ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΜΕ ΧΡΗΣΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ SDL

ΜΑΡΙΑ ΚΑΣΤΑΝΑΚΗ (ΑΜ 1364)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΘΑΝΑΣΙΟΣ ΚΟΥΤΡΑΣ

ΑΝΤΙΡΡΙΟ, 2018

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΜΗ ΛΟΓΟΚΛΟΠΗΣ

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Ακόμα δηλώνω ότι αυτή η γραπτή εργασία προετοιμάστηκε από εμένα προσωπικά και αποκλειστικά και ειδικά για την συγκεκριμένη πτυχιακή εργασία και ότι θα αναλάβω πλήρως τις συνέπειες εάν η εργασία αυτή αποδειχθεί ότι δεν μου ανήκει.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΣΠΟΥΔΑΣΤΗ	ΑΜ	ΥΠΟΓΡΑΦΗ
ΜΑΡΙΑ ΚΑΣΤΑΝΑΚΗ	1364	

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Αθανάσιο Κούτρα για την πολύτιμη βοήθειά του κατά τη διάρκεια της εκπόνησης της εργασίας μου.

Επίσης, ευχαριστώ την οικογένειά μου για την στήριξή της καθ'ολη τη διάρκεια των σπουδών μου, σε οποιαδήποτε δυσκολία αντιμετώπισα και ιδιαίτερα σε περιόδους έντονου άγχους.

Τέλος, τον αγαπημένο μου φίλο Dinu Mihai, ο οποίος ήταν δίπλα μου από τη πρώτη στιγμή και με βοήθησε να καταλάβω πως όταν πιστέψουμε πραγματικά στον εαυτό μας, μπορούμε να επιτύχουμε τους στόχους μας.

ΠΡΟΛΟΓΟΣ

Σκοπός αυτής της εργασίας, είναι να δείξουμε πώς μπορούμε να δημιουργήσουμε ένα video game σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης, χρησιμοποιώντας την βιβλιοθήκη SDL.

Επίσης, μέσα από αυτήν την εργασία, θα δούμε τι είναι τα video games, θα κάνουμε αναδρομή στην ιστορία τους και θα εξετάσουμε τις κατηγορίες στις οποίες διακρίνονται. Επιπλέον, θα μελετήσουμε τι πρέπει να λάβει υπόψη του ένας νέος developer για το σχεδιασμό ενός παιχνιδιού και τα στάδια ανάπτυξής του.

ΠΕΡΙΛΗΨΗ

Το πρώτο μέρος αυτής της πτυχιακής εργασίας, αναφέρεται στα video games, την ιστορία και τα είδη των video games, καθώς και στη διαδικασία ανάπτυξης ενός video game και το τι είναι απαραίτητο να έχει ένας game developer στο μυαλό του για την δημιουργία ενός παιχνιδιού.

Στο δεύτερο μέρος, αναφερόμαστε στην βιβλιοθήκη SDL και τη χρήση της για την δημιουργία ενός video game. Επίσης θα παρουσιάσουμε τα βήματα για τη δημιουργία του παιχνιδιού "Apple Cat".

ABSTRACT

In the first part of this thesis, we will analyze what a video game is, the history and the different genres of video games, as well as the procedure of a video game development and all the important steps a developer must follow and think before creating a video game.

In the second part, we will introduce the SDL library and refer to the advantages it offers in programming a video game. We will also present the steps of creating the video game "Apple Cat ".

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

game, video game, βιντεοπαιχνίδι, video game development, ανάπτυξη παιχνιδιού, game developer, SDL, κώδικας

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

Εδώ και αρκετά χρόνια, τα video games, γίνονται όλο και πιο δημοφιλή. Η εποχή που η ενασχόληση με ένα βιντεοπαιχνίδι ήταν υπόθεση των παιδιών, είναι πια παρελθόν. Πλέον, τα βιντεοπαιχνίδια, απευθύνονται και σε ενήλικες, κερδίζοντας όλο και μεγαλύτερο κοινό. Δεν είναι πια μια «σπατάλη» του ελεύθερου χρόνου μας, αλλά ένα μέσο ψυχαγωγίας.

Η ταχύτητα με την οποία εξελίσσονται τα video games είναι ραγδαία, η εμπειρία που προσφέρουν στον παίχτη γίνεται όλο και καλύτερη, όλο και πιο «γεμάτη». Δεν είναι τυχαίο ότι πολλά από τα παιχνίδια που κυκλοφορούν, ανταγωνίζονται επάξια ακόμα και ταινίες σε ορισμένες κατηγορίες, όσο αφορά το βαθμό ικανοποίησης. Για παράδειγμα τα horror games, έχουν φτάσει στο σημείο να προκαλούν έκκριση αδρεναλίνης περισσότερο από ότι αν κάποιος έβλεπε μια τρομαχτική ταινία.

Πέρα όμως από τους ανθρώπους που παίζουν video games, υπάρχουν και αυτοί που τα δημιουργούν. Το game development παλιότερα, ήταν "προνόμιο" μεγάλων εταιρειών, γνωστών ονομάτων στην βιομηχανία. Στις μέρες μας όμως, όλο και περισσότερες ανεξάρτητες ομάδες ή και μεμονομένα άτομα, μπαίνουν στον κόσμο του game design και development.

Με την εξέλιξη της τεχνολογίας και το διαδίκτυο, σήμερα μπορεί κάποιος πολύ πιο εύκολα από ότι στο παρελθόν, να κάνει τα πρώτα του βήματα στον κόσμο των video games, μιας και η πρόσβαση σε λογισμικά σχεδιασμού είναι εύκολη, άμεση και οικονομική, ενώ τα εργαλεία σχεδιασμού εύχρηστα.

ΜΕΡΟΣ ΠΡΩΤΟ: Η ΘΕΩΡΙΑ ΤΩΝ VIDEO GAMES

ΚΕΦΑΛΑΙΟ 1: ΓΝΩΡΙΖΟΝΤΑΣ ΤΑ VIDEO GAMES

Εικόνα 1: Video Games

1.1 Τι είναι το παιχνίδι

Αρχικά, πριν προσπαθήσουμε να δώσουμε ορισμό στο τι είναι video game, οφείλουμε να παραδεχτούμε ότι ένα video game είναι ένα game, δηλαδή ένα παιχνίδι.

Τι είναι λοιπόν το παιχνίδι;

Παιχνίδι είναι η δραστηριότητα, η οποία συνήθως περιλαμβάνει τους παράγοντες δεξιότητα, γνώση ή και τύχη, κατά την οποία ακολουθούμε συγκεκριμένους κανόνες με σκοπό να κερδίσουμε τον αντίπαλο ή να επιλύσουμε ένα γρίφο. Η δραστηριότητα αυτή γίνεται για διασκέδαση και δεν έχει αντίκτυπο στην πραγματική μας ζωή.

Ένα παιχνίδι έχει πάντα στοιχεία-συστατικά και κανόνες. Τα στοιχεία είναι το υλικό (hardware) , οι κανόνες είναι το λογισμικό (software). Και τα δύο καθορίζουν το παιχνίδι. Και τα δύο μπορούν να υπάρχουν ανεξάρτητα το ένα από το άλλο, αλλά ξεχωριστά δεν είναι παιχνίδι.

Τα στοιχεία και οι κανόνες μπορούν να συνδυαστούν. Ένα σύνολο στοιχείων μπορεί να χρησιμοποιηθεί με διαφορετικούς κανόνες. Αντιστρόφως, ένα σύνολο κανόνων μπορεί να χρησιμοποιηθεί με διαφορετικά στοιχεία.

1.2 Κριτήρια που καθιστούν ένα παιχνίδι

Οι κανόνες ωστόσο, δεν αρκούν για να ορίσουμε ένα παιχνίδι, αλλά είναι μέρος κριτηρίων που το καθιστούν.

Τα κριτήρια αυτά είναι:

- Κανόνες παιχνιδιού
- Στόχος παιχνιδιού
- Παράγοντας τύχη
- Ανταγωνισμός

1.2.1 Κανόνες παιχνιδιού

Όπως αναφέραμε παραπάνω, οι κανόνες και τα συστατικά καθορίζουν το παιχνίδι. Όλα όσα περιέχονται στους κανόνες είναι μέρος του παιχνιδιού. Όλα όσα δεν περιλαμβάνονται στους κανόνες δεν ανήκουν στο παιχνίδι. Οι κανόνες είναι τα σύνορα και η καρδιά του παιχνιδιού. Αναφέρονται μόνο στο παιχνίδι και δεν υπάρχουν ποτέ έξω από αυτό. Παρόλο που το παιχνίδι έχει κανόνες που είναι σαν

τους νόμους, το παιχνίδι είναι εθελοντικό. Όποιος παίζει ένα παιχνίδι, δεσμεύεται οικειοθελώς στους κανόνες.

1.2.2 Στόχος παιχνιδιού

Κάθε παιχνίδι έχει ένα στόχο. Για την επίτευξή του, ο παίχτης πρέπει να λάβει υπόψη του δύο πράγματα. Την προϋπόθεση ή την συνθήκη που θα του επιτρέψει να πραγματοποιήσει το στόχο, προσφέροντάς του έτσι τη νίκη, καθώς και την στρατηγική που απαιτείται να ακολουθήσει ο παίχτης για να φτάσει στον ζητούμενο στόχο.

1.2.3 Παράγοντας τύχη

Αν συγκρίνουμε όλα τα μέσα ψυχαγωγίας, θα διαπιστώσουμε ότι αυτό το χαρακτηριστικό το συναντάμε μόνο στα παιχνίδια. Κάποιος που διαβάζει ένα βιβλίο, παρακολουθεί μια ταινία ή ακούει μουσική, μπορεί να επαναλάβει την εμπειρία ανά πάσα στιγμή, αλλά η πορεία και το περιεχόμενο είναι πάντα το ίδιο.

Αντίθετα , μπορούμε να παίξουμε ένα παιχνίδι πολλές φορές και η πορεία για την επίτευξη του στόχου, να είναι πάντα διαφορετική. Επίσης, κάθε φορά που ξεκινάμε ένα παιχνίδι στο οποίο συμμετέχουν περισσότεροι από έναν παίχτες, είναι άγνωστο το ποιος θα κερδίσει. Η αβεβαιότητα και το άγνωστο είναι αυτά που κάνουν τα παιχνίδια τόσο συναρπαστικά και ευχάριστα.

Ο παράγοντας τύχη, παίζει μεγαλύτερο ή μικρότερο ρόλο, ανάλογα το είδος και τη φύση του παιχνιδιού. Από αυτόν εξαρτάται το πώς θα εξελιχθεί το παιχνίδι κάθε φορά που ξεκινάμε από την αρχή. Η τύχη κάνει τα παιχνίδια απρόβλεπτα και ενδιαφέροντα και προκαλεί την πορεία του παιχνιδιού να αναπτύσσεται διαφορετικά κάθε φορά.

Υπάρχουν διάφοροι τρόποι να εισάγουμε τον παράγοντα τύχη σε ένα παιχνίδι. Με «γεννήτρια» τυχαίων αριθμών (ζάρι), με διαφορετική κατάσταση εκκίνησης (τράπουλα), με ελλειπή στοιχεία (άγνωστη στρατηγική αντιπάλων), με πολύ μεγάλο αριθμό επιλογών κίνησης κ.α.

1.2.4 Ανταγωνισμός

Κάθε παιχνίδι επιδεικνύει ανταγωνισμό. Οι παίκτες ανταγωνίζονται μεταξύ τους και στο τέλος υπάρχουν νικητές και ηττημένοι. Ακόμη και σε παιχνίδια όπου οι παίκτες εργάζονται ως ομάδα (cooperative games ή coop games), υπάρχει ανταγωνισμός. Στην περίπτωση αυτή, οι παίκτες ανταγωνίζονται σε μία από τις προκαθορισμένες καταστάσεις, δηλαδή οι παίκτες παίζουν συνεργατικά εναντίον του παιχνιδιού.

Ακόμα και σε ένα παιχνίδι με έναν μόνο παίχτη (single player game), μπορεί να υπάρξει ανταγωνισμός, όπου ο παίχτης αγωνίζεται ενάντια στο παιχνίδι ή το χρόνο.

1.3 Τι είναι το video game

Η αλήθεια είναι ότι δεν υπάρχει ένας και μοναδικός ορισμός που να απαντά στην ερώτηση τι ειναι video game.

Βιντεοπαιχνίδι είναι ένα παιχνίδι που παίζεται χρησιμοποιώντας κουμπιά ή πλήκτρα με τα οποία εναλλάσουμε εικόνες που προβάλλονται σε μια οθόνη. Είναι ένα παιχνίδι στο οποίο χειριζόμαστε ηλεκτρονικά, εικόνες που παράγονται από ένα πρόγραμμα υπολογιστή και προβάλλονται σε μια οθόνη.

Ακόμα, video game είναι ένα παιχνίδι το οποίο παίζουμε χάρη σε μια οπτικοακουστική συσκευή και το οποίο μπορεί να βασιστεί στην διαδραστική αφήγηση μιας ιστορίας. Είναι ένα σύστημα στο οποίο οι παίχτες συμμετέχουν σε μια ψηφική «σύγκρουση», η οποία διέπεται από κανόνες και φέρει ένα μετρήσιμο αποτέλεσμα.

Θα μπορούσαμε επίσης να ορίσουμε ως video game, ένα ηλεκτρονικό παιχνίδι που περιλαμβάνει αλληλεπίδραση με μια διεπαφή χρήστη για τη δημιουργία οπτικής ανάδρασης σε μια συσκευή βίντεο όπως μια οθόνη τηλεόρασης, υπολογιστή, κινητού κτλ.

Συνδυάζοντας τα παραπάνω, καταλήγουμε στον εξής ορισμό ή περιγραφή ενός video game. Video game είναι ένα ηλεκτρονικό παιχνίδι, το οποίο διέπεται από κανόνες και περιλαμβάνει την αλληλεπίδραση με μια διεπαφή χρήστη, χρησιμοποιώντας κουμπιά ή πλήκτρα για τη δημιουργία μιας σειράς εναλλασόμενων εικόνων που προβάλλονται σε μια οθόνη, με σκοπό την ψυχαγωγία του χρήστη.

Τα video games εξελίσσονται με τόσο γρήγορους ρυθμούς, που είναι δύσκολο να δώσουμε ένα διαχρονικό ορισμό. Συνεχώς προστίθενται νέα χαρακτηριστικά, που τα κάνουν να βγαίνουν από το «καλούπι» των παραδοσιακών παιχνιδιών.

1.4 Γιατί τα video games είναι τόσο δημοφιλή

Εδώ και αρκετά χρόνια, τα video games δεν αποτελούν αποκλειστική ενασχόληση των παιδιών και των εφήβων. Πλέον οι ενήλικες αντιπροσωπεύουν ένα μεγάλο ποσοστό του πληθυσμού που ασχολείται με το gaming. Ακόμα και το ποσοστό των γυναικών που παίζουν video games έχει αυξηθεί. Σήμερα, ο καθένας μπορεί να είναι gamer, ανεξαρτήτως φύλου ή ηλικίας. Ποια είναι όμως τα χαρακτηριστικά εκείνα που καθιστούν τα video games τόσο ελκυστικά σε ένα μεγάλο εύρος αποδεκτών;

Ο λόγος που τα video games είναι τόσο δημοφιλή, είναι ότι καλύπτουν ορισμένες βασικές ανθρώπινες επιθυμίες. Οι επιθυμίες αυτές αφορούν: τον σκοπό, την ανταμοιβή, την αυτονομία, την δημιουργικότητα, την ιδιοκτησία, την κοινωνικότητα, την προσφορά, την σπανιότητα, το απρόβλεπτο.

1.4.1. Σκοπός

Όλοι αναζητούμε μια μορφή νοήματος, ένα σκοπό στη ζωή μας, είτε έχει τη μορφή θρησκείας, επαγγέλματος, οικογένειας ή οτιδήποτε άλλο. Έτσι και στα video games, προσφέρεται στον παίκτη αυτή η αναζήτηση, στα πλαίσια της επίτευξης του στόχου, είτε αυτός είναι η εξολόθρευση του εχθρού, η δημιουργία μιας κοινότητας ή αυτοκρατορίας, η λύση ενός γρίφου, ή γενικά η κατάκτηση ένός υψηλότερου επιτεύγματος. Ειδικότερα σε multiplayer games, όπου συμμετέχουν περισσότεροι παίχτες, δημιουργείται κλίμα συνεργασίας και ομαδικότητας, που ενθαρρύνει την έννοια αυτή ακόμα περισσότερο.

Στο game mode "Capture the Flag" του **Team Fortress 2**, οι παίκτες χωρίζονται σε 2 ομάδες, όπου η καθεμία έχει σκοπό να πάρει τη σημαία (χαρτοφύλακα) της αντίπαλης ομάδας.

1.4.2 Ανταμοιβή

Δεν υπάρχει καλύτερο συναίσθημα, μετά από ώρες ενασχόλησης πάνω σε ένα αντικείμενο, από το να νιώθουμε επιτυχημένοι. Γίνεται να αισθανθούμε το ίδιο παίζοντας ένα παιχνίδι; Η απάντηση είναι θετική. Όταν τα video games προσφέρουν ανταμοιβές για την προσπάθεια ή την επιτυχή ολοκλήρωση μιας αποστολής, ικανοποιούν την ανάγκη του παίχτη να αισθανθεί πετυχημένος. Κάθε τι που κερδίζει στην πορεία του παιχνιδιού, είτε αυτό είναι κάποιο καλύτερο όπλο για την εξόντωση του εχθρού, είτε επιπλέον ζωή για τον χαρακτήρα στο game, κάθετί που κερδίζει έπειτα από μια δύσκολη πρόκληση, τον φέρνει ένα βήμα πιο κοντά στο συναίσθημα της ολοκλήρωσης.

Στο παιχνίδι **Planetbase**, για να κατακτήσει ο παίκτης το ορόσημο του κάθε πλανήτη, πρέπει να αναπτύξει τη βάση με συγκεκριμένη στρατηγική, ώστε οι άποικοι να επιβιώσουν μακροχρόνια (σε game time), κάτι το οποίο γίνεται μέσω της trial and error μεθόδου.

1.4.3 Αυτονομία

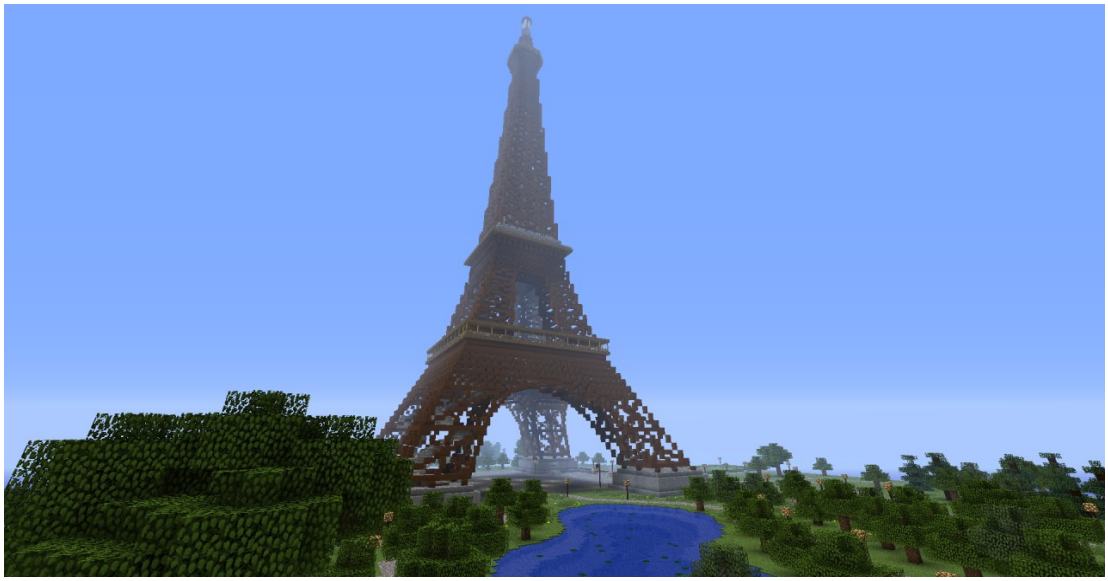
Η επιθυμία να αισθανόμαστε ανεξάρτητοι ή να έχουμε τον έλεγχο των πράξεών μας. Η πορεία προς την αυτονομία είναι ο λόγος για τον οποίο οι άνθρωποι δεν θέλουν να νιώθουν ότι χειραγωγούνται, ο λόγος που η φυλάκιση είναι μια μορφή τιμωρίας και ο λόγος που οι άνθρωποι νιώθουν μια έμφυτη ανάγκη για εξέγερση όταν νιώθουν σκλαβωμένοι. Αυτή η ανάγκη εξηγεί γιατί πολλά παιχνίδια προσφέρουν ελευθερία κινήσεων και ελεύθερων επιλογών.

Στη σειρά παιχνιδιών **Grand Theft Auto**, ο παίκτης, μπορεί να μετακινηθεί από το ένα μέρος στο άλλο, χρησιμοποιώντας διαφορετικές διαδρομές. Επίσης, μπορεί να παραβλέψει ή να αγνοήσει τις κύριες αποστολές και απλά να περιπλανηθεί σε διάφορα μέρη της πόλης, αλληλεπιδρώντας με NPCs.

1.4.4 Δημιουργικότητα

Όλοι αγαπάμε να δημιουργούμε. Είναι αναζωογονητικό και τείνει να μας θυμίζει την παιδική μας ηλικία. Video games που επιτρέπουν να προσαρμόσουμε το περιβάλλον του παιχνιδιού, ξυπνούν τη δημιουργικότητά μας . Είτε πρόκειται για την δημιουργία του χαρακτήρα/ήρωα μέσα στο παιχνίδι , είτε για το σχεδιασμό μιας πόλης , τα παιχνίδια μπορούν να είναι ένας πολύ καλός τρόπος να έρθει ο παίκτης σε επαφή με την εσωτερική του δημιουργικότητα.

Το παιχνίδι **Minecraft**, προσφέρει την δυνατότητα στον παίκτη να κατασκευάσει κτίρια, φανταστικά ή ακόμη και από την πραγματική ζωή.



Εικόνα 2: Minecraft (Πύργος του Άιφελ)

1.4.5 Ιδιοκτησία

Η απόκτηση αγαθών ή πλούτου, μας παρέχει μια αίσθηση δύναμης και υπεροχής. Στον πραγματικό κόσμο, αποκτούμε χρήματα, αγοράζουμε κοσμήματα, έπιπλα κλπ. Στον κόσμο των video games, ο παίκτης μπορεί να αποκτήσει πρόσβαση σε αυτά με τη συλλογή εικονικών νομισμάτων, αγαθών και ακινήτων, αλλά χωρίς να χρειαστεί να ξοδέψει πραγματικά χρήματα.

Στο παιχνίδι **The Sims**, ο παίκτης μπορεί να αποκτήσει το σπίτι των ονείρων του, ακόμα και να το σχεδιάσει και επιπλώσει όπως εκείνος επιθυμεί, ανεξαρτήτως κόστους.

Εικόνα 3: The Sims 4

1.4.6 Κοινωνικότητα

Όλοι έχουμε την ανάγκη να αλληλεπιδρούμε με άλλους ανθρώπους. Κάποιοι το επιθυμούν σε μεγαλύτερο βαθμό, ενώ άλλοι λιγότερο. Ανεξάρτητα όμως από το πόσο, όλοι μας επιζητούμε κοινωνική αλληλεπίδραση. Τα video games είτε μέσω ενός ανταγωνιστικού παιχνιδιού με άλλους παίκτες, είτε μέσω ενός coop game με φίλους ή ακόμα μέσω global scoreboard στα solo games, παρέχουν μια μεγάλη μέθοδο για να αξιοποιήσει ο παίκτης την έμφυτη ανθρώπινη ανάγκη του για κοινωνικοποίηση.

Αυτό επιτυγχάνεται πιο εύκολα σε MMORPGs (Massively Multiplayer Online Role-Playing Games), όπως το **World Of Warcraft**, μιας και του δίνεται η ευκαιρία να αλληλεπιδράσει με ένα μεγάλο αριθμό διαφορετικών παικτών. Μαθαίνει έτσι να συνεργάζεται με άλλους παίκτες και αναπτύσει ή εξασκεί τις κοινωνικές του δεξιότητες.

1.4.7 Προσφορά

Σε όλους μας αρέσει να νιώθουμε ότι έχουμε σημασία για τους άλλους και θέλουμε να αισθανόμαστε ότι συμβάλλουμε σημαντικά στην κοινωνία. Τα παιχνίδια εκπληρώνουν εύκολα αυτήν την ανάγκη, καθώς οι παίκτες μπορούν να παίξουν με φίλους τους online και να συμβάλλουν στην ομάδα, ή ακόμα και όταν οι παίκτες αλληλεπιδρούν με ανθρώπους που δεν είναι πραγματικοί. Παιχνίδια όπου ο παίκτης μιλά με έναν NPC (non-player character), ο οποίος του ζητά να τον βοηθήσει να βρει αυτό που ψάχνει ή να ανακαλύψει ένα νέο στοιχείο.

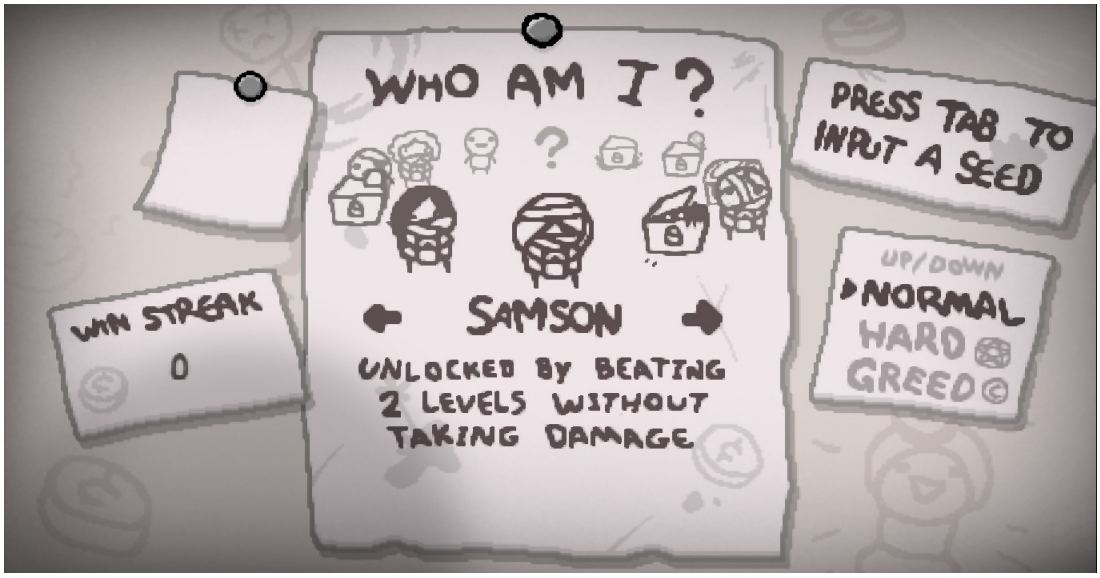
Για παράδειγμα στο παιχνίδι **The Incredible Adventures of Van Helsing**, ο παίκτης μιλά με NPCs, οι οποίοι του αναθέτουν αποστολές (quests).

Εικόνα 4: The Incredible Adventures of Van Helsing

1.4.8 Σπανιότητα

Τείνουμε να θέλουμε αυτό που δεν μπορούμε να έχουμε. Όσο πιο σπάνιο είναι ένα αντικείμενο ή όσο πιο δύσκολο είναι να το αποκτήσουμε, τόσο πιο ελκυστικό γίνεται. Έτσι και τα παιχνίδια χρησιμοποιούν αυτή την πτυχή της ψυχολογίας για να μας τραβήξουν το ενδιαφέρον. Για παράδειγμα, ένα παιχνίδι μπορεί να παρέχει στους παίκτες περιορισμένο αριθμό όπλων και στη συνέχεια να τους ζητάει να εκτελέσουν μια αποστολή ή να ξεπεράσουν μια πρόκληση πριν μπορέσουν να συγκεντρώσουν περισσότερα. Η επιθυμία να ξεπεραστεί η επόμενη πρόκληση είναι αυτό που διατηρεί τους παίκτες παθιασμένους για το παιχνίδι.

Στο παιχνίδι **The binding of Isaac: Rebirth**, ο παίκτης πρέπει να εκπληρώσει ορισμένες αποστολές που ικανοποιούν συνθήκες, προκειμένου να ξεκλειδώσει νέους χαρακτήρες και αντικείμενα που θα εμπλουτίσουν το gameplay.



Εικόνα 5: The Binding of Isaac Rebirth (character menu)

1.4.9 Απρόβλεπτο

Όλοι θέλουμε να μάθουμε τι θα συμβεί στη συνέχεια. Πώς θα τελειώσει ένα μυθιστόρημα ή μία ταινία. Είναι αυτό το αίσθημα της αγωνίας σχετικά με το αποτέλεσμα που μας τραβά πραγματικά το ενδιαφέρον. Τα παιχνίδια «παίζουν» με αυτό το συναίσθημα, επιφυλάσσοντας εκπλήξεις. Όταν εισάγονται νέα στοιχεία ή εμφανίζονται νέοι χαρακτήρες, δημιουργείται το αίσθημα της αγωνίας και της περιέργειας για το πώς θα εξελιχθεί η πλοκή του παιχνιδιού, κάνοντας έτσι τον παίχτη να θέλει να συνεχίζει να παίζει για να ανακαλύψει τι θα γίνει.

Στο παιχνίδι **The Wolf Among Us,** ο παίχτης παρακολουθεί μια ιστορία να εκτυλίσσεται, στην οποία συμμετέχει και ο ίδιος, με επιλογές που επηρεάζουν την πλοκή. Καθώς η ιστορία προχωρά, νέα στοιχεία εισάγονται στην υπόθεση καθώς και νέοι ύποπτοι, δημιουργώντας την περιέργεια στον παίκτη να συνεχίσει για να ανακαλύψει την αλήθεια.

Εικόνα 6: The Wolf Among Us

1.5 Η ιστορία των video games

Η ιστορία των video games ξεκινά στις αρχές της δεκαετίας του '50, ενώ έγιναν ευρέως αποδεκτά τη δεκαετία του '70 και του '80, όταν εισήχθησαν στο ευρύ κοινό τα arcade games, οι κονσόλες παιχνιδιών και τα pc games.

Σήμερα τα video games είναι δημοφιλής μορφή διασκέδασης και μέρος του παγκόσμιου πολιτισμού. Σχεδόν κάθε νοικοκυριό έχει ένα μέλος που ασχολείται με το gaming. Επίσης, η βιομηχανία βιντεοπαιχνιδιών είναι μία από τις πιο κερδοφόρες στον κόσμο, τα κέρδη της οποίας συνεχώς αυξάνονται.

Ως ορόσημα στην ιστορική διαδρομή, θα χρησιμοποιήσουμε κυρίως τις γενιές κονσολών.

1.5.1 Δεκαετία 1950

Αρχικά, να αναφέρουμε ότι το **1945**, κατασκευάστηκε ο πρώτος μεγάλης κλίμακας επαναπρογραμματιζόμενος ηλεκτρονικός υπολογιστής, ο **ENIAC (Electronic Numerical Integrator And Computer).** Αυτό ήταν το πρώτο τεχνολογικό βήμα, για την μετέπειτα δημιουργία των βιντεοπαιχνιδιών.

Το **1950**, το δίδυμο Claude Shannon και Alan Turning άρχισαν να προγραμματίζουν, το διάσημο επιτραπέζιο παιχνίδι **Chess**, σε υπολογιστή. Τους πήρε αρκετό χρόνο,

καθώς έπρεπε να δημιουργήσουν τεχνητή νοημοσύνη για να μπορεί ο παίκτης να παίζει μόνος του. Την ίδια χρονιά κυκλοφόρησαν το πρόγραμμα σκακιού.

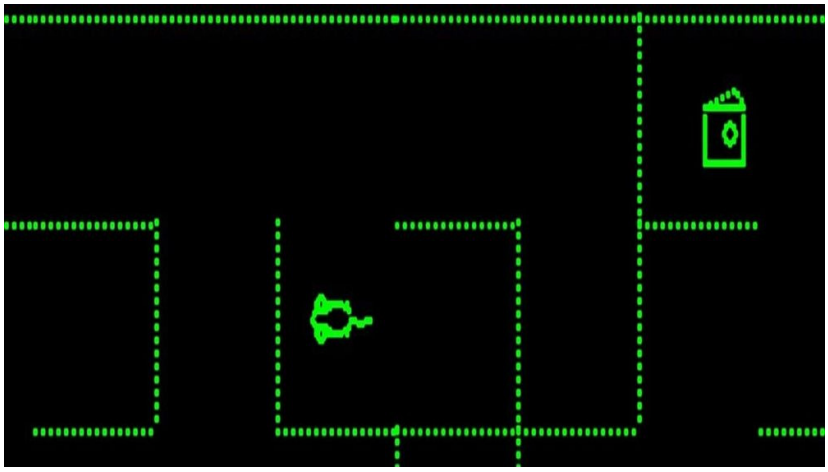
Το **1952**, ο Βρετανός καθηγητής A.S. Douglas, υποψήφιος για Ph.D. στην επιστήμη των υπολογιστών, δημιούργησε το παιχνίδι **OXO** ή αλλιώς tic-tac-toe, η γνωστή σε μας τρίλιζα, ως μέρος της διδακτορικής του διατριβής στο Πανεπιστήμιο του Cambridge. Το παιχνίδι αυτό "έτρεχε" στον υπολογιστή EDSAC (Electronic Delay Storage Automatic Calculator).

Το **1955**, ο αμερικανικός στρατός δημιούργησε ένα παιχνίδι προσομοίωσης πολέμου, το **Hutspiel**, στο οποίο η κόκκινη ομάδα αντιπροσώπευε το NATO και η μπλε ομάδα την USSR.

Το **1956**, ο Arthur Samuel ανέπτυξε ένα πρόγραμμα για το επιτραπέζιο παιχνίδι **Checkers** (Ντάμα), για τον υπολογιστη IBM 701 (Η σειρά IBM 700 έκανε ευκολότερους τους επιστημονικούς υπολογισμούς). Το checkers program θεωρείται ορόσημο της τεχνητής νοημοσύνης.

Το **1958**, ο Αμερικανός φυσικός William Higinbotham, δημιουργεί το παιχνίδι **Tennis for two**, το οποίο χαρακτηρίζεται από πολλούς ως το **πρώτο διαδραστικό βιντεοπαιχνίδι**. Το παιχνίδι χρησιμοποιεί έναν παλμογράφο ως οθόνη, ενώ οι παίκτες έχουν από ένα μοχλό με ένα κουμπί και έναν περιστρεφόμενο διακόπτη επιλογής.

Το **1959**, μαθητές του MIT, δημιουργούν το παιχνίδι **Mouse in the Maze** στον υπολογιστή TX-0 του MIT. Οι παίκτες χρησιμοποιώντας ένα ειδικό στυλό με φως ως συσκευή εισόδου, τοποθετούσαν τους τοίχους του λαβύρινθου και τις τελείες που αντιστοιχούσαν σε "τυρί" πάνω σε μία οθόνη τύπου αφής. Έπειτα ένα ψηφιακό ποντίκι προσπαθούσε να βρει τα αντικείμενα (τυριά), κινούμενο μέσα στο λαβύρινθο.



Εικόνα 7: Mouse in the Maze

1.5.2. Δεκαετία 1960

Το **1962**, δημιουργήθηκε το πρώτο computer based βιντεο παιχνίδι, το **Spacewar**, από ένα σπουδαστή του MIT, τον Steve Russeel. Στο παιχνίδι αυτό οι δύο παίκτες προσπαθούσαν να εκτοξεύσουν τορπίλες ο ένας στον άλλο και ταυτόχρονα να αποφύγουν την βαρύτητα του Ήλιου που τους τραβούσε προς τον "θάνατο".

Εικόνα 8: Spacewar

Το **1964**, ένας προγραμματιστής ηλεκτρονικών υπολογιστών με το όνομα John Kemeny είπε: «Ο καθένας είναι προγραμματιστής». Ο John, πρώην βοηθός έρευνας του Albert Einstein, συν-αναπτύσσει τη **γλώσσα προγραμματισμού BASIC** που επιτρέπει τη δημιουργία βιντεοπαιχνιδιών με ευκολία.

Το **1967**, οι προγραμματιστές της Sanders Associates, Inc., με επικεφαλής τον Ralph Baer, εφηύραν ένα πολυμεσικό σύστημα mutliplayer βιντεοπαιχνιδιού, το οποίο θα μπορούσε να παιχτεί σε μια τηλεόραση. Ήταν γνωστό με την ονομασία **"The Brown Box"**.

Το "Brown Box" χορηγήθηκε στην Magnavox, η οποία κυκλοφόρησε το σύστημα ως Magnavox Odyssey το 1972.

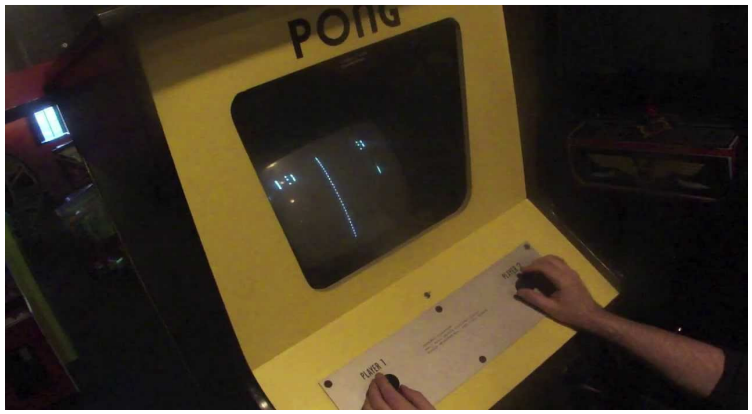
Το "Brown Box" μπορούσε να προγραμματιστεί για να παίξει διάφορα παιχνίδια, όπως πινγκ πονγκ, ντάμα και τέσσερα αθλητικά παιχνίδια. Χρησιμοποιούσε προηγμένη τεχνολογία για την εποχή του, με πρόσθετα αξεσουάρ που περιλάμβαναν ένα ελαφρύ όπλο για παιχνίδι σκοποβολής και ένα ειδικό εξάρτημα για παιχνίδι γκολφ.

1.5.3 Δεκαετία 1970

Το **1971**, Οι Nolan Bushnell και Ted Dabney, που αργότερα ίδρυσαν την Atari, δημιούργησαν το **Computer Space**, ένα arcade παιχνίδι, το οποίο ήταν το πρώτο εμπορικό βιντεοπαιχνίδι.

Την ίδια χρονιά, τρεις φοιτητές από τη Μινεσότα, ο Don Rawitsch, ο Bill Heinemann και ο Paul Dillenberger, δημιούργησαν το **The Oregon Trail** για τον τηλετύπο. Το 1973, προσαρμόζεται στον υπολογιστή και το 1978 διανέμεται σε δισκέτες. Στο παιχνίδι ο παίκτης ταξιδεύει με την ομάδα του στο Όρεγκον, ενώ παράλληλα πρέπει να προφυλάξει τα μέλη από την πείνα ή τις αρρώστιες. Το Oregon Trail υπήρξε ένα από τα πιο δημοφιλή games, ενώ η πιο πρόσφατη κυκλοφορία του ήταν το 2011.

Το **1972**, "γεννήθηκε" ο θρύλος των arcade games, το **Pong**. Οι προγραμματιστές της Atari, ο Nolan Bushnell και ο Al Alcorn ανέπτυξαν το επιτραπέζιο παιχνίδι πινγκ - πονγκ , για arcade μηχανή. Μόλις κυκλοφόρησε, έγινε αμέσως δημοφιλές σε μικρούς και μεγάλους, ενώ οι μηχανές arcade ξεκίνησαν να εμφανίζονται σε μπαρ, μπόουλινγκ και εμπορικά κέντρα σε όλο τον κόσμο.



Εικόνα 9: Pong

Το **1973**, κυκλοφόρησε το πρώτο βιντεοπαιχνίδι, που επιτρέπει το multiplayer σε ξεχωριστές οθόνες. Το **Empire**, ένα turn-based παιχνίδι στρατηγικής, με έως και 8 παίκτες, το οποίο δημιουργήθηκε για το σύστημα υπολογιστών δικτύου **PLATO (Programmed Logic for Automatic Teaching Operations)**, ένα από τα πρώτα γενικευμένα υπολογιστικά συστήματα διδασκαλίας.

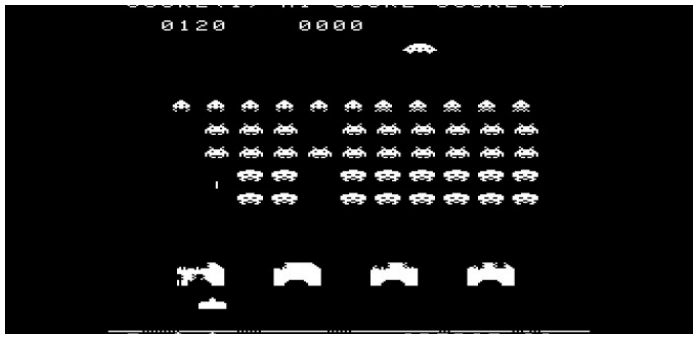
Το **1974**, ο Jim Bowery δημιούργησε το **Spasim**, το πρώτο videogame με 3D διανυσματικά γραφικά -για το σύστημα PLATO- με τη δυνατότητα συμμετοχής 32 παικτών, που περιελάμβανε 4 πλανητικά συστήματα με έως και 8 παίκτες ανά πλανητικό σύστημα.

Την ίδια χρονιά, κυκλοφόρησε το πρώτο first person shooter game, το **Maze War**, στο οποίο ο παίκτης κινείται σε ένα λαβύρινθο, προσπαθώντας να βρει άλλους

παίκτες και να τους πυροβολήσει. Ο παίκτης μπορεί να κινηθεί μπρος, πίσω, δεξιά και αριστερά (με γωνία περιστροφής 90 μοιρών).

Το **1977**, η Atari κυκλοφόρησε το Video Computer System, γνωστό ως **Atari 2600**, την πρώτη κονσόλα παιχνιδιών για το σπίτι. Το σύστημα ήταν πρωτοποριακό. Είχε κασέτες στις οποίες μπορούσε ο παίκτης να αποθηκεύσει πληροφορίες των παιχνιδιών, ένα joystick, χρωματική παλέττα, αυξανόμενο αριθμό παιχνιδιών που μπορούσε να "τρέξει"και τη δυνατότητα αλλαγής επιπέδων δυσκολίας.

Το **1978**, ο Niskikado Tomohiro, δημιούγησε το **Space Invaders**, ένα video game που αποτέλεσε ορόσημο των arcade games τη δεκαετία του '70. Το Space Invaders είχε μαζική απήχηση, καθώς ήταν πρωτοπόρο στα shooting games, εισάγωντας το top-down gameplay, όπου ο παίκτης πυροβολεί εξωγήινα διαστημόπλοια που εμφανίζονται από το πάνω μέρος της οθόνης.



Εικόνα 10: Space Invaders

Το **1979**, κυκλοφόρησε ένας από τους πιο δημοφιλείς τίτλους videogames της χρυσής εποχής των arcade games, το **Asteroids** από την Atari. Στο παιχνίδι αυτό, ο παίκτης ελέγχει ένα διαστημόπλοιο, κινούμενος σε ένα πεδίο αστεροειδών από το οποίο περνάνε περιοδικά ιπτάμενοι δίσκοι. Σκοπός του παιχνιδιού είναι ο παίκτης να πυροβολήσει και να καταστρέψει αστεροειδείς και ιπτάμενους δίσκους, χωρίς να συγκρουστεί ή να χτυπηθεί από αντίπαλα πυρά.

1.5.4 Δεκαετία 1980

Μια πίτσα από την οποία έλειπε ένα κομμάτι, έδωσε την έμπνευση στον Iwatani Toru να σχεδιάσει ένα από τα πιο διάσημα βιντεοπαιχνίδια στον κόσμο, το **Pac-Man**. Σκοπός του Iwatani ήταν να δημιουργήσει ένα μη βίαιο παιχνίδι που να απευθύνεται ακόμη και σε κορίτσια. Γι αυτό και είχε στοιχεία όπως λαβύρινθο και χαριτωμένα φαντασματάκια για εχθρούς. Επίσης, το σενάριο του να αποκτάς δύναμη τρώγοντας, το εμπνεύστηκε από τον Ποπάι! Το Pac-Man κυκλοφόρησε το **1980** από τη Namco,

κατακτώντας τον κόσμο των arcade βιντεοπαιχνιδιών, με τις μεγαλύτερες πωλήσεις. Γίνεται το πρώτο arcade παιχνίδι που βγάζει έκδοση για το Atari 2600 (home console) και το πρώτο παιχνίδι που έχει έναν κινούμενο και επώνυμο κύριο χαρακτήρα. Δύο χρόνια αργότερα κυκλοφορεί η θηλυκή έκδοσή του, Ms Pac-Man.

Στο παιχνίδι, ο παίκτης καθοδηγεί τον Pac-Man μέσα από ένα λαβύρινθο που περιέχει διάφορες κουκίδες, γνωστές ως Pac-Dots, ενώ τέσσερα πολύχρωμα φαντάσματα τον κυνηγούν. Ο στόχος του παιχνιδιού είναι ο παίκτης να συγκεντρώσει πόντους τρώγοντας όλα τα Pac-Dots στο λαβύρινθο, ολοκληρώνοντας το «στάδιο» του παιχνιδιού, περνώντας στο επόμενο.

Εικόνα 11: Pac-Man

Το **1981**, κυκλοφόρησε από την Nintendo το βίντεοπαιχνίδι **Donkey Kong**, το οποίο γίνεται ένα από τα πιο επιτυχημένα παιχνίδια όλων των εποχών. Σηματοδότησε την αρχή της σειράς παιχνιδιών Mario, εισάγοντας τον χαρακτήρα Jumpman, ο οποίος αργότερα έγινε γνωστός ως Mario.

Το **1982** κυκλοφόρησε ο **Commodore 64**, ένας 8-bit personal computer από την Commodore International, ο οποίος πήρε το όνομά του από την μνήμη RAM 64KB που διέθετε. Με χιλιάδες τίτλους παιχνιδιών σε δισκέτα (όπως **Bubble Bobble, Turrican, Paranoid** κ.α.) και εκατομμύρια πωλήσεις, το PC gaming κάνει τα πρώτα του βήματα προς τη δόξα.

Το **1983**, κυκλοφόρησε το videogame **Star Wars** από την Atari Inc. Το παιχνίδι ήταν ένα first person space simulator, που προσομοίωνε την επίθεση στο Death Star από την ταινία Star Wars του 1977. Το παιχνίδι αποτελούταν από 3D διανυσματικά γραφικά. Αναπτύχθηκε κατά τη διάρκεια της Χρυσής Εποχής των arcade games (1978 - 1983) και θεωρείται ένα από τα πιο δημοφιλή παιχνίδια όλων των εποχών.

Το **1985**, η Nintendo κυκλοφόρησε την κονσόλα βιντεοπαιχνιδιών **NES (Nintendo Entertainment System)** ή αλλιώς Famicom όπως ονομαζόταν στην Ιαπωνία. Το NES είχε βελτιωμένα 8-bit γραφικά, χρώματα, ήχο και gameplay από τις προηγούμενες

κονσόλες, με πάνω από 800 τίτλους παιχνιδιών όπως τα **Metroid, Megaman, Castlevania, Legend of Zelda, Ducktales, Mario Super Bros** κ.α.

Το **1988**, ο κόσμος του ποδοσφαίρου εισήλθε σε videogame, με την κυκλοφορία του **John Madden Football** από την Electronic Arts. Το παιχνίδι περιείχε πολλές προσαρμόσιμες παραμέτρους, όπως καιρικές συνθήκες (ζέστη, βροχή, χιόνι ή κρύο και άνεμος), χρονικές περίοδοι (8, 10, 12 ή 15 λεπτά), κόπωση παικτών, τραυματισμοί παικτών και ποινές. Κυκλοφόρησε για τους 8-bit υπολογιστές Commodore 64 και Apple II.

Το **1989**, ξεκινά η εποχή του handheld gaming (φορητού παιχνιδιού), με την κυκλοφορία του **Game Boy** από την Nintendo. Το 8-bit Game Boy, «εισβάλλει» στις παλάμες των παικτών, με το παιχνίδι **Tetris**. Η ονομασία του παιχνιδιού προήλθε από το ελληνικό πρόθημα "τέτρα", που σημαίνει τέσσερα, καθώς κι από το τένις, αγαπημένο άθλημα του Pajitnov. Το παιχνίδι χρησιμοποιεί επτά σχήματα τούβλων, τα οποία πέφτουν από το πάνω μέρος της οθόνης, χωρίς σταματημό. Ο παίκτης καλείται να τα τοποθετήσει με τρόπο τέτοιο, ώστε να ταιριάζουν και να μην υπάρχουν κενά μεταξύ τους. Όταν τα τούβλα σχηματίζουν πλήρεις γραμμές, τότε οι γραμμές αυτές εξαφανίζονται οριζοντίως. Αν δεν γίνει αυτό και οι συνθέσεις από τουβλάκια έχουν συνεχώς κενό χώρο, η οθόνη αναπόφευκτα θα γεμίσει και το παιχνίδι θα τερματιστεί με ήττα του παίκτη.

Εικόνα 12: Tetris

Την ίδια χρονιά, κυκλοφόρησε το **Sega Genesis**, μια 16-bit κονσόλα βιντεοπαιχνιδιών από την Sega Enterprises, Ltd. Η κονσόλα αυτή κυκλοφόρησε πρώτα στην Ιαπωνία το 1988 με το όνομα Mega Drive. Ένα χρόνο αργότερα, ακολούθησε η Nintendo με την κυκλοφορία της επίσης 16-bit κονσόλας **SNES** (**Super Nintendo Entertainment System**).

1.5.5 Δεκαετία 1990

Η πρόοδος πλέον στην υψηλή τεχνολογία δεν θα μπορούσε παρά να επηρεάσει τη βιομηχανία παιχνιδιών. Με τον καιρό, οι προγραμματιστές κέρδισαν περισσότερες ευκαιρίες για να δημιουργήσουν τα δικά τους παιχνίδια. Τα ίδια τα παιχνίδια, με τη σειρά τους, έγιναν πιο λεπτομερή. Ωστόσο, αυτή ήταν μόνο η αρχή της ανάπτυξης γραφικών στα παιχνίδια.

Το **1990**, η Microsoft κυκλοφόρησε τα windows 3.0, τα οποία συμπεριλάμβαναν την ηλεκτρονική έκδοση του card game **Solitaire**, την γνωστή σε μας πασιέντζα. Εκατομμύρια χρήστες που δεν είχαν στην κατοχή τους κονσόλα παιχνιδιών, άρχισαν να εξοικιώνονται και να απολαμβάνουν παιχνίδια στον υπολογιστή. Το Solitaire έγινε ένα από τα πιο δημοφιλή casual βιντεοπαιχνίδια και πρόσφερε ένα μοντέλο παιχνιδιών πάνω στο οποίο δημιουργήθηκαν παιχνίδια όπως το Bejeweled.

Το **1991**, κυκλοφόρησε το βιντεοπαιχνίδι **Sonic the Hedgehog**, που έγινε σήμα κατατεθέν της Sega. Στο παιχνίδι αυτό, το οποίο ήταν platform game, ο Sonic ήταν ένας μπλε ανθρωπόμορφος σκατζόχοιρος που έτρεχε απίστευτα γρήγορα και μάζευε δακτυλίδια. Την ίδια χρονιά κυκλοφόρησε το δημοφιλές βιντεοπαιχνίδι **Street Fighter II** από την Capcom, ένα fighting game, που απεικόνιζε το αίμα όταν ο ένας χαρακτήρας χτυπούσε τον άλλο, εισάγοντας το gore στα βιντεοπαιχνίδια.

Εικόνα 13: Sonic the Hedgehog

Στην ίδια λογική gameplay, το **1992**, κυκλοφόρησε το **Mortal Kombat** από τη Midway Games, το οποίο μαζί με το Street Fighter, αποτέλεσαν τους πιο δημοφιλείς τίτλους fighting παιχνιδιών.

Το **1993**, η id Software κατασκεύασε το βιντεοπαχνίδι **DOOM** για το PC. Θεωρείται ένας από τους πιο σημαντικούς τίτλους στην ιστορία των βιντεοπαιχνιδιών, καθώς πρωτοπόρησε ως first person shooter game, όπου δηλαδή ο παίχτης βλέπει και πυροβολεί μέσα από τα μάτια του χαρακτήρα.

Εικόνα 14: DOOM

Ωστόσο το βίαιο για την τότε εποχή περιεχόμενό του, μαζί με την κυκλοφορία του Mortal Kombat, που περιελάμβανε αιματοχυσίες, δημιούργησαν προβληματισμούς στην κοινωνία σχετικά με τη βία στα βιντεοπαιχνίδια, γεγονός που οδήγησε ένα χρόνο αργότερα, το **1994**, στην ίδρυση του **Entertainment Software Ratings Board** από την Entertainment Software Association (πρώην Interactive Digital Software Association), ενός οργανισμού για τον καθορισμό της επιτρεπόμενης ηλικίας και την αξιολόγηση του περιεχομένου των βιντεοπαιχνιδιών.

Η πραγματική επανάσταση στο gaming ήρθε όταν τα **δίκτυα LAN**, και αργότερα το **Internet**, άνοιξαν τους ορίζοντες για multiplayer παιχνίδια. Το multiplayer gaming "εκτόξευσε" την gaming κοινότητα σε νέο επίπεδο, επειδή επέτρεψε στους παίκτες να ανταγωνίζονται και να αλληλεπιδρούν από διαφορετικούς υπολογιστές, γεγονός που βελτίωσε την κοινωνική πλευρά των παιχνιδιών. Αυτό το βήμα-κλειδί έθεσε τα θεμέλια του διαδραστικού gaming μεγάλης κλίμακας που απολαμβάνουν σήμερα οι σύγχρονοι παίκτες. Στις 30 Απριλίου **1993**, το **CERN (European Organization for Nuclear Research)** έβαλε το λογισμικό του **World Wide Web** στον public domain, αλλά θα περνούσαν χρόνια προτού το Internet γίνει αρκετά ισχυρό για να φιλοξενήσει το gaming όπως το γνωρίζουμε σήμερα.

Το **1994**, η Blizzard κυκλοφόρησε το **Warcraft: Orcs & Humans**, ένα από τα πρώτα παιχνίδια στρατηγικής με λεπτομερείς αποστολές. Η σειρά Warcraft έγινε η πιο δημοφιλής στα massive multiplayer online παιχνίδια (MMO).

Το **1995**, με την κυκλοφορία 5ης γενιάς 32-bit κονσολών, όπως το **Sega Saturn** της Sega και το **PlayStation** της Sony, καθώς και την κυκλοφορία του Nintendo 64 (64bit) από την Nintendo, ένα χρόνο μετά, άνοιξαν οι δρόμοι για παιχνίδια με **τρισδιάστατα γραφικά** (**3D graphics**).

Μία νέα γενιά παιχνιδιών έρχεται να προσφέρει τρισδιάστατη εμπειρία στον παίκτη με διάσημους τίτλους, όπως: **Crash Bandicoot** (**1996**, Sony Interactive Entertainment), **Tomb Raider** (**1996**, Eidos Interactive), **Final Fantasy VII** (**1997**, Square), **Gran Turismo** (**1997**, Sony Interactive Entertainment), **Metal Gear Solid** (**1998**, Konami), **The Legend of Zelda: Ocarina of Time** (**1998**, Nintendo), **Medal**

of Honor (**1999**, Electronic Arts), **Silent Hill** (**1999**, Konami) κ.α. Ακόμη, κυκλοφόρησε το πρώτο τρισδιάστατο Super Mario παιχνίδι, το **Super Mario 64** (**1996**, Nintendo), ενώ αξίζει να σημειωθούν ακόμη, το **Half Life** (**1998**, Valve), το οποίο απέσπασε εξαιρετικές κριτικές για το έξυπνο σενάριο και gameplay και το **Everquest** (**1999**, Sony Online Entertainment), ένα massively multiplayer online role-playing game (MMORPG).

Εικόνα 15: Half Life

Τέλος, δεν μπορούμε να παραλείψουμε το πιο διάσημο mobile game, το **Snake**, το οποίο κυκλοφόρησε το **1997** στα κινητά τηλέφωνα **Nokia**.

1.5.6 Δεκαετία 2000

Το **2000**, κυκλοφόρησε το παιχνίδι **The Sims** από την Electronic Arts. Το The Sims είναι μια σειρά simulation παιχνιδιών, το οποίο έγινε ένα από τα πιο διάσημα και με τις μεγαλύτερες πωλήσεις παιχνίδια, καθώς επίσης είχε και μεγάλη απήχηση σε κορίτσια gamers. Στο παιχνίδι αυτό, ο παίκτης είχε τη δυνατότητα να επιλέξει την εμφάνιση του χαρακτήρα του, να χτίσει το σπίτι του, να εργαστεί και να κάνει οικογένεια.

Άλλοι αξιόλογοι τίτλοι παιχνιδιών που κυκλοφόρησαν το ίδιο έτος: **Spiderman** (Activision), **Pokemon Silver Version** (Nintendo), **Thief II** (Eidos Interactive), **Counter Strike** (Valve), **Diablo II** (Blizzard), **Hitman: Codename 47** (Eidos Interactive) κ.α.

Την ίδια χρονιά, κυκλοφόρησε η νέα κονσόλα 6ης γενιάς της Sony, το **PlayStation 2**, το οποίο στα 128 bit είχε καλύτερα γραφικά από ότι τα PC, ενώ ταυτόχρονα παρείχε τη δυνατότητα αναπαραγωγής DVD.

Το **2001**, η Microsoft κυκλοφόρησε το **Xbox**. Το Xbox διέθετε τεχνολογία H/Y και θύρα Ethernet. Ήταν η πρώτη κονσόλα βιντεοπαιχνιδιών με ενσωματωμένη μονάδα σκληρού δίσκου, με σκοπό κυρίως την αποθήκευση της προόδου των παιχνιδιών,

καθώς και περιεχομένου που είχει ληφθεί από το Xbox Live. Ένας χρήστης του Xbox μπορούσε να αντιγράψει μουσική από τα τυπικά CD ήχου στον σκληρό δίσκο και να την χρησιμοποιήσει ως soundtrack σε ορισμένα παιχνίδια.

Εικόνα 16: Halo Combat Evolved

Το **2003**, η Valve κυκλοφόρησε την ψηφιακή πλατφόρμα παιχνιδιών, το **Steam**, δίνοντας την δυνατότητα στους PC gamers, να "κατεβάσουν" και να παίξουν παιχνίδια στον υπολογιστή, χωρίς την ανάγκη CD ή DVD, καθώς επίσης και να κάνουν update τα παιχνίδια τους. Το Steam, μέχρι και σήμερα, αποτελεί τη μεγαλύτερη ψηφιακή πλατφόρμα παιχνιδιών, προσφέροντας πρόσβαση σε αμέτρητα βιντεοπαιχνίδια.

Το **2004**, εισάγονται στην αγορά το **Nintendo DS** από τη Nintendo και το **PlayStation Portable (PSP)** από τη Sony, φορητές κονσόλες παιχνιδιών. Διέθεταν ενσωματωμένη οθόνη, χειριστήριο και ηχεία. Όπως τα προηγούμενα χρόνια με την κυκλοφορία του Game Boy, πρόσφεραν στον παίκτη τη δυνατότητα να παίζει ακόμα και όταν δεν βρισκόταν στο σπίτι, μιας και ήταν εύκολες στη μεταφορά, λόγω του μικρού μεγέθους.

Από τις αρχές του 2000, οι δυνατότητες του Internet εκτοξεύθηκαν και η τεχνολογία των επεξεργαστών υπολογιστών άρχισε να βελτιώνεται με τόσο γρήγορους ρυθμούς, με αποτέλεσμα κάθε νέα παρτίδα παιχνιδιών, γραφικών και κονσολών να βγάζει εκτός συναγωνισμού την προηγούμενη. Το κόστος της τεχνολογίας, των διακομιστών (servers) και του διαδικτύου μειώθηκε τόσο, ώστε το Internet έγινε προσιτό στο ευρύ κοινό. Με το πέρασμα των χρόνων, το online multiplayer gaming έγινε αναπόσπαστο κομμάτι της εμπειρίας των παικτών.

Το 2004, κυκλοφόρησε το **World of Warcraft** από την Blizzard, ένα massive multiplayer online game, το οποίο τέσσερα χρόνια αργότερα, θα φτάσει τους 10 εκατομμύρια συνδρομητές, εκτοξεύοντάς το στην κορυφή των παιχνιδιών της κατηγορίας του. Δημιούργησε ολόκληρα εικονικά σύμπαντα για τους παίκτες

επαναπροσδιορίζοντας τον τρόπο με τον οποίο παίζουν και αλληλεπιδρούν ο ένας με τον άλλον, την κοινωνική πτυχή του gaming.

Το **2005**, περάσαμε πλέον στην 7η γενιά κονσολών με την κυκλοφορία του **Xbox 360** από τη Microsoft. Το Xbox 360 έφερε τα γραφικά υψηλής ανάλυσης **HD (High Definition)** στα παιχνίδια, καθώς και την επανάσταση στο online gaming με βελτιώσεις στο Xbox Live. Το 360 επίσης έφερε εφαρμογές σε κονσόλες όπως το Netflix και το Youtube, καθώς και συστήματα συνομιλίας με φίλους και άλλους παίκτες στο διαδίκτυο.

Το **2006**, η Sony κυκλοφόρησε το **PlayStation 3**, το οποίο παρείχε τη δυνατότητα στους χρήστες να παίζουν Blu-Rays, ταινίες και μουσική. Το PS3 μπορούσε να συνδεθεί επίσης με τις φορητές κονσόλες PlayStation και πρόσφερε δωρεάν online gameplay χωρίς συνδρομή.

Εικόνα 17: The Elder Scrolls IV

Την ίδια χρονιά, η Nintendo κυκλοφόρησε στην αγορά το **Nintendo Wii**, μια κονσόλα που συμπεριλάμβανε τηλεχειριστήρια με αισθητήρες κίνησης, προσφέροντας στον παίκτη μια νέα εμπειρία στο gameplay, καθώς ο παίκτης για να αλληλεπιδράσει με το παιχνίδι έπρεπε να κινείται.

Προς το τέλος της δεκαετίας, , τα βιντεοπαιχνίδια εξαπλώθηκαν σε πλατφόρμες κοινωνικών μέσων δικτύωσης όπως το Facebook και κινητές συσκευές όπως το iPhone. Η ταχεία εξέλιξη της τεχνολογίας , η κυκλοφορία των smartphones και των app stores, εκτόξευσαν το mobile gaming. Τα βιντεοπαιχνίδια πλέον έχουν απήχηση σε τεράστιο εύρος κοινού, ελκύοντας ακόμη και αυτούς που δεν αχολούνται με το gaming.

Το **2009**, κυκλοφόρησε στο facebook το **Farm Ville** από τη Zynga, ένα παιχνίδι προσομοίωσης φάρμας. Την ίδια χρονιά, το **Angry Birds** κυκλοφόρησε στα app stores από την Chillingo. Στο παιχνίδι αυτό, ο παίκτης εκτόξευε πτηνά με μια σφεντόνα με σκοπό να πετύχει χοίρους που βρίσκονταν πάνω ή μέσα σε διάφορες κατασκευές.

1.5.7 Δεκαετία 2010

Από το **2010** έως και σήμερα, διανύουμε την **8η γενιά κονσολών**, με την κυκλοφορία του **Nintendo Wii U** το **2012**, την πρώτη κονσόλα της Nintendo που υποστηρίζει HD γραφικά. Ακολούθησαν το **Xbox One** από τη Microsoft και το **PlayStation 4** από τη Sony το **2013**. Πλέον η εξέλιξη των 3D γραφικών στα games είναι τόσο μεγάλη, που τα κάνει να φαίνονται απίστευτα ρεαλιστικά, ενώ εισάγεται και η εικονική πραγματικότητα (**Virtual Reality**) στα games.

Εικόνα 18: Far Cry 4

Το **2016** κυκλοφόρησε το **Oculus Rift** από την Oculus VR, ένας εξοπλισμός headset εικονικής πραγματικότητας. Αποτελείται από μία συσκευή με οθόνη και ακουστικά, την οποία ο παίκτης φοράει στο κεφάλι του και από μοχλούς που κρατάει στα χέρια με τους οποίους αλληλεπιδρά στο παιχνίδι. Έκτοτε, πολλά παιχνίδια, ενσωμάτωσαν τη δυνατότητα χρήσης ενός Virtual Reality headset. Τίτλοι όπως το **Surgeon Simulator** (**2013**, Bossa Studios) και το **Superhot** (**2016**, Superhot Team), προσφέρουν στον παίκτη την εμπειρία του VR gaming.

Τι ακολουθεί;

Δεν μπορούμε να απαντήσουμε με βεβαιότητα πού μπορεί να φτάσει το gaming στο μέλλον, το μόνο σίγουρο είναι ότι όσο εξελίσσεται η τεχνολογία, θα αξιοποιείται από τις εταιρείες και τους developers για την δημιουργία καινοτομιών, με σκοπό την καλύτερη δυνατή εμπειρία στο gaming όσο αφορά το κομμάτι του ρεαλισμού και της διαδραστικότητας.

Εικόνα 19: Tomb Raider

Εικόνα 20: Grand Theft Auto

1.6 Τα είδη των video games

Ο αριθμός των βιντεοπαιχνιδιών που κυκλοφορούν στην αγορά είναι τεράστιος. Ένα video game μπορεί να ανήκει σε μία ή περισσότερες κατηγορίες (ή υποκατηγορίες) παιχνιδιών. Υπάρχουν πολλά είδη video games (game genres), τα οποία κατηγοριοποιούνται με βάση τα χακτηριστικά τους και τους υποκειμενικούς στόχους προς επίτευξη. Τα είδη, δηλώνουν τον τύπο των παιχνιδιών που ανήκουν στην κάθε κατηγορία. Με αυτόν τον τρόπο, δίνεται η δυνατότητα στον παίκτη να επιλέξει ένα βίντεοπαιχνίδι μιας κατηγορίας που ταιριάζει περισσότερο στα ενδιαφέροντά του.

Παρακάτω θα εξετάσουμε οκτώ γενικές κατηγορίες βιντεοπαιχνιδιών, καθώς και τις υποκατηγορίες της κάθε μίας (subgenre).

1.6.1 Action Games

Τα παιχνίδια δράσης, αποτελούν μία από τις πιο δημοφιλείς κατηγορίες παιχνιδιών. Ο παίκτης έχει τον έλεγχο και το κέντρο της δράσης αποτελείται από φυσικές προκλήσεις που πρέπει να ξεπεράσει. Επειδή σε αυτά τα παιχνίδια παίζουν ρόλο τα αντανακλαστικά, αρκετές φορές χρειάζεται εξάσκηση των αισθητήριων κινήσεων ματιών και χεριών του παίκτη, ώστε να πετύχει καλύτερο αποτέλεσμα ή σκορ.

Στην κατηγορία των **action games**, ανήκουν οι εξής υποκατηγορίες:

1.6.1.1 Platform

Πήραν το όνομά τους από το γεγονός ότι ο χαρακτήρας του παιχνιδιού αλληλεπιδρά με πλατφόρμες, τρέχοντας, πηδώντας ή πέφτοντας για να φτάσει στον προορισμό του, αποφεύγοντας ή και σκοτώνοντας εχθρούς. Η γωνία της κάμερας είναι σχεδόν πάντα η ίδια, κοιτάζοντας από την πλάγια πλευρά του χαρακτήρα.

Παραδείγματα: **Super Mario Bros**, **Bubble Bobble**, **Oddworld**, **Tom Raider**, **Limbo**, **Psychonauts, Bionicle Toa Mata game** κ.α.

Εικόνα 21: Bionicle Toa Mata game

1.6.1.2 Shooters

Επιτρέπουν στον παίκτη να χρησιμοποιήσει όπλα, με σκοπό να εξουδετερώσει εχθρούς ή αντίπαλους παίκτες. Διακρίνονται σε first-person, third-person και top-down shooters. Στα first-person shooters, ο παίκτης "βλέπει" μέσα από τα μάτια του χαρακτήρα. Στα third-person, ο παίκτης βλέπει ολόκληρο τον χαρακτήρα, η οπτική γωνία βρίσκεται λίγο πιο πάνω ή πίσω από αυτόν. Τέλος στα top-down shooters, η οπτική γωνία βρίσκεται από πάνω, ενώ αυτό που τα ξεχωρίζει από τα third-person, έχει να κάνει με την υγεία του χαρακτήρα. Στα thrid-person προβάλλεται με μορφή μπάρας η οποία αδειάζει ή γεμίζει. Στα top-down, η υγεία του χαρακτήρα προβάλλεται ως ένα σετ από ζωές, τις οποίες όταν χάσει είναι game over.

Παραδείγματα: **Space Invaders**, **Doom**, **Half Life**, **Bioshock**, **PlayerUnknown's Battlegrounds** κ.α.

Εικόνα 22: Half Life 2 (Fisrt-person), PUBG (Third-person) και Space Invaders (Top-down)

1.6.1.3 Fighting

Παιχνίδια πάλης που επικεντρώνονται στις περισσότερες περιπτώσεις στη μάχη χέρι με χέρι. Τα περισσότερα παιχνίδια μάχης χαρακτηρίζονται από ένα χαρακτήρα που έχει τις δικές του μοναδικές ικανότητες ή στυλ πάλης.

Παραδείγματα: **Street Fighter**, **Mortal Kombat**, **Tekken**, **Super Smash Bros**, **Injustice: Gods Among Us** κ.α.

Εικόνα 23: Street Fighter

1.6.1.4 Beat-em up

Εστιάζουν και αυτά στην πάλη, αλλά αντί να αντιμετωπίζουν έναν αντίπαλο, ο παίκτης έρχεται αντιμέτωπος με κύματα πολλαπλών εχθρών.

Παραδείγματα: **Double Dragon**, **God of War**, **Castle Crashers**, **Bayonetta**, **Devil May Cry** κ.α.

Εικόνα 24: Castle Crashers

1.6.1.5 Stealth

Σε αυτά τα παιχνίδια ο παίκτης ενθαρρύνεται να ξεπεράσει προκλήσεις, αποφεύγοντας την καταμέτωπο επίθεση, χωρίς δηλαδή να γίνει αντιληπτός από τους εχθρούς.

Παραδείγματα: **Dishonored**, **Hitman**, **Alien Isolation**, **Mark of the ninja**, **Thief** κ.α.

Εικόνα 25: Hitman

1.6.1.6 Survival

Παιχνίδια επιβίωσης, τα οποία λαμβάνουν χώρα σε ανοιχτό περιβάλλον. Ο παίκτης έχει πρόσβαση σε πόρους, όπως τροφή, όπλα, εργαλεία και καταφύγια και στόχος είναι να επιβιώσει όσο το δυνατό περισσότερο.

Παραδείγματα: **7 Days to Die**, **Minecraft**, **Subnautica**, **Dont Starve**, **Rust** κ.α.

Εικόνα 26: 7 Days to Die

1.6.1.7 Rhythm

Είναι παιχνίδια ρυθμού, που βασίζονται στη μουσική και προκαλούν τους παίκτες να συμβαδίζουν με το ρυθμό ενός τραγουδιού ή ενός soundtrack, πατώντας ένα αντίστοιχο κουμπί στο χειριστήριο μέσα σε συγκεκριμένο χρονικό περιθώριο, για να συγκεντρώσει πόντους. Θεωρούνται από τα πιο δύσκολα είδη παιχνιδιών, με εξιδεικευμένο περιεχόμενο, που δεν απευθύνεται σε όλους.

Παραδείγματα: **Beatmania, Intralism**, **Rock Band**, **Guitar Hero**, **Osu!** κ.α.

Εικόνα 27: Guitar Hero

1.6.2 Action-Adventure Games

Τα παιχνίδια δράσης-περιπέτειας περιλαμβάνουν συνήθως αποστολές ή εμπόδια που πρέπει να ξεπεραστούν, χρησιμοποιώντας αντικείμενα που συλλέγονται και εργαλεία. Εστιάζουν περισσότερο στην εξερεύνηση, την επίλυση γρίφων και την ανακάλυψη αντικειμένων, ενώ το κομμάτι της μάχης λειτουργεί υποστηρικτικά στη συνολική εμπειρία.

Στην κατηγορία των **action-adventure games**, ανήκουν οι εξής υποκατηγορίες:

1.6.2.1 Survival horror

Τα τρομαχτικά παιχνίδια επιβίωσης χρησιμοποιούν θεματολογία κατάλληλη κυρίως για ενήλικες, όπως το αίμα και το gore, ενώ το περιβάλλον είναι τρομαχτικό. Σκοπός του παίκτη είναι να επιβιώσει σε ένα περιβάλλον που περιλαμβάνει φανταστικά ή υπερφυσικά στοιχεία που είναι πολύ τρομακτικά και συχνά ενοχλητικά. Χαρακτηριστικό των παιχνιδιών αυτών είναι ο περιορισμένος αριθμός σφαιρών και όπλων που μπορεί να βρει και να χρησιμοποιήσει ο παίκτης.

Παραδείγματα: **Dead Space**, **Silent Hill**, **Amnesia**, **Resident Evil**, **Condemned** κ.α.

Εικόνα 28: Resident Evil Revelations

1.6.2.2 Metroidvania

Πήραν το όνομά τους από τα παιχνίδια τα οποία ενέπνευσαν το είδος, το Metroid και το Castlevania. Τα παιχνίδια τύπου Metroidvania είναι σαν τα βασικά **action-adventure games**, ωστόσο δεν είναι γραμμικά και συχνά απαιτούν ότι ο παίκτης δεν θα προχωρήσει, μέχρι να βρει ένα συγκεκριμένο στοιχείο ή ειδικό εργαλείο. Η αναβάθμιση του χαρακτήρα με νέα όπλα, ικανότητες και άλλες δυνάμεις, του επιτρέπει να αποκτήσει πρόσβαση σε "ειδικές" περιοχές του παιχνιδιού, αλλά όχι πριν από ένα bossfight, το οποίο είναι επίσης ένα σημαντικό χαρακτηριστικό των παιχνιδιών τύπου Metroidvania.

Παραδείγματα: **Axiom Verge**, **SteamWorld Dig**, **Dead Cells**, **Shadow Complex**, **Hollow Knight** κ.α.

Εικόνα 29: Dead Cells

1.6.3 Adventure Games

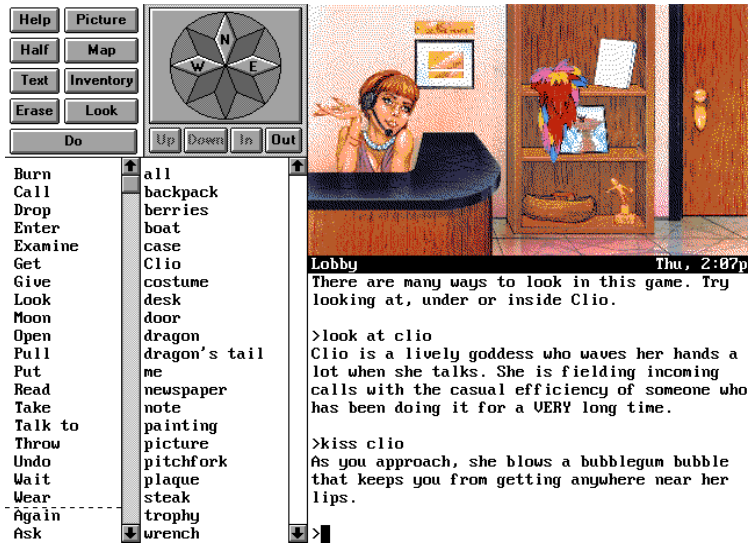
Τα παιχνίδια που ανήκουν σε αυτή την κατηγορία εστιάζουν στην πλοκή. Πολύ συχνά απαιτούν από τον παίκτη να σκεφτεί πριν πάρει μια απόφαση που μπορεί να αλλάξει την εξέλιξη της πλοκής. Ο παίκτης αλληλεπιδρά με το περιβάλλον και άλλους χαρακτήρες, επιλύει γρίφους, παζλ κτλ. Χαρακτηριστικό αυτού του είδους παιχνιδιών είναι η έλλειψη βίας.

Στην κατηγορία των **adventure games**, ανήκουν οι εξής υποκατηγορίες:

1.6.3.1 Text adventures

Όπως δηλώνει και το όνομα, το gameplay βασίζεται στο κείμενο, που σημαίνει ότι ο παίκτης χρησιμοποιεί το πληκτρολόγιό του για να εισάγει εντολές αλληλεπίδρασης, όπως "πάρε το φτυάρι", "σπάσε την πέτρα" ή "πάρε το κλειδί". Αξίζει να σημειωθεί ότι για αυτό το είδος παιχνιδιών απαιτείται αρκετός χρόνος ανάπτυξης, καθώς οι προγραμματιστές χρειάζεται να λάβουν υπόψη τους όλες τις πιθανές απαντήσεις που μπορεί να εισάγει ο παίκτης.

Παραδείγματα: **Spider and Web, The Dreamhold, Eric the Unready, Seedship, The Hitchhiker's Guide to the Galaxy** κ.α.



Εικόνα 30: Eric the Unready

1.6.3.2 Graphic adventures

Καθώς οι υπολογιστές έγιναν πιο ικανοί να δημιουργήσουν γραφικά για να υποστηρίξουν το κείμενο, τα παιχνίδια εξελίχθηκαν επίσης. Αρχικά χρησιμοποιήθηκαν απλές εικόνες για να υποστηρίξουν τα text adventure games. Αργότερα, καθώς το ποντίκι εξελίχθηκε σε χειριστήριο, τα παιχνίδια άρχισαν να αντικαθιστούν τις εντολές γραπτού κειμένου με την "point-and-click" λογική για να αλληλεπιδράσει ο παίκτης με ένα αντικείμενο στην οθόνη.

Παραδείγματα: **Grim Fandango**, **Indiana Jones and the Fate of Atlantis**, **The Riddle of the Sphinx**, **Deponia**, **Sam & Max Hit the Road** κ.α.

Εικόνα 31: The Riddle of the Sphinx

1.6.3.3 Visual novels

Ιδιαίτερα δημοφιλή στην Ιαπωνία, τα περισσότερα παιχνίδια του είδους απαιτούν την αναβάθμιση του χαρακτήρα, όπως τα στατιστικά του και τα γνωρίσματά του, ώστε να εξελιχθεί στο παιχνίδι. Συνήθως υπάρχουν περισσότερα από ένα πιθανά τέλη, τα οποία καθορίζονται από το πώς αλληλεπιδρά ο πάικτης σε καθορισμένα σημεία του παιχνιδιού.

Παραδείγματα: **Steins;Gate**, **Clannad**, **Mirror**, **Ace Academy**, **The Letter** κ.α.

Εικόνα 32: Clannad

1.6.3.4 Interactive movie

Τα παιχνίδα αυτής της υποκατηγορίας περιέχουν προκαθορισμένες σκηνές δράσης ή κινούμενων εικόνων. Ο παίκτης δεν είναι μέρος της πλοκής, συνήθως βλέπει από μια προοπτική τρίτου προσώπου και ελέγχει τη δράση κατά τη διάρκεια κομβικών σημείων της ιστορίας, για παράδειγμα, πιέζοντας πλήκτρα για να κινηθεί ο χαρακτήρας δεξιά ή αριστερά, ή πιέζοντας ένα κουμπί για να «πηδήσει» έξω από το δρόμο καθώς ένας βράχος έρχεται προς τα εμπρός. Χαρακτηριστικό αυτών των παιχνιδιών, τα quick time events, όπου ο παίκτης καλείται να πατήσει ένα πλήκτρο ή μια ακολουθία πλήκτρων μέσα σε ορισμένο μικρό χρονικό διάστημα.

Παραδείγματα: **The Walking Dead**, **Until Dawn**, **The Wolf Among Us**, **Life is Strange**, **Heavy Rain** κ.α.

Εικόνα 33: Until Dawn

1.6.4 Role-Playing Games (RPG)

Στα role-playing παιχνίδια, ο παίκτης παίρνει το ρόλο ενός χαρακτήρα και καθορίζει τη μοίρα του. Χαρακτηριστικό αυτών των παιχνιδιών είναι οι αποστολές που αναλαμβάνει να φέρει εις πέρας ο παίκτης, η διαδικασία εξέλιξης του χαρακτήρα, ανεβάζοντας στατιστικά όπως η εμπειρία (xp: experience), βελτιώνοντας τις ικανότητες του χαρακτήρα μέσα από ένα skill tree, ώστε να κατατροπώσει μεγαλύτερους εχθρούς και η αναζήτηση και διαχείριση αντικειμένων που κουβαλά και χρησιμοποιεί, όπως όπλα , πανοπλία, εργαλεία, φίλτρα κτλ.

Στην κατηγορία των **role-playing games**, ανήκουν οι εξής υποκατηγορίες:

1.6.4.1 Action RPG

Έχουν στοιχεία από action και action-adventure games. Ένα καθοριστικό χαρακτηριστικό των action RPG είναι ότι μάχη λαμβάνει χώρα σε πραγματικό χρόνο και εξαρτάται από την ταχύτητα και την ακρίβεια του παίκτη ενάντια στους εχθρούς, καθώς και από το πόσο αναβαθμισμένες είναι οι ιδιότητες του χαρακτήρα, όπως το χάρισμα και την επιδεξιότητα.

Παραδείγματα: **The Elder Scrolls**, **The Witcher**, **Mass Effect**, **Dark Souls**, **Fallout 4** κ.α.

Εικόνα 34: The Witcher

1.6.4.2 Massive Multiplayer Online Role-Playing Games (MMORPG)

Όπως δηλώνει και το όνομά τους, περιλαμβάνουν τεράστιο αριθμό παικτών οι οποίοι αλληλεπιδρούν μεταξύ τους στον ίδο κόσμο, όπου μοιράζονται τον ίδιο ή παρόμοιο στόχο. Εκατοντάδες παίκτες από όλο τον κόσμο, συνεργάζονται για να πετύχουν ένα σκοπό ή μάχονται ενάντια σε άλλους παίκτες.

Παραδείγματα: **World of Warcraft, Final Fantasy 14: A Realm Reborn, Guild Wars 2, Star Wars: The Old Republic, EVE Online** κ.α

Εικόνα 35: World of Warcraft

1.6.4.3 Rogue-like

Ο παίκτης επιλέγει ένα χαρακτήρα με συγκεκριμένα χαρακτηριστικά και εφόσον πολεμήσει εχθρούς και ξεπεράσει εμπόδια, του δίνεται η δυνατότητα να περάσει στο επόμενο επίπεδο, το οποίο δημιουργείται τυχαία (random generated). Καθώς ο παίκτης επιβιώνει και συνεχίζει από επίπεδο σε επίπεδο στο παιχνίδι, μπορεί να αναβαθμίσει τον χαρακτήρα του με αντικείμενα και όπλα. Επίσης άλλο ένα χαρακτηριστικό είναι το permanent death, δηλαδή από τη στιγμή που ο παίκτης χάσει όλες τις ζωές και ο χαρακτήρας πεθάνει, θα πρέπει να ξεκινήσει από την αρχή, από το πρώτο δηλαδή επίπεδο.

Παραδείγματα: **The Binding of Isaac, Crypt of the NecroDancer, Dead Cells**, **FTL: Faster Than Light, Spelunky** κ.α.

Εικόνα 36: Spelunky

1.6.4.4 Tactical RPG

Μοιάζουν με τα παραδοσιακά επιτραπέζια παιχνίδια, βασίζονται δηλαδή στη λογική της σειράς, όπου ο ένας χαρακτήρας κάνει την κίνησή του και στη συνέχεια ο επόμενος (turn based), σε ένα ισομετρικά χωρισμένο πλέγμα ή αρένα. Ο κάθε παίκτης έχει περιορισμένο αριθμό από στρατό, όπλα κτλ, με τα οποία πρέπει να κατατροπώσει τον εχθρό.

Παραδείγματα: **Tactical Monsters**, **The Banner Saga 3**, **Braveland**, **Forge of Gods**, **Armello** κ.α

Εικόνα 37: Tactical Monsters

1.6.5 Simulation Games

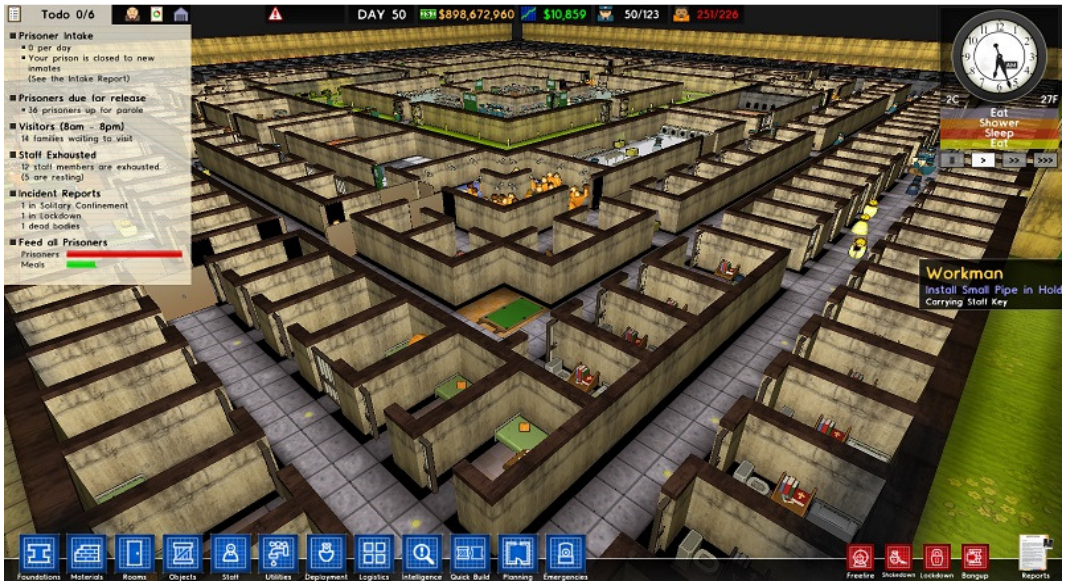
Παιχνίδια που είναι σχεδιασμένα να μιμούνται την πραγματική ζωή, προσπαθώντας να προσομοιώσουν με ακρίβεια καταστάσεις, εμπειρίες και γεγονότα της πραγματικότητας.

Στην κατηγορία των **simulation games**, ανήκουν οι εξής υποκατηγορίες:

1.6.5.1 Construction and management simulation

Ο παίκτης δημιουργεί και διαχειρίζεται κοινότητες, επιχειρήσεις ή έργα με περιορισμένους οικονομικούς πόρους.

Παραδείγματα: **Cities: Skylines**, **Prison Architect**, **Sim City**, **Planet Coaster**, **Factorio** κ.α.



Εικόνα 38: Prison Architect

1.6.5.2 Life simulation

Ο παίκτης χειρίζεται ένα χαρακτήρα, διαμορφώνοντας τα βιολογικά χαρακτηριστικά του και ελέγχοντας τις κινήσεις και τις αντιδράσεις του, καθώς και την αλληλεπίδρασή του με το περιβάλλον στο οποίο βρίσκεται. Πολλές φορές σε παιχνίδια τέτοιου είδος λένε ότι ο πάικτης αποκτά το ρόλο του Θεού.

Παραδείγματα: **The Sims**, **Goat simulator**, **Spore**, **Second Life**, **Imvu** κ.α.

Εικόνα 39: Spore

1.6.5.3 Vehicle simulation

Παιχνίδια που έχουν να κάνουν με προσομοίωση οχημάτων, από αυτοκίνητα και φορτηγά , μέχρι αεροσκάφη και διαστημόπλοια. Προσφέρουν στον παίκτη μια ρεαλιστική εμπειρία της λειτουργίας διάφορων οχημάτων. Ο παίκτης αναλαμβάνει το ρόλο του οδηγού, ενώ στα περισσότερα παιχνίδια, κινείται σε χάρτες που έχουν σχεδιαστεί βάση πραγματικών τοπίων.

Παραδείγματα: **Train simulator**, **Eurotrack simulator**, **World of Warplanes**, **City Car Driving simulator**, **Farming simulator** κ.α

Εικόνα 40: Eurotrack simulator 2

1.6.6 Strategy Games

Τα παιχνία αυτής της κατηγορίας. απαιτούν από τον παίκτη να αναπτύξει μια προσεκτική στρατηγική για να ξεπεράσει προκλήσεις. Οι αποφάσεις που παίρνει επηρεάζουν την εξέλιξη και το αποτέλεσμα του παιχνιδιού. Συνήθως υπάρχουν πολλοί τρόποι για να κερδίσει, ωστόσο πρέπει να επιλέξει τον καλύτερο δυνατό τρόπο για την κάθε περίπτωση, ανάλογα με τους πόρους που έχει διαθέσιμους. Αρκετά από τα παιχνίδια αυτού του είδους είναι turn-based και όχι realtime, με αποτέλεσμα ο παίκτης να έχει περισσότερο έλεγχο της κατάστασης και του δίνεται η δυνατότητα να προετοιμαστεί καλύτερα.

Στην κατηγορία των **strategy games**, ανήκουν οι εξής υποκατηγορίες:

1.6.6.1 4X

Τα παιχνίδια στρατιγικής 4X έχουν τέσσερις βασικούς στόχους: explore, expand, exploit, exterminate. Ο παίκτης πρέπει να εξερευνήσει στο χάρτη τις γύρω περιοχές, να επεκταθεί σε νέα εδάφη και να δημιουργήσει νέους οικισμούς, να συγκεντρώσει και να εκμεταλλευτεί τους διαθέσιμους πόρους και να εξοντώσει τους εχθρούς. Συνήθως παιχνίδια αυτού του είδους λαμβάνουν χώρα σε ιστορικές εποχές του παρελθόντος και πραγματικούς πολιτισμούς.

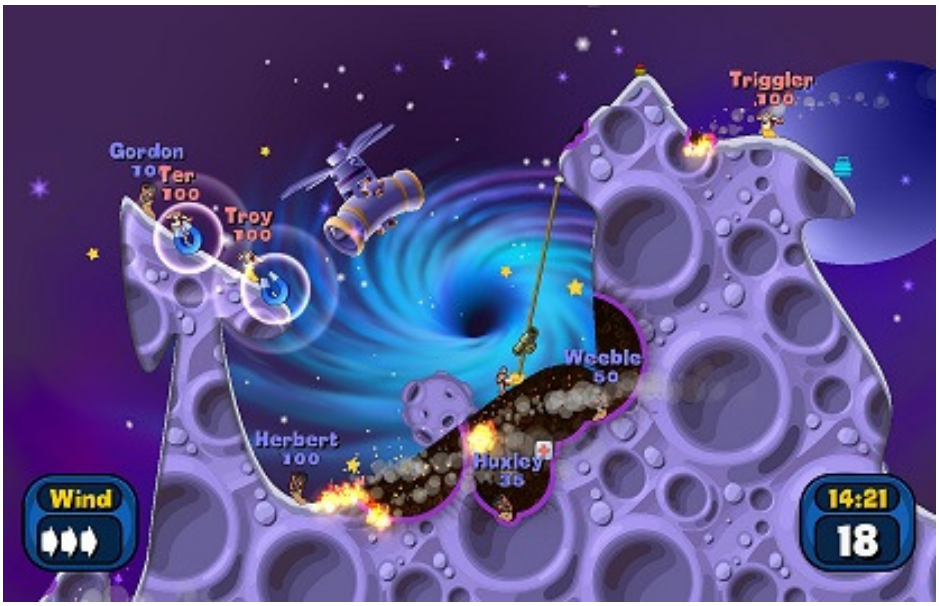
Παραδείγματα: **Stellaris**, **Sid Meier's Civilization**, **Endless Space**, **Master of Orion**, **Europa Universalis** κ.α.

Εικόνα 41: Sid Meier's Civilization

1.6.6.2 Artillery

Turn-based παιχνίδια στρατηγικής, μεταξύ δύο ή τριών παιχτών στα οποία η μάχη γίνεται μεταξύ τανκ ή στρατιωτών.

Παραδείγματα: **Gorillas**, **Worms**, **Pocket tanks**, **Warmux**, **Scroched tanks** κ.α.



Εικόνα 42: Worms Reloaded

1.6.6.3 Real-time strategy (RTS)

Τα παιχνίδια στρατηγικής σε πραγματικό χρόνο, απαιτούν από τον παίκτη να συλλέγει και να διατηρεί πόρους, ενώ ταυτόχρονα να αναπτύσσει ακόμα περισσότερους, καθώς και μαχητικές μονάδες. Επιτρέπουν ελάχιστο χρόνο στον παίκτη να αποφασίσει τι πρέπει να κάνει σε μια δεδομένη στιγμή. Απαιτείται γρήγορη λήψη αποφάσεων.

Παραδείγματα: **Starcraft**, **Age of Empires**, **Command and Conquer**, **Company of Heroes**, **Stronghold** κ.α.

Εικόνα 43: Age of Empires III

1.6.6.4 Turn-based strategy (TBS)

Αυτή η υποκατηγορία υπάρχει για να ξεχωρίσει τα παιχνίδια στρατιγικής που βασίζονται στη σειρά, με αυτά σε πραγματικό χρόνο. Στα turn-based παιχνίδια, ο παίκτης έχει περιθώριο κάποιο χρονικό διάστημα για να ετοιμαστεί και να λάβει δράση.

Παραδείγματα: **XCOM**, **Rome: Total War**, **Heroes of Might and Magic**, **Age of Wonders**, **Crusader Kings** κ.α.

Εικόνα 44: Crusader Kings

1.6.6.5 Multiplayer online battle arena (MOBA)

Αυτή η υποκατηγορία συνδυάζει action, roleplaying και real-time strategy. Στα παιχνίδια του είδους, ο παίκτης δεν χτίζει βάση και δεν συλλέγει πόρους, αλλά αντίθετα ελέγχει ένα χαρακτήρα σε μία από τις δύο ομάδες και μαζί με την ομάδα του προσπαθεί να καταστρέψει τη βάση της αντίπαλης ομάδας. Τα παιχνίδια αυτά απαιτούν όχι μόνο πολύ καλή στρατηγική αλλά και γρήγορα αντανακλαστικά.

Παραδείγματα: **Dota**, **League of Legends**, **Heroes of the Storm**, **Smite**, **Paladins** κ.α.

Εικόνα 45: Dota 2

1.6.6.6 Tower Defense

Στα παιχνίδια αυτά, ο παίκτης πρέπει να υπερασπιστεί τη βάση του ενάντια σε κύματα εχθρών, των οποίων η δυσκολία αυξάνεται. Χρησιμοποιεί πύργους με διαφορετικές ιδιότητες, όπως μαγεία, βέλη, δηλητήριο κτλ. Σκοτώνοντας εχθρούς ανταμοίβεται με χρήματα ή πόντους που χρησιμοποιεί για την αναβάθμιση των πύργων που ήδη υπάρχουν ή τη δημιουργία νέων.

Παραδείγματα: **Kingdom Rush**, **Sanctum**, **GemCraft**, **Cursed Treasure**, **Dungeon Defenders** κ.α.

Εικόνα 46: Kingdom Rush

1.6.7 Sports Games

Παιχνίδια που προσομοιώνουν αθλητικά σπορ, από ποδόσφαιρο και μπάσκετ μέχρι σκι και αγώνες αυτοκινήτων ή ακόμη και Ολυμπιακά αθλήματα. Συνήθως ο αντίπαλος είναι ο υπολογιστής, ωστόσο ο παίκτης μπορεί να αγωνιστεί ενάντια σε άλλους παίκτες online.

Στην κατηγορία των **sports games**, ανήκουν οι εξής υποκατηγορίες:

1.6.7.1 Racing

Όπως δηλώνει και το όνομα, είναι παιχνίδια αγώνων ταχύτητας με οχήματα. Ο παίκτης τρέχει ενάντια σε άλλους αντιπάλους ή ενάντια στο χρόνο.

Παραδείγματα: **Gran Turismo**, **Formula 1**, **Dirt**, **Test Drive**, **Forza Horizon** κ.α.

Εικόνα 47: Gran Turismo 6

1.6.7.2 Team Sports

Από τα πιο παλιά είδη παιχνιδιών, προσομοιώνουν ομαδικά αθλήματα. Άλλα έχουν σατυρικό και πιο ανάλαφρο χαρακτήρα και άλλα δημιουργούν όσο πιο ρεαλιστικά γίνεται την αίσθηση στον παίκτη ότι βρίσκεται στο γήπεδο ενός πραγματικού αγώνα.

Παραδείγματα: **FIFA**, **NBA**, **Madden NFL**, **Pro Evolution Soccer**, **Arch Rivals** κ.α.

Εικόνα 48: Pro Evolution Soccer 2015

1.6.8 Puzzle Games

Εστιάζουν στην επίλυση παζλ. Προκαλούν τον παίκτη να δοκιμάσει τις δυνατότητές του στην επίλυση λογικών προβλημάτων, αλληλουχιών ή αναγνώριση προτύπων. Σε οριμένα παιχνίδια, ο παίκτης έχει συγκεκριμένο χρόνο, μέσα στον οποίο θα πρέπει να τα καταφέρει, ώστε να προχωρήσει στο επόμενο επίπεδο ή στη συνέχεια του παιχνιδιού.

Στην κατηγορία των **puzzle games**, ανήκουν οι εξής υποκατηγορίες:

1.6.8.1 Logic games

Απαιτούν από τον παίκτη να επιλύσει ένα λογικό πρόβλημα ή παζλ, να περιηγηθεί σε ένα λαβύρινθο, βρίσκοντας την έξοδο, να επιλύσει ένα γρίφο όταν του δίνονται μερικά μόνο στοιχεία κτλ. Ορισμένα είναι εγκεφαλικά παιχνίδια που απαιτούν συγκέντρωση, ενώ άλλα είναι πιο απλά, καθημερινά παζλ.

Παραδείγματα: **The Room**, **The House of Da Vinci**, **Portal**, **Mahjong**, **The Talos Principle** κ.α.

Εικόνα 49: The Room

1.6.8.2 Trivia games

Όπως και τα γνωστά trivia games που γνωρίζουμε, έτσι και στα αντίστοιχα videogames , ο παίκτης πρέπει να απαντήσει σωστά στην ερώτηση πριν τελειώσει ο χρόνος.

Παραδείγματα: **Jeopardy**, **The Jackbox Pack**, **Majotori**, **Fibbage**, **Buzz! Quiz TV** κ.α.

Εικόνα 50: Jeopardy

1.6.9 Άλλα αξιοσημείωτα είδη video games

1.6.9.1 Casual games

Χαρακτηρίζονται από απλούς κανόνες και μειωμένες απαιτήσεις δεξιοτήτων. Δεν απαιτούν αφοσίωση και είναι ιδανικά ακόμη και για τον παίκτη που θέλει να παίζει περιστασιακά.

Παραδείγματα: **Candy Crash Saga**, **Plants vs Zombies**, **Stardew Valley**, **Overcooked**, **Bunny Minesweeper** κ.α

Εικόνα 51: Plants vs Zombies

1.6.9.2 Idle games

Γνωστά και ως clicker games, είναι απλοποιημένα παιχνίδια που απαιτούν ελάχιστη συμμετοχή από τον παίκτη, όπως το να κάνει επαναλαμβανόμενα κλικ σε μια εικόνα. Η ανταμοιβή που προσφέρουν μετά την ολοκλήρωση των στόχων, είναι αυτό που προσελκύει τον παίκτη.

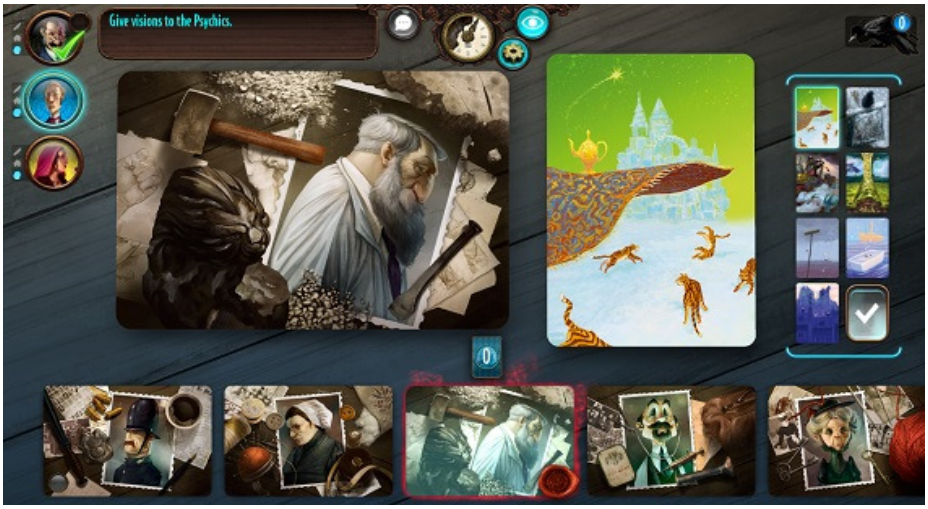
Παραδείγματα: **Clicker Heroes**, **Space Colonizers**, **AdVenture Capitalist**, **Crusaders of The Lost Idols**, **Realm Grinders** κ.α

Εικόνα 52: AdVenture Capitalist

1.6.9.3 Board/Card games

Όπως μαρτυρά και το όνομά τους, είναι παιχνίδια που παίζονται με κάρτες, είτε προσομοιώνουν επιτραπέζια παιχνίδια.

Παραδείγματα: **Magic: The Gathering**, **Pokemon Card Game**, **Hearthstone**, **100% Orange Juice**, **Mysterium** κ.α

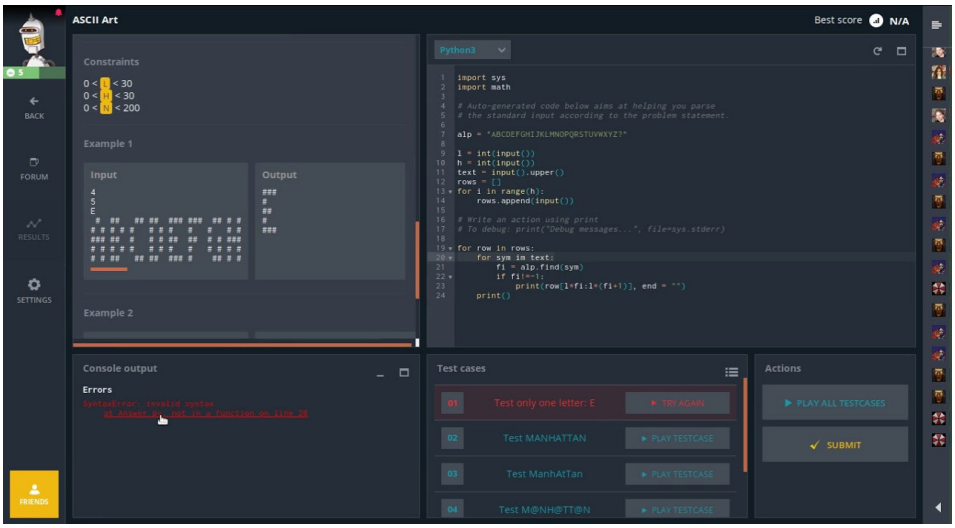


Εικόνα 53: Mysterium

1.6.9.4 Programming games

Παιχνίδια προγραμματισμού όπου ο παίκτης χρησιμοποιεί κώδικα, όπως C#, Java κτλ, για να ολοκληρώσει μια πρόκληση ή να ξεπεράσει ένα εμπόδιο.

Παραδείγματα: **Robocode**, **Codingame**, **TIS-100**, **Codehunt**, **While True: learn()** κ.α

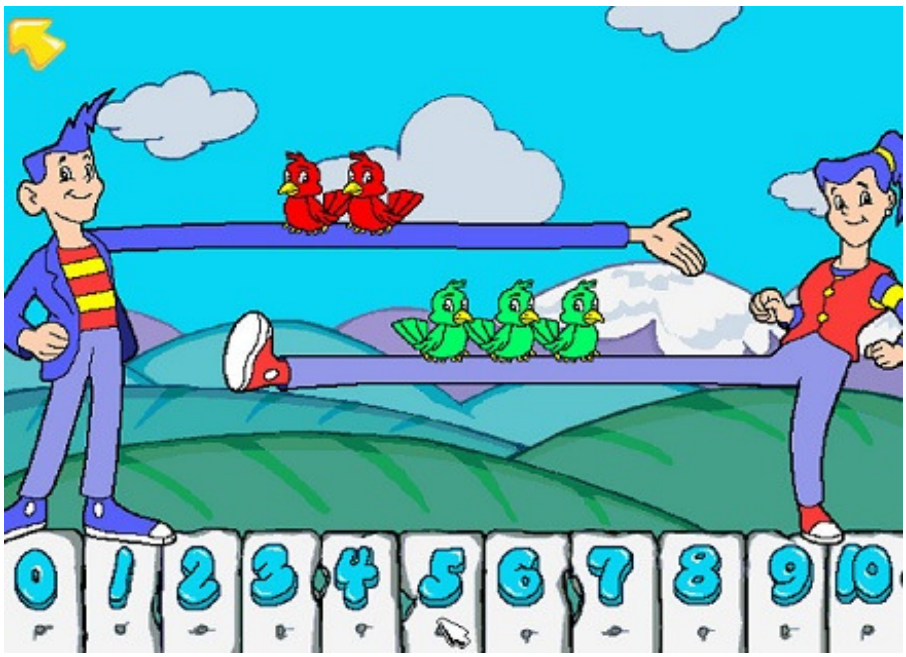


Εικόνα 54: Codingame

1.6.9.5 Educational games

Παιχνίδια που λειτουργούν κυρίως ως εκπαιδευτικό εργαλείο. Προσφέρουν γνώση μέσα από τη διασκέδαση. Συχνά, μπορούν να χρησιμοποιηθούν σε διδασκαλία μαθημάτων.

Παραδείγματα: **Human Resource Machine**, **Let's Explore the Jungle**, **Big Thinkers**, **Predynastic Egypt**, **The Counting Kingdom** κ.α



Εικόνα 55: Big Thinkers

1.7 Δημοφιλείς πλατφόρμες video games σήμερα

Για να παίξουμε ένα video game, χρειαζόμαστε μια συσκευή με μία ή περισσότερες μονάδες εισόδου (χειριστήριο, ποντίκι, πλήκτρα, οθόνη αφής κτλ) και μονάδες εξόδου εικόνας και ήχου (οθόνη, ηχεία κτλ). Υπάρχουν αρκετές πλατφόρμες που υποστηρίζουν βιντεοπαιχνίδια. Παρακάτω θα δούμε τις πιο δημοφιλείς video gaming πλατφόρμες που κυκλοφορούν την εποχή που διανύουμε.

1.7.1 Personal Computers

Οι προσωπικοί υπολογιστές είναι από τις πιο βολικές πλατφόρμες gaming σήμερα. Μπορεί να είναι είτε επιτραπέζιοι είτε φορητοί, ενώ με ειδική επιλογή και χρήση hardware, επιτρέπουν στους παίκτες να απολαμβάνουν τα βιντεοπαιχνίδια που επιθυμούν. Βασικές απαιτήσεις είναι μια αρκετά μεγάλη μνήμη RAM και μια δυνατή μονάδα επεξεργασίας γραφικών (GPU), κοινώς μια κάρτα γραφικών. Όσο πιο ισχυρή είναι η κάρτα γραφικών, τόσο καλύτερο θα είναι και το γραφικό αποτέλεσμα των βιντεοπαιχνιδιών που προβάλλεται στην οθόνη του παίκτη. Επομένως καλύτερη η video gaming εμπειρία. Απαραίτητα περιφερειακά η οθόνη, το ποντίκι και το πληκτρολόγιο. Οι προσωπικοί υπολογιστές, δίνουν τη δυνατότητα να συνδέσει ο παίκτης επιπλέον μονάδες εισόδου, όπως το joystick, που καθιστούν καλύτερη την εμπειρία του παιχνιδιού αναλόγως το είδος.

Το θετικό του να επιλέξει κάποιος τον προσωπικό υπολογιστή είναι η δυνατότητα αναβάθμισης του υλικού ανάλογα με τις απαιτήσεις των καινούργιων παιχνιδιών που κυκλοφορούν και η δυνατότητα για multiplayer gaming μεγάλου εύρους μέσω του διαδικτύου. Από την άλλη πλευρά, η συνεχής αναβάθμιση του υλικού ώστε να είναι συμβατό με τις απαιτήσεις των βιντεοπαιχνιδιών, καθιστά τον προσωπικό υπολογιστή μια ακριβή πλατφόρμα για gaming.

Η επιλογή του προσωπικού υπολογιστή για gaming, συνίσταται ιδιαιτέρως για strategy, real-time strategy και simulation παιχνίδια, ενώ αντίθετα τα sports games δεν είναι τόσο απολαυστικά σε αυτή την πλατφόρμα.

1.7.2 Video Game Consoles

Οι κονσόλες παιχνιδιών, είναι συσκευές ειδικά σχεδιασμένες για την αναπαραγωγή βιντεοπαιχνιδιών. Περιλαμβάνουν μια ή και περισσότερες συσκευές εισόδου, όπως το χεριστήριο και μια κύρια μονάδα επεξεργασίας. Δεν περιλαβάνουν μονάδες εξόδου, συνεπώς συνδέονται με οθόνες τηλεόρασης.

Σε αντίθεση με τους προσωπικούς υπολογιστές, το υλικό τους δεν αναβαθμίζεται, ωστόσο αποτελούν οικονομικότερη πλατφόρμα gaming.

Είναι ιδιαίτερα δημοφιλείς σε παίκτες που προτιμούν team sports και racing games, αλλά οι δυνατότητες για multiplayer gaming είναι περιορισμένες.

1.7.3 Mobile Gaming Platforms

Τα mobile games παίζονται κυρίως σε συσκευές κινητής τηλεφωνίας και tablets. Οι mobile πλατφόρμες παιχνιδιών έγιναν δημοφιλείς στα τέλη της περασμένης δεκαετίας, ενώ σήμερα χιλιάδες παιχνίδια είναι διαθέσιμα σε αυτές. Στις συγκεκριμένες πλατφόρμες, συνήθως τα παιχνίδια είναι διαθέσιμα δωρεάν ή έναντι ενός μικρού ποσού.

Οι παίκτες μπορούν να βρουν παιχνίδια κάθε είδους, πράγμα που καθιστά τις κινητές συσκευές, μια πολλά υποσχόμενη πλατφόρμα παιχνιδιών στο μέλλον, καθώς ήδη διαθέτουν παιχνίδια με εντυπωσιακά γραφικά.

ΜΕΡΟΣ ΠΡΩΤΟ: Η ΘΕΩΡΙΑ ΤΩΝ VIDEO GAMES

ΚΕΦΑΛΑΙΟ 2: ΔΗΜΙΟΥΡΓΩΝΤΑΣ VIDEO GAMES

Εικόνα 56: Game Design

2.1 Τι είναι το video game development

Η ανάπτυξη βιντεοπαιχνιδιών ή αλλιώς **video game development**, είναι η διαδικασία δημιουργίας ενός βιντεοπαιχνιδιού. Καλύπτει όλα τα στάδια που σχετίζονται με την παραγωγή, από την σύλληψη της ιδέας, την σχεδίαση, τον προγραμματισμό έως και την κυκλοφορία του τελικού προϊόντος στο κοινό. Συνήθως συγχέεται με τον όρο design, που είναι η σχεδίαση και αποτελεί μονάχα μέρος της διαδικασίας ανάπτυξης.

Η ανάπτυξη βιντεοπαιχνιδιών είναι μια εμπειρία πρόκληση. Το να δίνεις ζωή και να υλοποιείς μια ιδέα είναι μια δημιουργική και εθιστική διαδικασία. Πολλοί πιστεύουν ότι το να παίζεις τα παιχνίδια είναι αυτό που προσφέρει διασκέδαση, αλλά για ορισμένους ανθρώπους, το να δημιουργείς επιτυχημένα ένα περιβάλλον στο οποίο οι άλλοι διασκεδάζουν, είναι η μεγαλύτερη πηγή διασκέδασης και ικανοποίησης.

Την τελευταία δεκαετία, όλο και περισσότερες μικρές ανεξάρτητες ομάδες, που δεν ανήκουν δηλαδή σε μεγάλες και γνωστές εταιρείες ανάπτυξης βιντεοπαιχνιδιών που έχουν εδραιωθεί στην αγορά, ασχολούνται με το game development. Όλο και περισσότεροι νέοι developers επιχειρούν να δημιουργήσουν βιντεοπαιχνίδια, ορμώμενοι συνήθως από την αγάπη τους για το gaming.

Μέσα από τα μάτια του παίκτη, κάποια παιχνίδια φαίνονται εύκολα και κάποια δύσκολα. Ακόμη όμως και η ανάπτυξη ενός φαινομενικά απλού παιχνιδιού, απαιτεί σωστή και δομημένη οργάνωση από την πλευρά του developer.

2.2 Τι πρέπει να λάβει υπόψη του ένας νέος video game developer

Σκοπός ενός **video game developer** είναι να δημιουργήσει ένα βιντεοπαιχνίδι, που θα ψυχαγωγεί τους παίκτες. Ανεξάρτητα από το είδος του παιχνιδιού, το αν είναι μικρό ή μεγάλο, απλό ή σύνθετο, ο βασικός στόχος είναι η ικανοποίηση του κοινού που θα παίξει το παιχνίδι. Πρέπει λοιπόν το βιντεοπαιχνίδι να είναι διασκεδαστικό.

Διασκέδαση είναι το πρώτο πράγμα που σκέφτεται κάποιος όταν ακούει τη λέξη "video game". Όσο απλό και αν ακούγεται, το τι είναι διασκεδαστικό για τον καθένα, είναι τελείως υποκειμενικό. Όταν ένας νέος videogame developer αποφασίσει να δημιουργήσει ένα παιχνίδι, πρέπει να έχει αποδεκτεί το γεγονός, ότι σε κάποιους θα αρέσει, αλλά θα υπάρχει και ένα μέρος του κοινού που ίσως δεν εντυπωσιαστεί.

Πριν ωστόσο ξεκινήσει η διαδικασία ανάπτυξης ενός βιντεοπαινχιδιού, υπάρχουν κάποια πράγματα που πρέπει να λάβει υπόψη του ένας νέος developer, τα οποία θα τον βοηθήσουν να φτάσει στο στόχο του με μεγαλύτερη επιτυχία.

2.2.1 Το είδος του video game

Είναι σημαντικό ένας developer να γνωρίζει το είδος του βιντεοπαιχνιδιού που πρόκειται να δημιουργήσει, ώστε να ξέρει τον ανταγωνισμό, καθώς και το τι έχει προσφέρει το συγκεκριμένο είδος στους παίκτες μέχρι τη δεδομένη εκείνη στιγμή.

Το είδος πολλές φορές καθορίζει και την πλατφόρμα για την οποία θα υλοποιηθεί το βιντεοπαιχνίδι. Για παράδειγμα, τα first-person shooter games, είναι δημοφιλή κυρίως στους προσωπικούς υπολογιστές, καθώς χρησιμοποιούν τα πλήκτρα W,A,S και D για την κίνηση του χαρακτήρα. Οι παίκτες είναι εξοικιωμένοι με αυτό το χαρακτηριστικό και είναι αυτό που περιμένουν από το συγκεκριμένο είδος.

Γνωρίζοντας το είδος του video game, ο developer μπορεί να εντοπίσει τα σημεία που ίσως να χρειάζονται βελτίωση και να κάνει αλλαγές ή να εισάγει καινοτομίες, οι οποίες ωστόσο δε θα πρέπει να αποξενώσουν τους παίκτες από τις γενικές προσδοκίες και απαιτήσεις που έχουν σχετικά με το συγκεκριμένο είδος.

2.2.2 Το κοινό στο οποίο απευθύνεται

Ποιος αναμένεται να παίξει το βιντεοπαιχνίδι; Ο developer οφείλει να γνωρίζει το κοινό στο οποίο απευθύνεται. Αυτό μπορεί να είναι εύκολο, στην περίπτωση ενός video game που ανήκει σε δημοφηλείς κατηγορίες, όπως τα first-person shooter ή τα real-time strategy games. Όσο πιο πολύ παρεκκλίνει ένα βιντεοπαιχνίδι από τα συνηθισμένα, τόσο πιο δύσκολο είναι για τον developer να εντοπίσει το κοινό του.

Για να γνωρίσει το κοινό του, θα πρέπει να ψάξει είτε διαβάζοντας περιοδικά με κριτικές, είτε forums στο διαδίκτυο, ιδιαίτερα αν το βιντεοπαιχνίδι εμπεριέχει το online multiplayer στοιχείο.

Εφόσον βρει το κοινό στο οποίο απευθύνεται, πρέπει να δώσει προσοχή στις προσδοκίες του. Τι είναι αυτό που αρέσει στους παίκτες και τι αυτό που απεχθάνονται ή βαριούνται. Με αυτόν τον τρόπο, ο developer θα πάρει μία πρώτη γεύση για το τι πρέπει και τι όχι να συμπεριλάβει στο παιχνίδι του.

2.2.3 Η επίγνωση του εαυτού του

Αυτό μπορεί να ακούγεται λίγο φιλοσοφικό, αλλά για να φτιάξει ένα παιχνίδι το οποίο θα είναι διασκεδαστικό, χρειάζεται πραγματικά να γνωρίζει τον εαυτό του. Καθώς είναι δύσκολο να γνωρίζει πραγματικά το ακροατήριό του για να προβλέψει τι θα θεωρήσει διασκεδαστικό, τουλάχιστον να ξέρει τι θεωρεί ο ίδιος διασκεδαστικό σε ένα video game. Όταν παίζει ένα παιχνίδι, πρέπει να παρατηρεί και να προσπαθεί να εξηγήσει στον εαυτό του, ποιο είναι το στοιχείο εκείνο του βιντεοπαιχνιδιού που τον κάνει να χαίρεται ή να χαμογελά.

Το επόμενο βήμα είναι να απαντήσει γιατί αυτό που συνέβη ήταν τόσο διασκεδαστικό. Ο τρόπος που κινείται ο χαρακτήρας; Η ανταμοιβή όταν ξεπερνά μια πρόκληση; Η τελική νίκη; Ο διάλογος; Οι άλλοι παίκτες στο παιχνίδι; Τα παζλ; Όλα μαζί;

Πρέπει να βρει τον πυρήνα της διασκέδασης, να τον αναγνωρίσει και να τον εξετάσει πραγματικά. Έπειτα, να βρει ένα τρόπο να τον εφαρμόσει στο δικό του παιχνίδι.

Πολλοί άνθρωποι δεν μπορούν να απαντήσουν με ακρίβεια τι είναι αυτό που τους κάνει να διασκεδάζουν. Γνωρίζοντας ωστόσο τον εαυτό του κάποιος, μπορεί να εντοπίσει με αντικειμενικά κριτήρια, το λόγο για τον οποίο ένα βιντεοπαιχνίδι τον ψυχαγωγεί. Έτσι η γνώση αυτή θα συμβάλλει σημαντικά στο να αναπτύξει ένα παιχνίδι που θα προσφέρει ψυχαγωγία και στους άλλους.

2.2.4 Η πρωτοτυπία του video game

Τι είναι αυτό που θα προσελκύσει το κοινό; Ποια η ιδέα που θα ξεχωρίσει το συγκεκριμένο video game από τα υπόλοιπα του είδους; Ο developer θα πρέπει να εισάγει το στοιχείο εκείνο στο βιντεοπαιχνίδι του, που θα προσελκύσει τους παίκτες και θα τους κάνει να "γαντζωθούν" στο παιχνίδι.

Προκειμένου ο developer να ανακαλύψει το στοιχείο αυτό στο παιχνίδι του, θα πρέπει να εξετάσει το κόνσεπτ που έχει στο μυαλό του και να εντωπίσει το στοιχείο που το καθιστά μοναδικό. Έπειτα να απαντήσει στις εξής ερωτήσεις:

- Είναι όντως μοναδικό; Ή μήπως έχει υλοποιηθεί από κάποιον άλλο στο παρελθόν;
- Είναι πραγματικά διασκεδαστικό και θα παραμείνει καθ'όλη τη διάρκεια του gameplay;
- Το κοινό στο οποίο απευθύνεται είναι αρκετά μεγάλο, ώστε να αξίζει η υλοποίηση του συγκεκριμένου project;
- Είναι τεχνικά υλοποιήσιμο;

Το στοιχείο αυτό δεν αφορά απαραίτητα το κομμάτι του gameplay. Μπορεί να είναι το γραφικό στυλ, η μουσική, η πλοκή κτλ. Υπάρχουν πολλά πράγματα που μπορούν να κινήσουν το ενδιαφέρον του κοινού. Ο developer πρέπει να εντοπίσει ποιο είναι αυτό που κάνει το δικό του παιχνίδι να πρωτοστατεί και να επικεντρωθεί σε αυτό.

2.2.5 Η πρώτη εντύπωση

Η πρώτη εντύπωση που δίνει το video game στον παίκτη είναι πολύ σημαντική. Δέκα λεπτά αρκούν για να αποφασίσει αν του αρέσει ή αν θα το παρατήσει και δεν θα ξανασχοληθεί με αυτό.

Ο παίκτης όταν ξεκινά ένα καινούργιο παιχνίδι, εισέρχεται σε ένα νέο, άγνωστο κόσμο. Έχει μονάχα μια αφηρημένη ιδέα για το τι συμβαίνει. Αν δεν υπάρχει κάποιο στοιχείο να του τραβήξει το ενδιαφέρον ή να τον βοηθήσει να κατανοήσει το πώς πρέπει να κινηθεί στο παιχνίδι, είναι πολύ πιθανό να το παρατήσει, μιας και δεν έχει προλάβει να επενδύσει συναισθηματικά στο παιχνίδι, δεν υπάρχει κάτι που να τον δεσμεύει. Πολλές φορές κάποιος σταματά να παίζει ένα παιχνίδι από τα πρώτα λεπτά, επειδή δεν μπορεί να κατανοήσει τι πρέπει να κάνει.

Ο developer, πρέπει να εισάγει στο παιχνίδι του το στοιχείο εκείνο, που από τα πρώτα λεπτά θα μαγνητίσει τον παίκτη και παράλληλα θα τον "εκπαιδεύσει" για το τι πρέπει να κάνει.

Από την άλλη, αν ο παίκτης έχει παίξει στο παρελθόν παρόμοια παιχνίδια του είδους, τότε θα είναι εξοικιωμένος με τους βασικούς χειρισμούς και επομένως πιο εύκολο να προσαρμοστεί στο περιβάλλον του βιντεοπαιχνιδιού.

2.2.6 Ο ρόλος του παίκτη

Όλα τα παιχνίδια αφορούν την εκπλήρωση επιθυμιών. Όταν ο παίκτης παίζει ένα video game, μεταφέρεται στον κόσμο αυτό με ένα φανταστικό σενάριο, που ίσωσνα ήθελε να ζήσει στην πραγματική ζωή. Ένα σενάριο στο οποίο μπορεί να είναι βασιλιάς, δυνατός στρατιώτης, διάσημος αθλητής, ξωτικό με μαγικές δυνάμεις κτλ.

Όταν ο παίκτης νιώθει ότι έχει δύναμη με την οποία ξεπερνά δυσκολίες και προκλήσεις, τότε αρχίζει η διασκέδαση. Έτσι ο developer, πρέπει να δημιουργήσει ένα περιβάλλον στο παιχνίδι του, όπου ο παίκτης θα αναλάβει ένα ρόλο που θα τον κάνει να νιώσει δυνατό. Είτε με την έννοια της σωματικής δύναμης του χαρακτήρα, είτε με τον να έχει ο παίκτης τον έλεγχο μιας κατάστασης.

Ο developer μπορεί να σχεδιάσει ένα κόσμο στο παιχνίδι του, όπου οι προκλήσεις δε θα είναι ούτε εύκολες, αλλά και ούτε ακατόρθωτες για το μέσο παίκτη. Πρέπει ο παίκτης να βιώσει την πρόκληση, αλλά να μπορεί και να την ξεπεράσει με επιτυχία.

2.2.7 Ο σεβασμός στον παίκτη

Κάθε developer οφείλει να σέβεται τους παίκτες του βιντεοπαιχνιδιού του και να είναι ευγνώμων. Ο developer μπορεί να δείξει τον σεβασμό του απέναντι στον παίκτη μέσω του παιχνιδιού του με πολλούς τρόπους, όπως:

- Επιτρέποντάς του να αποχωρήσει από το παιχνίδι οποιαδήποτε στιγμή, χωρίς να χάνει την πρόοδο του.
- Δίνοντάς του την δυνατότητα να επιλέξει την εμφάνιση του χαρακτήρα του και την ελευθερία των κινήσεων ως προς το πώς θα αντιμετωπίσει τις προκλήσεις.
- Εισάγοντας ξεκάθαρους κανόνες με συνοχή, ώστε ο παίκτης να γνωρίζει τις συνέπειες σε περίπτωση αποτυχίας, καθώς και η άμεση δυνατότητα να προσπαθήσει ξανά κ.α.

Τέλος, ο developer πρέπει να θυμάται ότι ο παίκτης δεν είναι εχθρός του. Ακόμη και μια άσχημη κριτική για το video game μπορεί να λειτουργήσει θετικά στη βελτίωση του παιχνιδιού.

2.2.8 Η αγάπη για το video game

Για να δημιουργήσει ένας developer ένα καλό video game, θα πρέπει να του αρέσουν τα βιντεοπαιχνίδια, να αγαπά την ιδέα ενός video game, να αγαπά το ίδιο το video game.

Αν δεν του αρέσει αυτό που κάνει, αυτό θα επηρεάσει την ανάπτυξη του παιχνιδιού και θα φανεί στο τελικό αποτέλεσμα, οδηγώντας σε χαμηλή απήχηση στο κοινό. Σε περίπτωση που για τον οποιοδήποτε λόγο δεν είναι ικανοποιημένος ο ίδιος από το παιχνίδι που δημιουργεί, τότε είναι σχεδόν σίγουρο ότι δεν θα αρέσει και στους παίκτες.

Πέρα όμως από την αγάπη για το video game, θα πρέπει να αγαπά και ολόκληρη την διαδικασία ανάπτυξης. Να βλέπει κάθε γραμμή κώδικα και στάδιο σχεδιασμού ως πρόκληση και να μην απογοητευτεί όταν έρθει αντιμέτωπος με εμπόδια ή κάποιο λάθος. Η ανάπτυξη ενός video game, πρέπει να αντιμετωπίζεται ως μια δημιουργική δραστηριότητα και όχι ως μια δουλειά και μόνο.

2.2.9 Το design document

Ο developer, αφότου έχει συλλάβει την ιδέα για το video game που θέλει να αναπτύξει, είναι χρήσιμο να αρχίσει να κρατά σημειώσεις. Το design document είναι στην ουσία ένα οργανόγραμμα. Βοηθά τον developer να θυμάται τι πρέπει να κάνει, καθώς επίσης λειτουργεί και ως ημερολόγιο, ώστε ανά πάσα στιγμή να μπορεί να

ελέγχει το στάδιο της διαδικασίας καθώς και τι έχει υλοποιηθεί μέχρι τη δεδομένη στιγμή. Στην περίπτωση ομάδας από developers, ενημερώνει και συντονίζει τα υπόλοιπα μέλη για το στάδιο της ανάπτυξης και τις εργασίες που πρέπει να γίνουν.

Στο design document, ο developer σημειώνει αρχικά τις βασικές πληροφορίες του video game που πρόκειται να δημιουργήσει. Αυτές οι πληροφορίες αφορούν τον τίτλο, το είδος του παιχνιδιού, την πλατφόρμα για την οποία θα υλοποιηθεί, μια γενική περιγραφή της μηχανικής του παιχνιδιού, την πλοκή, το προφίλ του χαρακτήρα, το περιβάλλον στο οποίο θα λαμβάνει χώρα το βιντεοπαιχνίδι κ.α.

Στη συνέχεια, ξεδιπλώνει με λεπτομέρειες τα παραπάνω κομμάτια του βιντεοπαιχνιδιού, περιγράφει αναλυτικά το κάθε στάδιο υλοποίησης, ενώ παράλληλα ξεκινά η διαδικασία ανάπτυξης σε επίπεδο προγραμματισμού και σχεδίασης. Μπορεί ακόμα και να ζωγραφίσει σκίτσα που να απεικονίζουν το πώς φαντάζεται και επιθυμεί να λειτουργεί το παιχνίδι.

Από το σημείο αυτό και έπειτα, το design document, λειτουργεί ως καθοδηγής, χωρίς αυτό να σημαίνει ότι ο developer δεν μπορεί να κάνει αλλαγές. Ωστόσο το ιδανικό είναι η τελική μορφοποίηση του design document να ολοκληρωθεί προτού η διαδικασία ανάπτυξης του video game φτάσει στο 50%, ή να περιοριστεί σε μικρές αλλαγές που δεν θα αλλοιώσουν τον πυρήνα του βιντεοπαιχνιδιού.

2.3 Στοιχεία που δομούν ένα video game

Κάθε video game περιέχει στοιχεία τα οποία το καθιστούν μοναδικό. Στοιχεία όπως η βασική ιδέα πάνω στην οποία ξεδιπλώνεται η πλοκή, ο κεντρικός χαρακτήρας, τα γραφικά, η μουσική, ο στόχος που καλείται να επιτύχει ο παίκτης κτλ.

Υπάρχουν ωστόσο στοιχεία, τα οποία μοιράζονται όλα παιχνίδια. Στοιχεία που -αν όχι όλα, τα περισσότερα- είναι απαραίτητα για τη σωστή ανάπτυξη ενός βιντεοπαιχνιδιού.

2.3.1 Ο χώρος

Κάθε video game λαμβάνει χώρα σε κάποιο χώρο (**space**). Ο χώρος αυτός ορίζει τα διάφορα μέρη που υπάρχουν στο παιχνίδι, το μέγεθος καθώς επίσης την διάσταση (2D, 3D) και τα όρια του κόσμου στα οποία μπορεί να κινηθεί ο παίκτης.

2.3.2 Τα αντικείμενα

Ένας χώρος δίχως αντικείμενα (**objects**) είναι ένας κενός χώρος. Το video game περιλαμβάνει αντικείμενα. Χαρακτήρες, σκορ, μπάρα ζωής, ο,τιδήποτε που μπορεί να δει και αλληλεπιδράσει ο παίκτης, αποτελεί αντικείμενο του παιχνιδιού.

Κάθε αντικείμενο έχει ένα ή περισσότερα χαρακτηριστικά (attributes), τα οποία περιλαμβάνουν πληροφορίες στατικές ή δυναμικές, για την κατάσταση (state) του αντικειμένου. Η θέση του στο χώρο, το χρώμα, η τιμή που έχει μια δεδομένη στιγμή (π.χ. ταχύτητα αυτοκινήτου) κ.α.

2.3.3 Ο χειρισμός

Αφορά τα "εργαλεία" με τα οποία παίζεται το video game. Τα πλήκτρα που πρέπει να πατήσει ο παίκτης για να επιτύχει μια συγκεκριμένη δράση, κουμπιά-κλειδιά (**keys**) δηλαδή που θα χρησιμοποιήσει ως εντολές για το χειρισμό του χαρακτήρα ή τον έλεγχο μιας κατάστασης στο παιχνίδι.

2.3.4 Η δράση

Απαντά στο ερώτημα "Τι μπορεί να κάνει ο παίκτης;". Υπάρχουν δύο κατηγορίες δράσης (**action**) σε ένα video game.

Η ενεργητική δράση (operative action), η οποία αφορά τις βασικές κινήσεις του παίκτη στο παιχνίδι όπως το να προχωρά μπροστά, να κάνει άλμα, να σκύβει, να κολυμπά κτλ.

Η δεύτερη είναι η προκύπτουσα δράση (resultant action), η οποία αφορά κινήσεις που έχουν νόημα σε βάθος χρόνου στο παιχνίδι. Αφορά τον τρόπο που ο παίκτης χρησιμοποιεί την ενεργητική δράση για να πετύχει ένα στόχο. Θα μπορούσαμε να πούμε ότι είναι μια αλληλουχία από ενεργητικές δράσεις, την ο παίκτης έχει στρατηγικά επιλέξει να χρησιμοποιήσει, με συγκεκριμένο τελικό αποτέλεσμα.

2.3.5 Οι κανόνες

Όλα τα video games διέπονται από κανόνες (**rules**). Είναι θεμελιώδες στοιχείο, το οποίο ορίζει τον χώρο, τα αντικείμενα, τις δράσεις και τις συνέπειες των δράσεων, καθώς και τις μεταξύ σχέσεις όλων των στοιχείων ενός video game. Καθορίζουν τον τρόπο που ο παίκτης θα αλληλεπιδράσει με τον κόσμο του παιχνιδιού, τα δικαιώματα και τις ευθύνες του, καθώς και τον σκοπό του παιχνιδιού.

2.3.6 Τα επίπεδα και η δυσκολία

Αφορά τα επίπεδα (**levels**) ή τα κομμάτια στα οποία χωρίζεται το video game. Καθώς ο παίκτης ξεπερνά προκλήσεις στο παιχνίδι, περνά στο επόμενο επίπεδο, το οποίο μπορεί να σχετίζεται ή όχι με το προηγούμενο. Δεν είναι απαραίτητο τα επίπεδα να είναι ξεκάθαρα διακριτά από τον παίκτη, υπάρχουν ακόμα και σε μια συνεχή ροή gameplay. Λειτουργούν ως πρόοδος του παίκτη στο παιχνίδι, ενώ χαρακτηρίζονται από μια κλιμακούμενη αύξηση δυσκολίας (**difficulty**) με την οποία έρχεται αντιμέτωπος.

2.3.7 Η δεξιότητα

Το στοιχείο της δεξιότητας (**skill**) εστιάζει στον παίκτη. Κάθε video game απαιτεί από τον παίκτη να εξασκήσει ορισμένες δεξιότητες. Υπάρχουν χιλιάδες δεξιότητες που θα μπορούσαν να ζητηθούν, ωστόσο μπορούμε να τις χωρίσουμε σε τρεις κατηγορίες:

1. Φυσικές δεξιότητες (Physical skills): Περιλαμβάνουν δύναμη, αντοχή, επιδεξιότητα, συντονισμό. Απαιτούν από τον παίκτη τον αποτελεσματικό χειρισμό των πλήκτρων, ποντικιού, μοχλού κτλ.
2. Πνευματικές δεξιότητες (Mental skills): Περιλαμβάνουν μνήμη, παρατηρητικότητα, επίλυση γρίφων/παζλ και λήψη αποφάσεων.
3. Κοινωνικές δεξιότητες (Social skills): Μεταξύ άλλων, περιλαμβάνουν πρόβλεψη κινήσεων και αντίδρασης του αντιπάλου, μπέρδεμα ή αντιπερισπασμό του αντιπάλου και σωστό συντονισμό με τους συμπαίκτες.

2.3.8 Η τυχαιότητα

Η τυχαιότητα (**chance**) αφορά και επηρεάζει τις αλληλεπιδράσεις μεταξύ του χώρου, των αντικειμένων, των δράσεων, των κανόνων και των δεξιοτήτων, αλλάζοντας την εξέλιξη του παιχνιδιού και την όλη εμπειρία του gameplay. Μπορεί να είναι ένας αλγόριθμος που να κάνει τους αντιπάλους να κινούνται σε απρόβλεπτο μοτίβο, εμφάνιση τυχαίου αντικειμένου από μια κρυφή λίστα ανικειμένων κ.α. Είναι ουσιαστικό κομμάτι του video game, επειδή εισάγει την αβεβαιότητα, η αβεβαιότητα σημαίνει εκπλήξεις και οι εκπλήξεις συμβάλλουν στην διασκέδαση.

2.3.9 Οι παράγοντες της αποτυχίας

Η σκέψη της αποτυχίας στο video game, προκαλεί ένταση και ενθουσιασμό στον παίκτη, στην προσπάθειά του να την αποφύγει. Οι παράγοντες της αποτυχίας (**losing factors**) παίζουν σημαντικό ρόλο στο παιχνίδι, καθώς ενημερώνουν τον παίκτη για τις συνθήκες κάτω από τις οποίες χάνει, καθώς επίσης τον παρακινούν να προσπαθήσει περισσότερο και να συγκεντρωθεί στο στόχο.

Οι παράγοντες αυτοί μπορεί να είναι ένα ρολόι που μετρά αντίστροφα, μια παγίδα, μια μάχη με έναν πολύ ισχυρό αντίπαλο (boss fight) κ.α.

2.3.10 Η οθόνη φόρτωσης

Η οθόνη φόρτωσης (**loading screen**), είναι μια εικόνα που προβάλλεται στον παίκτη, κατά τη διάρκεια φόρτωσης των στοιχείων του video game, δηλαδή τον χώρο ή το επίπεδο (αν είναι χωρισμένο σε μέρη), τα αντικείμενα, τα γραφικά, τους ήχους κτλ. Μπορεί να δείχνει μια φωτογραφία, ένα animation, μια μπάρα με το ποσοστό ολοκλήρωσης της φόρτωσης, κείμενο με οδηγίες και tips κ.α.

2.3.11 Οι επιλογές αποθήκευσης και εγκατάλειψης

Απαραίτητο στοιχείο των video games είναι η δυνατότητα να αποθηκεύσει (**save**) την πρόοδό του ο παίκτης, ώστε να μπορεί να συνεχίσει από το σημείο που σταμάτησε, σε περίπτωση που αποφασίσει να εγκαταλείψει (**quit**) το παιχνίδι. Η δυνατότητα αποθήκευσης μπορεί να γίνει είτε οποιαδήποτε στιγμή ο παίκτης το αποφασίσει, είτε με την δημιουργία σημείων ελέγχου (check points) σε διάφορα σημεία του παιχνιδιού.

Η δυνατότητα να εγκαταλείψει το παιχνίδι οποιαδήποτε στιγμή το επιθυμεί, είναι από τα πιο σημαντικά στοιχεία ενός video game, καθώς ο παίκτης πρέπει να νιώθει ελεύθερος να αποχωρήσει όταν το επιθυμεί και όχι να "εγκλωβιστεί" στο παιχνίδι.

2.4 Οι ρόλοι σε μια development team

Ο αριθμός των ανθρώπων που απαρτίζουν μια ομάδα ανάπτυξης video game, εξαρτάται από το μέγεθος του project που έχουν αναλάβει να υλοποιήσουν. Παρ'όλα αυτά, οι ρόλοι είναι σταθεροί.

Ένας ρόλος μπορεί να ανατεθεί σε περισσότερα από ένα μέλη της ομάδας, αναλόγως του μεγέθους του project. Αντίστοιχα, ένα μέλος μπορεί να αναλάβει περισσότερους από ένα ρόλο, όπως για παράδειγμα στην περίπτωση ενός ανεξάρτητου developer, ο οποίος αναλαμβάνει και το ρόλο του designer.

Παρακάτω θα δούμε τους ρόλους που καλούνται να αναλάβουν τα μέλη μιας ομάδας ανάπτυξης βιντεοπαιχνιδιού.

2.4.1 Game Developer

Ο **game developer** ή **lead designer**, είναι το άτομο που επεξεργάζεται τις λεπτομέρειες του σχεδιασμού ενός παιχνιδιού, επιβλέπει τις δοκιμές του και αναθεωρεί το παιχνίδι, ανταποκρινόμενος στο feedback των παικτών.

Επιβλέπει το γενικό σχεδιασμό του video game, προκειμένου να βεβαιωθεί ότι το project θα ολοκληρωθεί με συνέπεια. Συχνά καλείται να εκπροσωπήσει το παιχνίδι ως project σε ομιλίες και παρουσιάσεις.

Ως επικεφαλής της ομάδας, δεν ασχολείται ο ίδιος σε μεγάλο βαθμό με το δημιουργικό κομμάτι της ανάπτυξης, αλλά παρακολουθεί και συντονίζει την δουλειά των υπόλοιπων μελών.

2.4.2 Game Designer

Ο **game designer** ορίζει και τεκμηριώνει πώς λειτουργεί το video game. Υλοποιεί δηλαδή την ιδέα, τους κεντρικούς μηχανισμούς και τους κανόνες του. Συχνά είναι το άτομο που δίνει τίτλο στο παιχνίδι και καθορίζει το θέμα του. Ασχολείται με όλες τις πτυχές του παιχνιδιού με τις οποίες αλληλεπιδρά ο παίκτης. Το μοτίβο του gameplay, τα χαρακτηριστικά του χαρακτήρα, τη ζωή του χαρακτήρα, τα όπλα, τις αναβαθμίσεις, τους εχθρούς, τα εμπόδια, τα παζλ κτλ. Ο game designer διεξάγει επίσης έρευνα και συλλέγει δεδομένα που μπορεί να χρειαστούν για την ανάπτυξη του παιχνιδιού.

Το κομμάτι του game design αφορά το μεγαλύτερο μέρος ενός videogame, γι' αυτό και τις περισσότερες φορές επιμέρους πιο εξειδικευμένοι ρόλοι ανατίθενται σε περισσότερα άτομα.

2.4.2.1 Level Designer

Ο **level designer** είναι υπεύθυνος για τον καθορισμό των μεμονομένων επιπέδων του video game. Παίρνει τα επιμέρους στοιχεία του παιχνιδιού που του παρέχουν οι άλλοι designers -τη διεπαφή χρήστη, τους βασικούς μηχανισμούς και το gameplay- και τα χρησιμοποιεί για να δημιουργήσει τα επίπεδα στα οποία θα αλληλεπιδράσει ο παίκτης. Επίσης είναι υπεύθυνος για τον συντονισμό των επιπέδων και φροντίζει ώστε το βασικό gameplay που αντιστοιχεί στον τίτλο του βιντεοπαιχνιδιού να αποδίδεται καταλλήλως σε κάθε επίπεδο.

2.4.2.2 User Interface Designer

Ο **user interface designer** είναι υπεύθυνος για τον σχεδιασμό και τη διάταξη της οθόνης σε διάφορα σημεία του video game, καθώς και για τον καθορισμό της λειτουργίας των συσκευών εισόδου. Αναλαμβάνει δηλαδή το κομμάτι της διεπαφής του χρήστη με το βιντεοπαιχνίδι. Για παράδειγμα το μενού του βιντεοπαιχνιδιού, το HUD (head-up display) στην οθόνη κατά τη διάρκεια του παιχνιδιού κ.α.

2.4.3 Programmer

Ο **programmer** είναι υπεύθυνος για την ενσωμάτωση όλων των στοιχείων που συλλέγει από τα υπόλοιπα μέλη της ομάδας και βεβαιώνεται ότι όλα λειτουργούν σωστά συνθέτοντας το τελικό αποτέλεσμα. Αναπτύσσει ή βελτιστοποιεί την μηχανική του παιχνιδιού. Κατασκευάζει ή χρησιμοποιεί τα εργαλεία για την δημιουργία του video game. Γράφει τον κώδικα που ενώνει τα κομμάτια και "δίνει ζωή" στο παιχνίδι. Ασχολείται με τη φυσική και το περιβάλλον του παιχνιδιού. Δημιουργεί τη βάση δεδομένων που αφορά χαρακτηριστικά των αντικειμένων του παιχνιδιού (hitpoints, weapon damage κ.α.) και τα στοιχεία ελέγχου (controls). Προγραμματίζει πότε πρέπει να ενεργοποιηθεί ένα ηχητικό εφέ ή ένα σενάριο στο gameplay (trap trigger, enemy spawn κ.α.)

2.4.4 Writer

Ο **writer** είναι υπεύθυνος για τη δημιουργία του εκπαιδευτικού (tutorial) ή φανταστικού περιεχομένου του παιχνιδιού. Αναλαμβάνει να γράψει το εισαγωγικό υλικό, την ιστορία πάνω στην οποία βασίζεται το σενάριο του video game, τους διαλόγους, τα cut-scenes κ.α.. Ο writer ωστόσο, σε γενικές γραμμές, δεν κάνει τεχνικό γράψιμο (design document). Αυτό το αναλαμβάνει κάποιος από τους game designers.

2.4.5 Art Director

Ο **art director** ή **lead artist**, διαχειρίζεται την παραγωγή όλων των οπτικών γραφικών στο video game. Πολλά γραφικά στοιχεία του παιχνιδιού δημιουργούνται από τους game designers, επιβλέπονται από τον developer και στη συνέχεια αποκτούν την τελική τους μορφή από τον art director, ο οποίος σχεδιάζει τα οπτικά γραφικά (visual art), τα οποία μπορεί να είναι και 3D μοντέλα. Τα σχέδια αυτά στη συνέχια "κουμπώνουν" πάνω στα αντικείμενα, όπως η εμφάνιση του χαρακτήρα, των όπλων, των εχθρων, του περιβάλλοντος, καθώς και οι υφές των αντικειμένων κτλ. Επιπλέον αναλαμβάνει να σχεδιάσει το πώς θα φαίνεται οπτικά το μενού και η οθόνη φόρτωσης του παιχνιδιού κ.α.

Έχει στενή συνεργασία με τον game developer και μαζί συμβάλλουν στον συντονισμό των εργασιών που υλοποιούνται από τα υπόλοιπα μέλη της ομάδας.

2.4.6 Audio Director

Ο **audio director** αναλαμβάνει την παραγωγή ακουστικού υλικού για το video game. Αυτό περιλαμβάνει την μουσική, τους ήχους περιβάλλοντος (ambient sound) , τα ηχητικά εφέ, τους διαλόγους, την αφήγηση κτλ. Ο ήχος παίζει καθοριστικό ρόλο στην συνολική εμπειρία του παίκτη που αλληλεπιδρά με το παιχνίδι.

2.4.7 Game Tester

Ο **game tester**, διασφαλίζει ότι το video game λειτουργεί τεχνικά, καθώς και σύμφωνα με το προτεινόμενο σχέδιο. Αυτό περιλαμβάνει δοκιμή όλων των στοιχείων που περιλαμβάνονται στον κόσμο του παιχνιδιού, της συμβατότητας κτλ.

Σκοπός είναι ο εντοπισμός και σημείωση τυχόν σφαλμάτων (bugs) και η ενημέρωση του game developer, ο οποίος με τη σειρά του θα αναθέσει την διόρθωση στα αρμόδια άτομα της ομάδας.

Ο ρόλος του game tester είναι πολύ σημαντικός. Επομένως, συνήθως ανατίθεται σε έμπειρα άτομα με τεχνογνωσία πληροφορικής και αναλυτική ικανότητα, κριτική δεξιότητα και αντοχή, μιας και αποτελεί μια πολύωρη διαδικασία που επαναλαμβάνεται αρκετές φορές.

2.5 Τα στάδια ανάπτυξης ενός video game

Η ανάπτυξη ενός video game, είναι μια διαδικασία ανάπτυξης λογισμικού, καθώς το βιντεοπαιχνίδι είναι στην ουσία ένα λογισμικό με διεπαφή χρήστη (παίκτη), οπτικά γραφικά, ήχο και αλληλεπιδράσεις. Ακολουθεί μια μεθοδολογία ανάπτυξης, της οποίας η μορφή μπορεί να διαφέρει αναλόγως την ομάδα και το project.

Η ανάπτυξη βιντεοπαιχνιδιού, είναι μια διαδικασία με επαναλαμβανόμενες φάσεις δοκιμών, αναθεώρησης και επανασχεδιασμού. Παρακάτω θα περιγράψουμε τα στάδια στα οποία διακρίνεται.

2.5.1 Απαιτήσεις (Requirements)

Στο αρχικό στάδιο καθορίζεται ο στόχος που πρέπει να επιτευχθεί από την ομάδα, οι διαθέσιμοι οικονομικοί πόροι (budget) και η ημερομηνία παράδοσης (deadline). Αποφασίζεται το είδος του παιχνιδιού και το κοινό στο οποίο θέλουν να απευθύνεται. Γίνεται αναφορά στη γενική ιδέα που έχει ο developer στο μυαλό του για το video game.

2.5.2 Προδιαγραφές (Specifications)

Σε αυτό το στάδιο, συζητείται με περισσότερες λεπτομέρειες, το τι πρέπει να κάνει το video game, ενώ συντάσσεται και το design document. Όπως είδαμε στην υποενότητα 2.2.9 του κεφαλαίου, στο design document καταγράφονται όλες οι πληροφορίες του παιχνιδιού (τίτλος, είδος κτλ) και τα στοιχεία που θα περιλαμβάνει το παιχνίδι (περιβάλλον, χαρακτήρες, αντικείμενα κτλ), την πλατφόρμα για την οποία θα υλοποιηθεί, την μηχανική του παιχνιδιού, το σενάριο, τις επιλογές και ρυθμίσεις, το ρόλο του παίκτη και γενικά όλα τα επιμέρους κομμάτια που θα χρησιμοποιηθούν για την δημιουργία του παιχνιδιού (γραφικά, μουσική/ ήχος, κείμενο κτλ). Επιπλέον περιγράφονται αναλυτικά οι αρμοδιότητες που θα αναλάβει κάθε μέλος της ομάδας.

Να σημειώθεί, ότι το design document μπορεί να υποστεί αλλαγές στα πρώτα στάδια της ανάπτυξης του video game, σε περίπτωση που ο developer αποφασίσει να αλλάξει στοιχεία του παιχνιδιού -που όμως δεν αφορούν τον πυρήνα. όπως το είδος, την κεντρική πλοκή ή τη μηχανική- ή σε περίπτωση που εντοπίσει ότι κάτι είναι αδύνατο να υλοποιηθεί τεχνικά έτσι όπως το είχε φανταστεί.

2.5.3 Σχεδιασμός (Design)

Στο στάδιο αυτό ξεκινά πρακτικά η δημιουργία του video game. Κάθε μέλος της ομάδας αναλαμβάνει το κομμάτι εκείνο που του αναλογεί. Οι designers σχεδιάζουν το περιβάλλον του παιχνιδιού, τον τρόπο με τον οποίο ο παίκτης θα αλληλεπιδρά με αυτόν, το gameplay, τα επίπεδα του παιχνιδιού, τους χαρακτήρες και όλα τα αντικείμενα που θα περιλαμβάνει το video game, την διεπαφή με το χρήστη, τους κανόνες, τις συνθήκες επιτυχίας ή αποτυχίας κτλ. Οι writers γράφουν τους διαλόγους, τις οδηγίες, το σενάριο και ο,τιδήποτε σχετίζεται με κείμενο που χρησιμοποείται στο παιχνίδι. Αντίστοιχα οι art και sound directors, αναλαμβάνουν σε επικοινωνία με τους designers να δημιουργήσουν τα οπτικά γραφικά και τους ήχους αντίστοιχα, που θα εισαχθούν στο παιχνίδι για να "ντύσουν" το περιβάλλον, τα αντικείμενα και την ατμόσφαιρα.

Σε αυτό το στάδιο, πάντα σε επικοινωνία με τον developer, μπορούν να πραγματοιποιηθούν αλλαγές σε σχέση με το design document, τις οποίες ο ίδιος επιβλέπει και εγκρίνει ή απορρίπτει.

2.5.4 Υλοποίηση (Implementation)

Το στάδιο αυτό αφορά τη δημιουργία κώδικα από τους programmers, ώστε να υλοποιηθούν όλα όσα έχουν σχεδιάσει οι designers. Το παιχνίδι αρχίζει να παίρνει πραγματική μορφή. Ιδανικά, πριν η ανάπτυξη του video game φτάσει στο στάδιο της υλοποίησης, πρέπει το design document να έχει "κλειδώσει", να μην υποστεί δηλαδή περαιτέρω αλλαγές. Μια κομβική αλλαγή στο παιχνίδι, σε αυτό το στάδιο, θα γυρνούσε πολλά βήματα πίσω την διαδικασία ανάπτυξης, θα καθυστερούσε χρονικά την ολοκλήρωση του παιχνιδιού, ρισκάροντας το ενδεχόμενο να μην είναι παραδόσιμο στην προκαθορισμένη ημερομηνία, ενώ τέλος θα αύξανε κατά πολύ το κόστος παραγωγής. Βέβαια, μικρές δευτερεύουσας σημασίας αλλαγές μπορούν ακόμα να γίνουν, που όμως να μην αλλάζουν τον πυρήνα του παιχνιδιού.

2.5.5 Ενσωμάτωση (Integration)

Σε αυτό το στάδιο, ενώνονται όλα τα επιμέρους κομμάτια του παιχνιδιού. Το video game πλέον διατίθεται στους testers για έλεγχο. Τυχόν λάθη που εντοπιστούν, σημειώνονται και ενημερώνονται τα αρμόδια μέλη για τη διόρθωσή τους, Στη συνέχεια το παιχνίδι υποβάλλεται ξενά σε testing, με την διαδικασία να επαναλαμβάνεται έως ότου η ομάδα βεβαιωθεί ότι δεν υπάρχουν λάθη που θα εμποδίσουν τους παίκτες να αλληλεπιδράσουν με το παιχνιδι σε οποιοδήποτε σημείο του gameplay.

2.5.6 Ολοκλήρωση (Completion)

Το video game πλέον είναι έτοιμο για κυκλοφορία στην αγορά. Από δω και πέρα μπορούν να πραγματοποιηθούν λειτουργίες συντήρησης ή αναβάθμισης (περισσότερα επίπεδα, νέοι εχθροί κτλ) ανάλογα με το είδος του παιχνιδιού και το feedback των παικτών.

ΜΕΡΟΣ ΔΕΥΤΕΡΟ: ΑΠΟ ΤΗ ΘΕΩΡΙΑ ΣΤΗΝ ΠΡΑΞΗ

ΚΕΦΑΛΑΙΟ 3: Η ΒΙΒΛΙΟΘΗΚΗ SDL

Εικόνα 57: SDL

3.1 Τι είναι η SDL

Η **Simple DirectMedia Layer (SDL)** είναι μια cross-platform βιβλιοθήκη ανάπτυξης λογισμικού, σχεδιασμένη να προσφέρει χαμηλού επιπέδου πρόσβαση σε ήχο, πληκτρολόγιο, ποντίκι, joystick και υλικό γραφικών μέσω OpenGL και Direct3D. Οι developers λογισμικού τη χρησιμοποιούν για να γράψουν υψηλής απόδοσης videogames και άλλες πολυμεσικές εφαρμογές που μπορούν να "τρέξουν" σε πολλά λειτουργικά συστήματα, όπως Android, iOS, Linux, Mac OS X και Windows. Χρησιμοποιείται επίσης από λογισμικό αναπαραγωγής βίντεο, εξομοιωτές και δημοφιλή παιχνίδια που συμπεριλαμβάνονται στον award winning catalog της Valve, καθώς και πολλά παιχνίδια στο Humble Bundle.

Η βιβλιοθήκη είναι εσωτερικά γραμμένη σε γλώσσα προγραμματισμού C, λειτουργεί εξ'ορισμού με C++ και παρέχει API σε C, ενώ υπάρχουν διαθέσιμα bindings για διάφορες άλλες γλώσσες, όπως την C# και την Python.

Μια διεπαφή προγραμματισμού εφαρμογών (**API, application programming interface**) είναι ένα σύνολο ορισμών υπορουτίνας, πρωτοκόλλων επικοινωνίας και εργαλείων για την δημιουργία λογισμικού. Σε γενικές γραμμές, πρόκειται για ένα σύνολο καθορισμένων μεθόδων επικοινωνίας μεταξύ των διαφόρων συνιστωσών. Μία καλή API διευκολύνει την ανάπτυξη ενός προγράμματος υπολογιστή, παρέχοντας όλα τα δομικά στοιχεία, τα οποία στη συνέχεια συνθέτει ο προγραμματιστής.

Κάθε πλατφόρμα έχει το δικό της τρόπο δημιουργίας και εμφάνισης παραθύρων και γραφικών, χειρισμού εισόδου χρήστη και πρόσβασης σε οποιοδήποτε υλικό χαμηλού επιπέδου, καθεμία με τις δικές της ιδιαιτερότητες και συντακτικό. Η βιβλιοθήκη SDL παρέχει έναν ομοιόμορφο τρόπο πρόσβασης σε αυτά τα ειδικά χαρακτηριστικά πλατφόρμας. Αυτή η ομοιομορφία διευκολύνει τον developer, δίνοντάς του τη δυνατότητα να αφιερώσει περισσότερο χρόνο για να τροποποιήσει το ίδιο το παιχνίδι, παρά να ανησυχεί για το πώς θα το κάνει συμβατό για μια συγκεκριμένη πλατφόρμα.

Ο προγραμματισμός video games μπορεί να είναι αρκετά δύσκολος και έχοντας μια βιβλιοθήκη όπως την SDL, καθιστά το παιχνίδι σε λειτουργία σχετικά γρήγορα. Η δυνατότητα να γράψει κάποιος ένα παιχνίδι στα Windows και στη συνέχεια να το κάνει compile σε OSX ή Linux με ελάχιστες ή και καθόλου αλλαγές στον κώδικα είναι εξαιρετικά ισχυρή και ιδανική για προγραμματιστές που θέλουν να στοχεύσουν όσο το δυνατόν σε περισσότερες πλατφόρμες παιχνιδιών.

Η SDL καθιστά αυτό το είδος cross-platform ανάπτυξης εύκολο. Παρόλο που η SDL είναι άκρως αποτελεσματική για την cross-platform ανάπτυξη λογισμικού, είναι επίσης μια εξαιρετική επιλογή για τη δημιουργία ενός παιχνιδιού για μία μόνο πλατφόρμα, λόγω της ευκολίας χρήσης και της αφθονίας των χαρακτηριστικών. Η SDL έχει μια μεγάλη βάση χρηστών, ενώ ενημερώνεται και συντηρείται ενεργά. Προσφέρει ένα εξαιρετικό περιβάλλον για να ξεκινήσει κάποιος να ασχολείται με την

ανάπτυξη παιχνιδιών, επιτρέποντάς του να εστιάσει στο ίδιο το παιχνίδι και να αγνοήσει την πλατφόρμα για την οποία το αναπτύσσει, έως ότου είναι απολύτως απαραίτητο.

Μια κοινή παρανόηση είναι ότι η SDL είναι game engine, αλλά αυτό δεν είναι αλήθεια. Ωστόσο, η βιβλιοθήκη είναι κατάλληλη για την άμεση δημιουργία παιχνιδιών, ενώ μπορεί να χρησιμοποιηθεί έμμεσα από engines που είναι κατασκευασμένες πάνω σε αυτή.

3.2 Αρχιτεκτονική λογισμικού της SDL

Η βιβλιοθήκη SDL είναι ένα "περιτύλιγμα" (wrapper library) γύρω από τις λειτουργίες που σχετίζονται με το λειτουργικό σύστημα τις οποίες χρειάζεται να έχει πρόσβαση το παιχνίδι. Ο μόνος σκοπός της SDL είναι να παρέχει ένα κοινό framework για την πρόσβαση σε αυτές τις λειτουργίες για πολλαπλά λειτουργικά συστήματα (cross-platform). Η SDL παρέχει υποστήριξη για λειτουργίες 2D pixel, ήχο, πρόσβαση σε αρχεία, διαχείριση συμβάντων (events), χρονισμό και threading. Συχνά χρησιμοποιείται για να συμπληρώσει την OpenGL, ρυθμίζοντας τη γραφική έξοδο και παρέχοντας είσοδο ποντικιού και πληκτρολογίου, αφού η OpenGL περιλαμβάνει μόνο rendering.

Ένα videogame που χρησιμοποιεί **Simple DirectMedia Layer** δεν θα εκτελεστεί αυτόματα σε κάθε λειτουργικό σύστημα, πρέπει να εφαρμοστούν περαιτέρω προσαρμογές. Αυτές ωστόσο μειώνονται στο ελάχιστο, καθώς η SDL περιέχει επίσης μερικά abstraction APIs για συχνές λειτουργίες που προσφέρονται από ένα λειτουργικό σύστημα.

Μια wrapper library όπως η SDL, είναι μια συλλογή από υπορουτίνες (subroutines) ή κλάσεις που χρησιμοποιούνται για την ανάπτυξη λογισμικού. Οι βιβλιοθήκες εκθέτουν τις διεπαφές που χρησιμοποιούν οι clients της βιβλιοθήκης για την εκτέλεση ρουτινών βιβλιοθηκών. Αποτελούνται από ένα λεπτό στρώμα κώδικα που μεταφράζει την υπάρχουσα διεπαφή μιας βιβλιοθήκης σε μια συμβατή διεπαφή. Αυτό γίνεται για διάφορους λόγους:

- Βελτιώνει μια ανεπαρκώς σχεδιασμένη ή περίπλοκη διεπαφή
- Επιτρέπει στον κώδικα να συνεργαστεί, ο οποίος διαφορετικά δεν μπορεί (π.χ. ασύμβατες μορφές δεδομένων)
- Ενεργοποιεί τη διαλειτουργικότητα μεταξύ γλωσσών και/ή χρόνου εκτέλεσης.

Το συντακτικό της SDL βασίζεται σε συναρτήσεις (functions): όλες οι λειτουργίες που πραγματοποιούνται στην SDL γίνονται με τη διέλευση παραμέτρων σε

subroutines. Ειδικές δομές χρησιμοποιούνται επίσης για την αποθήκευση των συγκεκριμένων πληροφοριών που χρειάζεται να χειρίζεται η SDL. Οι functions της SDL κατηγοριοποιούνται σε αρκετά διαφορετικά υποσυστήματα (subsystems).

Basics

- Initialization and Shutdown (SDL.h)
- Configuration Variables (SDL_hints.h)
- Error Handling (SDL_error.h)
- Log Handling (SDL_log.h)
- Assertions (SDL_assert.h)
- Querying SDL Version (SDL_version.h)

Video

- Display and Window Managemen. (SDL_video.h) - Creates and manages multiple windows
- 2D Accelerated Rendering. (SDL_render.h)
- Pixel Formats and Conversion Routines (SDL_pixels.h)
- Rectangle Functions (SDL_rect.h)
- Surface Creation and Simple Drawing (SDL_surface.h)
- Platform-specific Window Management (SDL_syswm.h)
- Clipboard Handling (SDL_clipboard.h)
- Vulkan Support (SDL_vulkan.h)

Input Events

- Event Handling (SDL_events.h)
- Keyboard Support (SDL_keyboard.h, SDL_keycode.h, SDL_scancode.h)
- Mouse Support (SDL_mouse.h)
- Joystick Support (SDL_joystick.h)
- Game Controller Support (SDL_gamecontroller.h)

Κάθε event μπορεί να ενεργοποιηθεί ή να απενεργοποιηθεί με την function SDL_EventState ()

Force Feedback

- Force Feedback Support (SDL_haptic.h)

To Force feedback υποστηρίζεται σε Windows, Mac OS X και Linux

Audio

- Audio Device Management, Playing and Recording (SDL_audio.h)

Threads

- Thread Management (SDL_thread.h)
- Thread Synchronization Primitives (SDL_mutex.h)
- Atomic Operations (SDL_atomic.h)

Timers

- Timer Support (SDL_timer.h)

File Abstraction

- Filesystem Paths (SDL_filesystem.h)

3.3 Πλατφόρμες στις οποίες "τρέχει" η SDL

Η SDL μπορεί να τρέξει σε διάφορες πλατφόρμες:

Windows

- Χρησιμοποιεί API Win32 για απεικόνιση, αξιοποιώντας το Direct3D για hardware acceleration
- Χρησιμοποιεί DirectSound και XAudio2 για ήχο

Mac OS X

- Χρησιμοποιεί Cocoa για απεικόνιση βίντεο, αξιοποιώντας την OpenGL για hardware acceleration
- Χρησιμοποιεί Core Audio για ήχο

Linux

- Χρησιμοποιεί X11 για απεικόνιση βίντεο, αξιοποιώντας την OpenGL για hardware acceleration
- Χρησιμοποιεί API ALSA, OSS και PulseAudio για ήχο

iOS

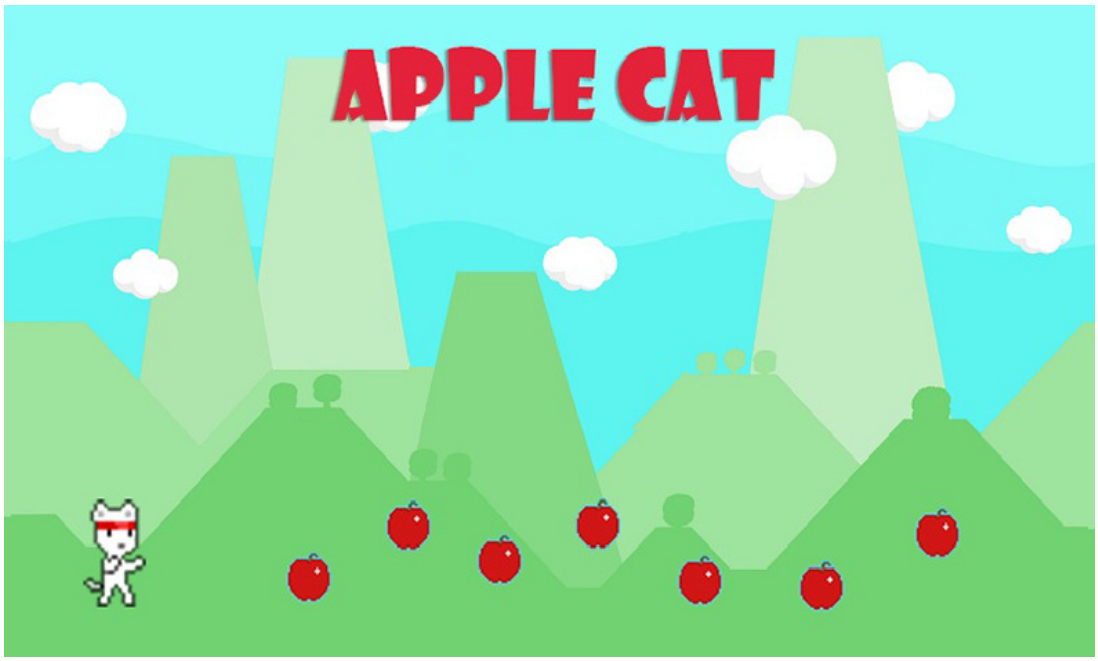
- Χρησιμοποιεί UIKit για απεικόνιση βίντεο, αξιοποιώντας την OpenGL ES 2.0 για hardware acceleration
- Χρησιμοποιεί Core Audio για ήχο

Android

- Χρησιμοποιεί JNI διασυνδέσεις για απεικόνιση βίντεο, αξιοποιώντας την OpenGL ES 1.1 και 2.0 για hardware acceleration
- Χρησιμοποιεί JNI audio callbacks για ήχο

ΜΕΡΟΣ ΔΕΥΤΕΡΟ: ΑΠΟ ΤΗ ΘΕΩΡΙΑ ΣΤΗΝ ΠΡΑΞΗ

ΚΕΦΑΛΑΙΟ 4: Η ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ "APPLE CAT"



Εικόνα 58: Apple Cat game

4.1 Γενικά χαρακτηριστικά του παιχνιδιού "APPLE CAT"

Ο τίτλος του παιχνιδιού, **APPLE CAT**, δηλώνει εξ' αρχής τους δύο πρωταγωνιστές του παιχνιδιού, την γάτα και τα μήλα. Ο χαρακτήρας του παιχνιδιού είναι μια λευκή γάτα με κόκκινη κορδέλα, η οποία πεινάει και προσπαθεί να πιάσει μερικά μήλα για να φάει. Σκοπός του παίκτη είναι να καθοδηγήσει την γάτα, βοηθώντας τη να πιάσει τα μήλα που πέφτουν από ψηλά.

Το **APPLE CAT**, είναι ένα top-down casual video game, εύκολο στην κατανόηση και το χειρισμό, ενώ απευθύνεται κυρίως σε μικρότερες ηλικίες παικτών, χωρίς αυτό να σημαίνει ότι δεν μπορεί να λειτουργήσει ψυχαγωγικά και για έναν ενήλικα.

Ο παίκτης μέσα από το συγκεκριμένο παιχνίδι, έχει τη δυνατότητα να τεστάρει τα αντανακλαστικά του, καθώς πρέπει να μετακινεί τον χαρακτήρα προς την κατεύθυνση που πέφτουν τα μήλα και να τα πιάσει, πριν αυτά αγγίξουν το έδαφος. Ιδιαίτερα, όσο προχωρά το επίπεδο, τα μήλα εμφανίζονται με μεγαλύτερη συχνότητα, κάτι που προσδίδει ένα βαθμό δυσκολίας στο παιχνίδι.

Ο παίκτης αλληλεπιδρά με το παιχνίδι με χρήση ποντικιού και πληκτρολογίου. Με το ποντίκι μπορεί να περιηγηθεί στα μενού του παιχνιδιού, ενώ η χρήση του πληκτρολογίου αφορά το gameplay. Υπάρχουν τέσσερις διαφορετικοί ήχοι στο παιχνίδι. Ένας background ήχος, ο οποίος ακούγεται όταν ο παίκτης βρίσκεται στην οθόνη του αρχικού μενού και κατά τη διάρκεια του gameplay, ένα ηχητικό εφέ όταν η γάτα πιάνει τα μήλα ή όταν αυτά αγγίζουν το έδαφος, ένα ηχητικό εφέ με παλαμάκια όταν ο παίκτης ολοκληρώνει με επιτυχία το επίπεδο και ένας ακόμη ήχος όταν ο παίκτης χάνει όλες τις ζωές του.

4.2 Περιγραφή του παιχνιδιού "APPLE CAT"

Όταν ο παίκτης πραγματοποιήσει εκκίνηση της εφαρμογής, του video game δηλαδή, βρίσκεται στην οθόνη με το μενού του παιχνιδιού, που αποτελεί και την αρχική οθόνη (Εικόνα 59). Εκεί έχει διαθέσιμες τρεις επιλογές με τη μορφή buttons. Οι επιλογές αυτές είναι Play, About και Exit. Χρησιμοποιώντας το ποντίκι, μπορεί να κάνει κλικ σε όποια από τις επιλογές αυτές θέλει.

Εάν κάνει κλικ στην επιλογή **About**, μεταφέρεται σε μια άλλη οθόνη όπου μπορεί να διαβάσει οδηγίες σχετικά με το σκοπό και το χειρισμό του παιχνιδιού (Εικόνα 60). Σκοπός, όπως αναφέραμε και στην προηγούμενη ενότητα, είναι να βοηθήσει την γάτα να πιάσει όλα τα μήλα, πριν αυτά αγγίξουν το έδαφος. Ο χειρισμός της γάτας γίνεται με τα βελάκια του πληκτρολογίου. Για να επιστρέψει στο προηγούμενο μενού, μπορεί να κάνει κλικ στην επιλογή **Back** ή πατώντας το πλήκτρο Esc.

Εάν ο παίκτης θέλει να αποχωρήσει από το παιχνίδι, μπορεί να κάνει κλικ στην επιλογή **Exit**, η οποία και θα τερματήσει την εφαρμογή. Διαφορετικά μπορεί να "εξαναγκάσει" οποιαδήποτε στιγμή το κλείσιμο του παιχνιδιού, πατώντας το X στο παράθυρο της εφαρμογής (Εικόνα 65).

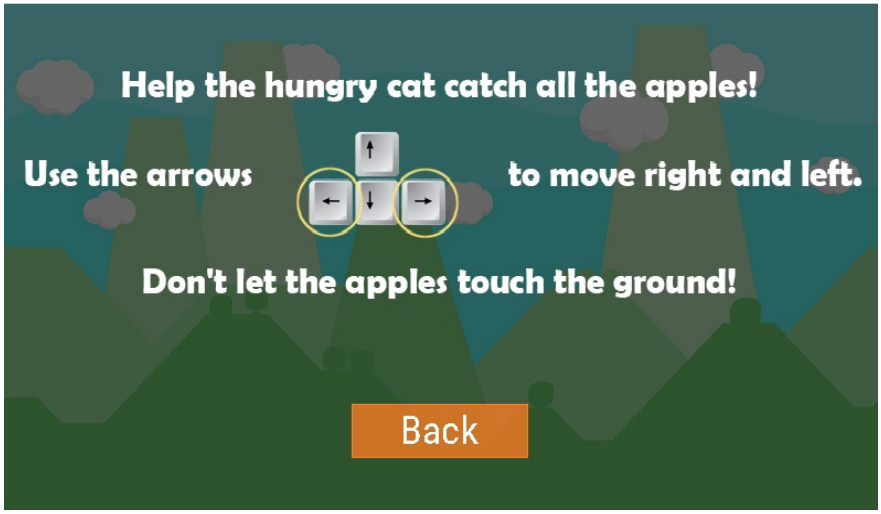
Πατώντας την επιλογή **Play**, ξεκινά το gameplay (Εικόνα 61). Στο κάτω μέρος της οθόνης, βρίσκεται ο χαρακτήρας, του οποίου την κίνηση ελέγχει ο παίκτης, είτε δεξιά είτε αριστερά. Πάνω αριστερά απεικονίζεται το Score, δείχνοντας πόσα μήλα πρέπει να μαζέψει ακόμα για να ολοκληρώσει με επιτυχία το επίπεδο, ενώ πάνω δεξιά απεικονίζονται με μορφή καρδιάς οι ζωές του χαρακτήρα που έχουν απομείνει. Από το πάνω μέρος της οθόνης και με κατεύθυνση προς τα κάτω, εμφανίζονται τα μήλα με τυχαίες θέσεις αφετηρίας, τα οποία πρέπει να πιάσει ο παίκτης. Για κάθε μήλο που πιάνει με επιτυχία, το σκορ αυξάνεται κατά ένα, ενώ για κάθε μήλο που αγγίζει το έδαφος, οι ζωές μειώνονται κατά μία. Όσο ανεβαίνει το σκορ, τα μήλα αρχίζουν να εμφανίζονται με μεγαλύτερη συχνότητα. Έτσι ο παίκτης θα πρέπει να είναι παρατηρητικός και πιο γρήγορος στις κινήσεις του.

Ο παίκτης μπορεί οποιαδήποτε στιγμή να διακόψει το παιχνίδι, πατώντας το πλήκτρο Esc, το οποίο θα πραγματοποιήσει παύση του gameplay, ενώ μεταφέρεται στην οθόνη paused (Εικόνα 62), όπου έχει διαθέσιμες τρεις επιλογές, Continue, Restart και Menu. Πατώντας **Continue** το παινχίδι συνεχίζεται από το σημείο στο οποίο το είχε αφήσει ο παίκτης. Πατώντας **Restart**, γίνεται επανεκκίνηση του επιπέδου. Τέλος, πατώντας **Menu**, ο παίκτης επιστρέφει στην αρχική οθόνη του παιχνιδιού (Εικόνα 59).

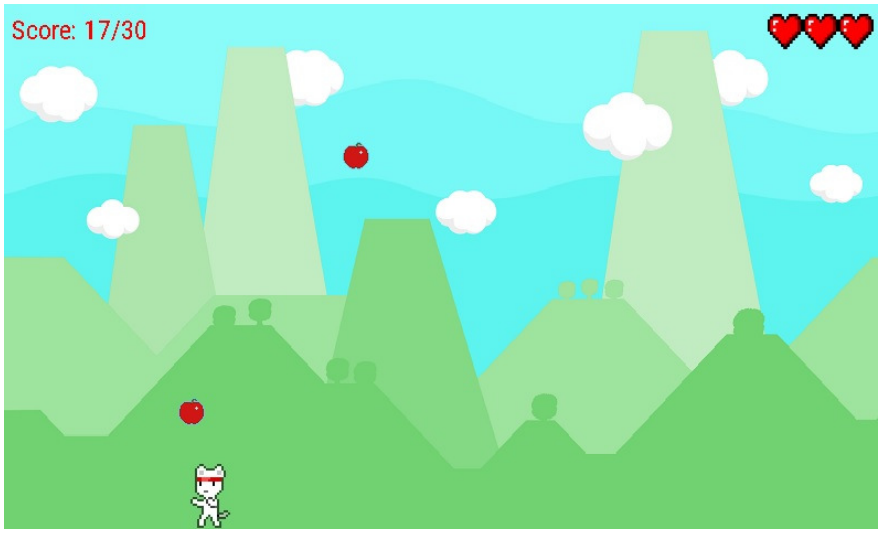
Στην περίπτωση που ο παίκτης καταφέρει με επιτυχία να μαζέψει όλα τα μήλα που απαιτούνται για την ολοκλήρωση του επιπέδου, μεταφέρεται στην οθόνη victory (Εικόνα 63), όπου επεφημείται με ένα ηχητικό εφέ (παλαμάκια) και εμφανίζονται οι επιλογές Restart και Menu.

Στην περίπτωση που ο παίκτης χάσει όλες τις ζωές, μεταφέρεται στην οθόνη defeat (Εικόνα 64), όπου ακούγεται ένα ηχητικό εφέ και εμφανίζονται οι επιλογές Restart και Menu.

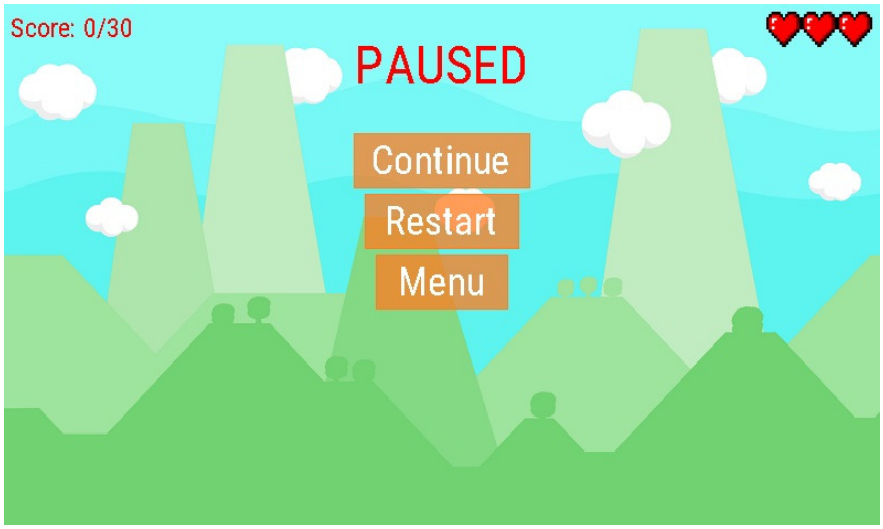
Εικόνα 59: Apple Cat menu screen



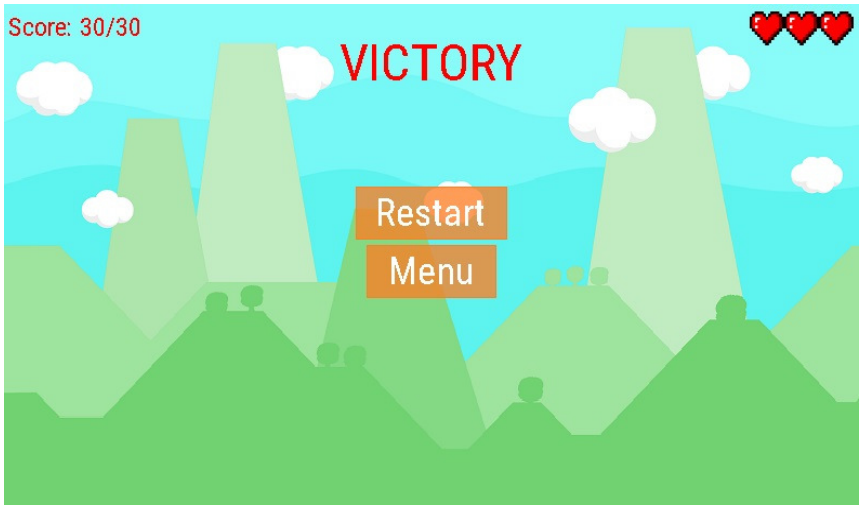
Εικόνα 60: Apple Cat about screen



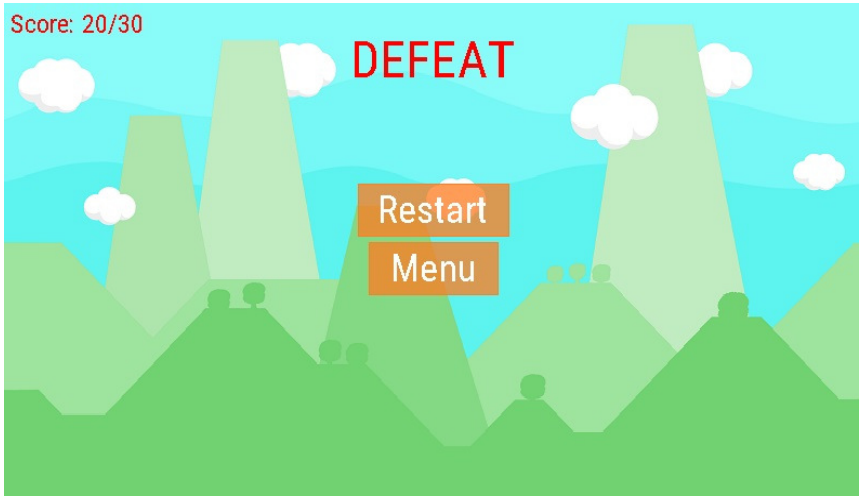
Εικόνα 61: Apple Cat gameplay



Εικόνα 62: Apple Cat paused screen



Εικόνα 63: Apple Cat victory screen



Εικόνα 64: Apple Cat defeat screen

Εικόνα 65: Apple Cat game window

4.3 Κώδικας του παιχνιδιού "APPLE CAT"

Το video game αυτό δημιουργήθηκε στο Visual Studio 2015 της Microsoft, σε γλώσσα προγραμματισμού C++, με τη χρήση της βιβλιοθήκης SDL.

Για τη δημιουργία του κώδικα, γράφτηκαν περισσότερα από ένα source files (.cpp), για τα οποία δημιουργήθηκαν τα αντίστοιχα header files (.h).

Όταν ένα project, στην προκειμένη περίπτωση το video game, είναι αρκετά μεγάλο όσο αφορά τον κώδικα, συνηθίζεται να χωρίζεται σε επιμέρους source files, για να μειώνεται ο χρόνος που χρειάζεται για το compile. Για την σύνδεση των επιμέρους αυτών αρχείων, ώστε τα κομμάτια αυτά του κώδικα να συνθέσουν το τελικό αποτέλεσμα, χρησιμοποιούνται τα header files. Τα header files (.h), δηλώνουν τα functions/classes/variables, ενώ τα αντίχτοιχα source files (.cpp), δίνουν τον ορισμό. Με απλά λόγια ένα header file δηλώνει το ποιες εξισώσεις, κλάσεις, μεταβλητές χρησιμοποιούνται στο αντίστοιχο source file, ενώ το source file δηλώνει το πώς χρησιμοποιούνται, δηλαδή περιέχει τον κύριο κώδικα. Έτσι αν για παράδειγμα το Alpha.cpp αρχείο χρειάζεται να καλέσει μια function από το Beta.cpp αρχείο, τότε μπορεί απλά να κάνει #include το Beta.h. Σημαντικό να σημειωθεί επίσης ότι κάθε source file πρέπει να κάνει #include το αντίστοιχό του header.

4.3.1 Κώδικας "Main"

Σε αυτό το κομμάτι του κώδικα (**Παράρτημα Ι**), ορίζουμε την function **main**, η οποία αποτελεί και την κύρια συνάρτηση του παιχνιδιού, η οποία περιέχει όλα τα components του video game. Το source file αυτό είναι και το μόνο στον κώδικά μας, που δεν έχει δικό του header, μιας και η συνάρτηση main, δεν πρόκειται να κληθεί από κανένα άλλο source file.

Στις γραμμές **9-22** του **main.cpp**, ορίζονται το όνομα της εφαρμογής, η ανάλυση της οθόνης του παιχνιδιού, καθώς και οι διαστάσεις της γάτας και του μήλου.

Στις γραμμές **29-52**, περιγράφονται τα βασικά components της SDL που χρησιμοποιεί η main. Αυτά είναι τα SDL_Window, SDL_Renderer και SDL_Event. Αρχικοποιείται η SDL library με τη χρήση της function SDL_Init, ενώ χρησιμοποιείται η function SDL_CreateWindowAndRenderer για την δημιουργία του παραθύρου της εφαρμογής και του renderer. Με τη χρήση της function SDL_SetWindowTitle, απεικονίζουμε το όνομα του video game ως τίτλο στο παράθυρο της εφαρμογής. Χρησιμοποιούμε την SDL_Surface δομή για να φορτώσουμε ένα icon μέσω της IMG_Load, το οποία θα ορίσουμε ως εικονίδιο της εφαρμογής μας με τη χρήση της SDL_SetWindowIcon. Αρχικοποιούνται επίσης η SDL ttf engine (αφορά τις γραμματοσειρές) και η SDL sound engine (αφορά τους ήχους). Τέλος περιγράφονται τα components του παιχνιδιού, δηλαδή το gameworld, ο χαρακτήρας (γάτα) και το μήλο.

Στις γραμμές **54-81**, περιγράφεται μια συνθήκη, κατά την οποία για όσο χρονικό διάστημα η εφαρμογή τρέχει, χρησιμοποιώντας την function SDL_PollEvent για ανίχνευση τυχόν συμβάντων. Αν τα συμβάντα αυτά είναι διαφορετικά από τον τερματισμό της εφαρμογής, τότε τα αναλαμβάνει ο κώδικας του **gameworld** που θα δούμε παρακάτω. Επιπλέον υπολογίζει το delta time και ανανεώνει και απεικονίζει το gameworld.

Τέλος, ελέγχει για τυχόν συμβάν τερματισμού και αν ανιχνευθεί, τότε με τη χρήση της function SDL_QUIT κάνει clear όλα τα components της SDL που αρχικοποιήθηκαν.

4.3.2 Κώδικας "GameObject"

Στο κομμάτι αυτό του κώδικα (**Παράρτημα ΙΙ**), ορίζουμε την **class GameObject** που περιέχει functions για τη δημιουργία ενός προτύπου ορθογώνιας περιοχής, την οποία αργότερα θα χρησιμοποιήσουμε τόσο για την δημιουργία της γάτας, όσο και για τη δημιουργία των μήλων. Για τη δημιουργία ορθογώνιας περιοχής, χρησιμοποιούμε την **SDL_Rect**, μια δομή που περιέχει τον ορισμό ενός ορθογωνίου. Οι παράμετροι παίρνουν ακέραιες αριθμητικές τιμές και αφορούν τις συντεταγμένες (x,y) της πάνω αριστερά γωνίας του ορθογωνίου, καθώς επίσης το μήκος και το πλάτος του. Οι

ορθογώνιες αυτές περιοχές θα είναι στην ουσία τα hitboxes των αντικειμένων του παιχνιδιού.

Στις γραμμές **5-71** του **gameobject.cpp**, περιγράφονται οι functions που σχετίζονται με την ορθογώνια περιοχή.

Στις γραμμές **11-29** του **gameobject.h**, δηλώνονται οι λειτουργίες αυτές, ενώ στη γραμμή **32**, χρησιμοποιώντας τη δομή **SDL_Renderer,** της οποίας ο ρόλος είναι να αναλάβει το rendering, ορίζεται μια εικονική function για το rendering, η οποία θα κληθεί αργότερα από τις classes Player και Apple.

4.3.3 Κώδικας "Player"

Σε αυτό το κομμάτι του κώδικα (**Παράρτημα III**), ορίζουμε τις μεταβλητές που αφορούν τις διαστάσεις του χαρακτήρα του παιχνδιού, δηλαδή της γάτας και τους δίνουμε τιμές. Επιπλέον, ορίζουμε την **class Player** και μάλιστα ως θυγατρική της GameObject. Η class αυτή περιέχει functions για τη δημιουργία της γάτας. Χρησιμοποιούμε την δομή **SDL_Texture** για να εισάγουμε εικόνες ως texture που θα "ντύσουν" τα αντικείμενά μας.

Στις γραμμές **3-14** του **player.cpp**, κάνουμε rendering το texture του παίκτη, φορτώνουμε δηλαδή την εικόνα player.png ως texture, ορίζουμε τις τιμές των παραμέτρων της ορθογώνιας επιφάνειας που θα καλύπτει, ενώ αρχικοποιούμε και τον προσανατολισμό του γραφικού.

Στις γραμμές **16-23** ορίζουμε τον constructor για τη δημιουργία της γάτας, ενώ στις γραμμές **25-30** ορίζουμε τον destructor για την διαγραφή του texture με τη χρήση της function **SDL_DestroyTexture.**

Στις γραμμές **32-45** περιγράφεται ο κώδικας για την εμφάνιση της γάτας στην οθόνη, πραγματοποιώντας έλεγχο για το αν έχει φορτωθεί το texture πριν γίνει το rendering, καθώς και αν χρειάζεται να εφαρμοστεί αλλαγή προσανατολισμού του γραφικού. Για τον προσανατολισμό χρησιμοποιούμε την **SDL_RendererFlip** η οποία απαριθμεί τα flags που μπορούν να χρησιμοποιηθούν για την παράμετρο flip της function SDL_RenderCopyEx. Τα flags μπορεί να είναι SDL_FLIP_HORIZONTAL, SDL_FLIP_VERTICAL και SDL_FLIP_NONE και δηλώνουν αντίστοιχα οριζόντια αλλαγή προσανατολισμού, κάθετη ή καθόλου. Για το rendering χρησιμοποιείται η function **SDL_RenderCopyEx**, η οποία αντιγράφει το texture στο συγκεκριμένο αντικείμενο (γάτα) που κάνουμε render και εφαρμόζει περιστροφή κατά γωνία ή αλλαγή προσανατολισμού κατά το πώς ορίζει η SDL_RendererFlip.

Τέλος στις γραμμές **47-71** περιγράφεται ο κώδικας για την κίνηση της γάτας Αν η κατεύθυνση είναι προς τα αριστερά, τότε η γάτα κινείται προς τα αριστερά, πραγματοποιώντας έλεγχο ώστε να μην ξεφύγει από τα αριστερά όρια της οθόνης,

ενώ το γραφικό αλλάζει αντίστοιχα προσανατολισμό προς τα αριστερά. Αν η κατεύθυνση είναι προς τα δεξιά, τότε η γάτα κινείται προς τα δεξιά, πραγματοποιώντας έλεγχο ώστε να μην ξεφύγει από τα δεξιά όρια της οθόνης, ενώ το γραφικό αλλάζει αντίστοιχα προσανατολισμό προς τα δεξιά.

4.3.4 Κώδικας "Apple"

Σε αυτό το κομμάτι του κώδικα (**Παράρτημα IV**), ορίζουμε τις μεταβλητές που αφορούν τις διαστάσεις του μήλου και τους δίνουμε τιμές. Επιπλέον, ορίζουμε την **class Apple** ως θυγατρική της GameObject. Η class αυτή περιέχει functions για τη δημιουργία του μήλου.

Στις γραμμές **3-12** του **apple.cpp**, κάνουμε rendering το texture του μήλου, φορτώνουμε δηλαδή την εικόνα apple.png ως texture και ορίζουμε τις τιμές των παραμέτρων της ορθογώνιας επιφάνειας που θα καλύπτει.

Στις γραμμές **14-21** ορίζουμε τον constructor για τη δημιουργία του μήλου, ενώ στις γραμμές **23-27** ορίζουμε τον destructor για την διαγραφή του texture.

Στις γραμμές **29-36** περιγράφεται ο κώδικας για την εμφάνιση του μήλου στην οθόνη, πραγματοποιώντας έλεγχο για το αν έχει φορτωθεί το texture πριν γίνει το rendering. Για το rendering χρησιμοποιείται η function **SDL_RenderCopy**, η οποία αντιγράφει το texture στο συγκεκριμένο αντικείμενο (μήλο) που κάνουμε render.

Τέλος στις γραμμές **38-42** περιγράφεται το βήμα κίνησης του μήλου στον άξονα y.

4.3.5 Κώδικας "TextEngine"

Στο κομμάτι αυτό του κώδικα (**Παράρτημα V**), ορίζουμε την class **TextEngine** που περιέχει functions σχετικά με την γραμματοσειρά του κειμένου που θα χρησιμοποιήσουμε στο παιχνίδι.

Στις γραμμές **29-37** του **textengine.h**, δηλώνονται οι δομές SDL_Renderer, SDL_Surface, SDL_Texture, SDL_Rect και SDL_Color. Η δομή **SDL_Surface** περιέχει πληροφορίες σχετικά με τα pixel, ενώ η λειτουργία της είναι παρόμοια με την δομή SDL_Texture, ωστόσο η δεύτερη πέρα από το να "κρατά" πληροφορίες pixel ενός στοιχείου, επιπλέον τις αναπαραστά. Η **SDL_Color** είναι μια δομή για την αναπαράσταση χρωμάτων.

Στις γραμμές **17-25** του **textengine.cpp** περιγράφεται η function **size** που υπολογίζει το μέγεθος του κειμένου και το είδος της γραμματοσειράς.

Στις γραμμές **27-41** περιγράφεται η function **draw** η οποία σχεδιάζει, απεικονίζει δηλαδή το κείμενο. Και η size και η draw φορτώνουν την γραμματοσειρά της επιλογής μας, την **robotoc.ttf** με τη χρήση της TTF_OpenFont.

Στις γραμμές **43-53** περιγράφονται οι functions που μας επιτρέπουν να αλλάξουμε το χρώμα του κειμένου, ενώ τέλος στις γραμμές **55-75** περιγράφονται οι functions που κάνουν στοίχιση του κειμένου οριζόντια και κάθετα.

4.3.6 Κώδικας "Button"

Στο κομμάτι αυτό του κώδικα (**Παράρτημα VI**), ορίζουμε την class **Button**, η οποία περιέχει functions για την δημιουργία ενός προτύπου button, το οποίο θα χρησιμοποιηθεί από την class GameWorld για την προσαρμογή των buttons που χρειαζόμαστε στο παιχνίδι.

Στις γραμμές **3-24** του **button.cpp**, ορίζουμε τον constructor για τη δημιουργία του button, αρχικοποιούνται οι member μεταβλητές, ενώ ορίζουμε και τον destructor για την διαγραφή της επιφάνειας και του texture.

Στις γραμμές **26-63**, περιγράφεται η συνάρτηση render που σχεδιάζει/απεικονίζει το button στην οθόνη. Σε αυτό το κομμάτι δημιουργείται η ορθογώνια περιοχή που θα καταλαμβάνει το button, ρυθμίζεται το blend mode με τη χρήση της **SDL_BlendMode** η οποία απαριθμεί τα blend modes που μπορούν να χρησιμοποιηθούν από την function **SDL_SetRenderDrawBlendMode**. Αυτά μπορεί να είναι SDL_BLENDMODE_NONE, SDL_BLENDMODE_BLEND, SDL_BLENDMODE_ADD και SDL_BLENDMODE_MOD τα οποία δηλώνουν αντίστοιχα no blending, alpha blending, additive blending και color modulate. Επίσης σχεδιάζεται το background και το περίγραμμα του button, με τη χρήση των functions SDL_SetRenderDrawColor, SDL_RenderFillRect και SDL_RenderDrawRect. Η **SDL_SetRenderDrawColor** δηλώνει τις παραμέτρους του χρώματος, η **SDL_RenderFillRect** "γεμίζει" την ορθογώνια περιοχή του button με το επιλεγμένο χρώμα, ενώ η **SDL_RenderDrawRect** σχεδιάζει το περίγραμμα γύρω από την ορθογώνια περιοχή που καλύπτει το button. Φορτώνεται η γραμματοσειρά που θα χρησιμοποιήσουμε για το κείμενο, ενώ δημιουργείται η επιφάνεια του κειμένου με τη χρήση της TTF_RenderText_Solid και το texture με την χρήση της function **SDL_CreateTextureFromSurface**, η οποία δημιουργεί texture από την υπάρχουσα επιφάνεια. Με τη χρήση της TTF_SizeText, λαμβάνεται το μέγεθος του κειμένου που χρειάζεται να σχεδιαστεί, ενώ τέλος κεντράρεται και σχεδιάζεται το κείμενο μέσα στο button.

Στις γραμμές **65-70**, περιγράφεται η function isClicked, η οποία ελέγχει αν οι συντεταγμένες του κέρσορα βρίσκονται μέσα στο button.

Τέλος, στις γραμμές **72-84**, περιγράφονται οι functions που ορίζουν το background και border color.

4.3.7 Κώδικας "GameWorld"

Σε αυτό το κομμάτι του κώδικα (**Παράρτημα VII**), ορίζεται η class **GameWorld**, στην οποία περιγράφονται όλα τα επιμέρους στοιχεία που δίνουν ζωή στο παιχνίδι. Εδώ δημιουργούνται όλες οι οθόνες που συνθέτουν το video game, τα buttons, εισάγεται η μουσική, τα πλήκτρα χειρισμού κτλ.

Στις γραμμές **17-43** του **gameworld.h**, ορίζουμε τις τιμές στις διαστάσεις της γάτας και του μήλου. Επίσης ορίζουμε και δίνουμε τιμές σε επιπλέον περιεχόμενα του παιχνιδιού όπως το τελικό σκορ που πρέπει να φτάσει ο παίκτης, τις ζωές που έχει, καθώς και τις διαστάσεις της καρδιάς που θα απεικονίζει τις ζωές, τα επίπεδα δυσκολίας καθώς και τον αρχικό χρόνο που πρέπει να περάσει για την εμφάνιση νέου μήλου στην οθόνη. Τέλος γίνεται απαρίθμηση των καταστάστεων του παιχνιδιού (gamestate). Οι καταστάσεις αυτές είναι οι: START, PLAY, PAUSE, WIN, LOSE, ABOUT, οι οποίες αντιστιχούν στην αρχική οθόνη με το κεντρικό μενού, την οθόνη που διαδραμματίζεται το gameplay, την οθόνη παύσης του παιχνιδιού, την οθόνη σε περίπτωση νίκης, την οθόνη σε περίπτωση αποτυχίας και την οθόνη με τις οδηγίες του παιχνιδιού.

Στις γραμμές **3-113** του **gameworld.cpp**, περιγράφεται ο κώδικας του constructor για τη δημιουργία του gameworld. Από τη γραμμή **5 έως 42** αρχικοποιούμε τα member data του gameworld. Φορτώνουμε επίσης τα εξής textures: την εικόνα που αποτελεί το φόντο του σκηνικού κατά τη διάρκεια του gameplay, το φόντο στην αρχική οθόνη, την εικόνα στην οθόνη about, καθώς και το εικονίδιο της καρδιάς. Επίσης φορτώνουμε την μουσική χρησιμοποιώντας την Mix_LoadMUS και τα ηχητικά εφέ (μήλου,νίκης και αποτυχίας) χρησιμοποιώντας την Mix_LoadWAV. Από τη γραμμή **44 έως 105** δημιουργούμε όλα τα buttons που θα απεικονίσουμε στο παιχνίδι. Τα buttons αυτά είναι τα: Play, Exit, Restart, Menu, Continue, About και Back τα οποία αντίστοιχα οδηγούν τον παίκτη στην εκκίνηση του level, στην έξοδο από την εφαρμογή, στην επανεκκίνηση του level, στο κεντρικό μενού, στην επιστροφή στο gameplay μετά από παύση, στην οθόνη με τις οδηγίες και πίσω στην αρχική οθόνη (ομοίως με το menu). Στην γραμμή **108** κάνουμε seed τον random generator ο οποίος θα χρησιμοποιηθεί αργότερα από την function getRandom για τη δημιουργία τυχαίου αριθμού. Τέλος στην γραμμή **111** χρησιμοποιώντας την Mix_VolumeMusic αλλάζουμε την ένταση της μουσικής.

Στις γραμμές **115-147**, ορίζουμε τον destructor του gameworld,όπου διαγράφουμε τα πάντα από την μνήμη. Τα textures, την μουσική (χρησιμοποιώντας την Mix_FreeMusic), το ηχητικό εφέ των μήλων (χρησιμοποιώντας την Mix_FreeChunk), την γάτα, τα μήλα, την textEngine και τα buttons.

Στις γραμμές **154-174** γίνεται έλεγχος για το αν ο παίκτης πατά το αριστερό ή το δεξί βελάκι στο πληκτρολόγιο χρησιμοποιώντας την function **SDL_GetKeyboardState** η οποία ενημερώνει για την δεδομένη κατάσταση του πληκτρολογίου, αν πατιέται δηλαδή κάποιο πλήκτρο). Αν πατιέται το αριστερό βελάκι, τότε η γάτα κινείται προς τα αριστερά, ενώ αν πατίεται το δεξί βελάκι, η γάτα κινείται προς τα δεξιά.

Στις γραμμές **176-196** γίνεται έλεγχος για το αν πρέπει να δημιουργηθεί νέο μήλο, καθώς και για το αν ικανοποιείται η συνθήκη για την αλλαγή της δυσκολίας του επιπέδου, όπου τα μήλα θα εμφανίζονται με μεγαλύτερη συχνότητα.

Στις γραμμές **203-256** γίνεται έλεγχος για το αν υπάρχει collision μεταξύ της γάτας και του μήλου ή αν το μήλο αγγίζει το έδαφος. Χρησιμοποιείται η function **SDL_IntersectRect**, η οποία υπολογίζει το σημείο τομής δύο ορθογωνίων περιοχών. Στην περίπτωση που το μήλο "συγκρούεται" με τη γάτα, το μήλο εξαφανίζεται, παίζει το ηχητικό εφέ του μήλου και το σκορ αυξάνεται κατά ένα. Αν το σκορ του παίκτη ισούται με το τελικό σκορ, τότε η κατάσταση του παιχνδιιού αλλάζει σε WIN, παίζει το ηχητικό της νίκης ενώ σταματά η μουσική. Αν το μήλο αγγίξει το έδαφος, τότε το μήλο εξαφανίζεται, παίζει το ηχητικό του μήλου και ο παίκτης χάνει μια ζωή. Αν η ζωή του παίκτη ισούται με το μηδέν, τότε η κατάσταση του παιχνιδιού αλλάζει σε LOSE, παίζει το ηχητικό εφέ της αποτυχίας και σταματά η μουσική.

Στις γραμμές **258-340**, περιγράφεται η δημιουργία και απεικόνιση της οθόνης ανάλογα με την κατάσταση του παιχνιδιού. Αν η κατάσταση του παιχνιδιού είναι **START**, τότε απεικονίζεται η εικόνα φόντου που αντιστοιχεί στην μεταβλητή **cover**, καθώς επίσης και τα buttons play, about και exit. Αν η κατάσταση του παιχνιδιού είναι PLAY, PAUSE, WIN ή LOSE, απεικονίζεται η εικόνα φόντου που αντιστοιχεί στη μεταβλητή **background**, καθώς επίσης και το σκορ του παίκτη όπως και οι ζωές του. Αν η κατάσταση είναι **PLAY**, τότε επιπλέον απεικονίζεται η γάτα καθώς και τα μήλα. Αν η κατάσταση είναι **WIN** ή **LOSE**,τότε απεικονίζονται τα buttons restart και menu, καθώς και το μήνυμα "victory" ή "defeat" αντίστοιχα. Αν η κατάσταση του παιχνιδιού είναι **PAUSE**, τότε απεικονίζεται το μήνυμα "paused", καθώς και τα buttons continue, restart και menu. Τέλος, αν η κατάσταση είναι ABOUT, τότε απεικονίζεται η εικόνα φόντου που αντιστοιχεί στη μεταβλητή **aboutCover**, καθώς και το button back.

Στις γραμμές **342-483**, περιγράφονται τα user input events, όταν δηλαδή ο παίκτης αλληλεπιδρά μέσω του πληκτρολογίου ή του ποντικιού σε διάφορες καταστάσεις του παιχνιδιού. Χρησιμοποιείται η δομή **SDL_Event**, η οποία είναι ο πυρήνας όλων των event handling στην SDL. Από τη γραμμή **344 έως 373**, περιγράφονται τα events όταν ο παίκτης πατά το πλήκτρο Escape σε κάθε μία από τις καταστάσεις του παιχνιδιού. Γίνεται έλεγχος αν πατιέται κάποιο πλήκτρο (SDL_KEYDOWN) και στην συνέχεια αν αυτό το πλήκτρο είναι το Escape (**SDLK_ESCAPE**). Εφόσον πατιέται, τότε:

- Αν το παιχνίδι βρίσκεται στην κατάσταση **PLAY**, η κατάσταση αλλάζει σε PAUSE και σταματά η μουσική.
- Αν το παιχνίδι βρίσκεται στην κατάσταση **PAUSE**, η κατάσταση αλλάζει σε PLAY και ξεκινά η μουσική.
- Αν το παιχνίδι βρίσκεται στην κατάσταση **WIN** ή **LOSE**, η κατάσταση αλλάζει σε PLAY, καλείται η function **reset** (*θα δούμε την λειτουργία της παρακάτω*) και ξεκινά η μουσική.
- Αν το παιχνίδι βρίσκεται στην κατάσταση **ABOUT**, η κατάσταση αλλάζει σε START.

Από τη γραμμή **375 έως 483**, περιγράφονται τα events όταν ο παίκτης πατά το αριστερό πλήκτρο στο ποντίκι (**SDL_BUTTON_LEFT**). Εφόσον πατιέται, τότε:

Όταν το παιχνίδι βρίσκεται στην κατάσταση **START**:

- Αν πατάει το play button, η κατάσταση αλλάζει σε PLAY, καλείται η function reset και ξεκινά η μουσική.
- Αν πατάει το about button, η κατάσταση αλλάζει σε ABOUT.
- Αν πατάει το exit button, τερματίζει η εφαρμογή.

Όταν το παιχνίδι βρίσκεται στην κατάσταση **PLAY**, δεν συμβαίνει απολύτως τίποτα, καθώς δεν υπάρχουν buttons.

Όταν το παιχνίδι βρίσκεται στην κατάσταση **PAUSE**:

- Αν πατάει το restart button, η κατάσταση αλλάζει σε PLAY, καλείται η function reset και ξεκινά η μουσική.
- Αν πατάει το menu button, η κατάσταση αλλάζει σε START, καλείται η function reset και ξεκινά η μουσική.
- Αν πατάει το continue button, η κατάσταση αλλάζει σε PLAY και ξεκινά η μουσική.

Όταν το παιχνίδι βρίσκεται στην κατάσταση **WIN** ή **LOSE**:

- Αν πατάει το restart button, η κατάσταση αλλάζει σε PLAY, καλείται η function reset και ξεκινά η μουσική.
- Αν πατάει το menu button, η κατάσταση αλλάζει σε START, καλείται η function reset και ξεκινά η μουσική.

Όταν το παιχνίδι βρίσκεται στην κατάσταση **ABOUT**:

- Αν πατάει το back button, η κατάσταση αλλάζει σε START.

Στις γραμμές **503-511**, περιγράφεται η function **pushApple** για δημιουργία καινούργιων μήλων σε τυχαία θέση στον άξονα x. Χρησιμοποιεί τη βοηθητική function **getRandom** (γραμμές **527-531**).

Στις γραμμές **513-525**, περιγράφεται η function **reset**, η οποία όταν καλείται, κάνει reset όλων των παραμέτρων που αφορούν το gameplay, όπως την θέση της γάτας, το σκορ, τις ζωές, την δυσκολία κτλ.

Τέλος, στις γραμμές **533-583**, περιγράφονται functions που αφορούν το σχεδιασμό κειμένου στον οθόνη, των καρδιών που απεικονίζουν τις ζωές του παίκτη, functions για τον έλεγχο της μουσικής, καθώς και functions που αλλάζουν τις καταστάσεις του παιχνιδιού.

4.3.8 Κώδικας "DeltaClock"

Στο κομμάτι αυτό του κώδικα (**Παράρτημα VIII**), ορίζουμε την class **DeltaClock**, μια απλή κλάση που κάνει track την delta time. Χρησιμοποιείται η function SDL_GetPerformanceCounter, η οποία λαμβάνει την τρέχουσα τιμή του μετρητή delta, καθώς και η function SDL_GetPerformanceFrequency, η οποία μετράει την συχνότητα του μετρητή.

4.4 Συμπεράσματα

Μέσα από αυτή την εργασία, μπήκαμε στον υπέροχο κόσμο των video games. Μάθαμε ότι τα παιχνίδια δεν είναι απλά μια δραστηριότητα για παιδιά, όπως πιστεύεται. Τα videogames προσφέρουν μια ολοκληρωμένη μορφή ψυχαγωγίας, με μεγάλη ποικιλία τίτλων στην αγορά, ακόμη και για τους πιο απαιτητικούς. Επίσης σε αυτή την εργασία, μπήκαμε στην διαδικασία να μελετήσουμε πώς δημιουργείται ένα video game, καθώς και να επιχειρήσουμε να αναπτύξουμε το δικό μας παιχνίδι, με την χρήση της βιβλιοθήκης SDL, η οποία παρέχει εύχρηστες functions και δομές, εύκολες στην κατανόηση λειτουργίας τους, που μας επιτρέπουν να δημιουργήσουμε ένα cross-platform video game.

4.5 Μελλοντικές επεκτάσεις

Μελλοντικά, θα μπορούσαν να εφαρμοστούν κάποιες διορθώσεις/βελτιώσεις στο video game **APPLE CAT** όπως:

- Δημιουργία buttons που θα κάνουν mute την μουσική και τα ηχητικά εφέ
- Περισσότερες επιλογές ανάλυσης του window της εφαρμογής.
- Περισσότερα επίπεδα, τα οποία θα είχαν το δικό τους ξεχωριστό φόντο, καθώς επίσης θα άλλαζε χρώμα η κορδέλα της γάτας.
- Αλλαγή της εμφάνισης του κέρσορα όταν κάνει mouse over σε ένα button.
- Δημιουργία ενός δεύτερου gameplay mode, το endless mode, όπου τα μήλα δεν θα σταματούσαν ποτέ να εμφανίζονται, έως ότου ο παίκτης χάσει όλες τις ζωές του. Σε αυτό το mode το επίπεδο δυσκολίας θα αυξάνεται με σαφώς μεγαλύτερο ρυθμό κ.α.

ΠΑΡΑΡΤΗΜΑ Ι: "Main" κώδικας

main.cpp

main.cpp*

```cpp
#include <SDL.h>
#include <SDL_image.h>

#include "player.h"
#include "apple.h"
#include "gameworld.h"
#include "deltaclock.h"

// APPLICATION INFO
const char *applicationName = "Apple Cat";

// Screen Resolution
#define SCREEN_WIDTH 800
#define SCREEN_HEIGHT 480

// PLAYER INFO
#define PLAYER_WIDTH 64
#define PLAYER_HEIGHT 64

// APPLE INFO
#define APPLE_WIDTH 32
#define APPLE_HEIGHT 32

int main(int argc, char **argv) {

    bool quit = false;
    DeltaClock deltaclock;

    // Basic SDL components
    SDL_Window *window = nullptr;
    SDL_Renderer *renderer = nullptr;
    SDL_Event evt;

    // Initialize SDL (window, renderer)
    SDL_Init(SDL_INIT_EVERYTHING);
    SDL_CreateWindowAndRenderer(SCREEN_WIDTH, SCREEN_HEIGHT, 0, &window, &renderer);
    SDL_SetWindowTitle(window, applicationName);

    // Change application's icon
    SDL_Surface* icon = IMG_Load("applecat.ico");
    SDL_SetWindowIcon(window, icon);

    // Initialize the sdl ttf engine
    TTF_Init();

    // Initialize the sdl sound engine
    Mix_OpenAudio(22050, MIX_DEFAULT_FORMAT, 2, 4096);

    // Game components
    GameWorld world = GameWorld(renderer, SCREEN_WIDTH, SCREEN_HEIGHT);
    world.getPlayer()->setPosition(400, SCREEN_HEIGHT - PLAYER_HEIGHT);
    world.pushApple();

    while ( !quit ) {

        // Detect Window Events (Keyboard, Mouse)
        while ( SDL_PollEvent(&evt) != 0 ) {

            // Check for quit status and let world handle the rest of the input
            quit = evt.type == SDL_QUIT || !world.handleInput(evt);

        }

        // Calculate the actual delta time ( in milliseconds )
        double delta = deltaclock.tick();

        // Set black for the background color and clear the screen
        SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
        SDL_RenderClear(renderer);

        // Update and render world (the actual game)
        world.update(delta);
```

```
73          world.render();
74
75          // Swap to window rendering
76          SDL_RenderPresent(renderer);
77
78          // Fixed to 60 FPS
79          SDL_Delay(1000 / 60);
80
81      }
82
83      // Stop and free SDL
84      SDL_FreeSurface(icon);
85      SDL_DestroyRenderer(renderer);
86      SDL_DestroyWindow(window);
87      SDL_Quit();
88      TTF_Quit();
89      Mix_CloseAudio();
90
91      return 0;
92
93  }
```

ΠΑΡΑΡΤΗΜΑ ΙΙ: "GameObject" κώδικας

gameobject.h

gameobject.h gameobject.cpp

```
1   #ifndef __GAMEOBJECT_H__
2   #define __GAMEOBJECT_H__
3
4   #include <SDL.h>
5
6   class GameObject {
7
8   public:
9
10      // Functions to set the object's position
11      void setPositionX(int x);
12      void setPositionY(int y);
13      void setPosition(int x, int y);
14
15      // Functions to get the object's position
16      int getPositionX();
17      int getPositionY();
18
19      // Functions to set object's size
20      void setWidth(int w);
21      void setHeight(int h);
22      void setSize(int w, int h);
23
24      // Functions to get the object's size
25      int getWidth();
26      int getHeight();
27
28      // Function to get the entire rect
29      SDL_Rect getRect();
30
31      // A pure virtual function for drawing
32      virtual void render(SDL_Renderer *renderer) = 0;
33
34  protected:
35
36      SDL_Rect rect;
37
38  };
39
40  #endif
```

gameobject.cpp

gameobject.h gameobject.cpp

```cpp
#include "gameobject.h"

#include "apple.h"

void GameObject::setPositionX(int x) {

    this->rect.x = x;

}

void GameObject::setPositionY(int y) {

    this->rect.y = y;

}

void GameObject::setPosition(int x, int y) {

    this->setPositionX(x);
    this->setPositionY(y);

}

int GameObject::getPositionX() {

    return this->rect.x;

}

int GameObject::getPositionY() {

    return this->rect.y;

}

void GameObject::setWidth(int w) {

}

void GameObject::setHeight(int h) {

    this->rect.h = h;

}

void GameObject::setSize(int w, int h) {

    this->setWidth(w);
    this->setHeight(h);

}

int GameObject::getWidth() {

    return this->rect.w;

}

int GameObject::getHeight() {

    return this->rect.h;

}

SDL_Rect GameObject::getRect() {

    return this->rect;

}
```

ΠΑΡΑΡΤΗΜΑ ΙΙΙ: "Player" κώδικας

player.h

player.h player.cpp

```
1   #ifndef __PLAYER_H__
2   #define __PLAYER_H__
3
4   #include <SDL.h>
5   #include <SDL_image.h>
6   #include "gameobject.h"
7
8   // PLAYER INFO
9   #define PLAYER_WIDTH 64
10  #define PLAYER_HEIGHT 64
11
12  class Player : public GameObject {
13
14  public:
15      Player(SDL_Renderer *renderer);
16      Player();
17      ~Player();
18
19      // Draw the player on the screen
20      void render(SDL_Renderer *renderer) override;
21
22      // Move the player
23      void move(int dir, int limitLeft, int limitRight, double delta);
24
25      enum {
26          DIR_LEFT,
27          DIR_RIGHT
28      };
29
30  private:
31
32      // Player's texture
33      SDL_Texture *texture;
34
35      // Player's flipped orientation (true when player's moves to the left)
36      bool isFlipped;
37
38  };
39
40  #endif
```

player.cpp

player.h player.cpp

```cpp
#include "player.h"

Player::Player(SDL_Renderer *renderer) {

    this->rect.x = 0;
    this->rect.y = 0;
    this->rect.w = PLAYER_WIDTH;
    this->rect.h = PLAYER_HEIGHT;

    this->texture = IMG_LoadTexture(renderer, "assets/player.png");

    this->isFlipped = false;

}

Player::Player() {

    this->rect.x = 0;
    this->rect.y = 0;
    this->rect.w = PLAYER_WIDTH;
    this->rect.h = PLAYER_HEIGHT;

}

Player::~Player() {

    // Free the texture
    SDL_DestroyTexture(this->texture);

}

void Player::render(SDL_Renderer *renderer) {

    // Check if texture is loaded before rendering
    if ( this->texture == nullptr )
        return;

    // Check if flip needs to be applied
    SDL_RendererFlip flip = this->isFlipped ?
        SDL_RendererFlip::SDL_FLIP_HORIZONTAL : SDL_RendererFlip::SDL_FLIP_NONE;

    // Render the player
    SDL_RenderCopyEx(renderer, this->texture, nullptr, &(this->rect), 0, nullptr, flip);

}

void Player::move(int dir, int limitLeft, int limitRight, double delta) {

    if ( dir == DIR_LEFT) {

        // Move the player to the left (check if out of bounds)
        if ( (this->rect.x -= static_cast<int>(0.68 * delta)) < limitLeft ) {
            this->rect.x = limitLeft;
        }

        // Change player's orientation
        this->isFlipped = true;

    } else if ( dir == DIR_RIGHT ) {

        // Move the player to the right (check if out of bounds)
        if ( (this->rect.x += static_cast<int>(0.68 * delta)) > limitRight ) {
            this->rect.x = limitRight;
        }

        // Change player's orientation
        this->isFlipped = false;

    }

}
```

ΠΑΡΑΡΤΗΜΑ IV: "Apple" κώδικας

apple.h

apple.h apple.cpp

```
1   #ifndef __APPLE_H__
2   #define __APPLE_H__
3
4   #include <SDL.h>
5   #include <SDL_image.h>
6   #include "gameobject.h"
7
8   // APPLE INFO
9   #define APPLE_WIDTH 32
10  #define APPLE_HEIGHT 32
11
12  class Apple : public GameObject {
13
14  public:
15      Apple(SDL_Renderer *renderer);
16      Apple();
17      ~Apple();
18
19      // Draw the apple on the screen
20      void render(SDL_Renderer *renderer) override;
21
22      // Step apple's position on axis Y
23      void step(double delta);
24
25  private:
26
27      SDL_Texture *texture;
28
29  };
30
31  #endif
```

apple.cpp

apple.h apple.cpp

```cpp
#include "apple.h"

Apple::Apple(SDL_Renderer *renderer) {

    this->rect.x = 0;
    this->rect.y = 0;
    this->rect.w = APPLE_WIDTH;
    this->rect.h = APPLE_HEIGHT;

    this->texture = IMG_LoadTexture(renderer, "assets/apple.png");

}

Apple::Apple() {

    this->rect.x = 0;
    this->rect.y = 0;
    this->rect.w = APPLE_WIDTH;
    this->rect.h = APPLE_HEIGHT;

}

Apple::~Apple() {

    SDL_DestroyTexture(this->texture);

}

void Apple::render(SDL_Renderer *renderer) {

    if ( this->texture == nullptr )
        return;

    SDL_RenderCopy(renderer, this->texture, nullptr, &(this->rect));

}

void Apple::step(double delta) {

    this->rect.y += static_cast<int>(0.2 * delta);

}
```

ΠΑΡΑΡΤΗΜΑ V: "TextEngine" κώδικας

textengine.h

textengine.h | textengine.cpp

```
1   #ifndef __TEXTENGINE_H__
2   #define __TEXTENGINE_H__
3
4   #include <string>
5   #include <SDL.h>
6   #include <SDL_ttf.h>
7
8   class TextEngine {
9
10  public:
11
12      TextEngine(SDL_Renderer *renderer);
13      ~TextEngine();
14
15      // Funtion to calculate the size of specific text and font
16      void size(const std::string& text, int fontSize, int& width, int& height);
17
18      // Function to draw some text
19      void draw(const std::string& text, int x, int y, int fontSize);
20
21      // Functions to change the color
22      void setColor(SDL_Color color);
23      void setColor(Uint8 r, Uint8 g, Uint8 b);
24
25      // Helper functions to align text
26      int getVerticalAlign(const std::string& text, int fontSize, int width);
27      int getHorizontalAlign(const std::string& text, int fontSize, int height);
28
29  private:
30
31      SDL_Renderer *renderer;
32      SDL_Surface *surface;
33      SDL_Texture *texture;
34      SDL_Rect rect;
35      SDL_Color color;
36
37  };
38
39  #endif
```

textengine.cpp

textengine.h | textengine.cpp

```cpp
#include "textengine.h"

TextEngine::TextEngine(SDL_Renderer *renderer) {

    this->renderer = renderer;
    this->color = { 255, 255, 255, 255 };

}

TextEngine::~TextEngine() {

    SDL_FreeSurface(this->surface);
    SDL_DestroyTexture(this->texture);

}

void TextEngine::size(const std::string& text, int fontSize, int& width, int& height) {

    TTF_Font *font = TTF_OpenFont("assets/robotoc.ttf", fontSize);

    TTF_SizeText(font, text.c_str(), &width, &height);

    TTF_CloseFont(font);

}

void TextEngine::draw(const std::string& text, int x, int y, int fontSize) {

    TTF_Font *font = TTF_OpenFont("assets/robotoc.ttf", fontSize);

    this->surface = TTF_RenderText_Solid(font, text.c_str(), this->color);
    this->texture = SDL_CreateTextureFromSurface(this->renderer, this->surface);

    SDL_Rect messageRect = { x, y, 0, 0 };
    TTF_SizeText(font, text.c_str(), &messageRect.w, &messageRect.h);

    SDL_RenderCopy(this->renderer, texture, NULL, &messageRect);

    TTF_CloseFont(font);

}

void TextEngine::setColor(SDL_Color color) {

    this->color = color;

}

void TextEngine::setColor(Uint8 r, Uint8 g, Uint8 b) {

    this->color = { r, g, b };

}

int TextEngine::getVerticalAlign(const std::string& text, int fontSize, int height) {

    int textWidth = 0;
    int textHeight = 0;

    this->size(text, fontSize, textWidth, textHeight);

    return (height - textHeight) / 2;

}

int TextEngine::getHorizontalAlign(const std::string& text, int fontSize, int width) {

    int textWidth = 0;
    int textHeight = 0;

    this->size(text, fontSize, textWidth, textHeight);

    return (width - textWidth) / 2;

}
```

ΠΑΡΑΡΤΗΜΑ VI: "Button" κώδικας

button.h

button.h button.cpp

```
#ifndef __BUTTON_H__
#define __BUTTON_H__

#include <string>
#include <SDL.h>
#include <SDL_ttf.h>

class Button {

public:

    // Constructor & Destructor
    Button(const std::string& caption, int x, int y, int w, int h, int fontSize, int r, int g, int b);
    ~Button();

    // Function to draw the button inside the window
    void render(SDL_Renderer *renderer);

    // Function to ckeck if button was clicked
    bool isClicked(int x, int y);

    // Function to set the background Color
    void setBackgroundColor(Uint8 r, Uint8 g, Uint8 b, Uint8 a);

    // Function to set the border Color
    void setBorderColor(Uint8 r, Uint8 g, Uint8 b, Uint8 a);

private:

    SDL_Surface *surface;
    SDL_Texture *texture;
    SDL_Rect rect;
    SDL_Color color;
    SDL_Color bgColor;
    SDL_Color borderColor;
    std::string caption;
    int posX;
    int posY;
    int width;
    int height;
    int fSize;

};

#endif
```

button.cpp

button.h button.cpp

```cpp
#include "button.h"

Button::Button(const std::string& text, int x, int y, int w, int h, int fontSize, int r, int g, int b) {

    // Initialize members
    this->caption = text;
    this->posX = x;
    this->posY = y;
    this->width = w;
    this->height = h;
    this->fSize = fontSize;
    this->color = { r, g, b };
    this->bgColor = { 255, 255, 255, 0 };
    this->borderColor = { 255, 255, 255, 0 };

}

Button::~Button() {

    // Free everything
    SDL_FreeSurface(this->surface);
    SDL_DestroyTexture(this->texture);

}

void Button::render(SDL_Renderer *renderer) {

    // Create button's rect
    SDL_Rect buttonRect = { this->posX, this->posY, this->width, this->height };

    // Set blend mode for alpha channel
    SDL_SetRenderDrawBlendMode(renderer, SDL_BLENDMODE_BLEND);

    // Render the background
    SDL_SetRenderDrawColor(renderer, this->bgColor.r, this->bgColor.g, this->bgColor.b, this->bgColor.a);
    SDL_RenderFillRect(renderer, &buttonRect);

    // Render the border
    SDL_SetRenderDrawColor(renderer, this->borderColor.r, this->borderColor.g, this->borderColor.b, this->borderColor.a);
    SDL_RenderDrawRect(renderer, &buttonRect);

    // Load the font
    TTF_Font *font = TTF_OpenFont("assets/robotoc.ttf", this->fSize);

    // Create the surface and the texture
    this->surface = TTF_RenderText_Solid(font, this->caption.c_str(), this->color);
    this->texture = SDL_CreateTextureFromSurface(renderer, this->surface);

    // Get the size of the text we need to render inside the button
    int textWidth = 0;
    int textHeight = 0;
    TTF_SizeText(font, this->caption.c_str(), &textWidth, &textHeight);

    // Center and render the text inside the button
    int textX = this->posX + (this->width - textWidth) / 2;
    int textY = this->posY + (this->height - textHeight) / 2;
    SDL_Rect messageRect = { textX, textY, textWidth, textHeight };
    SDL_RenderCopy(renderer, texture, NULL, &messageRect);

    // Close the font
    TTF_CloseFont(font);

}

bool Button::isClicked(int x, int y) {

    // Check if mouse coords are inside this button
    return (x >= this->posX) && (y >= this->posY) && (x <= (this->posX + this->width)) && (y <= (this->posY + this->height));

}

void Button::setBackgroundColor(Uint8 r, Uint8 g, Uint8 b, Uint8 a) {

    // Set the color
    this->bgColor = { r, g, b, a };

}

void Button::setBorderColor(Uint8 r, Uint8 g, Uint8 b, Uint8 a) {

    // Set the color
    this->borderColor = { r, g, b, a };

}
```

ΠΑΡΑΡΤΗΜΑ VII: "GameWorld" κώδικας

gameworld.h

gameworld.h gameworld.cpp

```
#ifndef __GAMEWORLD_H__
#define __GAMEWORLD_H__

#include <string>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_mixer.h>
#include <SDL_ttf.h>
#include "player.h"
#include "apple.h"
#include "textengine.h"
#include "button.h"

// PLAYER INFO
#define PLAYER_WIDTH 64
#define PLAYER_HEIGHT 64

// APPLE INFO
#define APPLE_WIDTH 32
#define APPLE_HEIGHT 32

// GAME LOGIC CONSTANTS
#define TARGET_SCORE 30
#define MAX_LIFES 3
#define HEART_WIDTH 32
#define HEART_HEIGHT 32
#define NEWDIFF_ROUNDS 3
#define INIT_ROUND_TIME 2000

// Game state enumerations
enum class GameState : unsigned char {

    START,
    PLAY,
    PAUSE,
    WIN,
    LOSE,
    ABOUT

};

class GameWorld {

public:

    // Constructor and destructor
    GameWorld(SDL_Renderer *renderer, int screenWidth, int screenHeight);
    ~GameWorld();

    // Functions to update, render and handle user input
    void update(double delta);
    void render();
    bool handleInput(SDL_Event evt);

    // Functions to set/get player/apple
    Player* getPlayer();
    Apple* getApple(int index);
    std::vector<Apple*> getApples();
    void pushApple();

    // Function to reset the game
    void reset();

private:

    // Game state
    GameState state;

    // The main renderer (where to draw)
```

```
    SDL_Renderer *renderer;

    // Textures
    SDL_Texture *background;
    SDL_Texture *cover;
    SDL_Texture *aboutCover;
    SDL_Texture *heart;

    // Rects for each texture (position and size)
    SDL_Rect backgroundRect;
    SDL_Rect coverRect;
    SDL_Rect aboutCoverRect;
    SDL_Rect heartRect;

    // Basic game sprites
    Player *player;
    std::vector<Apple*> apples;

    // Something to help us draw text easier and faster
    TextEngine *textEngine;

    // Sound components
    Mix_Music *backgroundMusic;
    Mix_Chunk *hitAppleSound;
    Mix_Chunk *winSound;
    Mix_Chunk *loseSound;

    // Game logic variables
    int sWidth;
    int sHeight;
    int score;
    int targetScore;
    int lifes;

    // Helper function for random numbers in range
    int getRandom(int min, int max);

    // Functions to draw text on the screen
    void drawScore();
    void drawResultMessage(std::string text);

    // Function to draw the hearts on the screen
    void drawHearts();

    // Function to control music
    void playMusic();
    void stopMusic();

    // Functions to switch between game states
    void winGame();
    void loseGame();
    void playGame();
    void pauseGame();

    // Buttons
    Button *playButton;
    Button *exitButton;
    Button *restartButton;
    Button *menuButton;
    Button *continueButton;
    Button *aboutButton;
    Button *backButton;

    // Apple creation timer
    double appleTimer;
    int roundTimer;
    int roundCounter;

};

#endif
```

gameworld.cpp

gameworld.h gameworld.cpp

```cpp
#include "gameworld.h"

GameWorld::GameWorld(SDL_Renderer *renderer, int screenWidth, int screenHeight) {

    // Initialize members
    this->background = IMG_LoadTexture(renderer, "assets/background.png");
    this->backgroundRect.x = 0;
    this->backgroundRect.y = 0;
    this->backgroundRect.w = screenWidth;
    this->backgroundRect.h = screenHeight;
    this->cover = IMG_LoadTexture(renderer, "assets/cover.png");
    this->coverRect.x = 0;
    this->coverRect.y = 0;
    this->coverRect.w = screenWidth;
    this->coverRect.h = screenHeight;
    this->aboutCover = IMG_LoadTexture(renderer, "assets/about.png");
    this->aboutCoverRect.x = 0;
    this->aboutCoverRect.y = 0;
    this->aboutCoverRect.w = screenWidth;
    this->aboutCoverRect.h = screenHeight;
    this->heart = IMG_LoadTexture(renderer, "assets/life.png");
    this->heartRect.x = 0;
    this->heartRect.y = 10;
    this->heartRect.w = 32;
    this->heartRect.h = 32;
    this->sWidth = screenWidth;
    this->sHeight = screenHeight;
    this->score = 0;
    this->targetScore = TARGET_SCORE;
    this->player = new Player(renderer);
    this->state = GameState::START;
    this->renderer = renderer;
    this->backgroundMusic = Mix_LoadMUS("assets/loop.wav");
    this->hitAppleSound = Mix_LoadWAV("assets/hit.wav");
    this->winSound = Mix_LoadWAV("assets/claps.wav");
    this->loseSound = Mix_LoadWAV("assets/lose.wav");
    this->playMusic();
    this->textEngine = new TextEngine(renderer);
    this->lifes = MAX_LIFES;
    this->appleTimer = 0.0;
    this->roundTimer = INIT_ROUND_TIME;
    this->roundCounter = 0;

    // Create the play button
    int playButtonWidth = 140;
    int playButtonHeight = 50;
    int playButtonX = (screenWidth - playButtonWidth) / 2;
    int playButtonY = 120;
    this->playButton = new Button("Play", playButtonX, playButtonY, playButtonWidth, playButtonHeight, 36, 255, 255, 255);
    this->playButton->setBackgroundColor(255, 125, 35, 195);
    this->playButton->setBorderColor(255, 125, 35, 255);

    // Create the exit button
    int exitButtonWidth = 120;
    int exitButtonHeight = 50;
    int exitButtonX = (screenWidth - exitButtonWidth) / 2;
    int exitButtonY = 230;
    this->exitButton = new Button("Exit", exitButtonX, exitButtonY, exitButtonWidth, exitButtonHeight, 36, 255, 255, 255);
    this->exitButton->setBackgroundColor(255, 125, 35, 195);
    this->exitButton->setBorderColor(255, 125, 35, 255);

    // Create the restart button
    int restartButtonWidth = 140;
    int restartButtonHeight = 50;
    int restartButtonX = (screenWidth - restartButtonWidth) / 2;
    int restartButtonY = 175;
    this->restartButton = new Button("Restart", restartButtonX, restartButtonY, restartButtonWidth, restartButtonHeight, 36, 255, 255, 255);
    this->restartButton->setBackgroundColor(255, 125, 35, 195);
    this->restartButton->setBorderColor(255, 125, 35, 255);

    // Create the menu button
    int menuButtonWidth = 120;
    int menuButtonHeight = 50;
    int menuButtonX = (screenWidth - menuButtonWidth) / 2;
    int menuButtonY = 230;
    this->menuButton = new Button("Menu", menuButtonX, menuButtonY, menuButtonWidth, menuButtonHeight, 36, 255, 255, 255);
    this->menuButton->setBackgroundColor(255, 125, 35, 195);
    this->menuButton->setBorderColor(255, 125, 35, 255);

    // Create the continue button
    int continueButtonWidth = 160;
    int continueButtonHeight = 50;
    int continueButtonX = (screenWidth - continueButtonWidth) / 2;
    int continueButtonY = 120;
    this->continueButton = new Button("Continue", continueButtonX, continueButtonY, continueButtonWidth, continueButtonHeight, 36, 255, 255, 255);
    this->continueButton->setBackgroundColor(255, 125, 35, 195);
    this->continueButton->setBorderColor(255, 125, 35, 255);

    // Create the about button
    int aboutButtonWidth = 160;
    int aboutButtonHeight = 50;
    int aboutButtonX = (screenWidth - aboutButtonWidth) / 2;
    int aboutButtonY = 175;
    this->aboutButton = new Button("About", aboutButtonX, aboutButtonY, aboutButtonWidth, aboutButtonHeight, 36, 255, 255, 255);
    this->aboutButton->setBackgroundColor(255, 125, 35, 195);
    this->aboutButton->setBorderColor(255, 125, 35, 255);

    // Create the back button
    int backButtonWidth = 160;
    int backButtonHeight = 50;
    int backButtonX = (screenWidth - backButtonWidth) / 2;
    int backButtonY = 375;
    this->backButton = new Button("Back", backButtonX, backButtonY, backButtonWidth, backButtonHeight, 36, 255, 255, 255);
    this->backButton->setBackgroundColor(255, 125, 35, 195);
    this->backButton->setBorderColor(255, 125, 35, 255);

    // Seed the random generator
    std::srand(std::time(0));
```

```
109
110         // Change music volume
111         Mix_VolumeMusic(50);
112
113     }
114
115     GameWorld::~GameWorld() {
116
117         // Clear everything
118         // Free textures
119         SDL_DestroyTexture(this->background);
120         SDL_DestroyTexture(this->heart);
121         SDL_DestroyTexture(this->cover);
122         SDL_DestroyTexture(this->aboutCover);
123
124         // Free sound components
125         Mix_FreeMusic(this->backgroundMusic);
126         Mix_FreeChunk(this->hitAppleSound);
127
128         // Free the player
129         delete this->player;
130
131         // Free the apples
132         for ( auto apple : this->apples )
133             delete apple;
134
135         // Free textengine
136         delete this->textEngine;
137
138         // Free buttons
139         delete this->playButton;
140         delete this->exitButton;
141         delete this->menuButton;
142         delete this->continueButton;
143         delete this->aboutButton;
144         delete this->restartButton;
145         delete this->backButton;
146
147     }
148
149     void GameWorld::update(double delta) {
150
151         if ( this->state != GameState::PLAY )
152             return;
153
154         // Check if left/right arrow key is pressed
155         // We move the character here and not on event listening
156         // because we need to avoid the initial delay and eventually
157         // move the player smoother
158         const Uint8* keystates = SDL_GetKeyboardState(NULL);
159
160         // Left/Arrow arrow keystates
161         bool isLeftArrowDown = keystates[SDL_SCANCODE_LEFT] == 1;
162         bool isRightArrowDown = keystates[SDL_SCANCODE_RIGHT] == 1;
163
164         if ( isLeftArrowDown ) {
165
166             // Move player to the left
167             player->move(Player::DIR_LEFT, 0, this->sWidth, delta);
168
169         } else if ( isRightArrowDown ) {
170
171             // Move player to the right
172             player->move(Player::DIR_RIGHT, 0, this->sWidth - this->player->getWidth(), delta);
173
174         }
175
176         // Check if it's time to create a new apple
177         if ((this->appleTimer += delta) > this->roundTimer) {
178
179             // Create the apple
180             this->pushApple();
181
182             // Reset the apple timer
183             this->appleTimer = 0.0;
184
185             // Increase round counter and check if it's time to change the difficulty
186             if ( (this->roundCounter += 1) >= NEWDIFF_ROUNDS ) {
187
188                 // Create new apples faster
189                 this->roundTimer -= 115;
190
191                 // Reset counter
192                 this->roundCounter = 0;
193
194             }
195
196         }
197
198         // Update the apples
199         for ( auto& apple : this->apples ) {
200             apple->step(delta);
201         }
202
203         // Now check for collisions
204         // if apple collides with the player
205         SDL_Rect inter = { 0 };
206         if ( this->apples.size() > 0 && SDL_IntersectRect(&(this->player->getRect()), &(this->apples.front()->getRect()), &inter) == SDL_TRUE ) {
207
208             // Remove this apple
209             this->apples.erase( this->apples.begin() );
210
211             // Increase score
212             ++(this->score);
213
214             // Play hit sound
215             Mix_PlayChannel(-1, this->hitAppleSound, 0);
216
```

```
        if ( this->score >= this->targetScore ) {

            // Switch to win state
            this->state = GameState::WIN;

            // Play win sound
            Mix_PlayChannel(-1, this->winSound, 0);

            // Stop background music
            this->stopMusic();

        }

    }

    // If apple get to the ground, player loses life
    if ( this->apples.size() > 0 && this->apples.front()->getPositionY() > (sHeight - this->apples.front()->getHeight()) ) {

        // Remove this apple
        this->apples.erase(this->apples.begin());

        // Play hit sound
        Mix_PlayChannel(-1, this->hitAppleSound, 0);

        if ( (this->lifes -= 1) == 0 ) {

            // Switch to lose state
            this->state = GameState::LOSE;

            // Play lose sound
            Mix_PlayChannel(-1, this->loseSound, 0);

            // Stop background music
            this->stopMusic();

        }

    }

}

void GameWorld::render() {

    if ( this->state == GameState::START ) {

        // Render cover
        SDL_RenderCopy(this->renderer, this->cover, nullptr, &(this->coverRect));

        // Render play button
        this->playButton->render(renderer);

        // Render about button
        this->aboutButton->render(renderer);

        // Render exit button
        this->exitButton->render(renderer);

    }

    if ( this->state == GameState::PLAY || this->state == GameState::PAUSE || this->state == GameState::WIN || this->state == GameState::LOSE ) {

        // Render background
        SDL_RenderCopy(this->renderer, this->background, nullptr, &(this->backgroundRect));

        // Render the score
        this->drawScore();

        // Render life hearts
        this->drawHearts();

    }

    if (this->state == GameState::PLAY) {

        // Render the player
        this->player->render(this->renderer);

        // Render the apples
        for (auto& apple : this->apples) {
            apple->render(this->renderer);
        }

    }

    if ( this->state == GameState::WIN || this->state == GameState::LOSE ) {

        // Render the result message
        this->drawResultMessage(this->state == GameState::WIN ? "VICTORY" : "DEFEAT");

        // Render restart button
        this->restartButton->render(renderer);

        // Render menu button
        this->menuButton->render(renderer);

    }

    if ( this->state == GameState::PAUSE ) {

        // Render the result message
        this->drawResultMessage("PAUSED");

        // Render continue button
        this->continueButton->render(renderer);

        // Render restart button
        this->restartButton->render(renderer);

```

```
325             // Render menu button
326             this->menuButton->render(renderer);
327 
328         }
329 
330         if ( this->state == GameState::ABOUT ) {
331 
332             // Render about cover
333             SDL_RenderCopy(this->renderer, this->aboutCover, nullptr, &(this->aboutCoverRect));
334 
335             // Render back button
336             this->backButton->render(renderer);
337 
338         }
339 
340     }
341 
342     bool GameWorld::handleInput(SDL_Event evt) {
343 
344         if ( evt.type == SDL_KEYDOWN ) {
345 
346             if ( evt.key.keysym.sym == SDLK_ESCAPE ) {
347 
348                 switch ( this->state ) {
349 
350                 case GameState::PLAY:
351                     this->state = GameState::PAUSE;
352                     this->stopMusic();
353                     break;
354 
355                 case GameState::PAUSE:
356                     this->state = GameState::PLAY;
357                     this->playMusic();
358                     break;
359 
360                 case GameState::WIN:
361                 case GameState::LOSE:
362                     this->state = GameState::PLAY;
363                     this->reset();
364                     this->playMusic();
365                     break;
366 
367                 case GameState::ABOUT:
368                     this->state = GameState::START;
369                     break;
370 
371                 }
372 
373             }
374 
375         } else if ( evt.type == SDL_MOUSEBUTTONUP && evt.button.button == SDL_BUTTON_LEFT ) {
376 
377             switch ( this->state ) {
378 
379             // If the game is on start screen
380             case GameState::START: {
381 
382                 if ( this->playButton->isClicked(evt.button.x, evt.button.y) ) {
383 
384                     // Start the game
385                     this->state = GameState::PLAY;
386                     this->reset();
387                     this->playMusic();
388 
389                 } else if ( this->aboutButton->isClicked(evt.button.x, evt.button.y) ) {
390 
391                     // Show about screen
392                     this->state = GameState::ABOUT;
393 
394                 } else if ( this->exitButton->isClicked(evt.button.x, evt.button.y) ) {
395 
396                     // Exit the game
397                     return false;
398 
399                 }
400 
401                 break;
402             }
403 
404             // If the game is on play screen
405             case GameState::PLAY: {
406 
407                 // Do nothing, no buttons on play screen
408 
409                 break;
410             }
411 
412             // If the game is on pause screen
413             case GameState::PAUSE: {
414 
415                 if ( this->restartButton->isClicked(evt.button.x, evt.button.y) ) {
416 
417                     // Restart the game
418                     this->state = GameState::PLAY;
419                     this->reset();
420                     this->playMusic();
421 
422                 } else if ( this->menuButton->isClicked(evt.button.x, evt.button.y) ) {
423 
424                     // Go to menu
425                     this->state = GameState::START;
426                     this->reset();
427                     this->playMusic();
428 
429                 } else if ( this->continueButton->isClicked(evt.button.x, evt.button.y) ) {
430 
431                     // Continue the game
432                     this->state = GameState::PLAY;
```

```
                this->playMusic();

            }

            break;
        }

        // If the game is on win/lose screen
        case GameState::LOSE:
        case GameState::WIN:{

            if ( this->restartButton->isClicked(evt.button.x, evt.button.y) ) {

                // Restart the game
                this->state = GameState::PLAY;
                this->reset();
                this->playMusic();

            } else if ( this->menuButton->isClicked(evt.button.x, evt.button.y) ) {

                // Go to menu
                this->state = GameState::START;
                this->reset();
                this->playMusic();

            }

            break;
        }

        // If the game is on about screen
        case GameState::ABOUT: {

            if ( this->backButton->isClicked(evt.button.x, evt.button.y) ) {

                // Go to start screen
               this->state = GameState::START;

            }

            break;
        }

        }


    }

    return true;

}

Player* GameWorld::getPlayer() {

    return this->player;

}

Apple* GameWorld::getApple(int index) {

    return this->apples[index];

}

std::vector<Apple*> GameWorld::getApples() {

    return this->apples;

}

void GameWorld::pushApple() {

    // Add a new apple
    this->apples.push_back(new Apple(this->renderer));

    // Move it to a random X position
    this->apples.back()->setPositionX(this->getRandom(0, this->sWidth - this->apples.back()->getWidth()));

}

void GameWorld::reset() {

    // Reset everything
    this->player = new Player(this->renderer);
    this->player->setPosition(400, sHeight - PLAYER_WIDTH);
    this->apples = std::vector<Apple*>();
    this->score = 0;
    this->lifes = MAX_LIFES;
    this->appleTimer = 0.0;
    this->roundTimer = INIT_ROUND_TIME;
    this->roundCounter = 0;

}

int GameWorld::getRandom(int min, int max) {

    return min + (std::rand() % static_cast<int>(max - min + 1));

}

void GameWorld::drawScore() {

    std::string text = std::string("Score: ") + std::to_string(this->score) + std::string("/") + std::to_string(this->targetScore);

    this->textEngine->setColor(255, 0, 0);
    this->textEngine->draw(text, 10, 10, 24);

}
```

```
541
542 void GameWorld::drawResultMessage(std::string text) {
543
544     int x = this->textEngine->getHorizontalAlign(text, 48, this->sWidth);
545
546     this->textEngine->setColor(255, 0, 0);
547     this->textEngine->draw(text, x, 30, 48);
548
549 }
550
551 void GameWorld::drawHearts() {
552
553     for ( unsigned i = 0; i < this->lifes; ++i ) {
554
555         // Calculate x position
556         this->heartRect.x = this->sWidth - (3 * 32 + 10) + (i * 32);
557
558         // Draw the heart
559         SDL_RenderCopy(this->renderer, this->heart, nullptr, &(this->heartRect));
560
561     }
562
563 }
564
565 void GameWorld::playMusic() {
566
567     if ( Mix_PlayingMusic() == 0 ) {
568
569         Mix_PlayMusic(this->backgroundMusic, -1);
570
571     }
572
573 }
574
575 void GameWorld::stopMusic() {
576
577     if ( Mix_PlayingMusic() != 0 ) {
578
579         Mix_HaltMusic();
580
581     }
582
583 }
```

ΠΑΡΑΡΤΗΜΑ VIII: "DeltaClock" κώδικας

deltaclock.h

deltaclock.h deltaclock.cpp

```
1   #ifndef __DELTACLOCK_H__
2   #define __DELTACLOCK_H__
3
4   #include <SDL.h>
5
6   // A simple class to keep track of delta time
7   class DeltaClock {
8
9   public:
10
11      DeltaClock();
12
13      double tick();
14
15  private:
16
17      Uint64 currDelta;
18      Uint64 lastDelta;
19
20  };
21
22  #endif
```

deltaclock.cpp

deltaclock.h deltaclock.cpp

```
1   #include "deltaclock.h"
2
3   DeltaClock::DeltaClock() {
4
5       currDelta = SDL_GetPerformanceCounter();
6       lastDelta = 0;
7
8   }
9
10  double DeltaClock::tick() {
11
12      lastDelta = currDelta;
13      currDelta = SDL_GetPerformanceCounter();
14
15      return ((currDelta - lastDelta) * 1000 / static_cast<double>(SDL_GetPerformanceFrequency()));
16
17  }
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Feil J. and Scattergoo M., (2005), Beginning Game Level Design, Boston: Thomson Course Technology PTR

Dille F. and Platten J.Z., (2007), The Ultimate Guide to Video Game Writing and Design, New York: Lone Eagle Publishing Company

Adams E., (2010), Fundamentals of Game Design 2nd Edition, Berkeley: New Riders

Schell J., (2008), The Art of Game Design A Book of Lenses, Burlington: Morgan Kaufmann Publishers

Crawford C., (1997), The Art of Computer Game Design, Washington State University

Mitchell S.R., (2013), SDL Game Development, Birmingham: Packt Publishing

Pazera E., (2003), Focus On SDL, Cincinnati: Premier Press, a division of Course Technology

Onuiri E., Awodele O., Udegbe A., Adepoju B., Wakama O., Okoro R. and Komolafe O., (2015), Independent Game Development, Ogun State; Nigeria: International Journal of Advance Research, IJOAR.org

ΗΛΕΚΤΡΟΝΙΚΕΣ ΠΗΓΕΣ

KEIMENA

Wolfgang Kramer (2000), What is a game?, [Online],
Available: http://www.thegamesjournal.com/articles/WhatIsaGame.shtml

The Psychology of Game Play: Why games have become so popular among players of all ages, (15 December 2005), [Online], Available: https://woobi.com/the-psychology-of-game-play-why-games-have-become-so-popular-among-players-of-all-ages/

Ben Reeves, (2012), Why We Play: How Our Desire For Games Shapes Our World, [Online], Available:
https://www.gameinformer.com/b/features/archive/2012/11/20/why-we-play-how-our-desire-for-games-shapes-our-world.aspx

History.com Editors, (2017), Video Game History, [Online], Available:
https://www.history.com/topics/history-of-video-games

Riad Chikhani, (2015), The History Of Gaming: An Evolving Community, [Online], Available: https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/?guccounter=1

Evolution of video game industry, [Online], Available: http://history-of-video-games.webflow.io/

Heather Newman, (2017), The History Of Video Games, In One Infographic, [Online], Available: https://www.forbes.com/sites/hnewman/2017/11/29/the-history-of-video-games-in-one-infographic/#4dc0c2431a5c

Eli Epstein, Tech Time Machine: The Evolution of Gaming, [Online], Available:
https://mashable.com/2015/01/08/gaming-tech-ces/?europe=true#KpMCnDrzcsqo

Jeff Desjardins, (2017), How video games became a $100 billion industry, [Online], Available: https://www.businessinsider.com/the-history-and-evolution-of-the-video-games-market-2017-1

raulochoa1, History and Evolution of Video Games, [Online], Available:
https://www.timetoast.com/timelines/history-of-video-games-9c232bc5-1ea8-440e-bf4a-8e7470a9259c

Video Game History Timeline, [Online], Available:
http://www.museumofplay.org/about/icheg/video-game-history/timeline

Ted Stahl, (2005), Video Game Genres, [Online], Available:
https://www.thocp.net/software/games/reference/genres.htm

Vince, (2018), The Many Different Types of Video Games & Their Subgenres, [Online], Available: https://www.idtech.com/blog/different-types-of-video-game-genres

Complete List of Game Genres, (18 October 2017), [Online], Available: https://mirillis.com/blog/en/complete-list-of-game-genres/

Different types of video game platform popular today, (31 May 2016), [Online], Available: https://pcdreams.com.sg/different-types-of-video-game-platforms-popular-today/

Michael Klappenbach, (2018), Choosing The Video Game Platform Best For You, [Online], Available: https://www.lifewire.com/choosing-video-game-platform-811337

What Is Video Game Development, [Online], Available: https://www.internationalstudent.com/study-video-game-development/what-is-video-game-development/

Mark Alexander, (2015a), The Foundations Of Good Game Design - Part 1, [Online], Available: https://www.yoyogames.com/blog/121/the-foundations-of-good-game-design-part-1

Mark Alexander, (2015b), The Foundations Of Good Game Design - Part 2, [Online], Available: https://www.yoyogames.com/blog/122/the-foundations-of-good-game-design-part-2

Ashish, (2018), Why Do Video Games Have ‘Loading Screens'?, [Online], Available: https://www.scienceabc.com/innovation/why-do-video-games-have-loading-screens.html

ΕΙΚΟΝΕΣ

Εικόνα 1: Video games https://thenextweb.com/contributors/2018/03/15/two-worlds-meet-thrive-unification-video-games-blockchain/

Εικόνα 2: Minecraft https://www.planetminecraft.com/project/eiffel-tower-schematic/

Εικόνα 3: The Sims 4 https://simsvip.com/2016/03/02/sims-4-gallery-spotlight-27/5/

Εικόνα 6: The Wolf Among Us http://www.game-over.com/content/2013/10/the-wolf-among-us-episode-1-faith/

Εικόνα 7: Mouse in the Maze http://database-of-characters.wikia.com/wiki/Mouse_in_the_Maze

Εικόνα 8: Spacewar https://www.thoughtco.com/history-of-spacewar-1992412

Εικόνα 9: Pong https://www.youtube.com/watch?v=YmzH4E3x1_g

Εικόνα 10: Space Invaders http://www.nigelwdavies.com/2018/08/killing-space-invaders-tango-goddess/

Εικόνα 11: Pac-Man https://www.youtube.com/watch?v=-CbyAk3Sn9I

Εικόνα 12: Tetris https://www.deviantart.com/sovietrussiaftw/art/Tetris-1989-Game-Boy-314916346

Εικόνα 13: Sonic the Hedgehog https://www.retrogames.cz/play_117-Genesis.php

Εικόνα 14: DOOM https://www.gamepressure.com/download.asp?ID=59098

Εικόνα 15: Half Life https://steamed.kotaku.com/half-life-npcs-can-smell-corpses-1818824392

Εικόνα 16: Halo Combat Evolved https://www.imdb.com/title/tt0309654/mediaviewer/rm12405504

Εικόνα 17: The Elder Scrolls IV https://www.mobygames.com/game/windows/elder-scrolls-iv-oblivion-game-of-the-year-edition-deluxe/promo/imageType,1/promoImageId,39258/

Εικόνα 18: Far Cry 4 https://www.geforce.com/games-applications/pc-games/far-cry-4/screenshots

Εικόνα 19: Tomb Raider https://www.reddit.com/r/gaming/comments/8o0qfx/tomb_raider_1996_vs_2017/

Εικόνα 23: Street Fighter https://www.mightyape.co.nz/product/street-fighter-v-pc-games/22996152

Εικόνα 24: Castle Crashers http://minimap.net/game/castle-crashers/review

Εικόνα 25 Hitman: https://www.dailymotion.com/video/x3rqjml

Εικόνα 26: 7 Days to Die https://www.gameskinny.com/zei01/7-days-to-die-beginner-tips-and-tricks

Εικόνα 27: Guitar Hero https://valleybugler.com/geek-speak/rock-like-guitar-hero/

Εικόνα 28: Resident Evil Revelations https://store.steampowered.com/agecheck/app/222480/

Εικόνα 29: Dead Cells http://www.gamers.at/pc/dead-cells-angespielt-22138/attachment/dead-cells-1

Εικόνα 30: Eric the Unready https://www.mobygames.com/game/dos/eric-the-unready/screenshots/gameShotId,733306/

Εικόνα 31: The Riddle of the Sphinx https://www.youtube.com/watch?v=YRKlfC7OUcs

Εικόνα 32: Clannad https://www.theodysseyonline.com/review-clannad-visual-novel

Εικόνα 33: Until Dawn https://www.kotaku.com.au/2015/10/until-dawn-is-much-shorter-when-youre-bad-at-quick-time-events/

Εικόνα 34: The Witcher https://www.youtube.com/watch?v=oiLMkf1COP0

Εικόνα 36: Spelunky https://store.steampowered.com/app/239350/Spelunky/

Εικόνα 37: Tactical Monsters https://store.steampowered.com/app/705220/Tactical_Monsters_Rumble_Arena/

Εικόνα 38: Prison Architect http://prison-architect.wikia.com/wiki/3D_Mode

Εικόνα 39: Spore http://spore.wikia.com/wiki/Early_Creature_Editor

Εικόνα 40: Eurotrack simulator Eurotrack simulator https://www.nuuvem.com/item/euro-truck-simulator-2-cabin-accessories

Εικόνα 41: Sid Meier's Civilization V https://store.silagames.com/game/1427/sid-meiers-civilization-v-the-complete-edition/

Εικόνα 42: Worms Reloaded https://steamcdn-a.akamaihd.net/steam/apps/22600/ss_5c4b7446d3d268cfbabd67fa0300e95cc98b9f5a.jpg?t=1478121750

Εικόνα 43: Age of Empires III https://superior-realities.com/2013/12/23/retro-review-age-of-empires-iii/

Εικόνα 44: Crusader Kings https://3dnews.ru/969957

Εικόνα 45: Dota 2 https://www.tuxboard.com/top-10-gamers-argent/dota-2-jeu-video/

Εικόνα 46: Kingdom Rush https://store.steampowered.com/app/246420/Kingdom_Rush/

Εικόνα 47: Gran Turismo 6 https://www.gamespot.com/articles/gran-turismo-6-passes-5-million-sales-series-climb/1100-6442540/

Εικόνα 48: Pro Evolution Soccer 2015 http://www.zougla.gr/games/article/pro-evolution-socer-2015-demo-ke-sta-pc

Εικόνα 49: The Room https://www.youtube.com/watch?v=BrDfTuNKQog

Εικόνα 50: Jeopardy https://www.youtube.com/watch?v=QZFOg6napqU

Εικόνα 51: Plants vs Zombies http://iosuniverse.com/apps/plants-vs-zombies-review-ios/

Εικόνα 52: AdVenture Capitalist https://www.youtube.com/watch?v=bffAclcS9_M

Εικόνα 53: Mysterium http://www.pocketgamer.co.uk/r/iPad/Mysterium/review.asp?c=72708

Εικόνα 54: Codingame https://www.youtube.com/watch?v=p8oCyM-ZjxA

Εικόνα 55: Big Thinkers https://www.wingamestore.com/product/4863/Big-Thinkers-Kindergarten/

Εικόνα 56: Game Design https://www.yoyogames.com/blog/121/the-foundations-of-good-game-design-part-1

Εικόνα 57: SDL https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer