Data Structure Deadheads

Lock Out Protocol

Last Date Modified: December 10, 2019

Authors: Ronald Abrams, Rinty Chowdhury, Joshua Crotts

Stakeholders: Data Structure Deadheads, Ike Quigley, CSC-340, UNC Greensboro

We have abided by the UNCG Academic Integrity Policy on this assignment.

Contents

1 Introduction

1.1 Purpose

Video games have existed since the early forties on some of the most primitive military technology to date. Ever since then, video games have evolved and transitioned to what we have today: modern three-dimensional graphics that astound kids and adults alike with their realism. However, for this game, we take a step back and experiment with the two-dimensional over-head style. These types of games are easy to pick up and play for a while, and most importantly, they are incredibly fun for most users, not just gamers.

1.2 Document Conventions

GUI - Graphical User Interface
SQL - Structured Query Language
HUD - Heads Up Display
API - Application Programming Interface.

1.3 Intended Audience

We want our game to be accessible by many types of users; not just hardcore gamers. Casual users, adults, and teenagers should all be able to enjoy this game. While it is on the more violent side of video games, it should still be casual enough for those who are not always on the edge of their toes, but intense enough to still give veteran gamers a run for their money.

1.4 Definitions

Video Game - Entertainment based product run-able on a computer.
Gamer - Any individual who utilizes video games on a common basis.
SQL Database - A relational database that stores information across multiple tables.
JRE - Java Runtime Environment
JDK - Java Development Kit
Standards - Medium/low-level Java gaming library created by Joshua Crotts & Andrew Matzureff.

Sprite - 2D representation of an image, typically resembling some object within the game such as a player, etc.

1.5 Project Scope

This 2D video game is intended to provide an enjoyable experience to a wide audience range, as well as incorporate a MySQL cloud storage for save and load functionality. For this MySQL database, we plan to use Google Cloud's hosting services, but without their authentication API. For the purposes of this project, it would be too intricate and moreover superfluous. With this in mind, for account validation, and serialization/deserialization, we will use the JBCrypt library. It is a straightforward yet powerful library to not only hash passwords, but to also salt them prior to storing them in the database.

Furthermore, this game will also support connections to the AWS IP Finder API, the IPStack Geolocation API, and the OpenWeatherMap API, which, in tandem, are used to generate certain graphical changes within the game. In addition to these, we plan to add support for various languages in the game. Writing custom language support for this game will be difficult in and of itself, so we rely on the Yandex API to translate our language given some string of text.

The intended product should support single-player interaction, and also store several option settings locally, such as resolution, game language, and volume. The product itself it will exist on a user to computer-application standpoint, and be as an individual product per each computer it is installed on. The game is intended to fit within the shooter video game genre, and is currently slated to be a 2D top-down video game. A library implemented by Joshua Crotts and Andrew Matzureff will also need to be incorporated for game engine functionality, but physics based interactions and the GUI structure will still need to be designed by the predominant functional code base. The total designated time for this product's completion is approximately three months, but it is expected that the game's minimum functional criteria will be completed after approximately two months into development. Any additional time remaining afterwards will go towards incorporating any supplementary optional features.

1.6 Technical Challenges

For starters, the primary difficulty in this project is the integration of the physics and collision detection. Secondary difficulties include cloud-save implementation and meaningful connection to the three APIs. For this game, we are using a library

provided by Joshua Crotts and Andrew Matzureff: [Referred to from here on out as Standards.] Standards abstracts much of the core logic and semantics behind game creation, so we can focus more on the overall design, appearance and gameplay, rather than investing time on how to create a perfectly micro-efficient game engine. Even with this, however, we will still need to write some of the physics for specific objects. Moreover, there exists the technical difficulty of connecting to a MySQL cloud database, thus allowing users to save and load their progress. So, if the user wants to play the game on a different computer without having to lose all their progress, they can create an account in the cloud, and tie their save data to their account. There are two main possibilities for hosting: Google Cloud, and Amazon Web Services. Both have their blatant pros and cons, and making sure there is a secure connection between the client and the server is very important (even though there's no sensitive information within the accounts, such as bank accounts or credit card numbers). Lastly, there's the issue with language localization. Many non-English character languages (such as Bengali, Arabic, Korean, etc.) are not supported by many custom fonts (in particular, the one used in the game). Therefore, we have to use a traditional font to ensure these fonts are renderable on the screen. However, even with this workaround in mind, there still exist languages that are unable to be displayed properly. Unfortunately, there is no available fix at this time.

1.7 References

As of the last date modified, no other articles are referenced within this SRS Doc.

2 Overall Description

2.1 Project Features

The game will contain many of the basic features found in similar video games; the core of which includes much of the following functionality:

- Fully supported single-player gameplay
- Keyboard+Mouse movement in a 2D world space
- Interaction with enemies and environment in said world spacet
- User account creation capability

- User login capability
- Ability to start a new game within the executable
- Ability to load or save games within the executable
- Ability to save and modify option settings within the executable
- Ability to choose avatar appearance within the game
- Ability to select difficulty level within the game
- Ability to pause the game's status while actively not in menus
- Capability of properly closing and exiting the application safely

2.2 User Characteristics/Classes

There will only be two types of access-able user from the product executable itself, that being the Admin User and the Player User. The Player User will be any user playing the game, and will not have any access to the code or under-the-hood mechanics existing in the video game's code-base. The user will only have access to play the video game as intended. The core gameplay scheme being a New Game, where enemies will attack the player once they load in. Said computer opponents will be built based on the game duration and difficulty level selected by the user. In contrast, the Admin User will have access to all the functionality of the Player User, but will additionally be able to alter the game's properties in a limited fashion while running the executable, as well as having access to debug-based commands for the purposes of stress-testing the game's systems.

2.3 Operating Environment

This particular product is intended to work on Windows 10, and ideally will feature backwards compatibility to at least Windows 8. Additional support is desired for both MacOS X, and any Linux distribution.

2.4 Design & Implementation Constraints

The product will be made utilizing Java Swing for the Game Engine, and will make use of MySQL commands for save data storage over Google Cloud. The game itself will make use of a functioning library created by the developers of this product

for it's implementation. The game will also make use of an external API called OpenWeatherMap API, whose commands will be incorporated into the game to check what the weather is like currently in the real world. Furthermore, the game will connect to the Amazon Web Services API, which returns their IP address. The user's IP address is sent to a separate API: IPStack for parsing. IPStack uses geolocation to give an approximation as to where the user is in the world. After this, OpenWeatherAPI is called with the aforesaid city.

2.5 Assumptions & Dependencies

The product will require a MySQL database, stored via a cloud-based system such as Google Cloud. It will also make use of OpenWeatherMap API for limited weather-tracking effects in-game via graphical change. In order for this product to continue functioning, it would require that such systems will remain in existence and are supported into the distant future.

3 Functional Requirements

3.1 Primary

The core gameplay requirements that must be implemented are as follows: Login account creation access-ability, save / load game functionality, single-player new game functionality, connection to external OpenWeatherMap API, working win-loss conditions for the game, and lastly; appropriate game executable closing procedure.

3.2 Secondary

Additional secondary feature considerations include: full options and settings modification via the user interacting with the GUI, multiple difficulty modes for enemies, multiple player customization options for the user, randomized levels or stat distribution for the game itself, high-level functioning artwork, advanced computer opponent intelligence, connection to Amazon Web Services API for IP checking, utilization of IPStack for real world position checking, linkages between the previous two mentioned API's and the OpenWeatherMap API, etc.

4 Technical Requirements

4.1 Operating Systems & Compatibility

This game will work on any modern operating system, such as Microsoft Windows 10, 8.1, 8, or 7, MacOS X, and any Linux distribution. However, due to the nature of some timing issues within the JDK and on UNIX-based systems, some of the animations will appear to have different timings on non-Windows computers. These cosmetic issues do not affect the gameplay; only the appearance.

4.2 Interface Requirements

4.2.1 User Interface

This product will support Mouse+Keyboard interaction from the user. The user will interact with the product via GUI Panels ran during the product's executable. As of current no other controller/input schemes are intended to be supported.

4.2.2 Hardware Interface

This product will require some basic standards from a computer, such as a functioning screen, Mouse+Keyboard, and some means to connect to the Internet.

4.2.3 Software Requirements

The primary caveat is the user must have Java 8 installed. Oracle no longer supports the downloading of legacy versions of Java. Therefore, the primary alternative to Oracle's Java is either OpenJDK 8, or Amazon Corretto's JDK 8.

4.2.4 Communications Interface

Connection to a user's account and save files located on the cloud will be handled over the internet utilizing the HTTP protocol. Further, using GET requests, we contact the three primary APIs, retrieve information, and process their returned information in the JSON format.

5 Non-functional Requirements

5.1 Performance Requirements

The game is projected to require the following System-Requirements in order to operate appropriately:

OS: Windows 7, 8/8.1, 10; MacOS X; Linux Mint
Processor: Dual Core 3.0 Ghz or Greater
Memory: 4GB or Greater
Hard Disk Space: 10 GB
Video Card: 256mb Video Memory, capable of Shader Model 2.0+
JRE: 8.0 Release; OpenJDK 8, or Amazon Corretto's JDK 8.

5.2 Safety & Recovery Requirements

Any breakdown or damage to the product's code-base, either due to user-interference, application crash, desktop crash etc, could cause save file corruption if done in the middle of a save. It can also cause the product to become unstable, and may require a full re-installation of said product on the computer to reset / fix any lingering issues. Any such reset should not negatively affect the ability to log back into a previously created account or access previous account cloud-stored saves.

5.3 Security Requirements

The product can be installed as is, but will require an account from the user's end in order to play the game. Said account will require a username and password, and this information would need to be stored securely on the cloud or via a separate database.

5.4 Policy Requirements

The game will come with an internal EULA, which the user will need to agree to during the account creation in order to make said account. The user must effectively agree to the terms of service listed within the EULA, as they will be unable to play the game otherwise.

5.5 Software Quality Attributes

5.5.1 Availability

The game should always be able to be played by the user once installed on the target machine. The only exceptions to this rule would be if they did not possess internet connection capacity, or if the Cloud service was temporarily down, or if the user was unable to access their account.

5.5.2 Correctness

The game's executable should always open from the Account Login screen, go into the Main Menu, and be ran all the way until the Active Game screen. When the user clicks Exit Game, or if the user force terminates the program, the game should terminate and close properly without leaving any of its background processes still running.

5.5.3 Maintainability

The developers will need to maintain the cloud service that stores the relevant account and save data information.

5.5.4 Reusability

The product can be installed and uninstalled within the same computer as many times as is needed. [Tentative]

5.5.5 Portability

The product should be able to be installed and ran on any computer or laptop that meets the Performance Requirements specified within this document.

5.6 Process Requirements

5.6.1 Development Process

The development of this product will start first with completing all the necessary wire-frames and documents required to solidify the creation process of the product. Time will be allocated towards learning difficult system functionality, and then priority will be placed into the feature list specified in the Primary Requirements section of this document. Any additional time will be put towards the Secondary Requirements also specified in this document.

5.6.2 Time Constraints

The product will be developed for approximately 4 months, starting from an initial date of 8/20/2019, to 12/10/2019.

5.6.3 Cost & Delivery Date

The product will have an effective cost of zero to the consumer. We have purchased several asset packages (images, sprites) totaling to $10. Its final deployment is to occur on 12/10/2019.