι ανά πάσα στιγμή.

Στη συνέχεια ακολούθησε το κομμάτι της προώθησης και της διαφήμισης. Το παιχνίδι είχε ως στόχο την διασκέδαση του χρήστη και την διαφήμιση της Κέρκυρας μέσα από αυτό. Είναι κατάλληλο για όλες τις ηλικίες καθώς δεν περιέχει ακατάλληλο περιεχόμενο ενώ ακόμα και οι εχθροί που εμφανίζονται είναι σχεδιασμένοι με τέτοιο τρόπο ώστε να είναι αποδεκτοί και από τα παιδιά δίχως να προκαλούν φόβο. Το πιο δύσκολο κομμάτι, αποτέλεσε ο προγραμματισμός που όταν ολοκληρώθηκε, το κομμάτι του Level design αποτέλεσε το πιο διασκεδαστικό τμήμα του παιχνιδιού.

Περιεχόμενα

Συνοπτική Περίληψη	2
Abstract	2
Κεφάλαιο 1: Εισαγωγή	7
Πεδίο Έρευνας	7
Στόχος της Εργασίας	8
Ορισμοί.....	8
Κεφάλαιο 2: Γενικές έννοιες και κατηγορίες Παιχνιδιών	9
Η έννοια του Παιχνιδιού	9
Κατηγορίες Βιντεοπαιχνιδιών	10
Παιχνίδια Περιπέτειας	10
Παιχνίδια Δράσης	12
Παιχνίδια Ρόλων (RPG)	14
Παιχνίδια Στρατηγικής	16
Παιχνίδια Προσομοίωσης.....	17
Παιχνίδια Αθλητισμού.....	19
Παιχνίδια Εκπαίδευσης	20
Παιχνίδια Διαδικτύου	21
Κεφάλαιο 3: Προ-Παραγωγή	24
Ιδέα	24
Σενάριο	25
Στόχος.....	27
Κεφάλαιο 4: Παραγωγή (Production).....	28
Σχεδιασμός	28
Game Design Document.....	28
Σχεδιασμός Επιπέδων	31
Προγραμματισμός	32
Κώδικας	32
User Interface	34
Προγραμματιστής Ήχου	36
Λογισμικό Προγραμματισμού	36
Γραφικά.....	38
Τμήμα Γραφικών	38
Art Director.....	38
2D Artist	39

Animation	40
Λογισμικά Γραφικών.....	42
Ήχος	44
Voice Over.....	44
Ηχητικά Εφέ	45
Μουσική – Soundtrack	45
Λογισμικό Ήχου	46
Κεφάλαιο 5: Μετα-Παραγωγή (Post Production).....	47
Δοκιμές.....	47
Έλεγχος Λειτουργικότητας (Debugging)	47
Playtesting	47
Αξιολόγηση	49
Ανάλυση & Συμπεράσματα Αξιολόγησης	53
Έκδοση.....	54
Ψηφιακή Μορφή.....	54
Εγχειρίδιο Χρήσης.....	55
Προώθηση	57
Δημιουργία Ιστοσελίδας (Web Design)	57
Digital Marketing	59
Κεφάλαιο 6: Συμπεράσματα	63
Κεφάλαιο 7: Προτάσεις	64
Βιβλιογραφία	65
Πηγές Εικόνων.....	66
Παραρτήματα.....	67
Παράρτημα Α: Προγραμματισμός	67

Κεφάλαιο 1: Εισαγωγή

Πεδίο Έρευνας

Η εργασία αποτελεί αντικείμενο μελέτης πολλών και διαφορετικών τμημάτων. Το πρώτο είναι η μετατροπή μιας ιδέας σε σενάριο ικανό να υλοποιηθεί για βιντεοπαιχνίδι. Το δεύτερο είναι η υλοποίηση του σεναρίου μέσα από κώδικα, γραφικά και animation. Στο πεδίο του προγραμματισμού η έρευνα αφορούσε την δημιουργία κώδικα στον οποίο εμπεριέχονται μεταβλητές, συναρτήσεις, πίνακες και αλγόριθμοι με σκοπό να δώσουν ζωή στα γραφικά και να τα καθιστούν. Τα γραφικά είναι ειδικά σχεδιασμένα ώστε να προσφέρουν την αισθητική που απαιτείται. Τα χρώματα του περιβάλλοντος, τα σκίτσα, οι χαρακτήρες ακόμα και οι εκφράσεις των ηρώων αποτελούν προϊόν μελέτης και σε συνδυασμό με τον ήχο δημιουργούν το επιθυμητό οπτικοακουστικό περιβάλλον. Ιδιαίτερη προσοχή δόθηκε και στο animation του ήρωα αφού μελετήθηκαν παραδείγματα κίνησης του ανθρώπινου σώματος για ένα όσο το δυνατόν πιο ρεαλιστικό αποτέλεσμα.

Η έκδοση για κινητές συσκευές αποτέλεσε ένα μικρό παράρτημα στο οποίο πραγματοποιήθηκε έρευνα ώστε να είναι συμβατή η έκδοση του παιχνιδιού σε όσο το δυνατόν περισσότερες συσκευές. Πέρα από τη συμβατότητα της έκδοσης και των γραφικών πρόκληση αποτέλεσε και η ορθή απεικόνιση λόγω της τεράστιας ποικιλίας που υπάρχει στις διαστάσεις οθονών των κινητών τηλεφώνων και tablet. Τέλος για τη προώθηση του παιχνιδιού πραγματοποιήθηκε μελέτη για την επιλογή των κατάλληλων μέσων κοινωνικής δικτύωσης σύμφωνα με το απευθυνόμενο κοινό. Για τις επιπλέον διαφημίσεις πραγματοποιήθηκε έρευνα βασισμένη στις ηλικιακές ομάδες, το φύλλο αλλά και τη γεωγραφική περιοχή των χρηστών στους οποίους απευθύνεται το παιχνίδι.

Στόχος της Εργασίας

Βασικός στόχος της εργασίας είναι ο σχεδιασμός, η ανάπτυξη και η έκδοση ενός βιντεοπαιχνιδιού που θα προσφέρει στιγμές διασκέδασης και θα λειτουργεί παράλληλα ως ταξιδιωτικός οδηγός παρακινώντας το χρήστη να γνωρίσει μέρη και τοποθεσίες της Κέρκυρας. Αν ο χρήστης δεν έχει επισκεφθεί τις τοποθεσίες αυτές, μέσα από το παιχνίδι κατά την ολοκλήρωση της κάθε πίστας θα έχει τη δυνατότητα να του παρέχονται πληροφορίες για να δει αυτά τα μέρη από κοντά, ενώ στην αντίθετη περίπτωση θα βλέπει τοπία γνώριμα με μια διαφορετική οπτική, μέσα από τον κόσμο του παιχνιδιού.

Ορισμοί

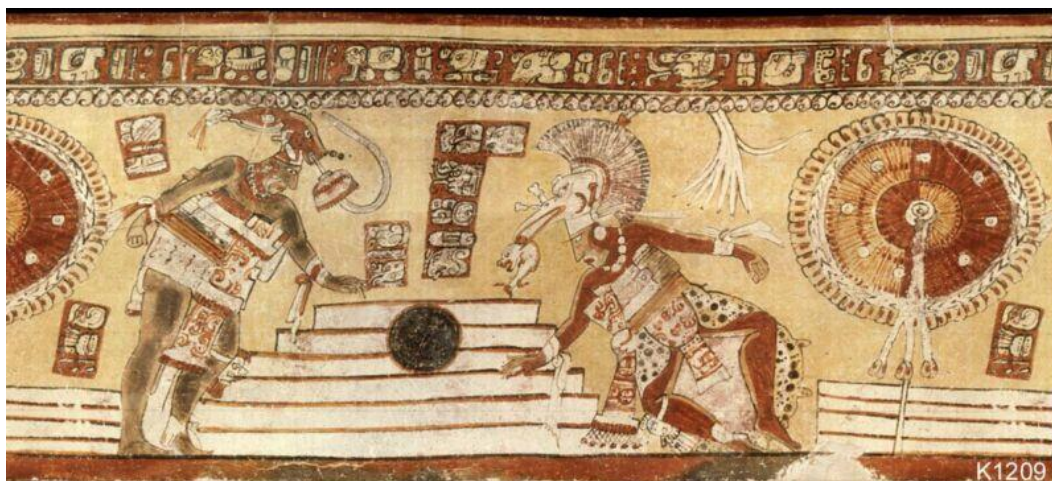
Rigging: Ονομάζεται η διαδικασία κατά την οποία η ομάδα του animation εφαρμόζει ένα ψηφιακό σκελετικό σύστημα επάνω στο μοντέλο.

FPS (Frame per Second): Είναι η συντομογραφία για τα πόσα καρέ, δηλαδή εικόνες, εμφανίζονται ανά δευτερόλεπτο. Αν για παράδειγμα ο ρυθμός μετάδοσης είναι 50 FPS, αυτό πρακτικά σημαίνει ότι 50 εικόνες εναλλάσσονται σε διάστημα ενός δευτερολέπτου.

Κεφάλαιο 2: Γενικές έννοιες και κατηγορίες Παιχνιδιών

Η έννοια του Παιχνιδιού

Τα παιχνίδια αποτελούν θεμελιώδες στοιχείο της ανθρώπινης φύσης. Αν και στην εποχή μας χαρακτηρίζεται πολλές φορές ως κάτι το ανούσιο ή επιπόλαιο και η σκέψη πάει στις νεαρές ηλικίες η πραγματικότητα είναι ότι αποτελεί ένα από τα κύρια στοιχεία της ανθρώπινης φύσης. Ακόμα και από την εποχή των σπηλαίων οι άνθρωποι έπαιζαν παιχνίδια. Δηλαδή πραγματοποιούσαν δραστηριότητες σε ατομικό είτε ομαδικό επίπεδο με σκοπό τη διασκέδαση τους. Για την έννοια του παιχνιδιού έχουν δοθεί πολλοί ορισμοί αλλά δεν υπάρχει κάποια επίσημη εκδοχή. Μία αρκετή επιστημονική προσέγγιση είναι αυτή που προσπάθησαν να δώσουν οι Elliot Avedon και Brian Sutton-Smith περιγράφοντας το παιχνίδι ως “ μία άσκηση εθελοντικών συστημάτων ελέγχου, όπου υπάρχει ένας διαγωνισμός μεταξύ δυνάμεων που περιορίζεται από κανόνες προκειμένου να υπάρχει στο τέλος ένα αποτέλεσμα”. Συνεπώς ένα παιχνίδι για να δημιουργηθεί χρειάζεται αρχικά τη συμμετοχή παικτών η οποία γίνεται με τη θέληση τους και όχι εξ αναγκαστικά. Επιπλέον ένα παιχνίδι πρέπει να έχει στόχους, σύγκρουση, κανόνες και ένα τελικό αποτέλεσμα.¹



Εικόνα 1.1 Άνθρωποι παίζουν παιχνίδι στην Μεσοποταμία πριν από 3500 χρόνια.

¹ Jesse Schell. *The Art of Game Design 2nd Edition*. (US: CRC Press, 2015), 41.

Κατηγορίες Βιντεοπαιχνιδιών

Παιχνίδια Περιπέτειας

Τα παιχνίδια περιπέτειας βασίζονται κατά κύριο λόγο πάνω σε μια ιστορία και ένα σενάριο. Πρόκειται για παιχνίδια όπου ο χρήστης καλείται να λύσει γρίφους και κουίζ για να προχωρήσει στα επόμενα στάδια. Τα πρώτα παιχνίδια σε αυτή τη κατηγορία αποτελούνταν μόνο από κείμενο με το χρήστη να πληκτρολογεί την δράση που έχει επιλέξει να κάνει. Στη συνέχεια εξελίχθηκαν σε παιχνίδια γραφικών δύο διαστάσεων και πλέον ακόμα περισσότερο με την έλευση των τρισδιάστατων γραφικών, όπου οι δράσεις πλέον γίνονται χρησιμοποιώντας το ποντίκι, δίχως όμως να εξαφανίσουν απαραίτητα τις πρωταρχικές μορφές οι οποίες και αυτές εξελίχθηκαν με τη πάροδο των χρόνων. Γενικά τα παιχνίδια αυτής της κατηγορίας δεν αποτελούν παιχνίδια πραγματικού χρόνου, καθώς ο χρήστης συνήθως έχει όσο χρόνο χρειάζεται να λύσει το γρίφο μέχρι να προχωρήσει στο επόμενο στάδιο. Πρόκειται για παιχνίδια που απευθύνονται σε κοινό που του αρέσει να εξερευνάει μεγάλες πίστεις, να αλληλεπιδρά με χαρακτήρες, να συλλέγει αντικείμενα και πληροφορίες μέχρι να φτάσει στο τελικό στόχο με όλα τα παραπάνω να βασίζονται σε μία καλογραμμένη ιστορία.²

Η ιστορία σε ένα παιχνίδι περιπέτειας αποτελεί το πρωταρχικό στοιχείο από το οποίο θα εξαρτηθεί το πόσο καλό θα είναι το τελικό αποτέλεσμα με το χρήστη να ταυτίζεται όσο το δυνατόν περισσότερο με τον ήρωα του παιχνιδιού. Αφού η ιστορία έχει γραφτεί, οι σχεδιαστές καλούνται να μεταφέρουν τα εμπόδια και τους γρίφους από το χαρτί στο κόσμο του παιχνιδιού. Κάθε γρίφος πρέπει να εμφανίζεται για συγκεκριμένο λόγο κατά τη διάρκεια του παιχνιδιού και να αποτελεί μία ευκαιρία για περεταίρω αφήγηση και εξέλιξη της ιστορίας, δίχως να έχουν σκοπό να προσφέρουν απλά ανούσιες περεταίρω ώρες gameplay. Κατά τη διάρκεια του παιχνιδιού ο παίκτης καλείται να λύσει γρίφους οι οποίοι είτε είναι αυτόνομοι είτε συνδυάζουν στοιχεία από προηγούμενους γρίφους που έλυσε ο ήρωας. Για αυτό το λόγο, ο χρήστης θα πρέπει να έχει πάντα τα απαραίτητα στοιχεία για να τους λύσει, με το επίπεδο δυσκολίας να κλιμακώνεται ανάλογα και με τη πρόοδο του παιχνιδιού, ενώ πάντα πρέπει να

² Crawford Cris. *The Art of Computer Game Design*. (Washington: Washington State University, 1997), 31.

λαμβάνεται υπόψιν και το κοινό στο οποίο απευθύνεται το παιχνίδι. Ένα παιχνίδι περιπέτειας για παιδιά ακολουθεί άλλα πρότυπα με διαφορετικό βαθμό δυσκολίας σε σχέση με ένα παιχνίδι που απευθύνεται σε ενήλικες. Για αυτό το λόγο θα πρέπει οι δοκιμασίες να είναι τέτοιου επιπέδου ώστε τα παιδιά να μην αποθαρρύνονται από τυχόν υπερβολικό βαθμό δυσκολίας και οι ενήλικες να μη το θεωρούν εύκολο και βαρετό σε περίπτωση απλοϊκών γρίφων.³



Εικόνα 2.1 Στιγμιότυπο από το παιχνίδι περιπέτειας “Monster Hunter World” .

³ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 49.

Παιχνίδια Δράσης

Πρόκειται για τη πιο δημοφιλή κατηγορία παιχνιδιών και με τους περισσότερους τίτλους με το κύριο χαρακτηριστικό τους είναι η δράση σε πραγματικό χρόνο με το χρήστη να δοκιμάζει τα γρήγορα αντανακλαστικά του. Οι κύριες δεξιότητες που απαιτούνται είναι ο όσο το δυνατόν μικρότερος χρόνος αντίδρασης σε ότι οπτικό ερέθισμα παρουσιάζεται στην οθόνη και ο σωστός συντονισμός ματιών και χεριών. Η αδρεναλίνη φτάνει στα ύψη καθώς Ιδιαίτερη έμφαση δίνουν οι κατασκευαστές στον ήχο και τα γραφικά για όσο το δυνατόν περισσότερο ρεαλιστικό αποτέλεσμα ενώ η χρήση χειριστηρίων η μοχλών είναι που έχει προβάδισμα έναντι του κλασικού πληκτρολογίου. Αν και σαν κατηγορία, μπορεί να ενταχθεί πλήθος παιχνιδιών, από αθλητικά, προσομοίωσης, πίστας κ.α. με τον όρο παιχνίδια δράσης αναφερόμαστε κυρίως σε παιχνίδια μάχης και έντονης δραστηριότητας με τη κάμερα κατά κύριο λόγο σε πρώτο η τρίτο πρόσωπο, που έχουν ως σκοπό να ο παίκτης να βρίσκεται διαρκώς σε κίνηση και να εξελίσσεται καθ' όλη τη διάρκεια του παιχνιδιού.

4

Η τοποθέτηση της κάμερας αποτελεί βασικό παράγοντα για την εμπειρία που θα βιώσει ο παίκτης παίζοντας το παιχνίδι. Όταν η κάμερα βρίσκεται σε πρώτο, πρόσωπο, είναι τοποθετημένη στο κεφάλι του χαρακτήρα, με το χρήστη να βλέπει στην οθόνη του μέσα από τα μάτια του ήρωα που ελέγχει. Αυτό έχει σαν αποτέλεσμα το παιχνίδι να είναι πιο γρήγορο, καθώς ο παίκτης γίνεται ένα με τον κόσμο του παιχνιδιού. Αντίθετα, τα παιχνίδια τρίτου προσώπου, όπου η κάμερα είναι τοποθετημένη πάνω από το παίκτη, δίνουν τη δυνατότητα στο χρήστη να δει το χαρακτήρα του εν δράση μέσα από μια πιο σφαιρική οπτική γωνία. Πέρα από την οπτική γωνία, η κάθε μία περίπτωση έχει τα θετικά και τα αρνητικά χαρακτηριστικά της. Στα παιχνίδια τρίτου προσώπου ο χρήστης έχει στραμμένη τη προσοχή του συνεχώς στον ήρωα που κινεί, αυτό απαιτεί ένα μοντέλο με περισσότερα πολύγωνα και εντυπωσιακά animation, με αποτέλεσμα ο υπόλοιπος κόσμος του παιχνιδιού να υστερεί σε γραφικά, κάτι που στα παιχνίδια πρώτου προσώπου συμβαίνει ακριβώς το αντίθετο.⁵

Πέρα από τη κάμερα και την έντονη ροή, ένα από τους κύριους μοχλούς σε ένα παιχνίδι δράσης αποτελούν τα όπλα και ο εξοπλισμός του εκάστοτε χαρακτήρα. Ο

⁴ Crawford Cris. *The Art of Computer Game Design*. (Washington: Washington State University, 1997), 26.

⁵ Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 178.

ήρωας ξεκινάει τις περισσότερες φορές με ένα απλό αντικείμενο για όπλο η ακόμα και με γυμνά χέρια ώστε να μάθει σιγά σιγά τον τρόπο με τον οποίο θα τα χειρίζεται. Το σύστημα των όπλων απαιτεί μεγάλη προσοχή στο τρόπο δόμησης του καθώς οφείλει να εξελίσσεται παράλληλα με τη πρόοδο του παίχτη ώστε να αποτελεί κίνητρο για τον ίδιο να ξεκλειδώσει όσα περισσότερα μπορεί. Ένα όπλο το οποίο εξολοθρεύει άμεσα έναν εχθρό καθιστά το παιχνίδι εύκολο με αποτέλεσμα ο παίκτης να χάσει το ενδιαφέρον του όπως και το ακριβώς αντίθετο όπου ένα αδύναμο όπλο δεν είναι ικανό να νικήσει τον εχθρό. Όσο προχωράει το παιχνίδι ο παίκτης καλείται να δημιουργήσει το κατάλληλο συνδυασμό όπλων και εξοπλισμού για να φέρει εις πέρας τη κάθε αποστολή καθώς και τη σωστή χρήση τους όσον αφορά την εξοικονόμηση πυρομαχικών και όποιον άλλων αντικειμένων διαθέτουν μαζί τους. Πέρα από το σχεδιαστικό κομμάτι όπου δίνεται ιδιαίτερη έμφαση στην εμφάνιση του κάθε όπλου εξίσου σημαντικό είναι και τα υπόλοιπα ειδικά εφέ όπως και ο ήχος που το συνοδεύει κάτι που απαιτεί τη συνεργασία πολλών τμημάτων μιας εταιρείας παραγωγής βιντεοπαιχνιδιών για ένα ολοκληρωμένο αποτέλεσμα.⁶



Εικόνα 2.2 Στιγμιότυπο από το παιχνίδι δράσης “Call of Duty: Black Ops”

⁶ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 41.

Παιχνίδια Ρόλων (RPG)

Στα παιχνίδια ρόλων σκοπός του παίκτη είναι να δημιουργήσει το δικό του χαρακτήρα από το μηδέν. Για να το πετύχει αυτό καλείται να ολοκληρώσει διάφορες αποστολές, να λύσει γρίφους και να ταξιδέψει σε ένα τεράστιο χάρτη. Έχει τη δυνατότητα να συλλέξει όπλα και πανοπλίες για κάθε μέρος του σώματος του, τα οποία καλείται να αναβαθμίζει συνεχώς. Στα παιχνίδια ρόλων ο χρήστης κατά το αρχικό στάδιο καλείται συνήθως να επιλέξει με ποια φυλή θέλει να πορευθεί (άνθρωποι, ξωτικά, νάνοι κτλπ) και στη συνέχεια να διαλέξει την ιδιότητα που θα ακολουθήσει ο χαρακτήρας του (μάγος, τοξότης, μονομάχος). Ανάλογα με τις επιλογές που θα κάνει θα έχει στη διάθεση του τις ανάλογες μαγικές ιδιότητες και δυνάμεις που θα τις χρησιμοποιεί για να φέρει εις πέρας τις αποστολές του. Ένα ακόμα χαρακτηριστικό των RPG παιχνιδιών είναι το σύστημα πόντων, οικονομίας και αναβαθμίσεων που διαθέτουν. Ο χρήστης, καλείται μέσα από τη σωστή διαχείριση των πόρων που διαθέτει, να επιλέξει μεθοδικά ανάμεσα σε διάφορες παραμέτρους ώστε να επιτύχει το μέγιστο δυνατό αποτέλεσμα. Για παράδειγμα θα χρειαστεί να συλλέξει αντικείμενα για να τα πουλήσει ώστε να αποκτήσει κάτι που έχει ανάγκη ή να αναβαθμίσει ένα από τα υπάρχοντα κομμάτια του πολεμικού του εξοπλισμού.⁷



Εικόνα 2.3 Το παιχνίδι Evil Lands διαθέτει πλούσιο UI και έχει εκδοθεί για κινητά

⁷ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 7.

Όσον αφορά το σχεδιαστικό κομμάτι τα παιχνίδια ρόλων αποτελούν μία από τις πιο σύνθετες κατηγορίες παιχνιδιών όπου για τη δημιουργία τους απαιτούνται συνήθως μεγάλες ομάδες σχεδιαστών, προγραμματιστών, animators κτλπ λόγω του τεράστιου όγκου δεδομένων και των ατελείωτων ωρών gameplay. Εξίσου απαιτητικό ρόλο έχουν και οι game designers καθώς ο πιο σημαντικός παράγοντας σε ένα RPPG παιχνίδι είναι το σενάριο. Όλα αυτά γίνονται ακόμα πιο απαιτητικά όταν τα τελευταία χρόνια η εμφάνιση των MMORPG (Massively Multiplayer Online RPG) γίνεται ακόμα πιο διαδεδομένη. Τα MMORPG είναι τα RPG παιχνίδια τα οποία είναι πλέον διαδικτυακά και ο κάθε χρήστης μπορεί να αλληλεπιδρά με άλλους μέσα από τον κόσμο του παιχνιδιού. Πλέον οι αποστολές μπορούν να γίνουν και ομαδικές πέρα από ατομικές για την επίτευξη των στόχων, ενώ ο χρήστης καλείται να ανταγωνιστεί τους άλλους χρήστες μέσα από συστήματα μάχης. Πέρα από το σύστημα μάχης υπάρχει αλληλεπίδραση μεταξύ των χρηστών και σε θέματα πώλησης ή αγοράς αντικειμένων, αλλά και η συμμετοχή σε τουρνουά ή ομάδες συμμαχίας.⁸



Εικόνα 2.4 Στιγμιότυπο από το MMORPG παιχνίδι Lineage 2

⁸ Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 188.

Παιχνίδια Στρατηγικής

Ο πιο σημαντικός παράγοντας στα παιχνίδια στρατηγικής είναι η ισορροπία. Όλα πρέπει να είναι μελετημένα και σχεδιασμένα ώστε να υπάρχει η απόλυτη ισορροπία μέσα στο παιχνίδι μεταξύ των αντιπάλων. Ο κάθε παίκτης ανεξάρτητα από το τι μονάδες έχει επιλέξει για να ξεκινήσει το παιχνίδι του πρέπει να έχει τις ίδιες πιθανότητες νίκης απέναντι στον αντίπαλο του. Όσον αφορά τα σενάρια στα παιχνίδια στρατηγικής, αναφέρονται συνήθως είτε σε πραγματικές μάχες του παρελθόντος, είτε σε ιστορίες μυθοπλασίας με συγκρούσεις ανάμεσα σε δυο στρατούς. Ο χρήστης επιλέγει με πιο στρατό θα παραταχθεί και στη συνέχεια καλείται να συλλέξει πόρους για τη δημιουργία των μονάδων του και των βάσεων του. Κάθε μονάδα έχει τα πλεονεκτήματα και τα μειονεκτήματά της τα οποία πρέπει να γνωρίζει ο χρήστης ώστε να επωφεληθεί για να εξουδετερώσει το απέναντι στράτευμα. Εκτός από τα παιχνίδια στρατηγικής που διαδραματίζονται σε πραγματικό χρόνο υπάρχουν και αυτά στα οποία ο χρήστης επιλέγει το σύνολο των μονάδων του πριν τη μάχη και δεν τις δημιουργεί κατά τη διάρκεια αυτής.⁹



Εικόνα 2.5 Χαρακτηριστικό στα παιχνίδια στρατηγικής είναι η πανοραμική θέα του χάρτη

⁹ Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 182.

Πέρα από την ισορροπία μεταξύ των ομάδων ο σχεδιαστής του παιχνιδιού έχει άλλες δύο ενότητες στις οποίες πρέπει να καλύψει με απόλυτη ακρίβεια, αυτή της τεχνητής νοημοσύνης και του Interface. Στη τεχνητή νοημοσύνη απαιτείται μια άριστη συνεργασία μεταξύ του προγραμματιστή και του game designer για να βγει το επιθυμητό αποτέλεσμα. Ο προγραμματιστής έχει πιο εύκολο έργο να του δοθούν λεπτομερώς όλες οι πληροφορίες για το πώς θα αντιδρούν οι εκάστοτε μονάδες σε διάφορες συνθήκες. Παρόλα αυτά ο προγραμματισμός των μονάδων απαιτεί μεγάλη ακρίβεια ώστε το παιχνίδι να είναι επιτυχημένο. Αν οι αντίπαλες μονάδες χάνουν πολύ εύκολα ο παίκτης θα βαρεθεί ενώ αν είναι εξαιρετικά δύσκολο να τις κερδίσει θα απογοητευτεί. Εκτός από την επιλογή του εκάστοτε στρατεύματος, εξίσου στρατηγικής σημασίας είναι και ο χώρος που λαμβάνεις χώρα η δράση. Λόφοι, ποτάμια, βουνά, δάση και λίμνες είναι καθοριστικοί παράγοντες στην εξέλιξη της μάχης και ο παίκτης οφείλει να τα εκμεταλλευτεί.¹⁰

Παιχνίδια Προσομοίωσης

Τα παιχνίδια προσομοίωσης συνδυάζουν τη διασκέδαση με εμπειρίες που προσφέρονται στην πραγματική ζωή αλλά παρόλα αυτά είναι δύσκολο για πολλούς να τις ζήσουν. Παιχνίδια με θέμα το free running, το ψάρεμα, το paintball μέχρι και το να βρίσκεσαι στο πιλοτήριο ενός αεροσκάφους είναι μερικοί τίτλοι που θα συναντήσει κανείς σε αυτή την ανερχόμενη κατηγορία των τελευταίων ετών. Για παράδειγμα σε μια προσομοίωση αυτοκινήτου, η ομάδα σχεδιασμού δίνει βάση στη μεταφορά δεδομένων του πραγματικού κόσμου, των νόμων της φυσικής και των υλικών κατασκευής του οχήματος στο προγραμματισμό του παιχνιδιού. Συνεπώς οι καιρικές συνθήκες για παράδειγμα μπορούν να καθορίσουν την ολισθηρότητα του δρόμου σε περίπτωση βροχής, το είδος των ελαστικών, οι αεροτομές, τα καύσιμα ή ο κινητήρας την απόδοση του αυτοκινήτου και το τερέν της πίστας αν είναι άσφαλτος ή χώμα το πώς αντιδράει το όχημα σε αυτό. Όλα αυτά είναι στοιχεία που πρέπει να μελετηθούν για την ορθότερη και ρεαλιστικότερη απόδοση του παιχνιδιού.¹¹

¹⁰ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 54.

¹¹ Bethke Erik, *Game Development and Production* (Texas: Wordware Publishing, 2003) 10.

Στα παιχνίδια προσομοίωσης το σενάριο δεν αποτελεί προτεραιότητα και κυρίως περνάει σε δεύτερη μοίρα. Η ιστορία συνήθως μεταφέρεται στο παίχτη μέσω ορισμένων cutscenes που τοποθετούνται κατά τη διάρκεια ή στο τέλος ενός επιπέδου. Ο κύριος εχθρός που έχει να αντιμετωπίσει ο παίκτης είναι ο χρόνος τον οποίο και αντιλαμβάνεται είτε άμεσα μέσα από μια αντίστροφη μέτρηση για ολοκλήρωση της πίστας είτε έμμεσα όταν θα πρέπει να επιτύχει καλύτερους χρόνους από τους πιθανούς αντιπάλους του. Για να το πετύχει αυτό πέρα από τις δεξιότητες που απαιτούνται, ο παίχτης μαζεύοντας πόντους μπορεί να αναβαθμίσει τον υπάρχοντα εξοπλισμό του ή να αγοράσει καλύτερο όσο ανεβαίνει το επίπεδο δυσκολίας. Πολλές φορές η κατανομή των παιχνιδιών προσομοίωσης συμπίπτει και με άλλες κατηγορίες όπως αυτή των σπορ και της εκπαίδευσης καθώς πολλά παιχνίδια έχουν έντονο το αθλητικό στοιχείο ενώ σε άλλες περιπτώσεις τα χρησιμοποιούν ακόμα και σε πιλότους κατά τη διάρκεια της εκπαίδευσης. Σημαντικό στοιχείο που έρχεται να προστεθεί είναι η επαυξημένη πραγματικότητα όπου με τη χρήση συγκεκριμένου εξοπλισμού ο χρήστης μπορεί να ζήσει την εμπειρία της προσομοίωσης σε ακόμα μεγαλύτερο επίπεδο.¹²



Εικόνα 2.6 Προσομοιωτής παιχνιδιού αυτοκινήτων

¹² Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 191

Παιχνίδια Αθλητισμού

Είναι μια κατηγορία παιχνιδιών για τους λάτρεις των αθλημάτων και των σπορ. Πρόκειται για παιχνίδια στα οποία ο χρήστης δεν παρακολουθεί απλά το αγαπημένο του άθλημα αλλά γίνεται ένα με αυτό είτε από τη θέση του μάνατζερ, είτε του προπονητή αλλά κυρίως από τη θέση του ίδιου του αθλητή. Στα παιχνίδια που αφορούν τα ομαδικά αθλήματα ο χρήστης έχει συνήθως μια πανοραμική θέαση του γηπέδου που εξελίσσονται ενώ στα ατομικά η κάμερα είναι σε πρώτο ή τρίτο πρόσωπο. Κύριο γνώρισμα αυτών των παιχνιδιών είναι η ρεαλιστικότητα που παρουσιάζει στις κινήσεις και τα σώματα των αθλητών και απώτερος σκοπός είναι να εντάξουν το χρήστη όσο το δυνατόν περισσότερο μέσα στο ίδιο το παιχνίδι. Εκτός από τα παιχνίδια που βασίζονται σε πραγματικά στατιστικά αθλημάτων υπάρχουν και αυτά όπου για λόγους περαιτέρω διασκέδασης ο αθλητής διαθέτει υπερφυσικές δυνατότητες και οι αγώνες γίνονται όχι στα γήπεδα αλλά σε φανταστικά ή ασυνήθιστα μέρη.¹³

Οι προγραμματιστές του παιχνιδιού θα πρέπει να γνωρίζουν όλους τους κανόνες και τις παραμέτρους του αθλήματος για την ορθή απόδοση του αν και υπάρχει η δυνατότητα στα περισσότερα πλέον παιχνίδια να δίνεται η επιλογή στο χρήστη να μπορεί να τροποποιεί τους κανόνες του παιχνιδιού στα δικά του θέλω. Τέτοιες αλλαγές μπορούν να γίνουν για παράδειγμα στο χρόνο των αγώνων, στους πόντους κτλπ. Λόγω ότι τα περισσότερα παιχνίδια σχετίζονται άμεσα με επίσημους αθλητικούς φορείς ιδιαίτερη προσοχή πρέπει να δοθεί από την ομάδα ανάπτυξης στα πνευματικά δικαιώματα των ομάδων και των αθλητών που εμφανίζονται στο παιχνίδι. Τέλος αλλά εξίσου σημαντικό είναι η κατάλληλη επιλογή του ηχητικού υλικού που μπορεί να γίνει μέσα από ηχογραφήσεις σταδίων ώστε να ακούγονται τα κατάλληλα επιφωνήματα η αποδοκίμασιες ανάλογα την εξέλιξη του παιχνιδιού και την απόδοση της ομάδας ή του αθλητή.¹⁴



Εικόνα 2.7 Στιγμιότυπο από το παιχνίδι ποδοσφαίρου FIFA 10

¹³ Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 190

¹⁴ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 62.

Παιχνίδια Εκπαίδευσης

Ο στόχος ενός εκπαιδευτικού παιχνιδιού είναι να δώσει ερεθίσματα στο χρήστη και να του διδάξει το περιεχόμενο για το οποίο έχει δημιουργηθεί. Ο Game Designer θα πρέπει να είναι σε πλήρη επικοινωνία με έναν ειδικό ώστε να τον κατευθύνει και να τον συμβουλεύει σε θέματα εκπαιδευτικού περιεχομένου αλλά και της εκάστοτε ενότητας. Εκπαιδευτικά παιχνίδια μπορούν να χρησιμοποιηθούν για όλες τις ηλικίες και για διαφορετικές περιστάσεις δίχως να προορίζονται αποκλειστικά και μόνο για παιδιά. Μια εταιρεία μπορεί να χρησιμοποιήσει ένα τέτοιο είδους παιχνίδι για να εκπαιδεύσει τους υπαλλήλους της, ένα παιχνίδι μπορεί να αποτελεί μέσο διαφήμισης ενός προϊόντος ή μέσο ενημέρωσης για ένα κοινωνικό ζήτημα ενώ μέχρι και ο στρατός τα χρησιμοποιεί για προσομοίωση ασκήσεων στους στρατιώτες.¹⁵

Η δομή ενός εκπαιδευτικού παιχνιδιού οφείλει να είναι ξεκάθαρη όχι μόνο στο εκπαιδευτικό υλικό αλλά και στο τρόπο λειτουργίας της. Τα κουμπιά πρέπει να είναι μεγάλα και ο χειρισμός εύκολος. Αν το παιχνίδι απευθύνεται σε παιδιά μικρής ηλικίας, τότε το περιβάλλον πρέπει να είναι έτσι διαμορφωμένο ώστε να νιώθουν ασφάλεια, αυτό επιτυγχάνεται με τη χρήση των κατάλληλων χρωμάτων και μουσικής. Επιπλέον, ο δημιουργός, είναι πολύ σημαντικό πριν την έκδοση του παιχνιδιού να δει από κοντά πως αλληλοεπιδρούν τα παιδιά με το παιχνίδι, ώστε να βγάλει τα απαραίτητα συμπεράσματα και να προβεί στις αναγκαίες διορθώσεις. Σε αυτά τα παιχνίδια το σύστημα πόντων δεν έχει τόσο μεγάλη αξία. Είναι προτιμότερο η επιβράβευση να γίνεται μέσα από ενδείξεις με συμβουλές και χρήσιμες πληροφορίες ενώ δεν ενδείκνυται στα λάθη να τον τιμωρεί αλλά να τον ενθαρρύνει να συνεχίσει.¹⁶



Εικόνα 2.8
Εκπαιδευτικό παιχνίδι
σε σχολική αίθουσα

¹⁵ Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 194

¹⁶ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 82.

Παιχνίδια Διαδικτύου

Με τον όρο Online Games αναφερόμαστε στη κατηγορία παιχνιδιών όπου οι χρήστες συνδέονται μεταξύ τους μέσω διαδικτύου από τη κονσόλα ή τον υπολογιστή τους. Πρόκειται περισσότερο για τρόπο σύνδεσης παρά για κατηγορία παιχνιδιών, χαρακτηρίζει δηλαδή με ποιο μέσο οι χρήστες αλληλοεπιδρούν μεταξύ τους και δεν αναφέρεται στο Gameplay του παιχνιδιού καθώς αυτό μπορεί να είναι το οτιδήποτε, από παιχνίδια στρατηγικής, RPG μέχρι και προσομοίωσης. Σε αυτή τη κατηγορία εντάσσονται και τα παιχνίδια τοπικού δικτύου με τη σύνδεση αυτή να είναι γνωστή στους χρήστες ως LAN (Local Area Network). Αξίζει να σημειωθεί ότι ο παραπάνω όρος δεν έχει καμία σχέση με τα παιχνίδια όπου πολλοί παίκτες είναι συνδεδεμένοι στην ίδια συσκευή και παρακολουθούν την ίδια οθόνη όλοι μαζί. Όλα αυτά διαμορφώνουν ορισμένα θετικά και αρνητικά χαρακτηριστικά για τους δημιουργούς των παιχνιδιών είτε για τους χρήστες.¹⁷



Εικόνα 2.9 Στο Lord of the Rings Online ο χρήστης έχει chat box και μια σειρά από εργαλεία και εντολές με τις οποίες μπορεί να αλληλεπιδρά με άλλους χρήστες που βρίσκονται online.

¹⁷ Adams Ernest. *Fundamentals of Game Design, 2nd Edition*. (Berkeley, CA: Pearson Education Inc, 2010), 591

Ένα από τα πλεονεκτήματα που προσφέρουν τα παιχνίδια διαδικτύου είναι οι ευκαιρίες για κοινωνικοποίηση μεταξύ των χρηστών. Η επικοινωνία πραγματοποιείται με ποικίλους τρόπους ανάλογα το είδος του παιχνιδιού αλλά και την ηλικιακή ομάδα στην οποία απευθύνεται. Στις μικρές ηλικίες η επικοινωνία γίνεται με προεπιλεγμένα μηνύματα απλών χαιρετισμών και εκφράσεων για να αποφευχθεί το ενδεχόμενο να χρησιμοποιηθούν απρεπείς χαρακτηρισμοί. Σε παιχνίδια μεγαλύτερης δυσκολίας υπάρχει που απαιτούν και περισσότερη συνεργασία μεταξύ των χρηστών η επικοινωνία γίνεται σε γραπτό λόγο μέσω chat, είτε ακόμα και με τη χρήση κάμερας και μικροφώνου για μεγαλύτερη ευκολία καθώς ενδέχεται ο χρήστης να μην έχει τη δυνατότητα να πληκτρολογεί και να παίζει συγχρόνως, ειδικά σε παιχνίδια δράσης όπου η ταχύτητα αποτελεί καθοριστικό παράγοντα. Αυτή η δυνατότητα αποτελεί και έναν από τους κύριους παράγοντες όπου οι γυναίκες προτιμάνε τα online Games έναντι των άλλων single player παιχνιδιών. Για τους σχεδιαστές των online Game η δομή του συστήματος επικοινωνίας των χρηστών αποτελεί ισάξιο κομμάτι με αυτό των γραφικών, του προγραμματισμού και της τεχνητής νοημοσύνης.¹⁸

Η τεχνητή νοημοσύνη των online Games αντικαθίσταται στις περισσότερες περιπτώσεις από τους ίδιους τους παίκτες καθώς έχουν να αναμετρηθούν μεταξύ τους και όχι με το σύστημα όπως γίνεται στα παιχνίδια για ένα χρήστη. Εξ αίρεση αποτελούν οι Non Player Characters (NPC), δηλαδή χαρακτήρες που ελέγχονται από τον υπολογιστή και είναι τοποθετημένοι για να παρέχουν υπηρεσίες και να καθοδηγούν το παίκτη κατά τη διάρκεια του παιχνιδιού. Πολλές φορές χαρακτήρες του υπολογιστή αναλαμβάνουν και το ρόλο του εχθρού όπου οι χρήστες καλούνται να συνεργαστούν, να ανταλλάξουν αντικείμενα και να καταστρώσουν στρατηγικά σχέδια μεταξύ τους για να τους κερδίσουν. Με αυτό τον τρόπο αναπτύσσεται η συνεργασία, η ομαδικότητα και η συλλογική προσπάθεια για την επίτευξη στόχων, ο κάθε χρήστης έχει το ρόλο του και συμβάλει διαφορετικά στην ολοκλήρωση των αποστολών. Τέλος, αλλά εξίσου σημαντικό είναι ότι σε παιχνίδια που υποστηρίζουν την συμμετοχή ακόμα και χιλιάδων παικτών ανά τον κόσμο ο χρήστης έχει τη δυνατότητα οποιαδήποτε στιγμή της ημέρας επιθυμεί να συνδεθεί να βρει άτομα συνδεδεμένα.¹⁹

¹⁸ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 89.

¹⁹ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 90.

Για την επίτευξη όλων αυτών η ομάδα ανάπτυξης και σχεδιασμού έρχεται αντιμέτωπη με δεκάδες προβλήματα που πρέπει να αντιμετωπίσει. Ένα από αυτά είναι οι ταχύτητες δικτύου που έχουν οι χρήστες, καθώς κάθε χρήστης ανάλογα τη χώρα και το πακέτο σύνδεσης που διαθέτει έχει και την ανάλογη ταχύτητα. Αυτό μεταφράζεται ως ταχύτερη μεταφορά δεδομένων και πληροφοριών. Συνεπώς η κατάλληλη κωδικοποίηση, η συμπίεση αρχείων και η δημιουργία ισχυρού Server αποτελεί σημείο αναφοράς για την δημιουργία του παιχνιδιού. Σε online παιχνίδια που ο κάθε παίκτης περιμένει τη σειρά του για να παίξει δεν αποτελεί πρόβλημα, δεν συμβαίνει το ίδιο όμως και στα παιχνίδια δράσης, ρόλων κτλ, όπου η ταχύτητα είναι σημαντικός παράγοντας ενάντια στους αντιπάλους. Δυσκολίες όμως αντιμετωπίζουν και οι χρήστες, καθώς όπως συνδέονται στο παιχνίδι ανά πάσα στιγμή έτσι μπορούν και να εξαφανιστούν. Μια τυχόν διακοπή της σύνδεσης δεν αποτελεί μεγάλο πρόβλημα σε ένα online παιχνίδι αγώνων ταχύτητας καθώς το όχημα του χρήστη απλά θα αφαιρεθεί από τη πίστα. Δεν συμβαίνει όμως το ίδιο όταν ο παίκτης αποτελεί μέρος ομάδας και εξαφανιστεί απρόσμενα αφήνοντας έτσι την ομάδα του σε μειονεκτική θέση.²⁰

²⁰ Adams Ernest. *Fundamentals of Game Design, 2nd Edition*. (Berkeley, CA: Pearson Education Inc, 2010), 598.

Κεφάλαιο 3: Προ-Παραγωγή

Ιδέα

Ο σχεδιασμός ενός Βιντεοπαιχνιδιού είναι ένας συνδυασμός δύο εντελώς διαφορετικών κόσμων, αυτών της τέχνης και της τεχνολογίας όπου ενώνονται για την υλοποίηση της κεντρικής ιδέας. Κύριος στόχος του σχεδιαστή βιντεοπαιχνιδιών είναι να μετατρέψει μια ιδέα σε ένα βιντεοπαιχνίδι δίνοντας της μορφή, πλοκή, στόχους, ροή και προκαλώντας συγχρόνως έντονα συναισθήματα στον παίκτη. Η κεντρική ιδέα μπορεί να είναι εμπνευσμένη από το οτιδήποτε. Προσωπικά βιώματα του σχεδιαστή, ιστορικά γεγονότα, βιβλία ή ακόμα και ιστορίες φανταστικού περιεχομένου είναι μερικά παραδείγματα που τείνουν να κυριαρχούν σε πολλούς τίτλους βιντεοπαιχνιδιών ως πυρήνας της ιστορίας που διαδραματίζεται μέσα από αυτά.²¹

Αφού ο σχεδιαστής έχει τη κεντρική ιδέα στο μυαλό του, το επόμενο στάδιο είναι να την αναπτύξει για να διαπιστώσει κατά πόσο είναι εφικτό να δημιουργηθεί ένα βιντεοπαιχνίδι μέσα από αυτή. Για να γίνει αυτό, το πρώτο βήμα είναι να συντάξει ένα κείμενο όπου θα παρουσιάζει με γραπτό λόγο τον ήρωα του παιχνιδιού, το χώρο όπου διαδραματίζεται καθώς και μια γενική περιγραφή από το περιβάλλον όπου επρόκειτο να συναντήσει (εχθρούς, δυσκολίες). Σε αυτό το πρωταρχικό στάδιο, μια συμβουλή από κοντινά άτομα είναι πάντα χρήσιμη στο δημιουργό για να λάβει τις πρώτες εντυπώσεις, θετικές είτε αρνητικές. Εφόσον έχει δομηθεί το βασικό σκέλος και μπορεί να θεωρηθεί υλοποιήσιμο, ακολουθεί το σενάριο όπου η ιδέα αποκτάει ροή, αρχή, μέση και τέλος.²²

Εικόνα 3.1

Όλα ξεκινάνε με μια ιδέα
και την αποτύπωση της στο χαρτί



²¹ Scott Rogers. *Level Up. The Guide to great video Game Design*. (United Kingdom: Wiley, 2014), 30.

²² Feil John, Scattergood Marc. *Beginning Game Level Design*. (Boston: Thomson Course Technology, 2005), 26.

Σενάριο

Το σενάριο έρχεται να παρουσιάσει και να αναλύσει την ιδέα με μεγαλύτερη σαφήνεια. Προσδίδει το βασικό ρόλο του χαρακτήρα, χωρίζει το παιχνίδι σε επιμέρους επίπεδα, παρουσιάζει αναλυτικά τους εχθρούς και τις δυσκολίες που καλείται να αντιμετωπίσει ο παίκτης και χωρίζεται σε 3 πράξεις. Σύμφωνα με το βιβλίο του Joseph Campbell, *Το ταξίδι του Ήρωα*, στη πρώτη πράξη γίνεται η αναχώρηση για τη περιπέτεια με τον ήρωα να είναι αντιμέτωπος με μια πρόκληση ή ένα πρόβλημα που πρέπει να λύσει. Η πρώτη πράξη απαιτεί την άμεση παρουσίαση της κατάστασης δίχως μεγάλη εισαγωγή με τη δράση να ξεκινάει σχεδόν άμεσα. Ακολουθεί η δεύτερη πράξη όπου ο ήρωας έχοντας αφήσει πίσω τη προηγούμενη ζωή του μπαίνει σε ένα νέο κόσμο όπου λαμβάνει δράση, σε αυτό φτάνει στο κατώτερο σημείο αλλά καταφέρνει να ανακάμψει και να ξεπεράσει όλα τα εμπόδια που θα του παρουσιαστούν. Σημαντικό σε αυτό το στάδιο είναι να φανεί η αλλαγή που έχει προκληθεί στο χαρακτήρα μέσα από τη πρόοδο του, για να ακολουθήσει η τρίτη και τελευταία πράξη όπου ο ήρωας αντιμετωπίζει την ύστατη και δυσκολότερη πρόκληση και αφού τα καταφέρει επιστρέφει στη πρωταρχική του κατάσταση, τελείως διαφορετικός όμως από ότι ήταν όταν ξεκίνησε το ταξίδι του.²³

Στο παιχνίδι, το οποίο διαδραματίζεται στη Κέρκυρα υπάρχουν τρία διαφορετικά επίπεδα όπου το καθένα προβάλλει και μια ξεχωριστή τοποθεσία του νησιού. Το πρώτο στάδιο ξεκινάει με το παίκτη στη περιοχή της Γαρίτσας, όπου ως πρώτη και αναγνωριστική πίστα είναι μικρής δυσκολίας, με τους εχθρούς να είναι είτε οι περαστικοί του δρόμου είτε πριόνια που αναδύονται από το έδαφος ενώ στο τέλος υπάρχει και ένα μικρό κανόνι. Σκοπό έχει να εισάγει το παίχτη στο περιβάλλον του παιχνιδιού με μικρής δυσκολίας εμπόδια και εύκολο μονοπάτι ώστε να εξοικειωθεί με το gameplay του παιχνιδιού. Τα αντικείμενα που μπορεί να συλλέξει είναι μικρά χρυσά νομίσματα για πόντους καθώς και ένα μπουκάλι κουμκουάτ για την μπάρα υγείας. Για να φτάσει ως το τέλος εκτός από τους εχθρούς και τα εμπόδια που καλείται να αποφύγει, ο παίχτης θα πρέπει να παραμείνει πάνω στη στεριά και να μην έρθει σε επαφή με τη

²³ Lee Sheldon. *Character development and storytelling*. (Boston: Thomson Course Technology, 2004), 25.

θάλασσα και τερματίζει όταν ακουμπήσει το μεγάλο χρυσό νόμισμα που βρίσκεται στον ανεμόμυλο.

Η δεύτερη πίστα που είναι και υψηλότερης δυσκολίας λαμβάνει χώρα στη περιοχή του Παλαιού Φρουρίου κατά τη διάρκεια της νύχτας. Η διαδρομή ξεκινάει από το υψηλότερο σημείο του φρουρίου και καταλήγει στο ναό του Αγίου Γεωργίου. Περισσότερα κανόνια, νυχτερίδες, φαντάσματα και καρφιά εδάφους είναι οι εχθροί που καλείται να αντιμετωπίσει πλέον ο ήρωας, ενώ το μονοπάτι πλέον απαιτεί μεγαλύτερη ακρίβεια στο στις κινήσεις του. Τα νομίσματα και το κουμκουάτ εμφανίζονται σε αυτό όπως και στα υπόλοιπα επίπεδα, δίνοντας πόντους και ενέργεια στο παίκτη σε κομβικά σημεία. Στη Τρίτη και τελευταία πίστα ο παίκτης εμφανίζεται σε μια από τις παραλίες του νησιού, ενώ σε αντίθεση με τη πρώτη, υπάρχει μια τροποποίηση στους κανόνες του χώρου, καθώς πλέον του δίνεται η δυνατότητα να κολυμπήσει και στη θάλασσα. Οι νέοι εχθροί αυτή τη φορά είναι ψάρια του βυθού και υποθαλάσσιες νάρκες που θα προσπαθήσουν να τον εμποδίσουν να φτάσει στο τρίτο και τελευταίο μεγάλο χρυσό νόμισμα.



Εικόνα 3.2 Στιγμιότυπο από το πρώτο επίπεδο του παιχνιδιού

Στόχος

Πέρα από το βασικό στόχο ο οποίος επικεντρώνεται στην ανακάλυψη τουριστικών και όχι μόνο προορισμών της Κέρκυρας το παιχνίδι χωρίζεται και σε επιμέρους στόχους. Αυτοί είναι ο χρήστης να τερματίσει τη πίστα, είτε συλλέγοντας όσο περισσότερα νομίσματα μπορεί, είτε στο γρηγορότερο δυνατό χρόνο, είτε έναν συνδυασμό των δύο παραπάνω όπου θα του αποφέρει το μεγαλύτερο άθροισμα πόντων. Επιπλέον θα πρέπει να ανακαλύψει τη πινακίδα με τον αναγραφόμενο κωδικό, ώστε να ξεκλειδώσει τις πληροφορίες που χρειάζεται στην επίσημη ιστοσελίδα του παιχνιδιού και να ξεκλειδώσει πληροφορίες για τη τοποθεσία της πίστας που μόλις έπαιξε.

Κεφάλαιο 4: Παραγωγή (Production)

Σχεδιασμός

Game Design Document

Το φύλλο σχεδιασμού του βιντεοπαιχνιδιού (GDD-Game Design Document) πρόκειται για σημείο αναφοράς που ανατρέχουν όλα τα μέλη των ομάδων που θα εργαστούν για τη δημιουργία του βιντεοπαιχνιδιού ενώ παράλληλα αποτελεί εργαλείο μνήμης και επικοινωνίας. Πολλοί είναι αυτοί που θα ψάξουν να βρουν φόρμες για το πώς δομείται ένα Design Document αλλά η πραγματικότητα είναι ότι κάτι τέτοιο δεν υπάρχει. Όπως κάθε παιχνίδι έχει διαφορετικά χαρακτηριστικά και διαφορετικές απαιτήσεις το ίδιο συμβαίνει και για το έγγραφο που το δομεί με αποτέλεσμα να μην υπάρχουν αυτοματοποιημένες φόρμες για κάτι τέτοιο. Φυσικά και υπάρχουν κάποια δομικά στοιχεία που συμβαδίζουν με τις περισσότερες περιπτώσεις δίχως όμως αυτό να θέτει όρια και περιορισμούς. Ενδεικτικά ένα Design Document αποτελείται από:

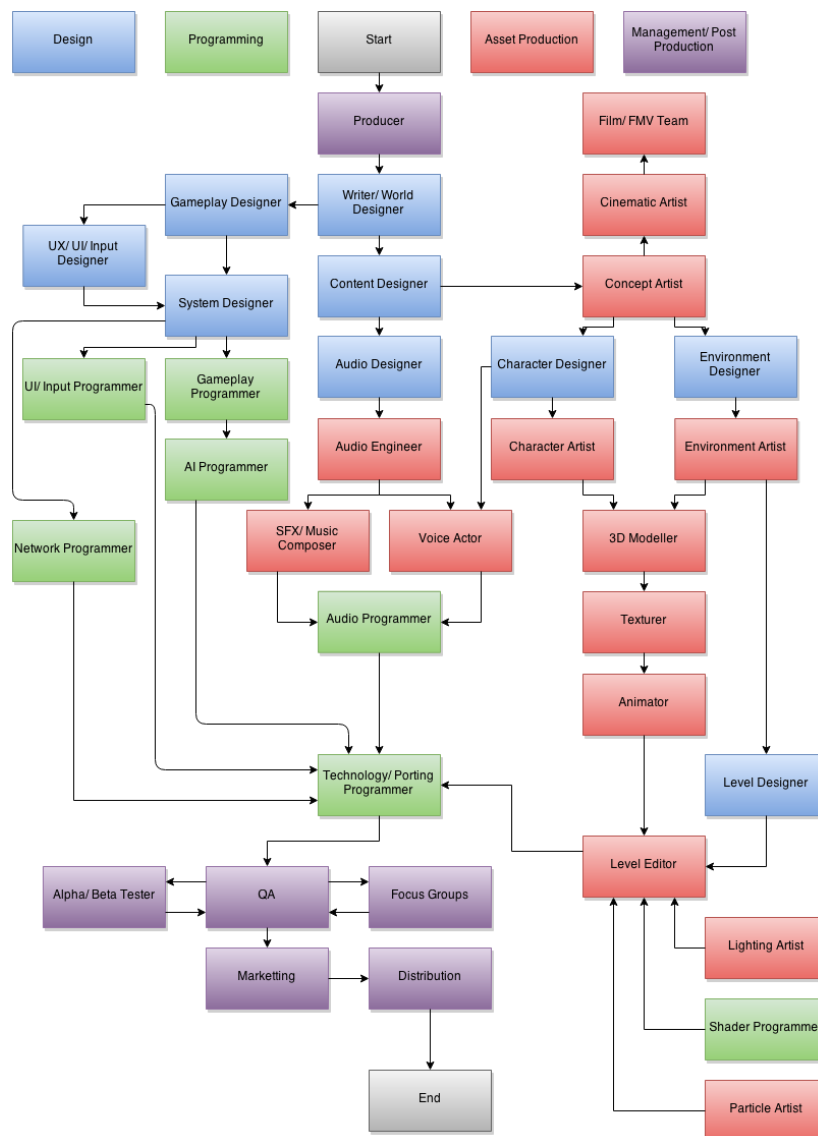
- *Περιεχόμενα:* Πολλές φορές κάτι που θεωρείται απλό και δεδομένο παραλείπετε. Ένας πίνακας περιεχομένων, ειδικά αν το Φύλλο σχεδιασμού αποτελείται από πολλές σελίδες, είναι άκρως απαραίτητος ώστε να μπορεί ο καθένας να ανατρέχει στο σημείο που τον ενδιαφέρει άμεσα, κυρίως όταν ο χρόνος πιέζει.
- *Περίληψη:* Πρέπει να περιορίζεται σε μία σελίδα και να παρουσιάζει μια γενική εικόνα του παιχνιδιού. Χρήσιμη για όποιον θέλει να κατανοήσει τα θέλω του σχεδιαστή, μέσα σε λίγα λεπτά ώστε να μπει άμεσα στο κλίμα του παιχνιδιού. Αποτελείται από μια σύντομη περιγραφή της ιστορίας που διαδραματίζεται και των κόσμων που θα εξερευνήσει ο ήρωας, του βασικούς μηχανισμούς του Gameplay και κλείνει με τρόπο δυναμικό και ενθουσιώδες ώστε να αποτελεί κίνητρο για οποιοδήποτε μέλος της ομάδας το διαβάσει ώστε να δώσει το καλύτερο του εαυτό.
- *Παραγωγή:* Παρουσιάζεται ένα αρχικό χρονοδιάγραμμα, το διαθέσιμο budget και οι στόχοι που έχουν ανατεθεί σε κάθε ομάδα να φέρει εις πέρας. Λόγω το

ότι στη δημιουργία ενός τόσο πολυδιάστατου project συνήθως υπάρχουν απρόοπτες αλλαγές, το αρχικό χρονοδιάγραμμα πιθανόν να ενημερώνεται συνεχώς όπου βρίσκεται αναρτημένο.

- *Gameplay*: Αποτελεί ίσως το κυριότερο δομικό στοιχείο του Design Document και περιγράφει πως παίζεται το παιχνίδι και τι επιτρέπεται στον παίκτη να κάνει. Σε τι πλατφόρμα θα λειτουργεί, πόσων διαστάσεων θα είναι, περιορισμούς και οριοθετήσεις των γραφικών, πως κινείται ο χαρακτήρας, με τι κουμπιά, πως χάνει και κερδίζει ενέργεια και πόντους, τι εχθρούς επρόκειτο να αντιμετωπίσει και πως μπορεί να τους εξολοθρεύσει.
- *Τεχνητή Νοημοσύνη*: Αν κατά το προηγούμενο κεφάλαιο του Gameplay γίνεται η περιγραφή για το πώς ο παίκτης αλληλεπιδρά με τον κόσμο, σε αυτό το κεφάλαιο περιγράφεται πως ο κόσμος και το περιβάλλον του παιχνιδιού αντιδρούν στις ενέργειες του παίκτη. Πως αντιδράει ο εχθρός όταν λαμβάνει επίθεση από το παίκτη, τι συμβαίνει στο κόσμο όταν ο χαρακτήρας μας δεν κινείται καθόλου
- *Στοιχεία του Παιχνιδιού*: Σε αυτή την ενότητα παρουσιάζονται τα στοιχεία που δομούν το παιχνίδι και χωρίζονται σε τρεις κατηγορίες: τους χαρακτήρες, τα αντικείμενα και τους μηχανισμούς του παιχνιδιού. Στους χαρακτήρες συμπεριλαμβάνονται όλοι οι εχθροί αλλά και οι προσωπικότητες με τις οποίες θα έρθει σε επαφή ο ήρωας δίχως όμως να μπορούν να ελεγχθούν από το χρήστη. Στα αντικείμενα κατατάσσονται όλα τα στοιχεία τα οποία ο παίκτης μπορεί να συλλέξει, να φορέσει ή να χρησιμοποιήσει γενικότερα, ενώ τέλος στους μηχανισμούς του παιχνιδιού αναφέρονται στοιχεία με τα οποία ο παίκτης μπορεί να αλληλεπιδράσει μαζί τους με κάποιες ενέργειες, όπως τραμπολίνα, πόρτες κτλπ, αλλά δεν είναι διαθέσιμα για να τα αποκτήσει.
- *Πρόοδος του Παιχνιδιού*: Σε ένα παιχνίδι πρέπει συνεχώς ο χαρακτήρας να εξελίσσεται μέσα από τη ροή του παιχνιδιού. Σε αυτή την ενότητα περιγράφεται για κάθε πίστα ξεχωριστά όλες οι προκλήσεις που πρόκειται να αντιμετωπίσει ο χαρακτήρας, πως διαμορφώνεται το περιβάλλον γύρω του, τι εχθρούς πρόκειται να αντιμετωπίσει, με τι χαρακτήρες θα έρθει σε επαφή, τι αντικείμενα πρόκειται να συλλέξει, σε ποιο σημείο θα τα εντοπίσει και πως θα τα αποκτήσει. Τι ρυθμό θα έχει η πίστα, θα πρόκειται για πίστα αναγνωριστική ή με έντονο συνεχόμενο περιεχόμενο; Πρόκειται για το πιο μακροσκελές τμήμα του Design

Document όπου περιγράφει πως το σενάριο και η ιστορία ζωντανεύουν μέσα από το βιντεοπαιχνίδι, ενώ δίνεται ιδιαίτερη βαρύτητα στα συναισθήματα που θα πρέπει να περάσουν στο χρήστη, οι ομάδες γραφικών και προγραμματισμού, όπως τα έχει φανταστεί ο δημιουργός.

- *Σύστημα-Μενού*: Αποτελεί μια ενότητα όπου αναφέρεται η λειτουργία του Μενού, το Interface όπως και το σύστημα πλοήγησης σε αυτό. Περιγράφει πως ο παίκτης μπορεί να αποθηκεύσει το παιχνίδι ή να κάνει ανάκτηση του από προηγούμενη αποθήκευση, πως μπορεί να αλλάξει τις ρυθμίσεις των γραφικών, του ήχου και οποιοδήποτε άλλων διαθέσιμων ρυθμίσεων. Αποτελεί μια ενότητα η οποία μπορεί να δημιουργηθεί και σε ένα ξεχωριστό έγγραφο, ξεχωριστά από το Design Document.



Εικόνα 4.1 Υπόδειγμα ενός από τα πολλά Game Design Documents που μπορεί να συναντήσει κανείς

Σχεδιασμός Επιπέδων

Τον σχεδιασμό των επιπέδων σε ένα βιντεοπαιχνίδι τον αναλαμβάνει ο Level Designer. Κύριο καθήκον του σε κάθε επίπεδο είναι η δημιουργία μιας γεωγραφικής περιοχής σε στα 3D παιχνίδια ή μιας ζώνης κύλισης στα 2D παιχνίδια. Πέρα από το γεωγραφικό σχεδιασμό είναι υπεύθυνος στην τοποθέτηση των NPC, των εχθρών, των κτηρίων, της χλωρίδας και της πανίδας. Ανάλογα το παιχνίδι κάθε πίστα έχει και έναν δικό της ξεχωριστό ρόλο. Οι πρωταρχικές πίστες είναι σχεδιασμένες ώστε να εγκλιματίσουν το παίκτη στο περιβάλλον του παιχνιδιού και στη συνέχεια να εξοικειωθεί με το χειρισμό και τα εργαλεία που διαθέτει. Τα επόμενα επίπεδα είναι σχεδιασμένα με τέτοιο τρόπο ώστε ο χρήστης να προσπαθήσει να αναπτύξει περεταίρω τις δεξιότητες του μέσα από διάφορες αποστολές. Σε ένα παιχνίδι αυτοκινήτων μια πίστα έχει στόχο ο παίκτης να τερματίσει με το καλύτερο χρόνο ενώ μια άλλη να προσπεράσει όλους τους υπόλοιπους αντιπάλους του. Σε ένα παιχνίδι δράσης ένα επίπεδο μπορεί να αποτελείται από αποφυγή παγίδων, το επόμενο από το να εξολοθρευτεί ο ήρωας όλους τους αντιπάλους και ένα τρίτο να πρέπει να κατακτήσει με την ομάδα του ένα οχυρό. Συνεπώς κάθε επίπεδο, το οποίο είναι βασισμένο πάνω στο σενάριο, σχεδιάζεται ανάλογα με τις προδιαγραφές που απαιτούνται ώστε να υπάρχει η κατάλληλη ροή μέσα στο παιχνίδι.²⁴

Ο σχεδιασμός των επιπέδων στο Corfu Runner βασίζεται στις παραπάνω τεχνικές. Η πρώτη πίστα είναι μικρής δυσκολίας, είναι σχεδιασμένη με απλούς αντιπάλους και δεν είναι μεγάλου βαθμού δυσκολίας. Στο δεύτερο επίπεδο ο σχεδιασμός δίνει έμφαση στην παρουσίαση νέων εχθρών ενώ τα μονοπάτια γίνονται ακόμα πιο δύσκολα. Στο τρίτο και τελευταίο επίπεδο εισάγεται ένα νέο είδος περιήγησης αυτό της θάλασσας όπου όταν ο παίκτης εισέρχεται μέσα έχει αλλάζουν οι κανόνες κίνησης με σκοπό να τείνει περισσότερο στο κολύμπι.



Εικόνα 4.2

Στιγμιότυπο από
το επίπεδο τρία

²⁴ Bethke Erik, *Game Development and Production* (Texas: Wordware Publishing, 2003), 40.

Προγραμματισμός

Κώδικας

Προγραμματισμός ή αλλιώς κωδικοποίηση του παιχνιδιού ονομάζεται η διαδικασία η οποία είναι υπεύθυνη για την ενοποίηση όλων των στοιχείων του παιχνιδιού. Σε αυτό το στάδιο δημιουργούνται όλα τα στοιχεία και οι βάσεις δεδομένων οι οποίες τρέχουν στο σύστημα του παιχνιδιού. Το κάθε τι που εμφανίζεται στο παιχνίδι χρειάζεται προγραμματισμό. Από τις φυσικές ιδιότητες των χαρακτήρων και του περιβάλλοντος, το σύστημα υγείας του παίκτη μέχρι το πώς και πότε ακούγονται τα ηχητικά εφέ, όλα αυτά χρειάζονται κώδικα. Πρόκειται για το σημείο όπου ότι έχει παρουσιαστεί στο χαρτί από το game designer αποκτάει ζωή μέσα από το προγραμματισμό. Το τμήμα του προγραμματισμού καθοδηγείται από τον προϊστάμενο του που αναγράφεται συνήθως με τον όρο Lead Programmer και έχει υπό την εποπτεία του όλες τις υποομάδες προγραμματισμού.²⁵

Ο υπεύθυνος που είναι και ο πιο έμπειρος προγραμματιστής της ομάδας, καλείται να είναι συνέχεια σε επαφή με τους αντίστοιχους αρμόδιους των άλλων τμημάτων, να συντονίζει όλες τις ομάδες του τμήματος του και αναλαμβάνει τα πιο απαιτητικά κομμάτια του παιχνιδιού. Κάθε ομάδα είναι αρμόδια για ένα συγκεκριμένο κομμάτι του παιχνιδιού. Για παράδειγμα, μια ομάδα είναι υπεύθυνη για τον κώδικα των εφέ, άλλη για τις παραμέτρους φυσικής, άλλη για τον ήχο, άλλη για το animation κοκ. Για κάθε παιχνίδι χρησιμοποιείται και η ανάλογη γλώσσα προγραμματισμού που καλύπτει τις ανάγκες του και είναι κοινή για όλες τις ομάδες εκτός από κάποιες εξαιρέσεις για λόγους λειτουργικότητας. Οι επικρατέστερες γλώσσες προγραμματισμού είναι η C++, C#, Java, HTML5, JavaScript SQL και Python.²⁶

Κατά τη δημιουργία του Corfu Runner ο προγραμματισμός αποτέλεσε το κομμάτι με τη μεγαλύτερη δυσκολία. Ατελείωτες ώρες αφιερώθηκαν για τη συγγραφή του κώδικα που αποτελείται από 35 αρχεία σε σύνολο 60 σελίδων. Κάθε script έχει έναν ξεχωριστό ρόλο και όλα είναι απαραίτητα για την ομαλή λειτουργία του παιχνιδιού. Σε περίπτωση λάθους στο κώδικα το παιχνίδι σταματάει να τρέχει μέχρι να

²⁵ Meigs Tom. *Ultimate Game Design: building game worlds*. (Emeryville, CA: McGraw-Hill, 2003), 253.

²⁶ Bethke Erik. *Game development and Production*. (Texas: Wordware Publishing Inc, 2003), 43.

βρεθεί η επίλυση του. Ο κώδικας του παιχνιδιού χωρίζεται σε δύο κατηγορίες, τον κώδικα συστήματος και τον κώδικα δράσεων. Ο κώδικας που αφορά το σύστημα και τη λειτουργία του παιχνιδιού δεν γίνεται έχει κάποιο οπτικό αποτέλεσμα στην οθόνη του χρήστη αλλά είναι απαραίτητος για να πραγματοποιηθούν δευτερεύοντες ενέργειες. Τέτοιου είδους κώδικα αποτελεί το script που οριοθετεί τα σημεία κίνησης του παίκτη, η κάμερα που τον ακολουθεί συνεχώς ή το σημείο μετάβασης από το ένα επίπεδο στο άλλο. Όσον αφορά το κώδικα που σχετίζεται με τις δράσεις που λαμβάνουν μέρος, τοποθετείται πάνω στα αντικείμενα και στους ήρωες ώστε να αποκτήσουν τις ιδιότητες που επιθυμούμε όπως για το αν θα πετάει σφαίρες έναν εχθρό, αν θα χρειάζεται ένα χτύπημα η περισσότερα για να εξοντωθεί ή αν η πλατφόρμα που βρίσκεται πάνω ο παίκτης είναι αναπήδησης, μετακινούμενη ή και τα δύο μαζί. (βλ. Παράρτημα Α.)



4.3 Φωτογραφία από την ομάδα προγραμματιστών της Ubisoft

User Interface

Ονομάζουμε το λογισμικό και τον εξοπλισμό που χρησιμοποιεί ο χρήστης για να αλληλοεπιδράσει με το παιχνίδι. Στον εξοπλισμό εντάσσονται τα πληκτρολόγια, το ποντίκι, τα joystick, τα χειριστήρια ή ολόκληρο το κινητό και το tablet όταν το παιχνίδι προβάλλεται σε αυτό. Το μέρος του λογισμικού που χρησιμοποιείται για το interface είναι τα σύμβολα στην οθόνη, τα εικονίδια και οποιαδήποτε άλλα σημεία της οθόνης τα οποία χρησιμοποιούνται για να πραγματοποιήσει ο παίκτης όλες τις ενέργειες που απαιτούνται. Το interface διαφοροποιείται ανάλογα με τη συσκευή αναπαραγωγής καθώς και με τις απαιτήσεις του παιχνιδιού. Διαφορετικές ανάγκες έχει για παράδειγμα ένα παιχνίδι τύπου puzzle και διαφορετικές ένα MMRPG που περιλαμβάνει κίνηση της κάμερας και το παίκτη, επιθέσεις, chat, quest κτλπ. Για αυτό το λόγο και τα πιο απαιτητικά παιχνίδια είναι πιο δημοφιλή για κονσόλες και υπολογιστή και όχι τόσο στις κινητές συσκευές.²⁷

Το Interface πρέπει να είναι έτσι σχεδιασμένο ώστε να είναι όσο το δυνατόν πιο ξεκάθαρο και κατανοητό και χωρίζεται σε δύο κατηγορίες. Τα πρωτεύοντα στοιχεία τα οποία καθορίζουν τη κίνηση του παίκτη και τις δράσεις τις οποίες εκτελεί και τα δευτερεύοντα τα οποία κάνουν το παιχνίδι πιο διασκεδαστικό προσφέροντας περισσότερες πληροφορίες η επιλογές. Ένα interface θεωρείται πετυχημένο όταν είναι διακριτικό και περνάει σχεδόν απαρατήρητο στο χρήστη κατά τη διάρκεια του παιχνιδιού αλλά συγχρόνως είναι συνέχεια ορατό, εύκολα προσβάσιμο και διακριτό. Για να πετύχει αυτό πρέπει τα στοιχεία στο interface της οθόνης να είναι τοποθετημένα στη κατάλληλη θέση ώστε να μην εμποδίζουν τη ροή του παιχνιδιού, να έχουν τα κατάλληλα χρώματα που θα τα κάνει να ξεχωρίζουν στο περιβάλλον και τα κατάλληλα απλοϊκά σύμβολα των δράσεων που αναφέρονται.²⁸

Στο παιχνίδι Corfu Runner το hardware interface είναι το πληκτρολόγιο του υπολογιστή και το ποντίκι όπου με τα πλήκτρα 'A' και 'D' ο παίκτης μετακινείται αριστερά και δεξιά αντίστοιχα, με το πλήκτρο space πηδάει και με το δεξί κλικ από το ποντίκι πυροβολάει. Το software interface που αφορά τις πληροφορίες που παρουσιάζονται στην οθόνη είναι οι πόντοι πάνω αριστερά που έχουμε συλλέξει, ο

²⁷ Marx Christy. *Writing for Animation, Comics and Games*. (Burlington, MA: Focal Press, 2007), 172.

²⁸ Jesse Schell. *The Art of Game Design 2nd Edition*. (US: CRC Press, 2015), 322.

μετρητής χρόνου και η μπάρα ενέργειας που βρίσκεται συνεχώς πάνω από το χαρακτήρα μας. Στην έκδοση για smartphones ο χειρισμός γίνεται με κουμπιά που εμφανίζονται στην οθόνη με τα αντίστοιχα σύμβολα για την ενέργεια που πραγματοποιούν. Τα πράσινα βέλη συμβολίζουν την κίνηση του ήρωα και το κόκκινο τον πυροβολισμό. Σε πρώτη φάση όλα τα σύμβολα μετακίνησης τοποθετήθηκαν στην αριστερή πλευρά και ο πυροβολισμός δεξιά, στη πορεία όμως αποδείχθηκε ότι δυσκόλευε τους χρήστες και το άλμα μετακινήθηκε στην αριστερή πλευρά της οθόνης πλάι από το κουμπί του πυροβολισμού. Επάνω δεξιά τοποθετήθηκε το κουμπί της παύσης όπου οδηγεί το χρήστη και στο αναδυόμενο μενού που του παρέχει επιλογές σχετικά με την σίγαση του ήχου, την επανεκκίνηση της πίστας και την έξοδο του από το παιχνίδι. Στο αριστερό μέρος της οθόνης παρουσιάζεται ο συνολικός χρόνος και οι πόντοι από τα νομίσματα που έχει συλλέξει.



Εικόνα 4.4 Στιγμιότυπο του παιχνιδιού όπου διακρίνεται σχεδόν όλο το User Interface

Προγραμματιστής Ήχου

Ο προγραμματιστής ήχου δημιουργεί το κώδικα για να υποστηρίξει τις ηχητικές απαιτήσεις του παιχνιδιού. Για παράδειγμα η μουσική που ακούγεται στο Background, ο ήχος που ακούγεται όταν ο παίκτης πυροβολεί ή πεθαίνει ή ακόμα και όταν ο κέρσορας του ποντικιού βρίσκεται πάνω σε μια επιλογή και ακούγεται η ηχητική ένδειξη, όλα αυτά απαιτούν το ανάλογο κώδικα για να ακουστούν όταν πρέπει και όπως πρέπει. Μπορεί εκ πρώτης όψεως να μην φαίνεται τόσο σημαντικό αλλά είναι μια ακόμα πινελιά στη ρεαλιστικότητα του παιχνιδιού. Δεν θα ήταν το ίδιο αν ο παίκτης ακούει έναν ήχο στην ίδια ένταση είτε είναι κοντά είτε μακριά από τη πηγή που τον εκπέμπει, κάτι τέτοιο θα φαινόταν ψεύτικο και δεν θα έδινε το βάθος που απαιτείται. Όταν όμως υπάρχει ο κατάλληλος προγραμματισμός και ο ήχος σταδιακά αυξάνεται όσο ο παίκτης πλησιάζει στη πηγή τόσο καλύτερα μπορεί να έχει την αίσθηση του χώρου καθώς και να προετοιμαστεί κατάλληλα για μια επικείμενη επίθεση του εχθρού.

29

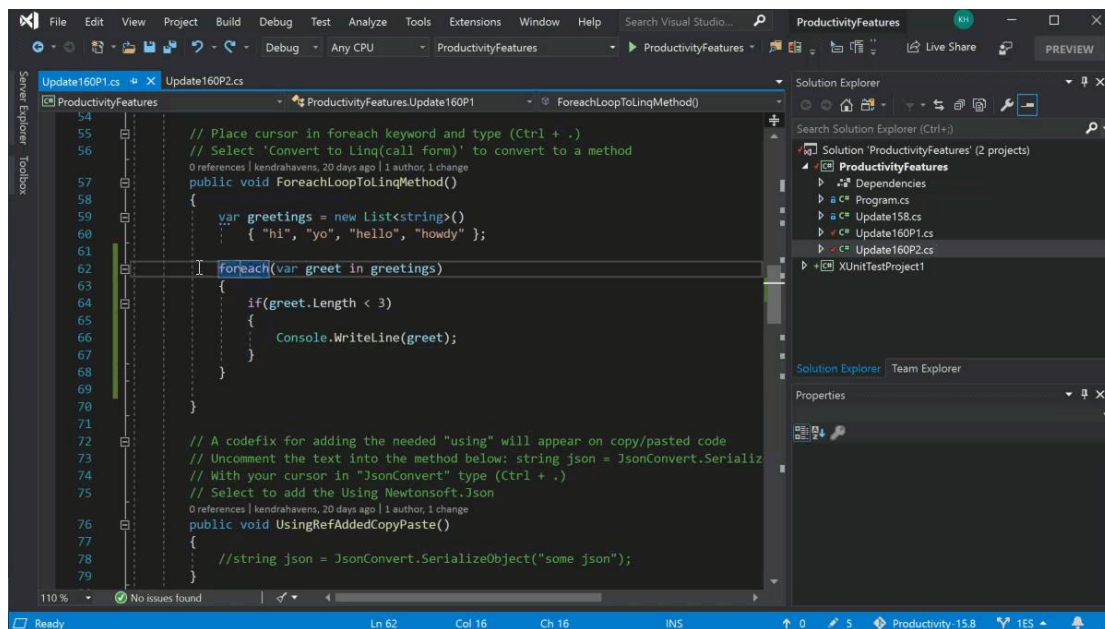
Λογισμικό Προγραμματισμού

Για την συγγραφή κώδικα υπάρχει μια τεράστια γκάμα λογισμικών με το καθένα από αυτά να αποτελείται από τα δικά του ξεχωριστά στοιχεία που το διαφοροποιούν από τα υπόλοιπα. Ο κάθε προγραμματιστής ανάλογα με τις απαιτήσεις του εκάστοτε πρότζεκτ και κατά κύριο λόγο τη γλώσσα προγραμματισμού που επρόκειτο να χρησιμοποιήσει επιλέγει και λογισμικό το οποίο εφαρμόζει καλύτερα στις απαιτήσεις και τα θέλω του. Τα πιο δημοφιλή είναι τα Microsoft Visual Studio, Notepad ++, Brackets, Atom, Bluefish και Sublime Text. Ορισμένα διατίθενται δωρεάν ενώ άλλα είναι μόνο με πληρωμή. Σημαντικός παράγοντας είναι και το λειτουργικό σύστημα που χρησιμοποιεί ο προγραμματιστής καθώς για Linux και MacOS προκύπτουν διαφορετικές απαιτήσεις.³⁰

²⁹ Meigs Tom. *Ultimate Game Design: building game worlds*. (Emeryville, CA: McGraw-Hill, 2003) 281.

³⁰ Dan Irish. *The Game Producer's Handbook*. (Boston: Thomson Course Technology, 2005), 170.

Για τη πτυχιακή εργασία ο κώδικας συντάχθηκε στο Microsoft Visual Studio. Κατά την ολοκλήρωση τη εργασίας αποδείχθηκε ότι ήταν η καλύτερη επιλογή λόγω της μεγάλης γκάμας εργαλείων που διαθέτει, της εύκολης χρήσης του και της γρήγορης λειτουργίας του δίχως να επιβαρύνει ιδιαίτερα το σύστημα. Το εν λόγω λογισμικό προσδιορίζει κάθε ομάδα εντολών με ένα συγκεκριμένο χρώμα ώστε να είναι εύκολα διακριτές και να ξεχωρίζουν η μία από την άλλη. Κάθε συντακτικό λάθος επισημαίνεται με μια κόκκινη υπογράμμιση και ενδείξεις εμφανίζονται στο κάτω μέρος της οθόνης και υποδεικνύουν τη γραμμή η οποία χρειάζεται επιδιόρθωση. Επιπλέον σου επιτρέπει την εναλλαγή μεταξύ πολλών παραθύρων για την συγγραφή πολλών κειμένων κώδικα σε παράλληλο χρόνο.



Εικόνα 4.5 Το περιβάλλον του Microsoft Visual Studio όπου δημιουργήθηκε ο κώδικας του παιχνιδιού

Γραφικά

Τμήμα Γραφικών

Κατά την έκδοση των πρώτων βιντεοπαιχνιδιών, υπεύθυνος για το προγραμματισμό, τα γραφικά και τα Animation ήταν ένα άτομο. Αργότερα όμως λόγω της μεγάλης ανάπτυξης και ζήτησης η δημιουργία ξεχωριστού τμήμα γραφικών ήταν επιτακτική και αποτελούταν από μικρές ομάδες περίπου των τριών ατόμων. Πλέον το τμήμα γραφικών χωρίζεται σε πολλές υποκατηγορίες ανάλογα τις απαιτήσεις, τον προϋπολογισμό και το μέγεθος του παιχνιδιού έχοντας ρόλο υψηλής βαρύτητας στην σχεδίαση του παιχνιδιού. Αποτελεί τη βιτρίνα του παιχνιδιού καθώς είναι αυτό που φαίνεται στο εξώφυλλο, σε screenshot και διαφημιστικά βίντεο ασκώντας μεγάλη επιρροή στο χρήστη για την απόκτηση του από τα πρώτα χρόνια έκδοσης βιντεοπαιχνιδιών μέχρι σήμερα. Πλέον δεν υπάρχει χρήστης που να μην εκτιμά τα όλο και καλύτερα γραφικά ανεξαρτήτου κατηγορίας παιχνιδιού. Ο ρεαλισμός είναι όλο και σε υψηλότερα επίπεδα χρόνο με το χρόνο με τα γραφικά να έχουν φτάσει σε σημείο πολλές φορές να μην μπορεί να διακρίνει ο μέσος χρήστης αν αυτό που βλέπει αποτελεί βίντεο ή animation.³¹

Art Director

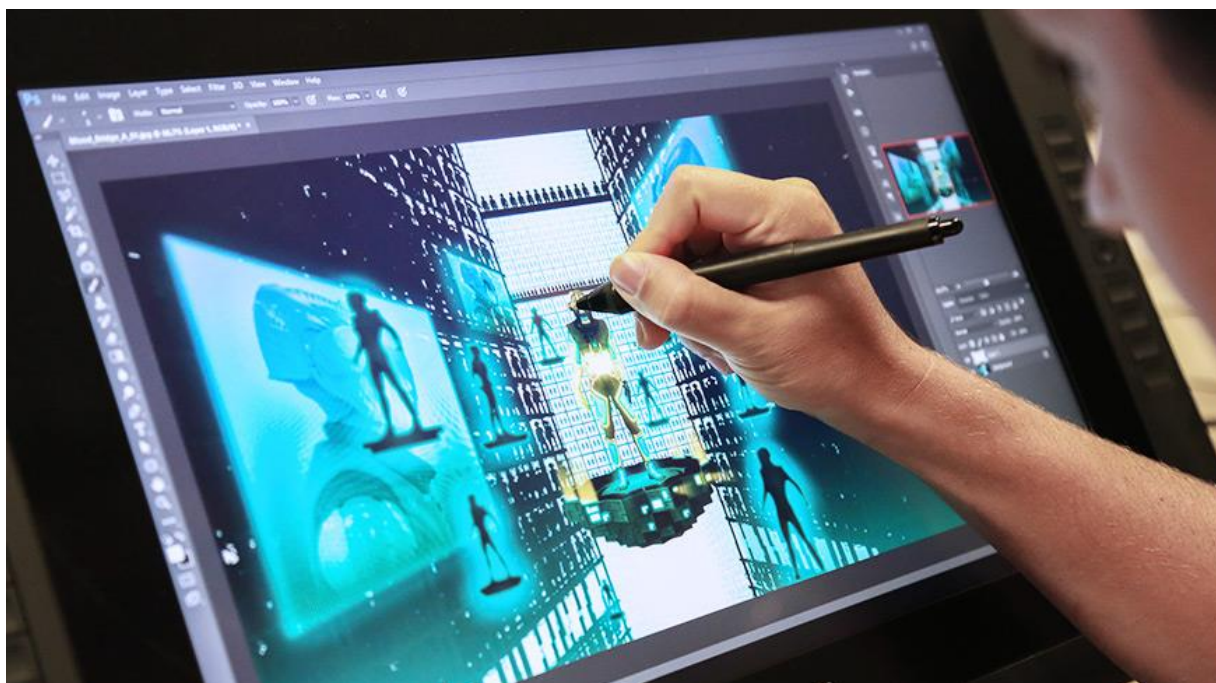
Υπεύθυνος συντονιστής για όλες τις ομάδες είναι ο Καλλιτεχνικός Διευθυντής, ένα άτομο με πλούσιο βιογραφικό, ηγετικές ικανότητες και με προϋπηρεσία όσο το δυνατόν πιο κοντά στο είδος και τις απαιτήσεις του παιχνιδιού. Συνεπώς αν το παιχνίδι το οποίο σχεδιάζεται είναι αθλητικού περιεχομένου ο καλλιτεχνικός διευθυντής έχει στις περισσότερες περιπτώσεις εργαστεί σε ένα παρόμοιο πρότζεκτ. Όλοι όσοι επρόκειτο να αποτελέσουν τις καλλιτεχνικές ομάδες πρέπει πρώτα να εγκριθούν από αυτόν ύστερα από την υπόδειξη βιογραφικού και μιας σύντομης συνέντευξης. Ευθύνη του είναι το τελικό αποτέλεσμα να έχει όσο το δυνατόν καλύτερη ποιότητα και να μην εμφανίζεται διαφορετικού τύπου αισθητική μεταξύ

³¹ Meigs Tom. *Ultimate Game Design: building game worlds*. (Emeryville, CA: McGraw-Hill, 2003), 268.

των καλλιτεχνικών ομάδων αλλά να κινούνται όλοι γύρω από τον βασικό άξονα που έχει σχεδιαστεί. Οι ομάδες οι οποίες εντάσσονται στο τμήμα γραφικών και καλλιτεχνικών είναι αυτές των 2D καλλιτεχνών, 3D καλλιτεχνών και Animators με τις αντίστοιχες υποομάδες τους η κάθε μία ξεχωριστά.³²

2D Artist

Οι 2D καλλιτέχνες ειδικεύονται στο σχέδιο και τη ζωγραφική και είναι άκρως απαραίτητοι ακόμα και σε παιχνίδια τριών διαστάσεων. Στα πλαίσια των καθηκόντων τους αναλαμβάνουν να σχεδιάσουν από χαρακτήρες και στοιχεία περιβάλλοντος μέχρι και το user interface. Η δυσκολία που αντιμετωπίζουν είναι η όσο το δυνατόν καλύτερη απεικόνιση αντικειμένων και χαρακτήρων του πραγματικού κόσμου στις δύο διαστάσεις με τις λιγότερες λεπτομέρειες. Στα παιχνίδια 3D καλύπτουν το κομμάτι του μενού, συμβόλων και γραφημάτων που εμφανίζονται κατά τη διάρκεια του παιχνιδιού και στη συνέχεια συμμετέχουν στα εξώφυλλα η ακόμα και στα γραφικά στοιχεία της ιστοσελίδας.³³



Εικόνα 4.6 Η γραφίδα αποτελεί βασικό εργαλείο για έναν 2D/3D Artist

³² Bethke Erik. *Game development and Production*. (Texas: Wordware Publishing Inc, 2003), 46.

³³ Bethke Erik. *Game development and Production*. (Texas: Wordware Publishing Inc, 2003), 47.

Animation

Η ομάδα του Animation δίνει ζωή σε οτιδήποτε έχει σχεδιαστεί από τις ομάδες των 2D και 3D γραφικών. Γνώσεις ανατομίας σχετικά με το μυϊκό σύστημα και το σκελετό είναι απαραίτητες για την ορθή απεικόνιση της κίνησης. Τα animation χωρίζονται σε δύο κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει τις κινήσεις των χαρακτήρων, των ζώων και των τεράτων και η άλλη κατηγορία τις κινήσεις όλων των άλλων στοιχείων, όπως αντικείμενα και στοιχεία της φύσης. Η ομάδα animation αφού λάβει τη σχετική λίστα με τις κινήσεις που απαιτούνται για το κάθε χαρακτήρα ξεκινάει πρώτα με τη διαδικασία του Rigging και στη συνέχεια με τις κινήσεις των χαρακτήρων.³⁴

Η κίνηση ενός ήρωα πέρα από τη μετακίνηση του μέσα στο χώρο προσδιορίζει το ύφος, του χαρακτήρα και τις προθέσεις του. Διαφορετικά περπατάει ένας ήρωας όταν είναι ήρεμος, διαφορετικά όταν είναι νευριασμένος, Άλλο βηματισμό έχει ένας στρατιωτικός που πάει στην βάση και άλλο ένας μαθητής που πάει άυπνος στο σχολείο. Συνεπώς οι animators πέρα από την ορθή στάση του σκελετού και των κινήσεων καλούνται να αποδώσουν μέσα από τη γλώσσα του σώματος και το ψυχολογικό προφίλ του εκάστοτε χαρακτήρα. Για αυτόν ακριβώς το λόγο είναι και πιο περίπλοκες οι κινήσεις ανθρώπων και ζώων σε σχέση με αυτές των αντικειμένων που έχουν μια πιο απρόσωπη κίνηση.³⁵

Για τη δημιουργία animation υπάρχουν τρεις βασικοί μέθοδοι. Η πρώτη με την οποία και ξεκίνησε το animation είναι η δημιουργία της κίνησης καρέ καρέ, δηλαδή σε κάθε FPS σχεδιάζεται και η κίνηση του παίκτη. Είναι μία αρκετά χρονοβόρα διαδικασία και πλέον εφαρμόζεται μόνο περιορισμένες περιπτώσεις λόγω των νέων εργαλείων που έχουν τεθεί στα χέρια των animators. Η δεύτερη τεχνική που είναι πλέον και η πιο δημοφιλής σχετίζεται με τα keyframes. Ο animator δημιουργεί δύο σημεία στη μπάρα του χρόνου όπου στο πρώτο σχεδιάζει την αρχική κίνηση και στο δεύτερο την τελική, με το λογισμικό να πραγματοποιεί αυτόματα όλα τα ενδιάμεσα καρέ από το ένα σημείο στο άλλο.³⁶

³⁴ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 175.

³⁵ Bethke Erik. *Game development and Production*. (Texas: Wordware Publishing Inc, 2003), 48.

³⁶ Franson David & Thomas Eric. *Game Character Design Complete*. (Boston: Thomson Course Technology, 2007), 15.

Η τρίτη τεχνική αν και αρκετά δημοφιλής απαιτεί μεγαλύτερο budget και χώρο για να εφαρμοστεί και είναι γνωστή με την ονομασία motion capture. Σε αυτή τη περίπτωση χρειάζεται ένα studio κάμερες και ηθοποιοί. Η διαδικασία ξεκινάει με την καταγραφή της κίνησης του ηθοποιού σε ένα σύστημα τριών αξόνων και στη συνέχεια γίνεται μετατροπή του αρχείου και εφαρμόζεται στο σκελετό του τρισδιάστατου μοντέλου. Είναι μία πιο περίπλοκη τεχνική και εφαρμόζεται σε περιπτώσεις όπου είναι πιο δύσκολη η καταγραφή της κίνησης όπως σε αθλητικά παιχνίδια. Το αποτέλεσμα είναι ακόμα πιο ρεαλιστικό, ενώ εφαρμόζεται και στο κινηματογράφο στα πλαίσια των Visual Effect.³⁷

Τα animation ήταν αναπόσπαστο κομμάτι και της πτυχιακής εργασίας. Τα animation των αντικειμένων και σε αυτή την περίπτωση δεν ήταν ιδιαίτερα δύσκολα να δημιουργηθούν λόγω του επαναλαμβανόμενου ρυθμού που διατηρούν. Σε αυτή τη κατηγορία ανήκουν τα νομίσματα που περιστρέφονται, τα βλήματα που εκτοξεύονται, οι παγίδες όπως και ορισμένοι εχθροί που είναι προγραμματισμένοι να ακολουθούν μια συγκεκριμένη πορεία. Αντίθετα η κίνηση του βασικού ήρωα χρησιμοποιεί πιο περίπλοκα, σε σχέση με τα υπόλοιπα στοιχεία του παιχνιδιού, animation, καθώς έχει δοθεί έμφαση ώστε το περπάτημα να έχει όσο το δυνατόν πιο ρεαλιστική κίνηση όπως και όταν αναπηδάει και πυροβολεί.

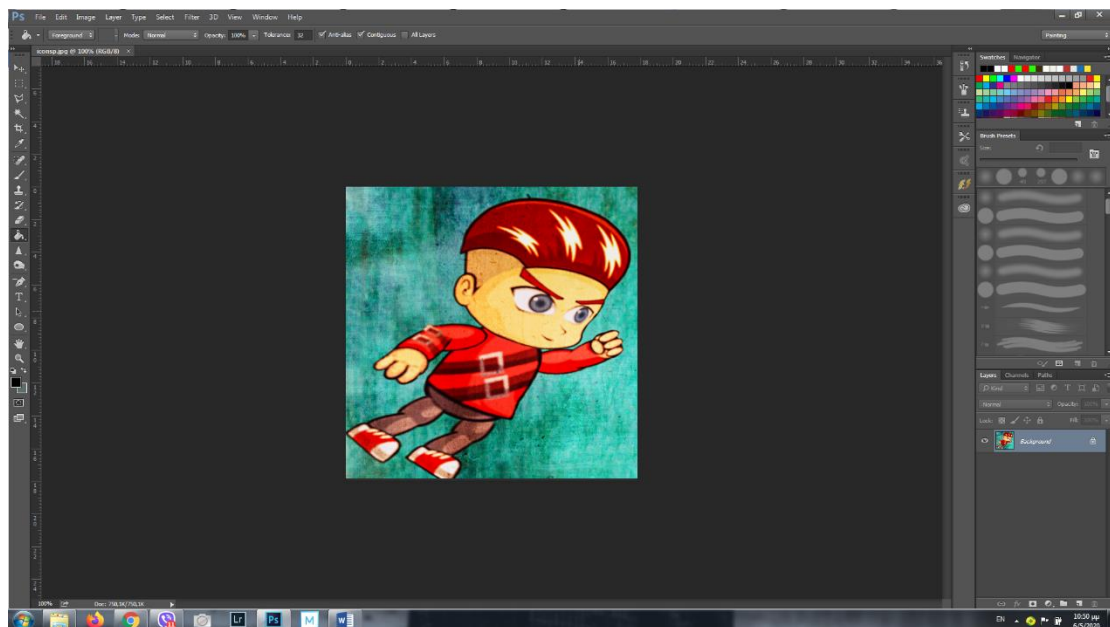


Εικόνα 4.7 Φωτογραφία από studio της Ubisoft κατά τη διάρκεια ενός motion capture

³⁷ Franson David & Thomas Eric. *Game Character Design Complete*. (Boston: Thomson Course Technology, 2007), 15.

Λογισμικά Γραφικών

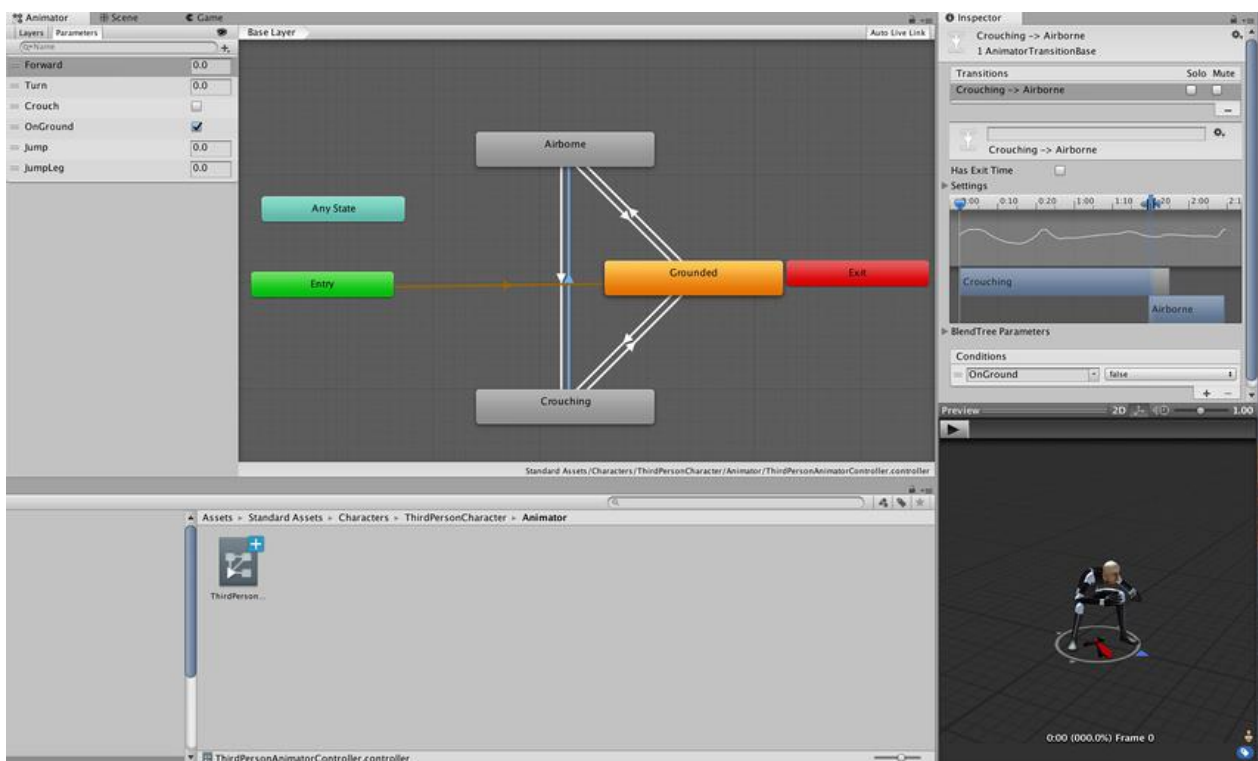
Το λογισμικό που χρησιμοποιήθηκε για την επεξεργασία των γραφικών είναι το Adobe Photoshop. Παρέχει όλα τα εργαλεία που χρειάζεται ο 2D σχεδιαστής όπως γόμες, μολύβια, πινέλα ψηφιακής μορφής και ότι άλλο χρειάζεται για την επεξεργασία της εικόνας. Αφού πραγματοποιήθηκε φωτογράφιση των στοιχείων τα οποία χρειάστηκαν για τη δημιουργία του παιχνιδιού από το νησί της Κέρκυρας στη συνέχεια έγινε εισαγωγή τους στο Photoshop για επεξεργασία. Οι φωτογραφίες δέχθηκαν παραμετροποίηση στη φωτεινότητα, την αντίθεση, τις χρωματικές αποχρώσεις και τοποθετήθηκε ειδικό φίλτρο ώστε να θυμίζουν καρτούν και όλα αυτά μέχρι να φτάσουν στο τελικό επιθυμητό αποτέλεσμα. Στο Photoshop δημιουργήθηκαν και το Interface όπως κουμπιά, μενού αλλά και στοιχεία της ιστοσελίδας. Επιπλέον εδώ έγινε και η βασική επεξεργασία του χαρακτήρα στο κάθε κομμάτι του σώματος του ξεχωριστά. Οι φωτογραφίες όπου αφορούν το περιβάλλον και το επίπεδο στο οποίο βαδίζει ο ήρωας έγιναν σε μορφή πλακιδίου. Αυτό σημαίνει ότι είναι έτσι σχεδιασμένα ώστε όταν ενώνονται μεταξύ τους να μην εμφανίζονται διαφορές μεταξύ τους αλλά να φαίνονται ως ένα ενιαίο επίπεδο.³⁸



Εικόνα 4.8 Στιγμιότυπο από το λογισμικό Adobe Photoshop

³⁸ Cairns George. *Teach yourself Photoshop*. (London: Future Publishing, 2014), 12.

Τα animation δημιουργήθηκαν εντός της εφαρμογής Unity. Η μία κατηγορία σχετίζεται με στοιχεία τα οποία δεν έχουν σταθερό σχήμα και μορφή και αφορά στοιχεία όπως η φωτιά, η βροχή, το χιόνι, οι σπίθες κτλπ. Η δημιουργία animation για αυτά τα στοιχεία ονομάζεται particle system animation και με ειδικές μπάρες ρυθμίζονται τα οι διαστάσεις, τα όρια και η συχνότητα όπου εντός αυτών των τιμών θα εμφανίζονται τα στοιχεία με τυχαία σειρά. Η επόμενη κατηγορία αφορά τα Sprites όπου ένα σύνολο από προσχεδιασμένα καρέ ενώνονται μεταξύ τους για τη δημιουργία ενιαίας κίνησης. Είναι μία τεχνική η οποία είναι κατάλληλη για παιχνίδια δύο διαστάσεων και ακόμα πιο συγκεκριμένα κατηγορίας πλατφόρμας όπως το παιχνίδι της πτυχιακής. Με αυτή τη τεχνική δημιουργήθηκαν σχεδόν όλα τα animation των χαρακτήρων του παιχνιδιού.³⁹



Εικόνα 4.9 Το περιβάλλον για animation που προσφέρεται στο Unity3D

³⁹ Thorn Alan. *Unity Animation Essentials*. (Birmingham: Packt Publishing, 2015), 33.

Ήχος

Voice Over

Σε ένα παιχνίδι οι ηχητικές απαιτήσεις ποικίλλουν και διαφέρουν λόγω της πληθώρας των οπτικών ερεθισμάτων που οφείλουν να συνοδεύονται και από το αντίστοιχο ακουστικό στοιχείο. Μέσα από το voice over ο σχεδιαστής του παιχνιδιού επιδιώκει να δώσει φωνή στους διαλόγους που έχει αποτυπώσει στο χαρτί, ενώ για τη σωστή απόδοση τους οφείλει να βρει τα κατάλληλα άτομα και όχι ερασιτέχνες από το φιλικό του περιβάλλον. Αυτή η διαδικασία ανατίθεται στον voice director ο οποίος αφού του δοθεί το σενάριο και τα μέρη με τους διαλόγους καλείται να βρει τους αντίστοιχους ηθοποιούς για το κάθε ρόλο. Η ηχογράφηση πραγματοποιείται σε στούντιο όσες φορές χρειαστεί μέχρι να υπάρχει το επιθυμητό αποτέλεσμα ενώ για τον ίδιο ρόλο πολλές φορές καλούνται και παραπάνω από ένας ηθοποιοί ώστε να υπάρχουν περισσότερες διαθέσιμες επιλογές κατά την παράδοση των ηχητικών αρχείων.⁴⁰

Στο συγκεκριμένο παιχνίδι το Voice Over ήταν αναγκαίο για την απόδοση του ηχητικού επιφωνήματος, που ακούγεται στις περιπτώσεις όπου ο παίκτης δέχεται επίθεση από κάποιο εχθρό, με αποτέλεσμα να μειώνεται η ενέργεια στη μπάρα υγείας. Η λέξη ‘ωρέ’, είναι χαρακτηριστικό γνώρισμα της Κερκυραϊκής διαλέκτου που ακούγεται σε κάθε τραυματισμό του παίκτη. Για την δημιουργία της κρίθηκε απαραίτητο να ηχογραφηθεί ηχητικό υλικό από την αρχή λόγω της ιδιαιτερότητας και της προφοράς που απαιτείται. Για το τελικό αποτέλεσμα επιλέχθηκε το ηχητικό κομμάτι που ταίριαζε περισσότερο στις απαιτήσεις του παιχνιδιού.



Εικόνα 4.10

Φωτογραφία από τα
Studio ήχου της Ubisoft

⁴⁰ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 185.

Ηχητικά Εφέ

Όπως στο πραγματικό κόσμο περιβαλλόμαστε από ήχους οι οποίοι δημιουργούν το ηχητικό περιβάλλον το οποίο βρισκόμαστε, ακριβώς το ίδιο συμβαίνει και στον εικονικό κόσμο του βιντεοπαιχνιδιού. Οι ήχοι που ακούει ο παίκτης κατά τη διάρκεια της πίστας συνθέτουν την ατμόσφαιρα που θέλει ο game designer να δώσει κάνοντας το παιχνίδι ακόμα πιο ρεαλιστικό. Αν και είναι ήχοι που ο παίκτης κατά κύριο λόγο δεν εστιάζει τη προσοχή του σε αυτούς, η τυχόν έλλειψη τους θα ήταν κάτι το οποίο δεν θα μπορούσε να περάσει απαρατήρητο κάνοντας το παιχνίδι πιο ‘φτωχό’. Πέρα από τη δημιουργία του περιβάλλοντος, τα ηχητικά εφέ παίζουν δραστικό ρόλο και στο gameplay του παιχνιδιού. Προειδοποιούν το παίκτη από τυχόν κινδύνους που πρόκειται να εμφανιστούν και συμπληρώνουν ή αναδεικνύουν ακόμα περισσότερο τα οπτικά ερεθίσματα. Στην όλη διαδικασία απαιτείται σωστός συγχρονισμός των δράσεων που απεικονίζονται σε σχέση με τους ήχους για ένα ακόμα πιο ορθολογικό αποτέλεσμα. Σημαντική σημείωση αποτελεί το γεγονός ότι κατά την προσθήκη των ηχητικά εφέ, το τελικό αποτέλεσμα που θα λαμβάνει ο χρήστης δεν θα του προκαλεί σύγχυση και αποσυντονισμό από το παιχνίδι και τους στόχους του, συνεπώς η χρήση τους οφείλει να γίνεται μόνο όπου κρίνεται απαραίτητη δίχως περιττούς ήχους.⁴¹

Μουσική – Soundtrack

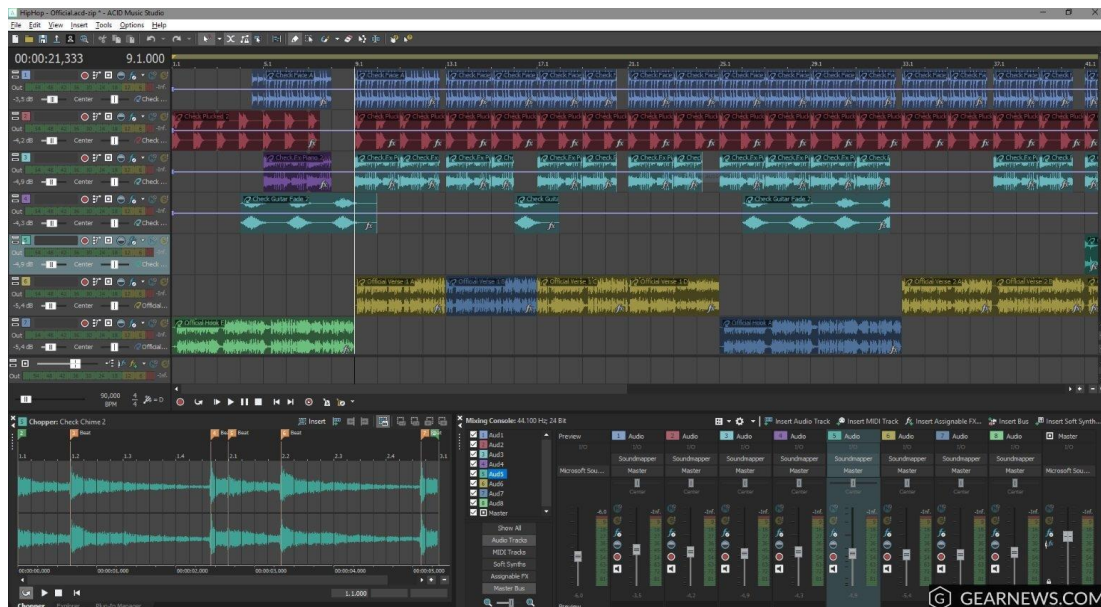
Αν τα ηχητικά εφέ συνθέτουν το περιβάλλον του παιχνιδιού, η μουσική δημιουργεί το συναίσθημα. Είναι ο κύριος μοχλός ώστε ο σχεδιαστής του παιχνιδιού να κατευθύνει υποσυνείδητα τα συναισθήματα και τη διάθεση του παίκτη ώστε να ενσωματώνει όσο το δυνατόν περισσότερο μέσα στο κόσμο του βιντεοπαιχνιδιού. Υπεύθυνος για το συγκεκριμένο κομμάτι είναι ο μουσικοσυνθέτης με τον οποίο έρχεται σε επικοινωνία ο game designer. ‘Όταν πλέον θα διαθέτει όλες τις απαραίτητες πληροφορίες του παιχνιδιού συνθέτει τα μουσικά κομμάτια τα οποία θα το απαρτίζουν. Κάθε επίπεδο μπορεί να διαθέτει είτε ένα συγκεκριμένο κομμάτι, είτε περισσότερα ανάλογα με τη δράση που εξελίσσεται όπως μουσική μάχης η δράσης, περιβάλλοντος, μενού, τίτλων τέλους, επίτευξης στόχων ή και αποτυχίας αποστολής. Για τη πρώτη

⁴¹ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 191.

πίστα του παιχνιδιού χρησιμοποιήθηκε μουσική που είχε ως σκοπό την εισαγωγή του χρήστη στο κλίμα του παιχνιδιού με ευδιάθετη και ήρεμη μελωδία. Στη δεύτερη πίστα η μουσική έρχεται να τονίσει το νυχτερινό περιβάλλον, το φρούριο και τα στοιχειωμένα μονοπάτια προκαλώντας υποσυνείδητα αγωνία και φόβο ενώ στη τρίτη και τελευταία στόχος είναι να παρουσιαστεί η καλοκαιρινή διάθεση και ξεγνοιασιά που συναντά κανείς στις ακρογιαλιές της Κέρκυρας.⁴²

Λογισμικό Ήχου

Όλο το ηχητικό υλικό τροποποιήθηκε σε δύο λογισμικά επεξεργασίας ήχου, το Wavelab και το Reaper. Στο Wavelab με το οποίο δουλεύτηκαν κυρίως τα ηχητικά εφέ, πραγματοποιήθηκαν όλες οι ηχητικές παραμορφώσεις όπως τονικές αλλαγές και τροποποίηση συχνοτήτων του υλικού που ηχογραφήθηκε η που προήλθε από πηγές του διαδικτύου, ώστε να συμβαδίζει με τις απαιτήσεις του παιχνιδιού. Το Reaper χρησιμοποιήθηκε κατά κύριο λόγο για τη δημιουργία του τελικού αρχείου μουσικής όπου με την απαραίτητη επεξεργασία επαναλαμβάνεται συνεχώς δίχως να δίνει στο χρήστη την εντύπωση ότι υπάρχει αρχή και τέλος του κομματιού, με αποτέλεσμα μια συνεχή ροή για όσο χρειάζεται μέχρι να τερματίσει το εκάστοτε επίπεδο.



Εικόνα 4.11 Το λογισμικό Reaper όπου έγινε ένας μέρος της ηχητικής Επεξεργασίας

⁴² Dan Irish. *The Game Producer's Handbook*. (Boston: Thomson Course Technology, 2005), 245-46.

Κεφάλαιο 5: Μετα-Παραγωγή (Post Production)

Δοκιμές

Έλεγχος Λειτουργικότητας (Debugging)

Σε αυτό το στάδιο ελέγχονται όλα τα τεχνικά μέρη του παιχνιδιού όσον αφορά το θέμα λειτουργικότητας. Τυχόν λάθη προγραμματισμού διορθώνονται, επανεξετάζονται μεταβλητές οι οποίες πιθανόν να προκαλούν σύγχυση στην εμπειρία του παίκτη και ελέγχονται τα όρια πλοήγησης ώστε να μην υπάρχουν κενά σημεία που οδηγούν έξω από τα όρια δράσης του παιχνιδιού. Αν και το συγκεκριμένο στάδιο αποτελεί κυρίως κομμάτι εργασίας για την ομάδα προγραμματισμού υπάρχουν περιπτώσεις όπου τα γραφικά συγχέονται, τρεμοπαίζουν ή μοιάζουν ασύνδετα, αυτό έχει ως κατάληξη να επανασχεδιάζονται από το τμήμα γραφιστικής ώστε το τελικό αποτέλεσμα να είναι πλήρως λειτουργικό.⁴³

Playtesting

Πρόκειται για τη διαδικασία όπου το παιχνίδι έχει πλέον σε πρώτη φάση ολοκληρωθεί και δίνεται η ευκαιρία σε διάφορους χρήστες να το παίξουν και να μοιραστούν τις εμπειρίες που τους προσέφερε, θετικές είτε αρνητικές, ώστε να τελειοποιηθεί πριν τη τελική του έκδοση. Αν και πολλές φορές η διαδικασία αυτή συγχέεται με το debugging, η πραγματική προσέγγιση που πρέπει να δοθεί είναι εντελώς διαφορετική, καθώς σκοπός της είναι όχι η διόρθωση τυχόν προγραμματιστικών λαθών αλλά να διαπιστωθεί αν ο χρήστης διασκεδάζει μέσα από αυτό, αν το βρίσκει ενδιαφέρον, αν στο κοινό το οποίο απευθύνεται θεωρείται ικανό να το παίξει και πολλές ακόμα ερωτήσεις που καλούνται οι δημιουργοί να δώσουν λύσεις όπου χρειαστεί.⁴⁴

⁴³ Richard Rouse. *Game Design: Theory & Practise*. (Texas: Wordware Publishing Inc, 2005), 484.

⁴⁴ Richard Rouse. *Game Design: Theory & Practise*. (Texas: Wordware Publishing Inc, 2005), 485-6.

Οι πρώτοι χρήστες που θα δοκιμάσουν τη πρώτη έκδοση του παιχνιδιού συνήθως αποτελούνται από επαγγελματίες gamers, φίλους, αγνώστους ή ακόμα και από τους ίδιους τους κατασκευαστές με τη κάθε κατηγορία να έχει τα θετικά και τα αρνητικά της. Οι κατασκευαστές που γνωρίζουν το παιχνίδι εις βάθος παρέχουν χρήσιμο feedback σε συγκεκριμένες περιπτώσεις αλλά λόγω των ατελειών των ωρών εργασίας πάνω σε αυτό θεωρείται συνήθως δύσκολο να βγουν κάποια συμπεράσματα. Οι φίλοι που θα δοκιμάσουν να παίξουν το παιχνίδι, λόγω οικειότητας θα μιλήσουν πιο ανοιχτά όμως υπάρχει και η περίπτωση να αποκρύψουν αρνητικά στοιχεία που παρατήρησαν. Οι επαγγελματίες gamers μπορούν να σου παρέχουν μια πλήρη αξιολόγηση παρατηρώντας ακόμα και μικρές λεπτομέρειες, επειδή όμως τείνουν να είναι περισσότερο εξοικειωμένοι με πολύ περίπλοκα συστήματα η εμπειρία τους διαφέρει από το μέσο χρήστη ο οποίος αποτελεί και το μεγαλύτερο ποσοστό της αγοράς.⁴⁵



Εικόνα 5.1 Φωτογραφία ενός χρήστη κατά τη διάρκεια Playtesting σε studio της Ubisoft

⁴⁵ Jesse Schell. *The Art of Game Design 2nd Edition*. (US: CRC Press, 2015), 436.

Αξιολόγηση

Κατά την αξιολόγηση του παιχνιδιού οι πρώτοι χρήστες που το δοκίμασαν απάντησαν σε μια σειρά από ερωτήσεις. Δεν έφτασαν όλοι οι χρήστες μέχρι το τερματισμό συνεπώς ο καθένας το αξιολόγησε ανάλογα με τη πρόοδο που είχε. Στο ερωτηματολόγιο συμμετείχαν άτομα και των δύο φύλων και τα ηλικιακά όρια κυμαίνονταν από 10 έως 50 ετών με διαφορετική εμπειρία στα παιχνίδια ο καθένας. Οι πρώτες ερωτήσεις ήταν γενικού περιεχομένου για να προσδιοριστεί το φύλο, η ηλικία και η σχέση του χρήστη με τα παιχνίδια. Οι ερωτήσεις της δεύτερης ενότητας είχαν αποκλειστικό σκοπό την αξιολόγηση της εμπειρίας των χρηστών με το συγκεκριμένο παιχνίδι. Όλες οι ερωτήσεις είναι αποτυπωμένες στους δύο παρακάτω πίνακες που ακολουθούν.⁴⁶

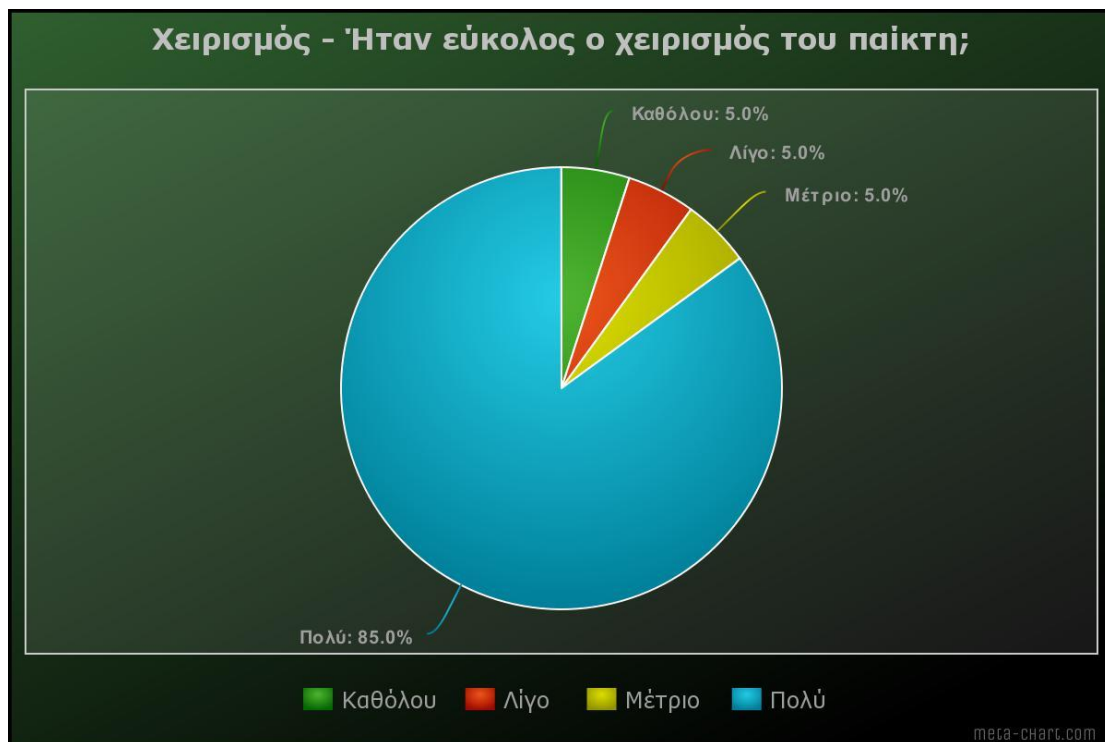
Γενικές Ερωτήσεις

No.1	Παίζεις Βιντεοπαιχνίδια ;
No.2	Παίζεις παιχνίδια 2D (δύο διαστάσεων) ;
No.3	Είσαι Άντρας η Γυναίκα ;
No.4	Πόσο χρονών είσαι ;
No.5	Ποια είναι η αγαπημένη σου κατηγορία Βιντεοπαιχνιδιών;
No.6	Πόσες φορές τη βδομάδα παίζεις Βιντεοπαιχνίδια;
No.7	Τι συσκευή χρησιμοποιείς περισσότερο για τα Βιντεοπαιχνίδια ;

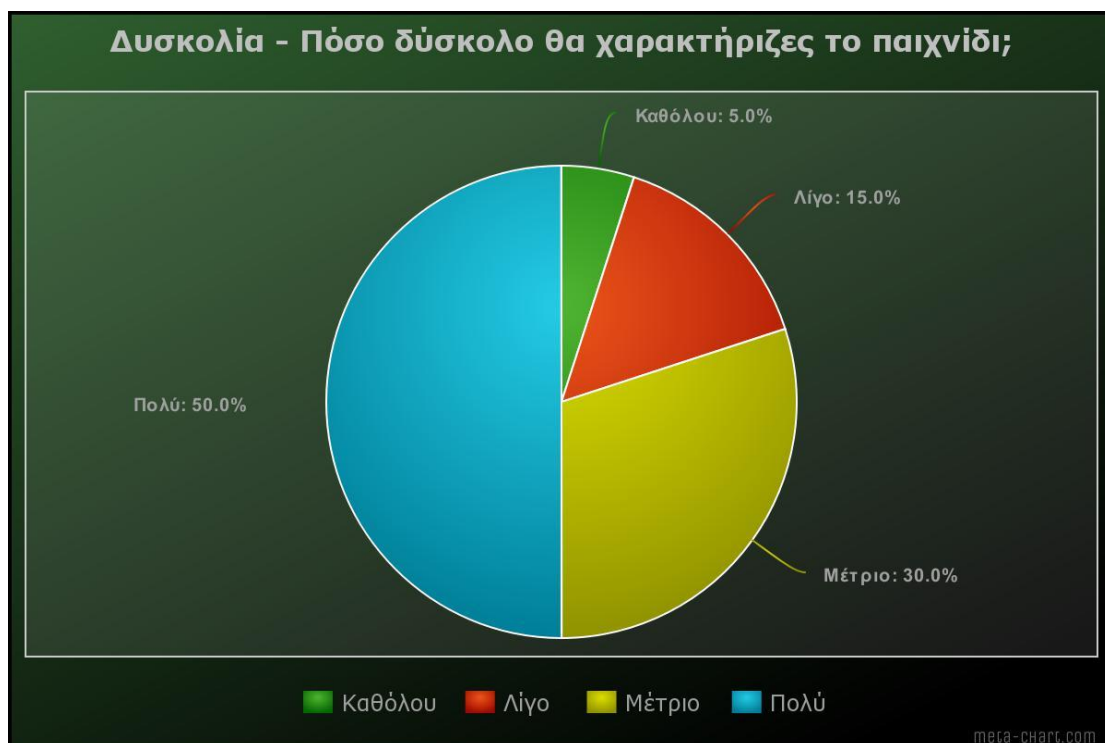
Ερωτήσεις Αξιολόγησης Corfu Runner

Θεματική Ενότητα	No	Ερωτήσεις
Χειρισμός	1	Ήταν εύκολος ο χειρισμός του παίκτη;
Δυσκολία	2	Πόσο δύσκολο θα χαρακτήριζες το παιχνίδι;
Ψυχαγωγία	3	Ήταν διασκεδαστικό;
Γραφικά	4	Σου άρεσαν τα γραφικά και οι χαρακτήρες;
Πρόκληση	5	Ήθελες να παίζεις το παιχνίδι ξανά για καλύτερο σκορ;
Εξοικείωση	6	Ήταν το περιβάλλον του παιχνιδιού οικείο για σένα;
Ικανοποίηση	7	Θα πρότεινες το παιχνίδι σε κάποιον φίλο σου;

⁴⁶ Meigs Tom. *Ultimate Game Design: building game worlds*. (Emeryville, CA: McGraw-Hill, 2003), 319.



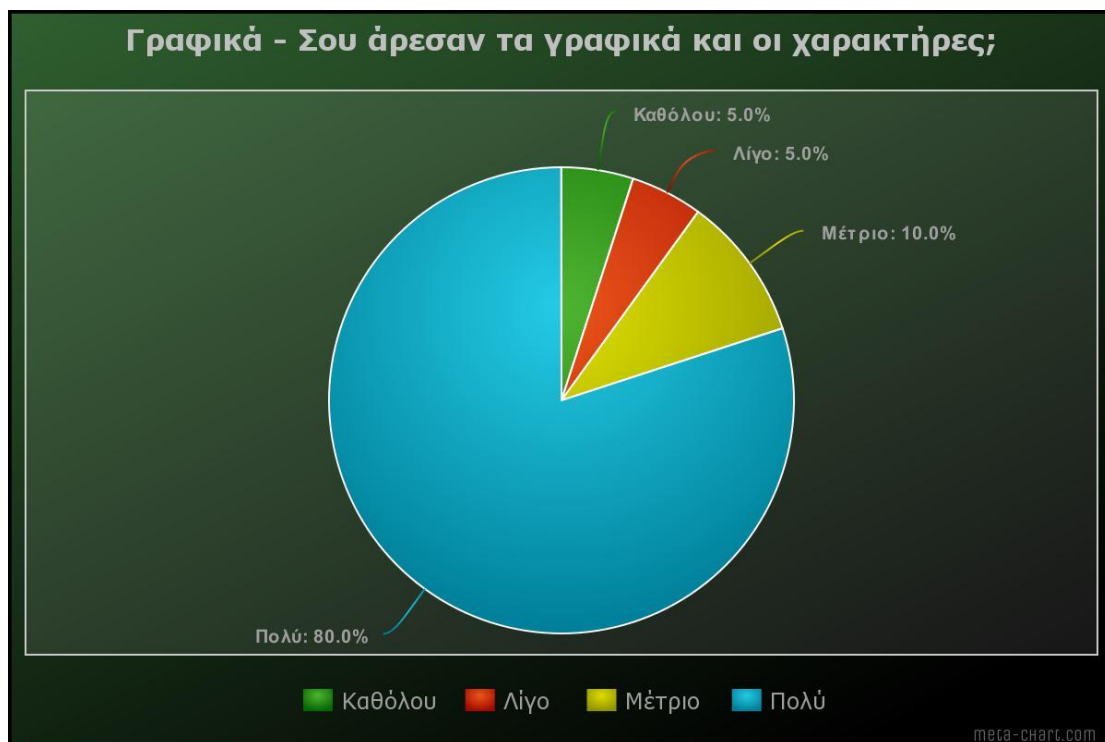
Εικόνα 5.2 Γράφημα 1 – Αποτελέσματα αξιολόγησης



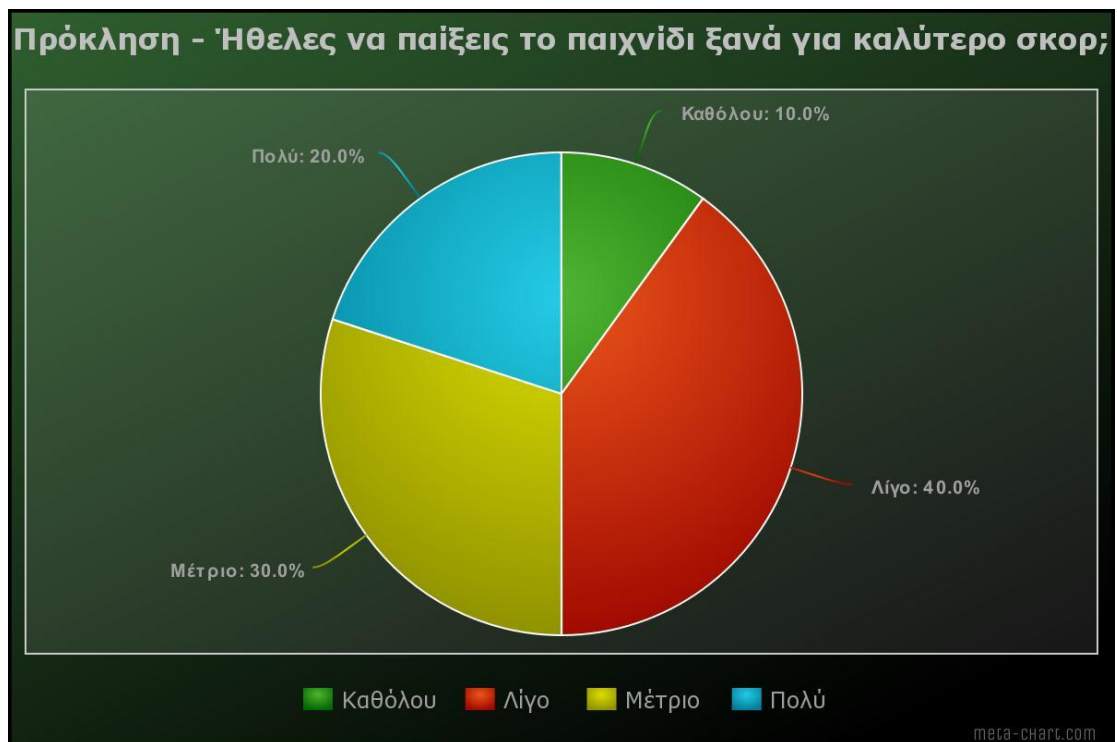
Εικόνα 5.3 Γράφημα 2 – Αποτελέσματα αξιολόγησης



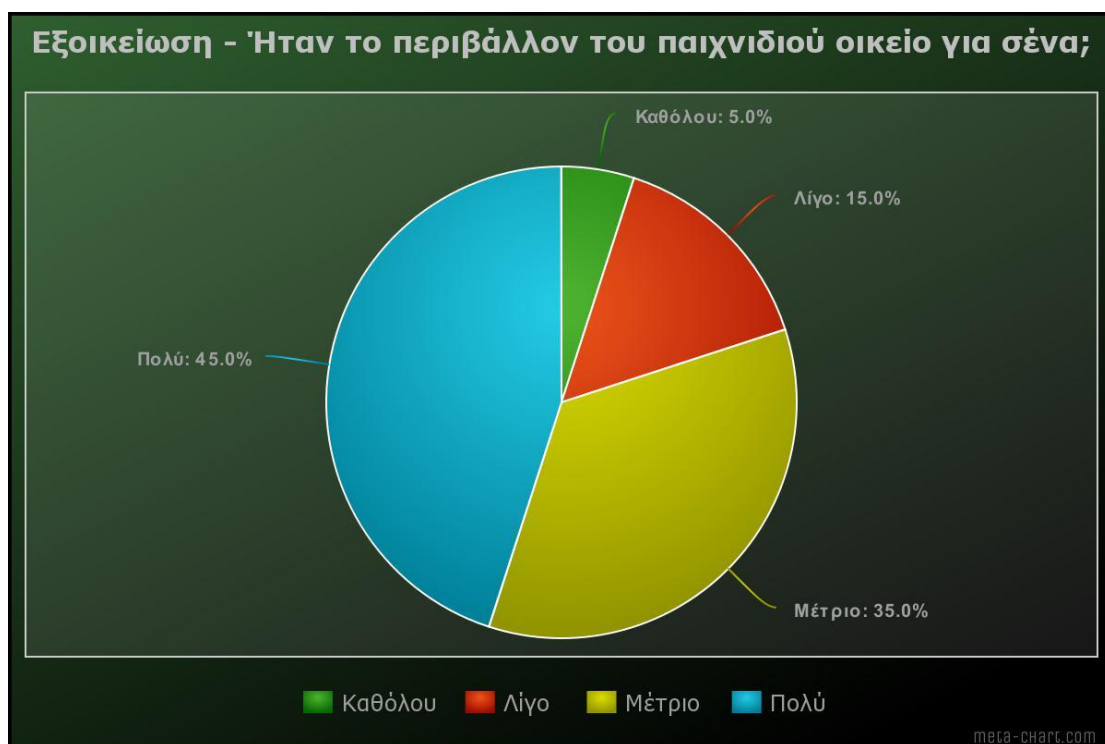
Εικόνα 5.4 Γράφημα 3 – Αποτελέσματα αξιολόγησης



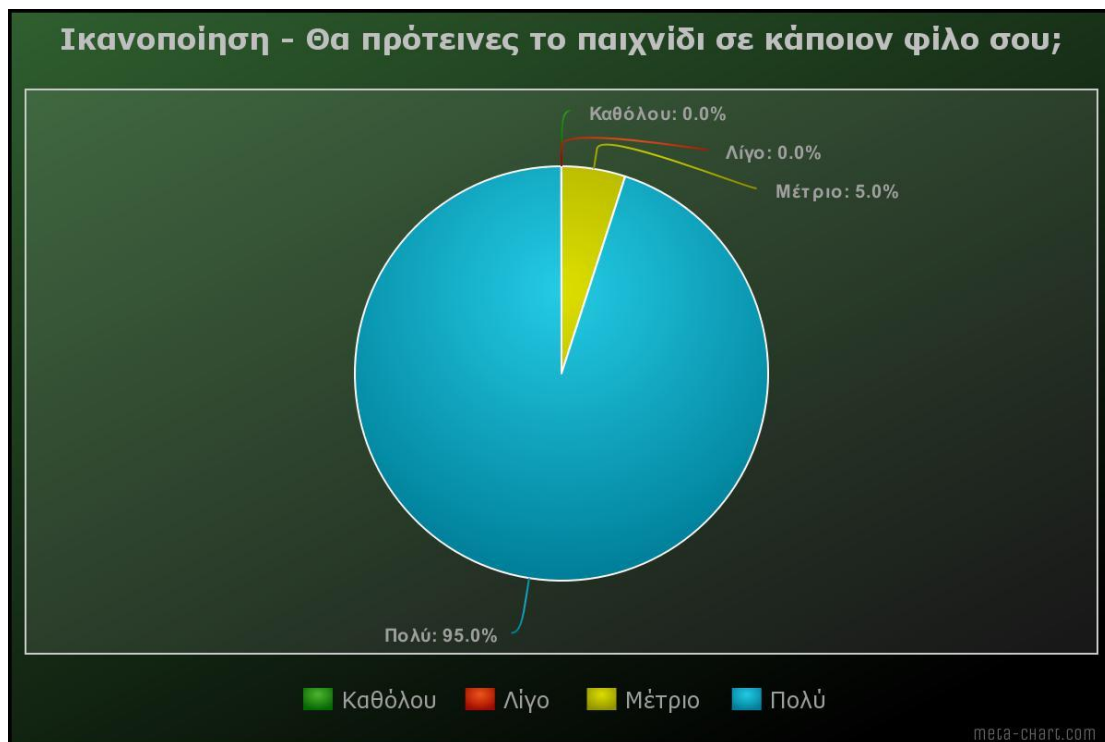
Εικόνα 5.5 Γράφημα 4 – Αποτελέσματα αξιολόγησης



Εικόνα 5.6 Γράφημα 5 – Αποτελέσματα αξιολόγησης



Εικόνα 5.7 Γράφημα 6 – Αποτελέσματα αξιολόγησης



Εικόνα 5.8 Γράφημα 7 – Αποτελέσματα αξιολόγησης

Ανάλυση & Συμπεράσματα Αξιολόγησης

Το γενικό συμπέρασμα που προκύπτει ύστερα από την ανάλυση της αξιολόγησης είναι ότι το παιχνίδι άφησε θετικές εντυπώσεις στο μεγαλύτερο σύνολο των χρηστών. Ο χειρισμός κρίθηκε εύκολος κατά γενικές γραμμές εκτός από τις περιπτώσεις όπου μικρότερες ηλικίες, κυρίως κάτω των δώδεκα ετών, δυσκολεύτηκαν αν το μέσο αναπαραγωγής ήταν tablet λόγω των αποστάσεων μεταξύ των πλήκτρων. Ο βαθμός δυσκολίας προκύπτει ότι ήταν αρκετά απαιτητικός καθώς οι μισοί χρήστες χαρακτήρισαν το παιχνίδι ως δύσκολο. Αυτό είναι ένα στοιχείο δίχως απαραίτητα αρνητικό πρόσημο καθώς ούτε αποτρέπει το χρήστη από το να συνεχίσει να παίζει ούτε είναι πολύ εύκολο ώστε να το τερματίσει αμέσως.

Η διασκέδαση ήταν μία παράμετρος η οποία διαμορφώθηκε από τα γραφικά και τα χιουμοριστικά ηχητικά εφέ και είχε ένα άκρως ικανοποιητικό αποτέλεσμα καθώς η πλειοψηφία των χρηστών διασκέδασαν με το παιχνίδι όπου ήταν και ο απώτερος σκοπός κατά τη δημιουργία του. Σημαντικό ρόλο έπαιξαν και τα γραφικά για αυτό το κομμάτι τα οποία χαρακτηρίστηκαν πολύ καλά αν και ένα μικρό ποσοστό θα τα προτιμούσε λίγο πιο ρεαλιστικά και όχι τόσο της μορφής κόμικς. Όσο αφορά την

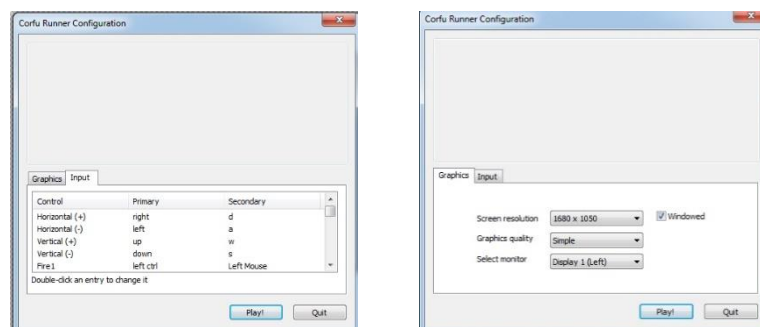
ερώτηση που τέθηκε για καλύτερο σκορ οι απαντήσεις ήταν σχετικά μοιρασμένες καθώς όπως εξήγησαν ορισμένοι χρήστες τους ενδιέφερε περισσότερο να εξερευνήσουν τις πίστες και να ξεκλειδώσουν νέα επίπεδα για να δουν νέα μέρη του νησιού παρά να συλλέξουν νομίσματα και να κάνουν καλύτερο χρόνο.

Το περιβάλλον του παιχνιδιού φάνηκε να είναι οικείο σε ένα αρκετά μεγάλο ποσοστό χρηστών λόγω της διαδεδομένης κατηγορίας παιχνιδιών πλατφόρμας αλλά και του εύκολου και διακριτικού interface. Στην τελευταία ερώτηση που αφορούσε την ικανοποίηση των χρηστών και αν αυτοί θα το πρότειναν σε κάποιον φίλο τους υπήρχε ένα συντριπτικό αποτέλεσμα όπου σχεδόν όλοι οι χρήστες θα πρότειναν ή είχαν προτείνει ήδη το παιχνίδι σε φίλους τους. Ένα στοιχείο άκρως ενθαρρυντικό καθώς ένας ικανοποιημένος χρήστης είναι όχι μόνο το αποτέλεσμα της δουλειάς του game designer αλλά και η καλύτερη διαφήμιση για το ίδιο το παιχνίδι.

Έκδοση

Ψηφιακή Μορφή

Η έκδοση του παιχνιδιού πραγματοποιήθηκε μέσω της ιστοσελίδας και του Playstore. Ο χρήστης θα μπορεί να κατεβάσει το κατάλληλο αρχείο για να το παίξει σε υπολογιστή αλλά και το αντίστοιχο application αν είναι κάτοχος Smartphone με λογισμικό Android. Για χρήση σε υπολογιστή ο χρήστης λαμβάνει ένα αρχείο τύπου .exe και αφού πραγματοποιηθεί η εγκατάσταση του εμφανίζεται η παρακάτω οθόνη όπου μπορεί να ρυθμίσει τις διαστάσεις προβολής του παιχνιδιού αλλά και η ρύθμιση των πλήκτρων για τις ενέργειες που απαιτούνται.



Εικόνα 5.9

Η Έναρξη του παιχνιδιού για υπολογιστή.

Κάθε παιχνίδι που παρέχεται με τη μορφή CD συνοδεύεται από ένα μικρό εγχειρίδιο χρήσης που έχει σκοπό να εξηγήσει το χρήστη τις βασικές λειτουργίες του παιχνιδιού. Οι πρώτες δύο σελίδες είναι απαραίτητο να αναφέρουν το περιεχόμενο του παιχνιδιού δηλαδή αν αυτό για παράδειγμα παρουσιάζει στοιχεία βίας, ναρκωτικά, αν χρειάζεται online σύνδεση και το γονικό έλεγχο που απαιτείται. Επιπλέον αναγράφονται λεπτομέρειες σχετικά με τη χρήση του δίσκου και της κονσόλας, πνευματικών δικαιωμάτων όπως και συστάσεις ορθής χρήσης για την αποτροπή προβλημάτων υγείας. Στη τελευταία σελίδα υπάρχει το customer service της εταιρείας και τρόπους επικοινωνίας με το εκάστοτε γραφείο ανάλογα τη χώρα που βρίσκεται ο χρήστης. Σε περίπτωση όπου η διάθεση του παιχνιδιού γίνεται μόνο online το εγχειρίδιο μπορεί να περιλαμβάνεται σε μια από τις υποσελίδες της επίσημης ιστοσελίδας ⁴⁷

Λόγω ότι οι θήκες των δίσκων γίνονται όλο και μικρότερες το ίδιο ισχύει και για τα εγχειρίδια όπου οι πληροφορίες πρέπει να είναι ξεκάθαρες και συρρικνούμενες ταυτόχρονα. Όπως τα περισσότερα έντυπα, έτσι και σε αυτή τη περίπτωση ακολουθεί ο πίνακας περιεχομένων όπου ο χρήστης μπορεί να ανατρέξει για να βρει αυτό που επιθυμεί. Αν η εγκατάσταση του παιχνιδιού είναι αρκετά περίπλοκη οι πρώτες ενότητες παρέχει πληροφορίες για την ολοκλήρωση της. Στη συνέχεια αν τα παιχνίδια είναι βασισμένα σε μια ιστορία υπάρχει μια σύντομη περίληψη για το τι έχει προηγηθεί για να γνωρίζει ο χρήστης γενικές πληροφορίες του κόσμου όπου θα εισέλθει. Ακολουθεί η ενότητα που αφορά το χειρισμό και συνοδεύεται με το ανάλογο σχεδιάγραμμα στο οποίο αναγράφεται η δράση του κάθε πλήκτρου και μοχλού. Ακολουθεί η ενότητα του μενού, του Gameplay και αυτές των multiplayer και της online σύνδεσης όπου υπάρχουν. Η παραπάνω ταξινόμηση είναι ενδεικτική και η σειρά μπορεί να αλλάξει ανάλογα τον εκδότη και το παιχνίδι του. ⁴⁸

⁴⁷ Marx Christy. *Writing for Animation, Comics and Games*. (Burlington, MA: Focal Press, 2007), 190

⁴⁸ Bob Bates. *Game Design, 2nd Edition*. (Boston: Thomson Course Technology, 2004), 197.

Στη συγκεκριμένη περίπτωση όπου το παιχνίδι είναι διαθέσιμο και στο Playstore ο κατασκευαστής συμπληρώνει ένα ερωτηματολόγιο ώστε να δημιουργηθεί ένας πίνακας με τα χαρακτηριστικά του παιχνιδιού. Αυτές οι πληροφορίες προβάλλονται ως εξής:

Λεπτομέρειες βαθμολογίας			✕
 ΠΑΙΧΝΙΔΙ Η εφαρμογή είναι παιχνίδι.			
Σύστημα αξιολόγησης	Κατηγορία αξιολόγησης	Περιγραφείς	
Australian Classification Board (ACB) Αυστραλία	 Γενικά	Για όλους	
Classificação Indicativa (ClassInd) Βραζιλία	 Κατάλληλο για όλες τις ηλικίες		
Entertainment Software Rating Board (ESRB) Βόρεια Αμερική	 Κατάλληλο για όλους		
Game Rating and Administration Committee (GRAC) Νότια Κορέα	 Κατάλληλο για όλες τις ηλικίες		
Pan-European Game Information (PEGI) Ευρώπη	 PEGI 3		
Unterhaltungssoftware Selbstkontrolle (USK) Γερμανία	 USK: Κατάλληλο για όλες τις ηλικίες		
IARC Generic Υπόλοιπος κόσμος	 Αξιολογήθηκε ως κατάλληλο για ηλικίες άνω των 3 ετών		
Google Play Ρωσία	 Αξιολογήθηκε ως κατάλληλο για ηλικίες άνω των 3 ετών		

Εικόνα 5.10 Τα αποτελέσματα αξιολόγησης για τις κατηγορίες στις οποίες εντάσσεται το παιχνίδι.

Προώθηση

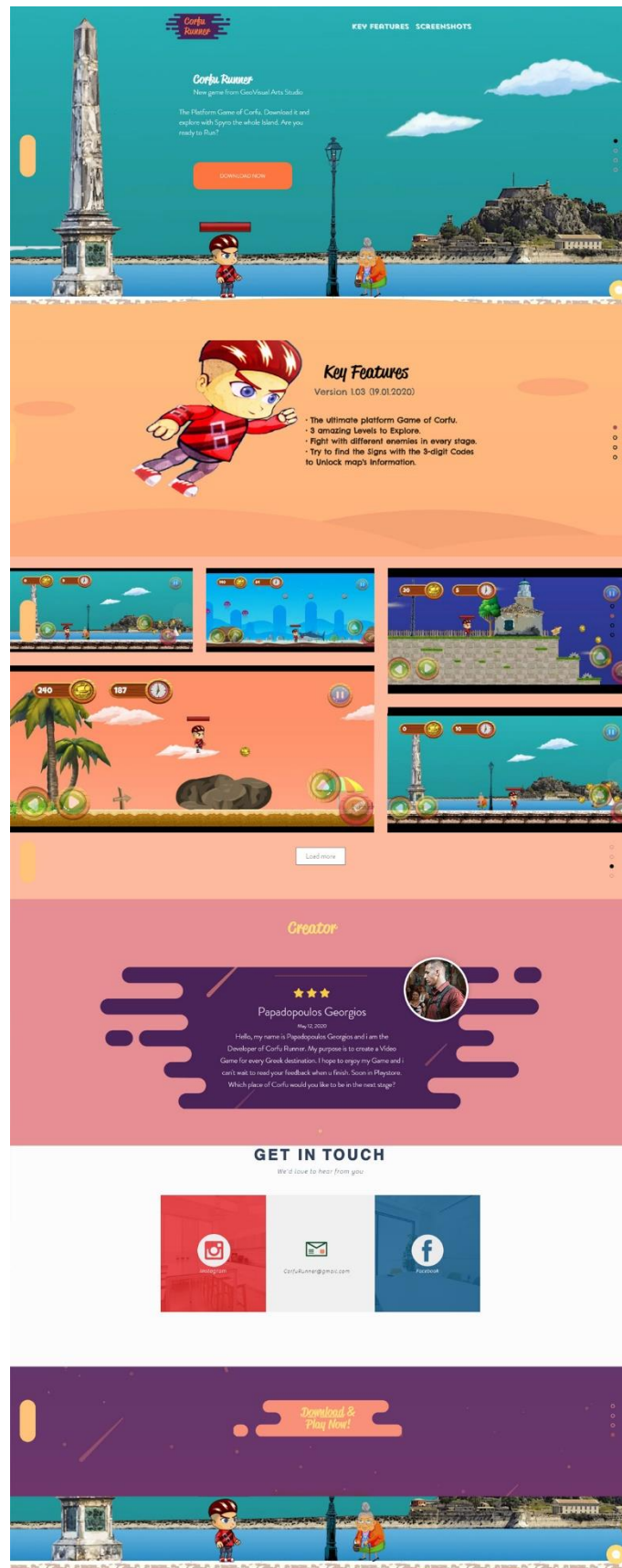
Δημιουργία Ιστοσελίδας (Web Design)

Διανύουμε την ψηφιακή εποχή όπου σχεδόν κάθε υπηρεσία και προϊόν χρειάζεται την επίσημη ιστοσελίδα του όπου ο χρήστης μπορεί να ανατρέξει για όλη την πληροφορία που χρειάζεται. Ακόμα μεγαλύτερη είναι η ανάγκη όταν το προϊόν είναι ένα βιντεοπαιχνίδι το οποίο αποτελεί μέσο ψηφιακής διασκέδασης. Η ιστοσελίδα ενός βιντεοπαιχνιδιού έχει όλα όσα χρειάζεται ο χρήστης και του παρέχει ανά πάσα στιγμή πληροφορίες, οπτικοακουστικό υλικό αλλά και λύσεις σε πιθανές ερωτήσεις και προβλήματα. Πιθανή ενότητα η οποία εμφανίζεται όλο και περισσότερο στα παιχνίδια τα τελευταία χρόνια είναι τα online-stores όπου ο χρήστης μπορεί να αγοράσει το παιχνίδι σε ψηφιακή μορφή καθώς τα CD βρίσκονται σε πτωτική πορεία όλο και περισσότερο.⁴⁹

Μία ιστοσελίδα για να θεωρείται επιτυχημένη πέρα από το καλλιτεχνικό κομμάτι οφείλει να είναι και πλήρως λειτουργική. Να φορτώνει γρήγορα στον περιηγητή του χρήστη και να αποτελείται από μία δομή ξεκάθαρη για την όσο δυνατόν καλύτερη πλοήγηση του χρήστη. Η μουσική δεν προτείνεται καθώς πολλές φορές αποσυντονίζει το χρήστη κατά την αναζήτηση του περιεχομένου που επιθυμεί. Η κύρια γλώσσα προγραμματισμού που απαιτείται είναι η HTML για μία βασική ιστοσελίδα με απλά χαρακτηριστικά ενώ για περεταίρω δυνατότητες χρησιμοποιούνται οι CSS, Javascript, Java, PHP και MySQL. Πλέον υπάρχουν ειδικές πλατφόρμες όπου ο χρήστης μπορεί να δημιουργήσει τη δική του ιστοσελίδα δίχως να γνωρίζει κάποια από τις παραπάνω γλώσσες, χρησιμοποιώντας τη μέθοδο Drag n Drop όπου σέρνει τα στοιχεία που επιθυμεί και τα τροποποιεί. Μια από αυτές είναι και η πλατφόρμα Wix στην οποία στήθηκε και η ιστοσελίδα του παιχνιδιού.⁵⁰

⁴⁹ Marx Christy. *Writing for Animation, Comics and Games*. (Burlington, MA: Focal Press, 2007), 190.

⁵⁰ Rollings Andrew & Morris Dave. *Game Architecture and Design: A New Edition*. (Indianapolis IN: New Riders, 2004), 713.



Εικόνα 5.11 Η ιστοσελίδα του παιχνιδιού στην έκδοση για κινητά.

Διανύουμε μια εποχή όπου τα μέσα κοινωνικής δικτύωσης αποτελούν αναπόσπαστο κομμάτι για το μεγαλύτερο μέρος της κοινωνίας μας. Σχεδόν όλοι καθημερινά συνδέονται σε μια από τις πλέον δημοφιλείς πλατφόρμες με σκοπό την αλληλεπίδραση με άλλους χρήστες. Αυτή τη στιγμή στο διαδίκτυο μπορεί κανείς να ανακαλύψει πλατφόρμες με διαφορετικά χαρακτηριστικά η μία από την άλλη, τρεις όμως είναι αυτές με τους περισσότερους χρήστες και οι πλέον διαδεδομένες. Στη πρώτη θέση και με τους περισσότερους χρήστες παγκοσμίως βρίσκεται το Facebook με το συνολικό αριθμό των χρηστών να αγγίζει τα δύομιση δισεκατομμύρια. Πρόκειται για μια πλατφόρμα όπου διαθέτει εργαλεία με τα οποία ο χρήστης μπορεί να συνδεθεί με χρήστες από όλο τον κόσμο, να αναρτήσει φωτογραφίες, βίντεο, κείμενα, να συνομιλήσει μέσα από ατομικά και ομαδικά chat room, να συμμετέχει σε ομάδες καθώς και πολλές άλλες λειτουργίες οι οποίες ανανεώνονται συνεχώς με καινούργιες.⁵¹

Το επόμενο σε σειρά χρηστών μέσω κοινωνικής δικτύωσης είναι το Youtube με το περιεχόμενο να είναι βασισμένο πάνω στα βίντεο. Ο κάθε χρήστης μπορεί να δημιουργήσει το δικό του κανάλι και να ανεβάζει τα βίντεο που δημιουργεί ή ακόμα να προβάλει και περιεχόμενο σε ζωντανή αναμετάδοση. Κάτω από κάθε βίντεο υπάρχει ο αριθμός προβολών, πληροφορίες του δημιουργού σχετικά με το βίντεο καθώς και κουμπιά αλληλεπίδρασης με τα οποία οι χρήστες μπορούν να δηλώσουν εάν τους αρέσει το βίντεο ή όχι καθώς και να αφήσουν κάποιο σχόλιο. Το τρίτο δίκτυο και αυτό με πάνω από ένα δισεκατομμύριο χρήστες είναι το Instagram. Πρόκειται για πλατφόρμα η οποία αρχικά ήταν βασισμένη στις φωτογραφίες αλλά πλέον οι χρήστες μπορούν να ανεβάσουν και βίντεο στο προφίλ τους. Ο χρήστης έχει τη δυνατότητα να ανεβάσει μια εικόνα προφίλ, να γράψει ένα σύντομο βιογραφικό, να ανεβάζει όσες φωτογραφίες επιθυμεί στην συλλογή του και να ακολουθεί άλλους χρήστες ώστε να του εμφανίζονται στην αρχική σελίδα του το πρόσφατο περιεχόμενο τους. Όλα αυτά τα δίκτυα πέρα από την αλληλεπίδραση και την ψυχαγωγία που προσφέρουν στο χρήστη αποτελούν και εργαλεία marketing τα οποία μπορεί ο καθένας να εκμεταλλευτεί.

⁵¹ Humphreys Ashlee. *Social Media*. (New York: Oxford University Press, 2016), 84.

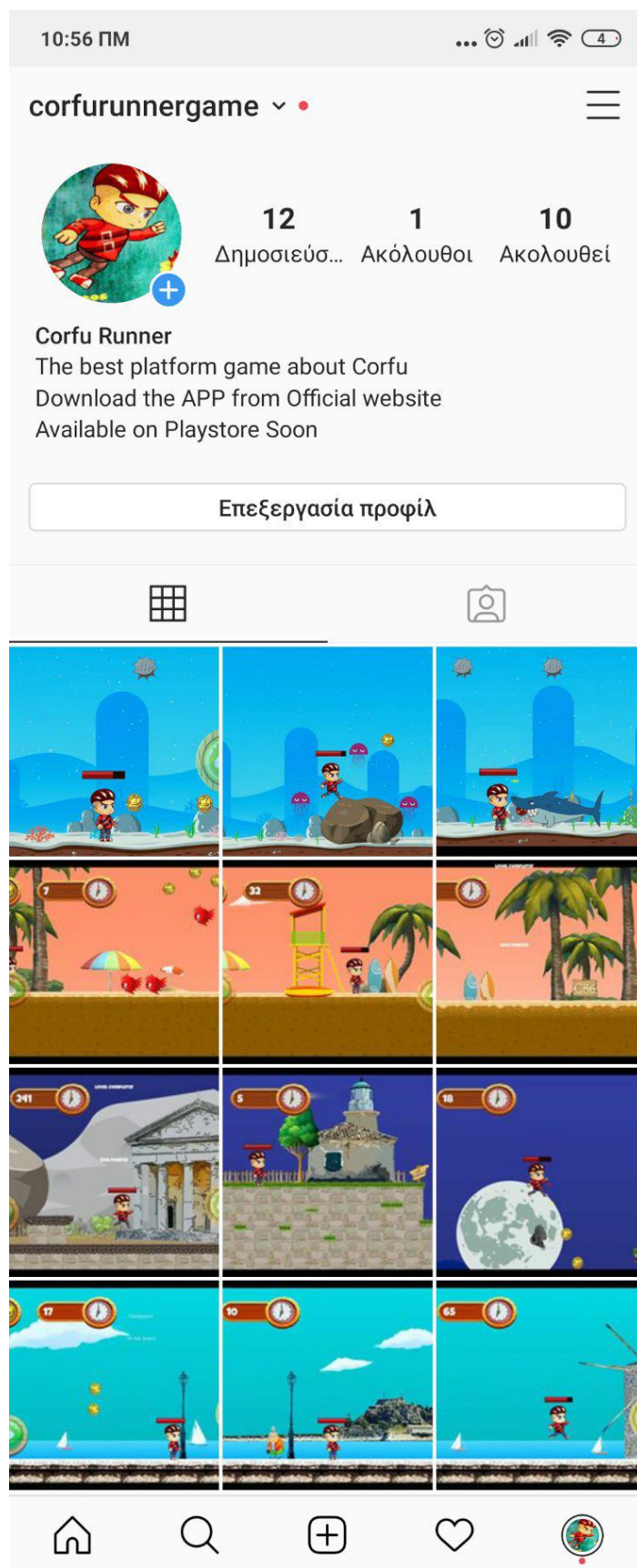
Ένα προϊόν μπορεί να διαφημιστεί στα μέσα κοινωνικής δικτύωσης είτε άμεσα από την ίδια τη πλατφόρμα είτε έμμεσα ως περιεχόμενο που ανεβάζει ο χρήστης. Η άμεση διαφήμιση για να πραγματοποιηθεί, χρειάζεται να ορίσει ο χρήστης το ποσό που επιθυμεί να διαθέσει και στη συνέχεια η πλατφόρμα προβάλλει το προϊόν με βάση την τιμή που έχει οριστεί. Πέρα από το ποσό ο χρήστης μπορεί να ορίσει τη γεωγραφική περιοχή των χρηστών που θα προβάλλεται η διαφήμιση, το φύλο, την ηλικία, την ώρα προβολής ενώ η επιλογή μπορεί να γίνεται ακόμα και με βάση τις δραστηριότητες και τις αναζητήσεις τους. Στο Facebook και στο Instagram οι διαφημίσεις προβάλλονται ανάμεσα στις αναρτήσεις των χρηστών που εμφανίζονται στην αρχική σελίδα ή ανάμεσα στα stories, ενώ Youtube η διαφήμιση εμφανίζεται είτε στην αρχή του βίντεο είτε σε ενδιάμεσα σημεία αν έχει μεγάλη διάρκεια.⁵²

Η άλλη κατηγορία με την οποία μπορεί να δημιουργηθεί μια διαφημιστική καμπάνια είναι η έμμεση ή αλλιώς Word of mouth. Πρόκειται για την ένα είδος επικοινωνίας και διαφήμισης όπου το προϊόν μεταδίδεται από στόμα σε στόμα μεταξύ των χρηστών. Είναι μια τεχνική η οποία εμπνέει μεγαλύτερη εμπιστοσύνη καθώς ο χρήστης-καταναλωτής μαθαίνει για το προϊόν από έναν άλλο χρήστη ο οποίος το προτείνει και πιθανόν να το έχει δοκιμάσει κιόλας. Αυτή η τεχνική βασίζεται στο ότι θα πρέπει το διαφημιζόμενο αγαθό να εμφανίζεται στις αναρτήσεις των χρηστών. Αυτό έρχεται ως αποτέλεσμα διάφορων τεχνικών και μεθόδων που ακολουθούν όσοι ασχολούνται με το Digital Marketing. Η χρήση των Hashtags, online διαγωνισμοί και η δημιουργία ομάδων αλληλεπίδρασης είναι μόνο μερικές από τις τεχνικές που χρησιμοποιούνται για την προβολή και ανάδειξη των προϊόντων μέσα από τους ίδιους τους χρήστες.⁵³

Για τη προώθηση του παιχνιδιού με τη χρήση των Social Media δημιουργήθηκε μία σελίδα στο Facebook, ένα προφίλ στο Instagram και ένα βίντεο για το Youtube. Στις δύο πρώτες πλατφόρμες ανέβηκαν εικόνες με περιεχόμενο του παιχνιδιού και δημιουργήθηκαν περιγραφές σχετικά με το τι προβάλλεται διαδραστικού χαρακτήρα. Έγινε έρευνα για τα κατάλληλα hashtag όπως επίσης και για το κοινό που απευθύνεται το βιντεοπαιχνίδι.

⁵² Humphreys Ashlee. *Social Media*. (New York: Oxford University Press, 2016), 195.

⁵³ Humphreys Ashlee. *Social Media*. (New York: Oxford University Press, 2016), 196.



Εικόνα 5.12 Η σελίδα του παιχνιδιού στο Instagram.



Corfu Runner ▼



Επισκόπηση



Διαφημίσεις



Ειδοποιή



Corfu Runner

Δημιουργήστε ένα όνομα Σελίδας σε μορφή
@username

ΠΡΟΣΘΗΚΗ ΚΟΥΜΠΙΟΥ



Δημοσίευση



Φωτογραφία



Πρωώθηση



Προβολή ως



Δεν υπάρχει ακόμα αξιολόγηση • Βιντεοπαιχνίδι

Αρχική σελίδα

Δημοσιεύσεις

Εκδηλώσεις

Κριτικές



Αναπτύξτε την επιχείρησή σας



Πρωθήστε τη Σελίδα σας για 2 €

Προσελκύστε περισσότερα άτομα από την τοποθεσία

Κεφάλαιο 6: Συμπεράσματα

Η εργασία αφού ολοκληρώθηκε είχε εκπληρώσει όλες τις απαιτούμενες προσδοκίες. Το παιχνίδι είναι πλήρως λειτουργικό όπως και η ιστοσελίδα. Οι χρήστες που το δοκίμασαν άφησαν θετικές κριτικές τόσο για τον εύκολο χειρισμό όσο και για τη διασκέδαση που προσφέρει. Εξίσου σημαντικό είναι ότι όσοι είναι γονείς θα επέτρεπαν στα παιδιά τους να το παίζουν λόγω του κατάλληλου περιβάλλοντος. Σημαντικό στοιχείο το οποίο δεν πέρασε απαρατήρητο από τους χρήστες ήταν η κλιμακωτή αύξηση δυσκολίας στο κάθε επίπεδο προσφέροντας εξοικείωση τόσο στο χειρισμό όσο και στο περιβάλλον του παιχνιδιού.

Εκ πρώτης όψεως το παιχνίδι φαίνεται κάτι απλοϊκό, λόγω της τεράστιας ανάπτυξης των γραφικών και των παιχνιδιών των τελευταίων ετών. Όμως δεν παύει να χρειάζονται πολλές ώρες ακόμα και για τη παραμικρή λεπτομέρεια που μπορεί να μοιάζει ασήμαντη. Κάθε αρχείο κώδικα αποτελεί και μία πρόκληση καθώς μπορεί να εμφανιστούν διάφορα προβλήματα λειτουργικότητας. Αν και η αρχική εκτίμηση ήταν πως το παιχνίδι θα προοριζόταν αποκλειστικά για υπολογιστή, κατά την ολοκλήρωση αποφασίστηκε η έκδοση του και για κινητές συσκευές όπου τα παιχνίδια δύο διαστάσεων είναι ιδιαίτερα δημοφιλή λόγω των χαμηλών απαιτήσεων που χρειάζονται για την εκτέλεση τους αλλά και για τον εύκολο χειρισμό τους. Όπως αποδείχθηκε ήταν μία σωστή απόφαση όπως φάνηκε από τις αξιολογήσεις των χρηστών στο αντίστοιχο ερώτημα που τους τέθηκε.

Η χρήση της ιστοσελίδας και των social media πρόσφεραν τη δυνατότητα στο παιχνίδι να διαδοθεί με μεγαλύτερη ευκολία αλλά να αποτελούν και συγχρόνως μια πηγή πληροφοριών και όχι μόνο, προσφέροντας τη δυνατότητα στο χρήστη να το κατεβάσει ή να το μοιραστεί με τους φίλους του ανά πάσα ώρα και στιγμή. Επιπλέον πέρα από τα διαφημιστικά οφέλη αποτελούν έναν διάυλο επικοινωνίας ανάμεσα στο χρήστη και τον κατασκευαστή όπου ο πρώτος μπορεί να αφήσει feedback, να αναφέρει τεχνικά προβλήματα ή και βελτιώσεις.

Κεφάλαιο 7: Προτάσεις

Σε ένα τόσο πολυδιάστατο πρότζεκτ υπάρχουν περιθώρια βελτίωσης σε κάθε επιμέρους κατηγορία. Όσον αφορά το κομμάτι της ιστορίας και του σεναρίου, το παιχνίδι πέρα από επιπλέον πίστες από το νησί της Κέρκυρας μπορεί να γίνει διαθέσιμο σαν παιχνίδι-τουριστικός οδηγός και για άλλα μέρη της Ελλάδας ή και του κόσμου όπου ο χρήστης να μπορεί να επιλέγει τη πόλη που επιθυμεί και να την εξερευνάει μέσα από τον τοπικό της ήρωα/χαρακτήρα. Αλλαγές μπορούν να υπάρξουν και στη ροή του παιχνιδιού ώστε να μην υπάρχει μια συγκεκριμένη ροή από το ένα μέρος του νησιού στο άλλο αλλά μέσα από μια μη γραμμική διαδρομή ο παίκτης να μπορεί να επιλέγει ανάμεσα σε διακλαδώσεις για τον επόμενο σταθμό όπου επρόκειτο να επισκεφθεί.

Το Gameplay μπορεί να γίνει ακόμα πιο εντυπωσιακό και ρεαλιστικό. Η προσθήκη νέων χαρακτήρων, περισσότερων animation, η δυνατότητα συλλογής περεταίρω αντικειμένων πέρα από τα νομίσματα, η χρήση επιπλέον όπλων ή ικανοτήτων όπως και η εμφάνιση ενός Big-Boss ανά τρία με πέντε επίπεδα θα πρόσφεραν στο χρήστη περισσότερη διασκέδαση. Συγκεκριμένος αριθμός πυρομαχικών θα έδινε έναν επιπλέον βαθμό δυσκολίας στο παιχνίδι ώστε ο παίκτης να τα χρησιμοποιεί μόνο όταν κρίνεται απαραίτητο. Τέλος η εμφάνιση του χάρτη του νησιού και η μετακίνηση από το ένα μέρος στο άλλο ανάμεσα στην εναλλαγή των επιπέδων θα έδινε στο χρήστη μία πιο ολοκληρωμένη εικόνα για το σύνολο των επιπέδων και τον κόσμο του παιχνιδιού.

Στο κομμάτι της διαφήμισης και του Marketing, σύμφωνα με τη τοποθεσία που αντιστοιχεί σε κάθε πίστα, θα μπορούν να εμφανίζονται και οι ανάλογες επιχειρήσεις της περιοχής μέσα στη πίστα του παιχνιδιού πληρώνοντας το ανάλογο αντίτιμο για να διαφημιστούν. Όσον αφορά τα social Media ο χρήστης θα μπορούσε να συνδέεται και να μοιράζεται τα στοιχεία του από τα μέσα κοινωνικής δικτύωσης και να φαίνεται η πρόοδος του στο χάρτη έναντι αυτή των φίλων του και να ανταλλάζουν αντικείμενα τα οποία έχουν συλλέξει κατά τη διάρκεια των αποστολών για να ξεκλειδώσουν νέες διαδρομές, αντικείμενα και στολές ώστε να αλλάζουν την εμφάνιση του χαρακτήρα σε ρούχα, αξεσουάρ και όπλα. Τέλος μια τρισδιάστατη έκδοση του παιχνιδιού σίγουρα θα ήταν μια πρόκληση αλλά είναι κάτι που για την υλοποίηση του απαιτείται η συγκρότηση μίας ομάδας με ειδικευμένα στελέχη κυρίως στο θέμα των τρισδιάστατων γραφικών και του προγραμματισμού λόγω των μεγαλύτερων απαιτήσεων.

Βιβλιογραφία

- ❖ Adams Ernest. *Fundamentals of Game Design, 2nd Edition*. Berkeley, CA: Pearson Education Inc, 2010.
- ❖ Bethke Erik. *Game development and Production*. Texas: Wordware Publishing Inc, 2003.
- ❖ Bob Bates. *Game Design, 2nd Edition*. Boston: Thomson Course Technology, 2004.
- ❖ Cairns George. *Teach yourself Photoshop*. London: Future Publishing, 2014.
- ❖ Crawford Cris. *The Art of Computer Game Design*. Washington: Washington State University, 1997.
- ❖ Dan Irish. *The Game Producer's Handbook*. Boston: Thomson Course Technology, 2005.
- ❖ Feil John, Scattergood Marc. *Beginning Game Level Design*. Boston: Thomson Course Technology, 2005.
- ❖ Franson David & Thomas Eric. *Game Character Design Complete*. Boston: Thomson Course Technology, 2007.
- ❖ Jesse Schell. *The Art of Game Design 2nd Edition*. US: CRC Press, 2015.
- ❖ Lee Sheldon. *Character development and storytelling*. Boston: Thomson Course Technology, 2004.
- ❖ Marx Christy. *Writing for Animation, Comics and Games*. Burlington, MA: Focal Press, 2007.
- ❖ Meigs Tom. *Ultimate Game Design: building game worlds*. Emeryville, CA: McGraw-Hill, 2003.
- ❖ Richard Rouse. *Game Design: Theory & Practise*. Texas: Wordware Publishing Inc, 2005.
- ❖ Rollings Andrew & Morris Dave. *Game Architecture and Design: A New Edition*. Indianapolis: New Riders, 2004.
- ❖ Scott Rogers. *Level Up. The Guide to great video Game Design*. United Kingdom: Wiley, 2014.
- ❖ Thorn Alan. *Unity Animation Essentials*. Birmingham: Packt Publishing, 2015.
- ❖ Humphreys Ashlee. *Social Media*. New York: Oxford University Press, 2016.

Πηγές Εικόνων

- Εικόνα 1.1 <https://arstechnica.com/>
- Εικόνα 2.1 <https://creativecommons.org/> - ReinoGame
- Εικόνα 2.2 <https://creativecommons.org/> - PlayStation Europe
- Εικόνα 2.3 <https://appscharger.com/evil-lands-online-action-rpg/>
- Εικόνα 2.4 <https://www.l2vanir.com/>
- Εικόνα 2.5 <https://www.strategygamer.com/>
- Εικόνα 2.6 <https://commons.wikimedia.org/>
- Εικόνα 2.7 <https://creativecommons.org/> - PlayStation Europe
- Εικόνα 2.8 <https://naaweb.org/>
- Εικόνα 2.9 <https://creativecommons.org/> - arcticpenguin
- Εικόνα 3.1 <https://unsplash.com/> - Med Badr Chemmaoui
- Εικόνα 4.1 <https://visualstudio.microsoft.com/>
- Εικόνα 4.3 <https://www.ubisoft.com/>
- Εικόνα 4.5 <https://visualstudio.microsoft.com/>
- Εικόνα 4.6 <https://www.ubisoft.com/>
- Εικόνα 4.7 <https://www.ubisoft.com/>
- Εικόνα 4.9 <https://docs.unity3d.com/Manual/AnimationOverview.html>
- Εικόνα 5.1 <https://www.ubisoft.com/>
- Εικόνα 5.10 <https://play.google.com/apps/publish>
- Εικόνα 5.11 <https://geopapavartist.wixsite.com/mysite>
- Εικόνα 5.12 <https://www.instagram.com/corfurunnergame/>
- Εικόνα 5.13 <https://www.facebook.com/>

Παραρτήματα

Παράρτημα Α: Προγραμματισμός

Στο Παρακάτω παράρτημα παρουσιάζονται όλα τα αρχεία κώδικα τα οποία χρησιμοποιήθηκαν κατά τη διάρκεια της ανάπτυξης του παιχνιδιού. Το όνομα του κάθε αρχείου προσδιορίζει και τη λειτουργικότητα του κώδικα που εμπεριέχει.

A1. Auto Destroy Particle System

```
using UnityEngine;

class AutoDestroyParticleSystem : MonoBehaviour
{
    private ParticleSystem _particleSystem;

    public void Start()
    {
        _particleSystem = GetComponent<ParticleSystem>();
    }

    public void Update()
    {
        if (_particleSystem.isPlaying)
            return;

        Destroy(gameObject);
    }
}
```

A2. Background Parallax

using UnityEngine;

```
public class BackgroundParallax : MonoBehaviour
{
    public Transform[] Backgrounds;
    public float ParallaxScale;
    public float ParallaxReductionFactor;
    public float Smoothing;

    private Vector3 _lastPosition;

    public void Start()
    {
        _lastPosition = transform.position;
    }

    public void Update()
    {
        var parallax = (_lastPosition.x - transform.position.x) * ParallaxScale;

        for (var i = 0; i < Backgrounds.Length; i++)
        {
            var backgroundTargetPosition = Backgrounds[i].position.x + parallax * (i *
ParallaxReductionFactor + 1);
            Backgrounds[i].position = Vector3.Lerp(
                Backgrounds[i].position,
                new Vector3(backgroundTargetPosition, Backgrounds[i].position.y,
Backgrounds[i].position.z),
                Smoothing * Time.deltaTime);
        }

        _lastPosition = transform.position;
    }
}
```

A3. Camera control

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour
{
    public Transform Player;

    public Vector2
        Margin,
        Smoothing;

    public BoxCollider2D Bounds;
    public Camera camera;

    private Vector3
        _min,
        _max;

    public bool IsFollowing { get; set; }

    public void Start()
    {
        _min = Bounds.bounds.min;
        _max = Bounds.bounds.max;
        IsFollowing = true;
        camera = GetComponent<Camera>();
    }

    public void Update()
```

```

{
    var x = transform.position.x;
    var y = transform.position.y;

    if (IsFollowing)
    {
        if (Mathf.Abs(x - Player.position.x) > Margin.x)
        {
            x = Mathf.Lerp(x, Player.position.x, Smoothing.x * Time.deltaTime);
        }

        if (Mathf.Abs(y - Player.position.y) > Margin.x)
        {
            y = Mathf.Lerp(y, Player.position.y, Smoothing.y * Time.deltaTime);
        }
    }

    var cameraHalfWidth = camera.orthographicSize * ((float)Screen.width /
Screen.height);

    x = Mathf.Clamp(x, _min.x + cameraHalfWidth, _max.x - cameraHalfWidth);

    y = Mathf.Clamp(y, _min.y + camera.orthographicSize, _max.y -
camera.orthographicSize);

    transform.position = new Vector3(x, y, transform.position.z);
}
}

```

A4. Centered Text Positioner

using UnityEngine;

```
public class CenteredTextPositioner : IFloatingTextPositioner
{
    private readonly float _speed;
    private float _textPosition;

    public CenteredTextPositioner (float speed)
    {
        _speed = speed;
    }

    public bool GetPosition(ref Vector2 position, GUIContent content, Vector2 size)
    {
        _textPosition += Time.deltaTime * _speed;
        if (_textPosition > 1)
            return false;

        position = new Vector2(Screen.width / 2f - size.x / 2f, Mathf.Lerp(Screen.height
/ 2f + size.y, 0, _textPosition));
        return true;
    }
}
```

A5. Character Controller 2D

```
using System.Collections;
```

```
using UnityEngine;
```

```
public class CharacterController2D : MonoBehaviour
```

```
{
```

```
    private const float SkinWidth = .02f;
```

```
    private const int TotalHorizontalRays = 8;
```

```
    private const int TotalVerticalRays = 4;
```

```
    private static readonly float SlopeLimitTangent = Mathf.Tan(75 *  
    Mathf.Deg2Rad);
```

```
    public LayerMask PlatformMask;
```

```
    public ControllerParameters2D DefaultParameters;
```

```
    public ControllerState2D State { get; private set; }
```

```
    public Vector2 Velocity { get { return _velocity; } }
```

```
    public bool HandleCollisions { get; set; }
```

```
    public ControllerParameters2D Parameters { get { return _overrideParameters ??  
    DefaultParameters; } }
```

```
    public GameObject StandingOn { get; private set; }
```

```
    public Vector3 PlatformVelocity { get; private set; }
```

```
    public bool CanJump
```

```
{
```

```
    get
```

```
{
```

```
        if (Parameters.JumpRestrictions ==  
        ControllerParameters2D.JumpBehavior.CanJumpAnywhere)
```

```
            return _jumpIn <= 0;
```

```

        if (Parameters.JumpRestrictions ==
ControllerParameters2D.JumpBehavior.CanJumpOnGround)
            return State.IsGrounded;

        return false;
    }
}

private Vector2 _velocity;
private Transform _transform;
private Vector3 _localScale;
private BoxCollider2D _boxCollider;
private ControllerParameters2D _overrideParameters;
private float _jumpIn;
private GameObject _lastStandingOn;

private Vector3
    _activeGlobalPlatformPoint,
    _activeLocalPlatformPoint;

private Vector3
    _raycastTopLeft,
    _raycastBottomRight,
    _raycastBottomLeft;

private float
    _verticalDistanceBetweenRays,
    _horizontalDistanceBetweenRays;

public void Awake()
{
    HandleCollisions = true;

```

```

    State = new ControllerState2D();

    _transform = transform;

    _localScale = transform.localScale;

    _boxCollider = GetComponent<BoxCollider2D>();

    var colliderWidth = _boxCollider.size.x * Mathf.Abs(transform.localScale.x) -
(2 * SkinWidth);

    _horizontalDistanceBetweenRays = colliderWidth / (TotalVerticalRays - 1);

    var colliderHeight = _boxCollider.size.y * Mathf.Abs(transform.localScale.y) -
(2 * SkinWidth);

    _verticalDistanceBetweenRays = colliderHeight / (TotalHorizontalRays - 1);
}

public void AddForce(Vector2 force)
{
    _velocity = force;
}

public void SetForce(Vector2 force)
{
    _velocity += force;
}

public void SetHorizontalForce(float x)
{
    _velocity.x = x;
}

public void SetVerticalForce(float y)
{

```

```

        _velocity.y = y;
    }

    public void Jump()
    {
        //TODO: Moving Platform Support
        AddForce(new Vector2(0, Parameters.JumpMagnitude));
        _jumpIn = Parameters.JumpFrequency;
    }

    public void LateUpdate()
    {
        _jumpIn -= Time.deltaTime;
        _velocity.y += Parameters.Gravity * Time.deltaTime;
        Move(Velocity * Time.deltaTime);
    }

    private void Move(Vector2 deltaMovement)
    {
        var wasGrounded = State.IsCollidingBelow;
        State.Reset();

        if (HandleCollisions)
        {
            HandlePlatforms();
            CalculateRayOrigins();

            if (deltaMovement.y < 0 && wasGrounded)
                HandleVerticalSlope(ref deltaMovement);
        }
    }

```

```

    if (Mathf.Abs(deltaMovement.x) > .001f)
        MoveHorizontally(ref deltaMovement);

    MoveVertically(ref deltaMovement);

    CorrectHorizontalPlacement(ref deltaMovement, true);
    CorrectHorizontalPlacement(ref deltaMovement, false);
}

_transform.Translate(deltaMovement, Space.World);

if (Time.deltaTime > 0)
    _velocity = deltaMovement / Time.deltaTime;

_velocity.x = Mathf.Min(_velocity.x, Parameters.MaxVelocity.x);
_velocity.y = Mathf.Min(_velocity.y, Parameters.MaxVelocity.y);

if (State.IsMovingUpSlope)
    _velocity.y = 0;

if (StandingOn != null)
{
    _activeGlobalPlatformPoint = transform.position;
    _activeLocalPlatformPoint =
StandingOn.transform.InverseTransformPoint(transform.position);

    if (_lastStandingOn != StandingOn)
    {
        if (_lastStandingOn != null)
            _lastStandingOn.SendMessage("ControllerExit2D", this,
SendMessageOptions.DontRequireReceiver);

```

```

        StandingOn.SendMessage("ControllerEnter2D", this,
SendMessageOptions.DontRequireReceiver);
        _lastStandingOn = StandingOn;
    }
    else if (StandingOn != null)
        StandingOn.SendMessage("ControllerStay2D", this,
SendMessageOptions.DontRequireReceiver);
    }
    else if (_lastStandingOn != null)
    {
        _lastStandingOn.SendMessage("ControllerExit2D", this,
SendMessageOptions.DontRequireReceiver);
        _lastStandingOn = null;
    }
}

private void HandlePlatforms()
{
    if (StandingOn != null)
    {
        var newGlobalPlatformPoint =
StandingOn.transform.TransformPoint(_activeLocalPlatformPoint);
        var moveDistance = newGlobalPlatformPoint - _activeGlobalPlatformPoint;

        if (moveDistance != Vector3.zero)
            transform.Translate(moveDistance, Space.World);

        PlatformVelocity = (newGlobalPlatformPoint - _activeGlobalPlatformPoint) /
Time.deltaTime;
    }
    else

```

```

        PlatformVelocity = Vector3.zero;

        StandingOn = null;
    }

    private void CorrectHorizontalPlacement(ref Vector2 deltaMovement, bool
isRight)
    {
        var halfWidth = (_boxCollider.size.x * _localScale.x) / 2f;
        var rayOrigin = isRight ? _raycastBottomRight : _raycastBottomLeft;

        if (isRight)
            rayOrigin.x -= (halfWidth - SkinWidth);
        else
            rayOrigin.x += (halfWidth - SkinWidth);

        var rayDirection = isRight ? Vector2.right : -Vector2.right;
        var offset = 0f;

        for (var i = 1; i < TotalHorizontalRays - 1; i++)
        {
            var rayVector = new Vector2(deltaMovement.x + rayOrigin.x,
deltaMovement.y + rayOrigin.y + (i * _verticalDistanceBetweenRays));

            //      Debug.DrawRay(rayVector, rayDirection * halfWidth, isRight ?
Color.cyan : Color.magenta);

            var raycastHit = Physics2D.Raycast(rayVector, rayDirection, halfWidth,
PlatformMask);

            if (!raycastHit)
                continue;
        }
    }

```

```

        offset = isRight ? ((raycastHit.point.x - _transform.position.x) - halfWidth) :
(halfWidth - (_transform.position.x - raycastHit.point.x));
    }

```

```

        deltaMovement.x += offset;
    }

```

```

private void CalculateRayOrigins()
{
    var size = new Vector2(_boxCollider.size.x * Mathf.Abs(_localScale.x),
_boxCollider.size.y * Mathf.Abs(_localScale.y)) / 2;

    var center = new Vector2(_boxCollider.offset.x * _localScale.x,
_boxCollider.offset.y * _localScale.y);

    _raycastTopLeft = _transform.position + new Vector3(center.x - size.x +
SkinWidth, center.y + size.y - SkinWidth);

    _raycastBottomRight = _transform.position + new Vector3(center.x + size.x -
SkinWidth, center.y - size.y + SkinWidth);

    _raycastBottomLeft = _transform.position + new Vector3(center.x - size.x +
SkinWidth, center.y - size.y + SkinWidth);
}

```

```

private void MoveHorizontally(ref Vector2 deltaMovement)
{
    var isGoingRight = deltaMovement.x > 0;
    var rayDistance = Mathf.Abs(deltaMovement.x) + SkinWidth;
    var rayDirection = isGoingRight ? Vector2.right : -Vector2.right;
    var rayOrigin = isGoingRight ? _raycastBottomRight : _raycastBottomLeft;

    for (var i = 0; i < TotalHorizontalRays; i++)
    {
        var rayVector = new Vector2(rayOrigin.x, rayOrigin.y + (i *
_verticalDistanceBetweenRays));
    }
}

```

```

        Debug.DrawRay(rayVector, rayDirection * rayDistance, Color.red);

        var rayCastHit = Physics2D.Raycast(rayVector, rayDirection, rayDistance,
PlatformMask);

        if (!rayCastHit)

            continue;

        if (i == 0 && HandleHorizontalSlope(ref deltaMovement,
Vector2.Angle(rayCastHit.normal, Vector2.up), isGoingRight))

            break;

        deltaMovement.x = rayCastHit.point.x - rayVector.x;
        rayDistance = Mathf.Abs(deltaMovement.x);

        if (isGoingRight)
        {
            deltaMovement.x -= SkinWidth;
            State.IsCollidingRight = true;
        }
        else
        {
            deltaMovement.x += SkinWidth;
            State.IsCollidingLeft = true;
        }

        if (rayDistance < SkinWidth + .0001f)

            break;
    }
}

private void MoveVertically(ref Vector2 deltaMovement)

```

```

{
    var isGoingUp = deltaMovement.y > 0;
    var rayDistance = Mathf.Abs(deltaMovement.y) + SkinWidth;
    var rayDirection = isGoingUp ? Vector2.up : -Vector2.up;
    var rayOrigin = isGoingUp ? _raycastTopLeft : _raycastBottomLeft;

    rayOrigin.x += deltaMovement.x;

    var standingOnDistance = float.MaxValue;
    for (var i = 0; i < TotalVerticalRays; i++)
    {
        var rayVector = new Vector2(rayOrigin.x + (i *
            _horizontalDistanceBetweenRays), rayOrigin.y);
        Debug.DrawRay(rayVector, rayDirection * rayDistance, Color.red);

        var raycastHit = Physics2D.Raycast(rayVector, rayDirection, rayDistance,
PlatformMask);
        if (!raycastHit)
            continue;

        if (!isGoingUp)
        {
            var verticalDistanceToHit = _transform.position.y - raycastHit.point.y;
            if (verticalDistanceToHit < standingOnDistance)
            {
                standingOnDistance = verticalDistanceToHit;
                StandingOn = raycastHit.collider.gameObject;
            }
        }
    }

    deltaMovement.y = raycastHit.point.y - rayVector.y;

```

```

rayDistance = Mathf.Abs(deltaMovement.y);

if (isGoingUp)
{
    deltaMovement.y -= SkinWidth;
    State.IsCollidingAbove = true;
}
else
{
    deltaMovement.y += SkinWidth;
    State.IsCollidingBelow = true;
}

if (!isGoingUp && deltaMovement.y > .0001f)
    State.IsMovingUpSlope = true;

if (rayDistance < SkinWidth + .0001f)
    break;
}
}

private void HandleVerticalSlope(ref Vector2 deltaMovement)
{
    var center = (_raycastBottomLeft.x + _raycastBottomRight.x) / 2;
    var direction = -Vector2.up;

    var slopeDistance = SlopeLimitTangant * (_raycastBottomRight.x - center);
    var slopeRayVector = new Vector2(center, _raycastBottomLeft.y);

    Debug.DrawRay(slopeRayVector, direction * slopeDistance, Color.yellow);

```

```
var raycastHit = Physics2D.Raycast(slopeRayVector, direction, slopeDistance, PlatformMask);
```

```
if (!raycastHit)
```

```
    return;
```

```
// ReSharper disable CompareOfFloatsByEqualityOperator
```

```
var isMovingDownSlope = Mathf.Sign(raycastHit.normal.x) ==  
Mathf.Sign(deltaMovement.x);
```

```
if (!isMovingDownSlope)
```

```
    return;
```

```
var angle = Vector2.Angle(raycastHit.normal, Vector2.up);
```

```
if (Mathf.Abs(angle) < .0001f)
```

```
    return;
```

```
State.IsMovingDownSlope = true;
```

```
State.SlopeAngle = angle;
```

```
deltaMovement.y = raycastHit.point.y - slopeRayVector.y;
```

```
}
```

```
private bool HandleHorizontalSlope(ref Vector2 deltaMovement, float angle, bool  
isGoingRight)
```

```
{
```

```
    if (Mathf.RoundToInt(angle) == 90)
```

```
        return false;
```

```
    if (angle > Parameters.SlopeLimit)
```

```
    {
```

```
        deltaMovement.x = 0;
```

```

        return true;
    }

    if (deltaMovement.y > .07f)
        return true;

    deltaMovement.x += isGoingRight ? -SkinWidth : SkinWidth;
    deltaMovement.y = Mathf.Abs(Mathf.Tan(angle * Mathf.Deg2Rad) *
deltaMovement.x);
    State.IsMovingUpSlope = true;
    State.IsCollidingBelow = true;
    return true;
}

public void OnTriggerEnter2D(Collider2D other)
{
    var parameters =
other.gameObject.GetComponent<ControllerPhysicsVolume2D>();
    if (parameters == null)
        return;
    _overrideParameters = parameters.Parameters;
}

public void OnTriggerExit2D(Collider2D other)
{
    var parameters =
other.gameObject.GetComponent<ControllerPhysicsVolume2D>();
    if (parameters == null)
        return;
    _overrideParameters = null;
}
}

```

A6. Checkpoint

```
using UnityEngine;

using System.Collections;
using System.Collections.Generic;

public class Checkpoint : MonoBehaviour
{
    private List<IPlayerRespawnListener> _listeners;

    public void Awake()
    {
        _listeners = new List<IPlayerRespawnListener>();
    }

    public void PlayerHitCheckpoint()
    {
        StartCoroutine(PlayerHitCheckpointCo(LevelManager.Instance.CurrentTimeBonus));
    }

    private IEnumerator PlayerHitCheckpointCo(int bonus)
    {
        FloatingText.Show("Checkpoint!", "CheckpointText", new
        CenteredTextPositioner(.30f));

        yield return new WaitForSeconds(.30f);

        FloatingText.Show(string.Format("+{0} time bonus!", bonus),
        "CheckpointText", new CenteredTextPositioner(.30f));
    }

    public void PlayerLeftCheckpoint()
    {
    }
}
```

```

public void SpawnPlayer (Player player)
{
    player.RespawnAt(transform);

    foreach (var listener in _listeners)
        listener.OnPlayerRespawnInThisCheckpoint(this, player);
}

public void AssignObjectToCheckpoint(IPlayerRespawnListener listener )
{
    _listeners.Add(listener);
}
}

```

A7. Controller Parameter 2D

```
using System.Collections;
```

```
using UnityEngine;
```

```
using System;
```

```
[Serializable]
```

```
public class ControllerParameters2D
```

```
{
```

```
    public enum JumpBehavior
```

```
    {
```

```
        CanJumpOnGround,
```

```
        CanJumpAnywhere,
```

```
        CantJump
```

```
    }
```

```
    public Vector2 MaxVelocity = new Vector2(float.MaxValue, float.MaxValue);
```

```
    [Range(0, 90)]
```

```
    public float SlopeLimit = 30;
```

```
    public float Gravity = -25f;
```

```
    public JumpBehavior JumpRestrictions;
```

```
    public float JumpFrequency = .25f;
```

```
    public float JumpMagnitude = 12;
```

```
}
```

A8. Controller Physics Volume 2D

```
using System.Collections;
```

```
using UnityEngine;
```

```
public class ControllerPhysicsVolume2D : MonoBehaviour
{
    public ControllerParameters2D Parameters;
}
```

A9. Controller State 2D

```
using System.Collections;
```

```
using UnityEngine;
```

```
public class ControllerState2D
{
    public bool IsCollidingRight { get; set; }
    public bool IsCollidingLeft { get; set; }
    public bool IsCollidingAbove { get; set; }
    public bool IsCollidingBelow { get; set; }
    public bool IsMovingDownSlope { get; set; }
    public bool IsMovingUpSlope { get; set; }
    public bool IsGrounded { get { return IsCollidingBelow; } }
    public float SlopeAngle { get; set; }

    public bool HasCollisions { get { return IsCollidingRight || IsCollidingLeft ||
IsCollidingAbove || IsCollidingBelow; } }

    public void Reset()
    {
        IsMovingUpSlope =
```

```

        IsMovingDownSlope =
        IsCollidingLeft =
        IsCollidingRight =
        IsCollidingAbove =
        IsCollidingBelow = false;

        SlopeAngle = 0;
    }

    public override string ToString()
    {
        return string.Format("(controller: r:{0} 1:{1} a:{2} b:{3} down-slope: {5}
angle: {6})",
            IsCollidingRight,
            IsCollidingLeft,
            IsCollidingAbove,
            IsCollidingBelow,
            IsMovingDownSlope,
            IsMovingUpSlope,
            SlopeAngle);
    }
}

```

A10. Finish Level

using UnityEngine;

```
public class FinishLevel : MonoBehaviour
{
    public string LevelName;

    public void OnTriggerEnter2D(Collider2D other)
    {
        if (other.GetComponent<Player>() == null)
            return;

        LevelManager.Instance.GotoNextLevel(LevelName);
    }
}
```

A11. Floating Text

using UnityEngine;

```
public class FloatingText : MonoBehaviour
{
    private static readonly GUISkin Skin = Resources.Load<GUISkin>("GameSkin");

    public static FloatingText Show(string text, string style, IFloatingTextPositioner
positioner)
    {
        var go = new GameObject("Floating Text");
        var floatingText = go.AddComponent<FloatingText>();
        floatingText.Style = Skin.GetStyle(style);
        floatingText._positioner = positioner;
    }
}
```

```

        floatingText._content = new GUIContent(text);
        return floatingText;
    }

    private GUIContent _content;
    private IFloatingTextPositioner _positioner;

    public string Text { get { return _content.text; } set { _content.text = value; } }
    public GUIStyle Style { get; set; }

    public void OnGUI()
    {
        var position = new Vector2();
        var contentSize = Style.CalcSize(_content);
        if (!_positioner.GetPosition(ref position, _content, Style.CalcSize(_content)))
        {
            Destroy(gameObject);
            return;
        }

        GUI.Label(new Rect(position.x, position.y, contentSize.x, contentSize.y),
        _content, Style);
    }
}

```

A12. Follow Object

using UnityEngine;

```
public class FollowObject : MonoBehaviour
{
    public Vector2 Offset;
    public Transform Following;

    public void Update()
    {
        transform.position = Following.transform.position + (Vector3) Offset;
    }
}
```

A13. Follow Path

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FollowPath : MonoBehaviour

```
{
    //This is where we create the two types of Movements that we want.

    //Fun fact, this is also EXACTLY the code that has the options show up in the
    Unity editor itself.

    //If you were to make this private, the options 'lerp' and 'movetowards' would not
    show up in Unity's Inspector.

    public enum FollowType
    {
        MoveTowards,
        Lerp
    }
}
```

//This creates an INSTANCE of FollowType. Instances are a BIG deal in C#, so if you're new,

//Understanding them more in depth is paramount to moving past the beginner stages.

```
public FollowType Type = FollowType.MoveTowards;
```

//This answers the question 'What Path do I need to follow, mr/mrs game developer person?'

//This is where we will click and drag out path to in the Unity Inspector.

```
public PathDefinition Path;
```

//This is a PUBLIC variable, therefore it shows up in the Unity Inspector.

```
public float speed = 1; // How fast the platform will move.
```

//If we are within a .1f radius of the point, we cool. Let's move to the next one.

//This way, if for some reason our platform doesn't hit the target exactly on the dot, it won't break the whole game.

//An example might be, if we have a lot of physics in our game, and our platform somehow gets hit out of alignment unexpectedly.

```
public float MaxDistanceToGoal = .1f;
```

//Notice that this is a PRIVATE variable, so it will not show up in the Unity Editor.

```
private IEnumerator<Transform> _currentPoint; //_currentPoint is of TYPE  
IEnumerator
```

```
public void Start()
```

```
{
```

//This if statement says 'If There isn't a Path, Yell at the developer (or player) that something went wrong

//So that they know Why something is going wrong, rather than stuff just not working and nobody understands why

//and everyone cries.

```
if (Path == null)
```

```
{
```

```
    Debug.LogError("Path Cannot be null", gameObject);
```

```
    return;
```

```
}
```

```

//Remember, _currentPoint is of TYPE 'IEnumerator'. GetEnumerator()
returns an Enumerator.

//Let me take a sec to make sure you got this.

//In the while (true) loop, it says 'yield return Points[index];'

//I emphasized over there that this stops the code and waits, which is true. BUT!

//It also returns 'Points[index]'.

//This is where this is important.

//We are taking the result of that loop, and setting it to _currentPoint.


//currentPoint = Path.GetEnumerator() is essentially doing the same thing as
//_currentPoint = Points[index].

//This is HOW we know for sure that _currentPoint is actually what it says... The
current point.

//Because if you look at the function GetEnumerator, figuring out what our
current point is, and where we go

//from there is its entire job!

//The Path part tells the GetEnumerator function how many points there are.

//GetEnumerator uses that number that the path tells it,

//and gets an enumerator(a point), and determines if the object should be going
forwards or backwards from that point.

_currentPoint = Path.GetEnumerator();

//Remember in that while(true) loop, where we had a line of code that said STOP
UNTIL WE GET TO THE NEXT POINT?

//This is the 'Alright, we're here, now move along you slackers.'

_currentPoint.MoveNext();

//Just another 'if things aint the way they're supposed to be, then run awaaaayyyy'
if (_currentPoint.Current == null)

    return;

//Sets our position to the Start (Remember, currentPoint = Points[index]. Our
index starts at 0, so we are at the start.

//If we don't have this, we can't do any of the fancy stuff down below.

transform.position = _currentPoint.Current.position;

```

```

    }

    public void Update()
    {
        if (_currentPoint == null || _currentPoint.Current == null) //if object has no path
        or no points:

            return; //early exit out of update: There is
        nothing to do here.

        //For Clarity, Here are 2 different explanations of the following code:

        //If Type == Move Towards

        //(transform.position) Move Towards the object

        //Vector3.MoveTowards (1st paramater == CURRENT position, 2nd paramater
        == TARGET position, 3rd paramater = MAX distance)

        //deltaTime helps make animation smooth

        //_currentPoint.Current.position references the current point that we are
        LOOKING at

        //This if statement basically says:

        //if (The user picks 'MoveTowards' from the options in the Unity interface

        //Run this code

        //To emphasize this point, That one line of code following the if statement
        Moves an object from a position,

        //Towards another position, at a certain distance, in a certain way (The
        'MoveTowardsy' Way.) Pretty cool.

        if (Type == FollowType.MoveTowards)
        {
            transform.position = Vector3.MoveTowards(transform.position,
            _currentPoint.Current.position, Time.deltaTime * speed);
        }

        //For Clarity, Here are 2 different explanations of the following code:

        //If Type == Lerp

        //(transform.position) Move Towards the object

        //Everything is the same, but is instead in a linear interpolation method rather
        than a move towards method.

        //This if statement basically says:

```

```

//if (The user picks 'Lerp' from the options in the Unity interface
//Run this code
else if (Type == FollowType.Lerp)
{
    transform.position = Vector3.Lerp(transform.position,
    _currentPoint.Current.position, Time.deltaTime * speed);
}

//Also, if you're confused about Vector 3:

//Vector 3 = 3 coordinates. X, Y, Z. He could have probably used Vector 2 and
have been fine, but there ya go.

//Although I've seen a few other people do it this way in 2D, so maybe there's
something to it.

//There's more to it, but just understand that fundamental element and you should
be okay for now.

//Yeah, this is where he lost me. But if you're new (like me), and grasp all the
other stuff, i'd call this a win.

var distanceSquared = (transform.position -
_currentPoint.Current.position).sqrMagnitude;

if (distanceSquared < MaxDistanceToGoal * MaxDistanceToGoal)
    _currentPoint.MoveNext(); //runs the while(true) block again.
}
}

```

A14. From World Point Text Positioner

using UnityEngine;

```
public class FromWorldPointTextPositioner : IFloatingTextPositioner
{
    private readonly Camera _camera;
    private readonly Vector3 _worldPosition;
    private readonly float _speed;
    private float _timeToLive;
    private float _yOffset;

    public FromWorldPointTextPositioner(Camera camera, Vector3 worldPosition,
float timeToLive, float speed)
    {
        _camera = camera;
        _worldPosition = worldPosition;
        _timeToLive = timeToLive;
        _speed = speed;
    }

    public bool GetPosition(ref Vector2 position, GUIContent content, Vector2 size)
    {
        if ((_timeToLive -= Time.deltaTime) <= 0)
            return false;

        var screenPosition = _camera.WorldToScreenPoint(_worldPosition);
        position.x = screenPosition.x - (size.x / 2);
        position.y = Screen.height - screenPosition.y - _yOffset;

        _yOffset += Time.deltaTime * _speed;
        return true;
    }
}
```

```

    }
}

A15. Game Hud
using UnityEngine;

public class GameHud : MonoBehaviour
{
    public GUISkin Skin;

    public void OnGUI()
    {
        GUI.skin = Skin;

        GUILayout.BeginArea(new Rect(0, 0, Screen.width, Screen.height));
        {
            GUILayout.BeginVertical(Skin.GetStyle("GameHud"));
            {
                GUILayout.Label(string.Format("Points: {0}",
                GameManager.Instance.Points), Skin.GetStyle("PointsText"));

                var time = LevelManager.Instance.RunningTime;
                GUILayout.Label(string.Format(
                    "{0:00}:{1:00} with {2} bonus",
                    time.Minutes + (time.Hours * 60),
                    time.Seconds,
                    LevelManager.Instance.CurrentTimeBonus),
                Skin.GetStyle("TimeText"));
            }
            GUILayout.EndVertical();
        }
        GUILayout.EndArea();
    }
}

```

```
    }  
}
```

A16. Game Manager

using UnityEngine;

```
public class GameManager
```

```
{
```

```
    private static GameManager _instance;
```

```
    public static GameManager Instance { get { return _instance ?? (_instance = new  
GameManager()); } }
```

```
    public int Points { get; private set; }
```

```
    private GameManager()
```

```
    {
```

```
    }
```

```
    public void Reset()
```

```
    {
```

```
        Points = 0;
```

```
    }
```

```
    public void ResetPoints(int points)
```

```
    {
```

```
        Points = points;
```

```
    }
```

```
    public void AddPoints(int pointsToAdd)
```

```
    {
```

```
        Points += pointsToAdd;
```

```
    }
```

```
}
```

A17. Give Damage to Player

using UnityEngine;

```
public class GiveDamageToPlayer : MonoBehaviour
{
    public int DamageToGive = 10;

    private Vector2
        _lastPosition,
        _velocity;
    public void LateUpdate()
    {
        _velocity = (_lastPosition - (Vector2) transform.position) / Time.deltaTime;
        _lastPosition = transform.position;
    }
    public void OnTriggerEnter2D(Collider2D other)
    {
        var player = other.GetComponent<Player>();
        if (player == null)
            return;
        player.TakeDamage(DamageToGive, gameObject);
        var controller = player.GetComponent<CharacterController2D>();
        var totalVelocity = controller.Velocity + _velocity;

        controller.SetForce(new Vector2(
            -1 * Mathf.Sign(totalVelocity.x) * Mathf.Clamp(Mathf.Abs(totalVelocity.x) *
        6, 10, 40),
            -1 * Mathf.Sign(totalVelocity.y) * Mathf.Clamp(Mathf.Abs(totalVelocity.y) *
        6, 5, 30)));
    }
}
```

A18. Give Health

using UnityEngine;

```
public class GiveHealth : MonoBehaviour, IPlayerRespawnListener
{
    public GameObject Effect;
    public int HealthToGive;

    public void OnTriggerEnter2D(Collider2D other)
    {
        var player = other.GetComponent<Player>();
        if (player == null)
            return;

        player.GiveHealth(HealthToGive, gameObject);
        Instantiate(Effect, transform.position, transform.rotation);

        gameObject.SetActive(false);
    }

    public void OnPlayerRespawnInThisCheckpoint(Checkpoint checkpoint, Player
player)
    {
        gameObject.SetActive(true);
    }
}
```

A19. HealthBar

using UnityEngine;

```
public class HealthBar : MonoBehaviour
{
    public Player Player;
    public Transform ForegroundSprite;
    public SpriteRenderer ForegroundRenderer;
    public Color MaxHealthColor = new Color(255 / 255f, 63 / 255f, 63 / 255f);
    public Color MinHealthColor = new Color(64 / 255f, 137 / 255f, 255 / 255f);

    public void Update()
    {
        var healthPercent = Player.Health / (float)Player.MaxHealth;

        ForegroundSprite.localScale = new Vector3(healthPercent, 1, 1);
        ForegroundRenderer.color = Color.Lerp(MaxHealthColor, MinHealthColor,
healthPercent);

    }
}
```

A20. IFloating Text Positioner

using UnityEngine;

```
public interface IFloatingTextPositioner
{
    bool GetPosition(ref Vector2 position, GUIContent content, Vector2 size);
}
```

A21. InstaKill

```
using UnityEngine;

public class InstaKill : MonoBehaviour
{
    public void OnTriggerEnter2D(Collider2D other)
    {
        var player = other.GetComponent<Player>();
        if (player == null)
            return;

        LevelManager.Instance.KillPlayer();
    }
}
```

A22. IPlayerRespawnListener

```
public interface IPlayerRespawnListener
{
    void OnPlayerRespawnInThisCheckpoint(Checkpoint checkpoint, Player player);
}
```

A23. ITakeDamage

```
using UnityEngine;

public interface ITakeDamage
{
    void TakeDamage(int damage, GameObject instigator);
}
```

A24. Jump Platform

```
using System.Collections;
using UnityEngine;

public class JumpPlatform : MonoBehaviour
{
    public float JumpMagnitude = 20;
    public AudioClip JumpSound;

    public void ControllerEnter2D(CharacterController2D controller)
    {
        if (JumpSound != null)
            AudioSource.PlayClipAtPoint(JumpSound, transform.position);

        controller.SetVerticalForce(JumpMagnitude);
    }
}
```

A25. Level Manager

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

public class LevelManager : MonoBehaviour
{
    public static LevelManager Instance { get; private set; }

    public Player Player { get; private set; }
```

```

public CameraController Camera { get; private set; }

public TimeSpan RunningTime { get { return DateTime.UtcNow - _started; } }

public int CurrentTimeBonus
{
    get
    {
        var secondDifference = (int)(BonusCutoffSeconds -
RunningTime.TotalSeconds);
        return Mathf.Max(0, secondDifference) * BonusSecondMultiplier;
    }
}

private List<Checkpoint> _checkpoints;
private int _currentCheckpointIndex;
private DateTime _started;
private int _savedPoints;

public Checkpoint DebugSpawn;
public int BonusCutoffSeconds;
public int BonusSecondMultiplier;

public void Awake()
{
    _savedPoints = GameManager.Instance.Points;
    Instance = this;
}

public void Start()
{

```

```

        _checkpoints = FindObjectsOfType<Checkpoint>().OrderBy(t =>
t.transform.position.x).ToList();

        _currentCheckpointIndex = _checkpoints.Count > 0 ? 0 : -1;

Player = FindObjectOfType<Player>();
Camera = FindObjectOfType<CameraController>();

_started = DateTime.UtcNow;

var listeners =
FindObjectsOfType<MonoBehaviour>().OfType<IPlayerRespawnListener>();
foreach (var listener in listeners)
{
    for (var i = _checkpoints.Count - 1; i >= 0; i--)
    {
        var distance = ((MonoBehaviour)listener).transform.position.x -
_checkpoints[i].transform.position.x;
        if (distance < 0)
            continue;

        _checkpoints[i].AssignObjectToCheckpoint(listener);
        break;
    }
}

#if UNITY_EDITOR
    if (DebugSpawn != null)
        DebugSpawn.SpawnPlayer(Player);
    else if (_currentCheckpointIndex != -1)
        _checkpoints[_currentCheckpointIndex].SpawnPlayer(Player);
#else

```

```

        if (_currentCheckpointIndex != -1)
            _checkpoints[_currentCheckpointIndex].SpawnPlayer(Player);
    #endif
    }

    public void Update()
    {
        var isAtLastCheckpoint = _currentCheckpointIndex + 1 >= _checkpoints.Count;
        if (isAtLastCheckpoint)
            return;

        var distanceToNextCheckpoint = _checkpoints[_currentCheckpointIndex + 1].transform.position.x - Player.transform.position.x;
        if (distanceToNextCheckpoint >= 0)
            return;

        _checkpoints[_currentCheckpointIndex].PlayerLeftCheckpoint();
        _currentCheckpointIndex++;
        _checkpoints[_currentCheckpointIndex].PlayerHitCheckpoint();

        GameManager.Instance.AddPoints(CurrentTimeBonus);
        _savedPoints = GameManager.Instance.Points;
        _started = DateTime.UtcNow;
    }

    public void GotoNextLevel(string levelName)
    {
        StartCoroutine(GotoNextLevelCo(levelName));
    }

    private IEnumerator GotoNextLevelCo(string levelName)

```

```

{
    Player.FinishLevel();
    GameManager.Instance.AddPoints(CurrentTimeBonus);

    FloatingText.Show("Level Complete!", "CheckpointText", new
CenteredTextPositioner(.2f));

    yield return new WaitForSeconds(1);

    FloatingText.Show(string.Format("{0} points!",
GameManager.Instance.Points), "CheckpointText", new
CenteredTextPositioner(.1f));

    yield return new WaitForSeconds(5f);

    if (string.IsNullOrEmpty(levelName))
        Application.LoadLevel("StartScreen");
    else
        Application.LoadLevel(levelName);
}

public void KillPlayer()
{
    StartCoroutine(KillPlayerCo());
}

private IEnumerator KillPlayerCo()
{
    Player.Kill();
    Camera.IsFollowing = false;
    yield return new WaitForSeconds(2f);
}

```

```

        Camera.IsFollowing = true;

        if (_currentCheckpointIndex != -1)
            _checkpoints[_currentCheckpointIndex].SpawnPlayer(Player);

        _started = DateTime.UtcNow;
        GameManager.Instance.ResetPoints(_savedPoints);
    }
}

```

A26. Path Definition

```

using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

public class PathDefinition : MonoBehaviour
{
    public Transform[] Points;

    public IEnumerator<Transform> GetPathEnumerator()
    {
        if (Points == null || Points.Length < 1)
            yield break;

        var index = 0;
        var direction = 1;
        while (true)
        {

```

```

        yield return Points[index];

    if (Points.Length == 1)
        continue;

    if (index <= 0)
        direction = 1;
    else if (index >= Points.Length - 1)
        direction = -1;

    index = index + direction;
}
}

public void OnDrawGizmos()
{
    if (Points == null || Points.Length < 2)
        return;

    var points = Points.Where(t => t != null).ToList();
    if (points.Count < 2)
        return;

    for (var i = 1; i < points.Count; i++)
    {
        Gizmos.DrawLine(points[i - 1].position, points[i].position);
    }
}
}

```

A27. Pathed Projectile

using UnityEngine;

```
public class PathedProjectile : MonoBehaviour, ITakeDamage
{
    private Transform _destination;
    private float _speed;

    public GameObject DestroyEffect;
    public int PointsToGivePlayer;
    public AudioClip DestroySound;

    public void Initialize(Transform destination, float speed)
    {
        _destination = destination;
        _speed = speed;
    }

    public void Update()
    {
        transform.position = Vector3.MoveTowards(transform.position,
        _destination.position, Time.deltaTime * _speed);

        var distanceSquared = (_destination.transform.position -
        transform.position).sqrMagnitude;
        if (distanceSquared > .01f * .01f)
            return;

        if (DestroyEffect != null)
            Instantiate(DestroyEffect, transform.position, transform.rotation);
    }
}
```

```

        if (DestroySound != null)
            AudioSource.PlayClipAtPoint(DestroySound, transform.position);

        Destroy(gameObject);

    }

    public void TakeDamage(int damage, GameObject instigator)
    {
        if (DestroyEffect != null)
            Instantiate(DestroyEffect, transform.position, transform.rotation);

        Destroy(gameObject);

        var projectile = instigator.GetComponent<Projectile>();
        if (projectile != null && projectile.Owner.GetComponent<Player>() != null &&
            PointsToGivePlayer != 0)
        {
            GameManager.Instance.AddPoints(PointsToGivePlayer);

            FloatingText.Show(string.Format("+{0}!", PointsToGivePlayer),
                "PointStarText", new FromWorldPointTextPositioner(Camera.main,
                    transform.position, 1.5f, 50));
        }
    }
}

```

A28. Pathed Projectile Spawner

using UnityEngine;

```
public class PathedProjectileSpawner : MonoBehaviour
{
    public Transform Destination;
    public PathedProjectile Projectile;

    public GameObject SpawnEffect;
    public float Speed;
    public float FireRate;
    public AudioClip SpawnProjectileSound;
    public Animator Animator;

    private float _nextShotInSeconds;

    public void Start()
    {
        _nextShotInSeconds = FireRate;
    }

    public void Update()
    {
        if ((_nextShotInSeconds -= Time.deltaTime) > 0)
            return;

        _nextShotInSeconds = FireRate;

        var projectile = (PathedProjectile)Instantiate(Projectile, transform.position,
transform.rotation);
        projectile.Initialize(Destination, Speed);
    }
}
```

```

        if (SpawnEffect != null)
            Instantiate(SpawnEffect, transform.position, transform.rotation);

        if (SpawnProjectileSound != null)
            AudioSource.PlayClipAtPoint(SpawnProjectileSound, transform.position);

        if (Animator != null)
            Animator.SetTrigger("Fire");
    }

    public void OnDrawGizmos()
    {
        if (Destination == null)
            return;

        Gizmos.color = Color.red;
        Gizmos.DrawLine(transform.position, Destination.position);
    }
}

```

A29. Player

```

using System.Collections;
using UnityEngine;

public class Player : MonoBehaviour, ITakeDamage
{
    private bool _isFacingRight;
    private CharacterController2D _controller;
    private float _normalizedHorizontalSpeed;

```

```

public float MaxSpeed = 8;
public float SpeedAccelerationOnGround = 10f;
public float SpeedAccelerationInAir = 5f;
public int MaxHealth = 100;
public GameObject OuchEffect;
public Projectile Projectile;
public float FireRate;
public Transform ProjectileFireLocation;
public GameObject FireProjectileEffect;
public AudioClip PlayerHitSound;
public AudioClip PlayerShootSound;
public AudioClip PlayerHealthSound;
public Animator Animator;

public int Health { get; private set; }
public bool IsDead { get; private set; }

public float _canFireIn;

public void Awake()
{
    _controller = GetComponent<CharacterController2D>();
    _isFacingRight = transform.localScale.x > 0;
    Health = MaxHealth;
}

public void Update()
{
    _canFireIn -= Time.deltaTime;

```

```

    if (!IsDead)
        HandleInput();

    var movementFactor = _controller.State.IsGrounded ?
SpeedAccelerationOnGround : SpeedAccelerationInAir;

    if (IsDead)
        _controller.SetHorizontalForce(0);
    else
        _controller.SetHorizontalForce(Mathf.Lerp(_controller.Velocity.x,
_normalizedHorizontalSpeed * MaxSpeed, Time.deltaTime * movementFactor));

    Animator.SetBool("IsGrounded", _controller.State.IsGrounded);
    Animator.SetBool("IsDead", IsDead);
    Animator.SetFloat("Speed", Mathf.Abs(_controller.Velocity.x) / MaxSpeed);
}

public void FinishLevel()
{
    enabled = false;
    _controller.enabled = false;
    this.gameObject.GetComponent<Collider2D>().enabled = false;
}

public void Kill()
{
    _controller.HandleCollisions = false;
    this.gameObject.GetComponent<Collider2D>().enabled = false;
    IsDead = true;
    Health = 0;
}

```

```

        _controller.SetForce(new Vector2(0, 20));
    }

    public void RespawnAt(Transform spawnPoint)
    {
        if (!_isFacingRight)
            Flip();

        IsDead = false;
        this.gameObject.GetComponent<Collider2D>().enabled = true;
        _controller.HandleCollisions = true;
        Health = MaxHealth;

        transform.position = spawnPoint.position;
    }

    public void TakeDamage(int damage, GameObject instagator)
    {
        AudioSource.PlayClipAtPoint(PlayerHitSound, transform.position);

        FloatingText.Show(string.Format("-{0}", damage), "PlayerTakeDamageText",
new FromWorldPointTextPositioner(Camera.main, transform.position, 2f, 60f));

        Instantiate(OuchEffect, transform.position, transform.rotation);
        Health -= damage;

        if (Health <= 0)
            LevelManager.Instance.KillPlayer();
    }

    public void GiveHealth(int health, GameObject instagator)

```

```

{

    AudioSource.PlayClipAtPoint(PlayerHealthSound, transform.position);

    FloatingText.Show(string.Format("+{0}", health), "PlayerGotHealthText", new
    FromWorldPointTextPositioner(Camera.main, transform.position, 2f, 60f));

    Health = Mathf.Min(Health + health, MaxHealth);
}

private void HandleInput()
{
    if (Input.GetKey(KeyCode.D))
    {
        _normalizedHorizontalSpeed = 1;
        if (!_isFacingRight)
            Flip();
    }
    else if (Input.GetKey(KeyCode.A))
    {
        _normalizedHorizontalSpeed = -1;
        if (_isFacingRight)
            Flip();
    }
    else
    {
        _normalizedHorizontalSpeed = 0;
    }

    if (_controller.CanJump && Input.GetKeyDown(KeyCode.Space))
    {
        _controller.Jump();
    }
}

```

```

        if (Input.GetMouseButtonDown(0))
            FireProjectile();
    }

    private void FireProjectile()
    {
        if (_canFireIn > 0)
            return;

        if (FireProjectileEffect != null)
        {
            var effect = (GameObject)Instantiate(FireProjectileEffect,
            ProjectileFireLocation.position, ProjectileFireLocation.rotation);
            effect.transform.parent = transform;
        }

        var direction = _isFacingRight ? Vector2.right : -Vector2.right;
        var projectile = (Projectile)Instantiate(Projectile, ProjectileFireLocation.position,
        ProjectileFireLocation.rotation);

        projectile.Initialize(gameObject, direction, _controller.Velocity);
        _canFireIn = FireRate;
        AudioSource.PlayClipAtPoint(PlayerShootSound, transform.position);
        Animator.SetTrigger("Fire");
    }

    private void Flip()
    {
        transform.localScale = new Vector3(-transform.localScale.x,
        transform.localScale.y, transform.localScale.z);
        _isFacingRight = transform.localScale.x > 0;
    }
}

```

A30. Player Bounds

using UnityEngine;

```
public class PlayerBounds : MonoBehaviour
{
    public enum BoundsBehavior
    {
        Nothing,
        Constrain,
        Kill
    }

    public BoxCollider2D Bounds;
    public BoundsBehavior Above;
    public BoundsBehavior Below;
    public BoundsBehavior Left;
    public BoundsBehavior Right;

    private Player _player;
    private BoxCollider2D _boxCollider;

    public void Start()
    {
        _player = GetComponent<Player>();
        _boxCollider = GetComponent<BoxCollider2D>();
    }

    public void Update()
    {
        if (_player.IsDead)
            return;
    }
}
```

```

var colliderSize = new Vector2(
    _boxCollider.size.x * Mathf.Abs(transform.localScale.x),
    _boxCollider.size.y * Mathf.Abs(transform.localScale.y)) / 2;

if (Above != BoundsBehavior.Nothing && transform.position.y + colliderSize.y
> Bounds.bounds.max.y)

    ApplyBoundsBehavior(Above, new Vector2(transform.position.x,
Bounds.bounds.max.y - colliderSize.y));

if (Below != BoundsBehavior.Nothing && transform.position.y - colliderSize.y
< Bounds.bounds.min.y)

    ApplyBoundsBehavior(Below, new Vector2(transform.position.x,
Bounds.bounds.min.y + colliderSize.y));

if (Right != BoundsBehavior.Nothing && transform.position.x + colliderSize.x
> Bounds.bounds.max.x)

    ApplyBoundsBehavior(Right, new Vector2(Bounds.bounds.max.x -
colliderSize.x, transform.position.y));

if (Left != BoundsBehavior.Nothing && transform.position.x - colliderSize.x <
Bounds.bounds.min.x)

    ApplyBoundsBehavior(Left, new Vector2(Bounds.bounds.min.x +
colliderSize.x, transform.position.y));
}

private void ApplyBoundsBehavior(BoundsBehavior behavior, Vector2
constrainedPosition)
{
    if (behavior == BoundsBehavior.Kill)
    {
        LevelManager.Instance.KillPlayer();
        return;
    }

    transform.position = constrainedPosition;
}
}

```

A31. Point Star

using UnityEngine;

```
public class PointStar : MonoBehaviour, IPlayerRespawnListener
{
    public GameObject Effect;
    public int PointsToAdd = 10;
    public AudioClip HitStarSound;
    public Animator Animator;

    private bool _isCollected;

    public void OnTriggerEnter2D(Collider2D other)
    {
        if (_isCollected)
            return;

        if (other.GetComponent<Player>() == null)
            return;

        if (HitStarSound != null)
            AudioSource.PlayClipAtPoint(HitStarSound, transform.position);

        GameManager.Instance.AddPoints(PointsToAdd);
        Instantiate(Effect, transform.position, transform.rotation);
        FloatingText.Show(string.Format("+{0}!", PointsToAdd), "PointStarText", new
        FromWorldPointTextPositioner(Camera.main, transform.position, 1.5f, 50));

        _isCollected = true;

        gameObject.SetActive(false);
    }
}
```

```

    }

    public void OnPlayerRespawnInThisCheckpoint(Checkpoint checkpoint, Player
player)
    {
        _isCollected = false;
        gameObject.SetActive(true);
    }
}

```

A32. Projectile

using UnityEngine;

```

public abstract class Projectile : MonoBehaviour
{
    public float Speed;
    public LayerMask CollisionMask;

    public GameObject Owner { get; private set; }
    public Vector2 Direction { get; private set; }
    public Vector2 InitialVelocity { get; private set; }

    public void Initialize(GameObject owner, Vector2 direction, Vector2
initialVelocity)
    {
        transform.right = direction;

        Owner = owner;
        Direction = direction;
        InitialVelocity = initialVelocity;
    }
}

```

```

        OnInitialized();
    }

    protected virtual void OnInitialized()
    {
    }

    public virtual void OnTriggerEnter2D(Collider2D other)
    {

        if ((CollisionMask.value & (1 << other.gameObject.layer)) == 0)
        {
            OnNotCollideWith(other);
            return;
        }

        var isOwner = other.gameObject == Owner;
        if (isOwner)
        {
            OnColliderOwner();
            return;
        }

        var takeDamage = (ITakeDamage)other.GetComponent(typeof(ITakeDamage));
        if (takeDamage != null)
        {
            OnCollideTakeDamage(other, takeDamage);
            return;
        }
    }

```

```

        OnCollideOther(other);
    }

    protected virtual void OnNotCollideWith(Collider2D other)
    {
    }

    protected virtual void OnColliderOwner()
    {
    }

    protected virtual void OnCollideTakeDamage(Collider2D other, ITakeDamage
takeDamage)
    {
    }

    protected virtual void OnCollideOther(Collider2D other)
    {
    }
}

```

A33. Simple Enemy AI

```
using UnityEngine;
```

```
public class SimpleEnemyAi : MonoBehaviour, ITakeDamage,
IPlayerRespawnListener
```

```

{
    public float Speed;
    public float FireRate = 1;
    public Projectile Projectile;
    public GameObject DestroyedEffect;
}

```

```

public int PointsToGivePlayer;

public AudioClip ShootSound;


private CharacterController2D _controller;
private Vector2 _direction;
private Vector2 _startPosition;
private float _canFireIn;


public void Start()
{
    _controller = GetComponent<CharacterController2D>();
    _direction = new Vector2(-1, 0);
    _startPosition = transform.position;
}

public void Update()
{
    _controller.SetHorizontalForce(_direction.x * Speed);

    if ((_direction.x < 0 && _controller.State.IsCollidingLeft) || (_direction.x > 0
&& _controller.State.IsCollidingRight))
    {
        _direction = -_direction;

        transform.localScale = new Vector3(-transform.localScale.x,
transform.localScale.y, transform.localScale.z);
    }

    if ((_canFireIn -= Time.deltaTime) > 0)
        return;


    var raycast = Physics2D.Raycast(transform.position, _direction, 10, 1 <<
LayerMask.NameToLayer("Player"));

    if (!raycast)
        return;

```

```

        var projectile = (Projectile)Instantiate(Projectile, transform.position,
transform.rotation);

        projectile.Initialize(gameObject, _direction, _controller.Velocity);
        _canFireIn = FireRate;
        if (ShootSound != null)
            AudioSource.PlayClipAtPoint(ShootSound, transform.position);
    }

    public void TakeDamage(int damage, GameObject instigator)
    {
        if (PointsToGivePlayer != 0)
        {
            var projectile = instigator.GetComponent<Projectile>();
            if (projectile != null && projectile.Owner.GetComponent<Player>() != null)
            {
                GameManager.Instance.AddPoints(PointsToGivePlayer);

                FloatingText.Show(string.Format("+{0}!", PointsToGivePlayer),
"PointStarText", new FromWorldPointTextPositioner(Camera.main,
transform.position, 1.5f, 50));
            }
        }

        Instantiate(DestroyedEffect, transform.position, transform.rotation);
        gameObject.SetActive(false);
    }

    public void OnPlayerRespawnInThisCheckpoint(Checkpoint checkpoint, Player
player)
    {
        _direction = new Vector2(-1, 0);
        transform.localScale = new Vector3(1, 1, 1);
        transform.position = _startPosition;
        gameObject.SetActive(true);
    }
}

```

A34. Simple Projectile

using UnityEngine;

```
public class SimpleProjectile : Projectile, ITakeDamage
{
    public int Damage;
    public GameObject DestroyedEffect;
    public int PointsToGiveToPlayer;
    public float TimeToLive;
    public AudioClip DestroySound;

    public void Update()
    {
        if ((TimeToLive -= Time.deltaTime) <= 0)
        {
            DestroyProjectile();
            return;
        }

        transform.Translate(Direction * ((Mathf.Abs(InitialVelocity.x) + Speed) *
Time.deltaTime), Space.World);
    }

    public void TakeDamage(int damage, GameObject instigator)
    {
        if (PointsToGiveToPlayer != 0)
        {
            var projectile = instigator.GetComponent<Projectile>();
            if (projectile != null && projectile.Owner.GetComponent<Player>() != null)
            {
                GameManager.Instance.AddPoints(PointsToGiveToPlayer);
            }
        }
    }
}
```

```

        FloatingText.Show(string.Format("+{0}!", PointsToGiveToPlayer),
"PointStarText", new FromWorldPointTextPositioner(Camera.main,
transform.position, 1.5f, 50));

```

```

    }
}

```

```

    DestroyProjectile();
}

```

```

protected override void OnCollisionOther(Collider2D other)
{
    DestroyProjectile();
}

```

```

protected override void OnCollisionTakeDamage(Collider2D other, ITakeDamage
takeDamage)
{
    takeDamage.TakeDamage(Damage, gameObject);
    DestroyProjectile();
}

```

```

private void DestroyProjectile()
{
    if (DestroyedEffect != null)
        Instantiate(DestroyedEffect, transform.position, transform.rotation);

    if (DestroySound != null)
        AudioSource.PlayClipAtPoint(DestroySound, transform.position);
    Destroy(gameObject);
}
}

```

A35. Start Screen

```
using UnityEngine;
```

```
public class StartScreen : MonoBehaviour
{
    public string FirstLevel;

    public void Update()
    {
        if (!Input.GetMouseButtonDown(0))
            return;

        Application.LoadLevel(FirstLevel);
    }
}
```