Software Requirements Specification

for

“Drago’s Inferno”

Version 1.4 approved

Prepared by: Elizabeth Collins

Mike Enriquez

Scott Hodge

Bryan Tanner

John Thrasher

Little Endian Entertainment, LLC

19 March 2013

Table of Contents

Table of Contents ii

Revision History iii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Document Intended Audience and Reading Suggestions 2

1.4 Product Scope 2

1.5 References 3

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions 4

2.3 Game Intended Audience 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 5

2.6 Available User Documentation 5

2.7 Assumptions and Dependencies 5

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 6

3.3 Software Interfaces 6

3.4 Communications Interfaces 7

4. System Features 7

4.1 Game Start 7

4.2 Game End 9

4.3 Playing Game 11

4.4 Attacks and Avoidance 12

4.5 Projectiles 14

4.6 Collision Detection 15

4.7 A\* Pathfinding 16

4.8 Heads Up Display 17

4.9 Audio Effects 19

4.10 Peripheral Input 19

4.11 Binary Space Partitions 20

4.12 Dynamic Portals 20

5. Non-Functional Requirements 21

5.1 Performance Requirements 21

5.2 Safety Requirements 22

5.3 Security Requirements 22

5.4 Software Quality Attributes 22

5.5 Business Rules 24

6. Future Requirements 24

6.1 Health, Life and Death 24

6.2 NPCs 26

4.10 Visual Effects/Graphics 26

Appendix A: Glossary 27

Appendix B: Analysis Models 28

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Bryan Tanner | 12/3/2012 | Consolidating sections of SRS from individuals | 1.0.0.0 |
| John Thrasher | 12/5/2012 | Proof reading and editing | 1.1.0.0 |
| Bryan Tanner | 12/6/2012 | Final changes and preparation for submission | 1.2.0.0 |
| Mike Enriquez | 3/19/2013 | Revision from grading comments | 1.3.0.0 |
| Liz Collins | 5/2/2013 | Final revisions in preparation for submission | 1.4.0.0 |
| Bryan Tanner | 5/2/2013 | Corrected game control keys in section 3.1 | 1.4.1.0 |

# Introduction

## Purpose

This document provides for the recommended requirements, specifications, and designs for the implementation of the initial release (version 1.0) of the game “Drago’s Inferno”. The document is a report on the initial review of the “Drago’s Inferno” project and recommendations of the game designs, animations, graphics, controls, software, and hardware requirements to fully implement the game. The recommendations provided in this document are for all parties involved in the implementation and development of “Drago’s Inferno”, more specifically for the Little Endian Entertainment, LLC, working group.

## Document Conventions

The document conventions of this SRS are outlined in the Table of Contents, and each higher level requirement shall inherit attributes from the detailed sub-requirements.

All section headings shall be numbered by the next conventional whole number and appear in bold face, “Times”, number 18 font.

Example:

# 1. Introduction

All subsection headings shall be numbered by the heading number, a period, and the next conventional subsection number and appear in bold face, “Times”, number 14 font.

Example:

**1.1 Purpose**

Further requirements to a subsection shall be indented, and numbered by the heading number, a period, the subheading number, a period, and the next conventional requirement number and appear in “Times”, number 12 font.

Example:

4.1.1 Description and Priority

In Section 3, red text denotes keys on the keyboard, while green text denotes actions that can be triggered by key presses.

Requirement Prefixes/Naming Convention

The functional and non-functional requirements shall be presented with the following format in accordance with other Little Endian Entertainment Games and Applications:

Game Prefix \_ Game Play Functionality – Requirement Number

Example:

Game Prefix

Drago’s Inferno 🡪 DI

Game Play/Section

Game Start 🡪 GS

Requirement Number

1

DI\_GS-1

## Document Intended Audience and Reading Suggestions

This document is intended to provide the project manager and the customer with the necessary details to assess the development of the Drago video game. In addition, it provides documentation for users to understand the features and settings of the game. Information is also provided to allow game developers to understand the classes and methods used in this software so that they can make improvements or assess existing features.

Section 1 of this document contains general information regarding the scope of the software and who it is intended for. Section 2 generalizes the software functions and user features of the game and the operating environments it was intended for. Users will find Section 2.3 and 2.6 helpful for determining the available features for different types of users and the resources provided by the designers. The license holder for this game will require information from sections 2.1, 2.2 and 2.7 to enhance their knowledge of the product design and assist in marketing for the game. Developers can use sections 2.1-2.5 and 2.7 to familiarize themselves with the inner-workings of this product and its basic feature outline.

Section 3 details the hardware and software needed for the game to function with. Section 3.1 also gives some more detail about how the user will interact with the game. This section will be most helpful to users or other game developers if they would like to make improvements to the user interface. Section 3.2 and 3.3 detail what hardware and software the user will need in order for the game to function properly.

Section 4 goes into details regarding the features of the program. The features are detailed and the priority of each is listed. This will allow future developers to change features to improve the game. Some features may need to be enhanced to improve the user’s experience or some features may need to be removed if their functions are no longer needed in future releases of the game.

Finally, Section 5 goes through some non-functional requirements, such as performance, safety and security requirements. This section is not intended for users. It is primarily for the customer to see that their requests have been met in the creation of this software. Future developers can also gain some insight from these areas.

References are provided in Section 1.5 so that the reader can find additional details for the information discussed in this document. These references will provide educational information regarding some of the functions employed in this software as well as specific features discussed in this document.

## Product Scope

This game has been developed to entertain the casual gamer who enjoy medieval or adventure type games. It designed to be reminiscent of a classic style, 2-D adventure game similar to Zelda. The objective of the game is for the main character (Drago) to find all of the stolen dragon eggs and save them from the humans who have captured them. The software needs to be enjoyable and engaging in order to draw in users of all ages. The purpose of the software is solely for its entertainment value. There will be no educational functions or add-ins for this game, however, the content will be appropriate for all ages.

## References

Brackeen, D. (2003). *Developing Games in Java*.  
Retrieved December 2012, from www.brackeen.com:  
<http://www.brackeen.com/javagamebook/>

(2003). In D. Brackeen, B. Barker, & L.   
Vanhelsuwe, *Developing Games in Java* (p. 1008). New Riders.

(2005). In A. Davison, *Killer Game Programming*  
(p. 998). O'Reilly.

Deleon, C. (n.d.). *2D Platformer Advanced  
Collision Detection*. Retrieved December 2012, from www.hobbygamedev.com:  
<http://www.hobbygamedev.com/adv/2d-platformer-advanced-collision-detection/>

Lambert, S. (2012, September 3). *Use Quadtrees to  
Detect Likely Collisions in 2D Space*. Retrieved December 2012, from  
gamedev.tutsplus.com:  
[http://gamedev.tutsplus.com/tutorials/implementation/quick-tip-use-quadtrees-to-detect- likely-collisions-in-2d-space/](http://gamedev.tutsplus.com/tutorials/implementation/quick-tip-use-quadtrees-to-detect-%20%20%20%20%20%20%20%20likely-collisions-in-2d-space/)

*System Requirements for Java 7.* (n.d.). Retrieved December 2012, from www.java.com:  
<http://www.java.com/en/download/help/sysreq.xml>

# Overall Description

## Product Perspective

“Drago’s Inferno” is an overhead, exploratory-based game that may resemble more popular games from the Super Nintendo Entertainment System era (circa 1990). Some games that may present similar gameplay include *The Legend of Zelda: A Link to the* Past (1991), *Final Fantasy III* (1990), and *Chrono Trigger* (1995).

*The* *Legend of Zelda: A Link to the Past* (commonly referred to as ALTTP)is known as one of the greatest video games of all time. It was a classic that captured the hearts of many kids, and even adults, with its release in 1991. “Drago’s Inferno” wants to get back to the roots of video games and present a simple style of gameplay to the user with diverse and challenging experiences. Like *ALTTP,* “Drago’s Inferno” only presents itself as a simple game. The underlying mechanics are what make the experience enjoyable.

“Drago’s Inferno” will be an independent software project but will borrow several elements from these classic games. The overhead view will be familiar to fans of classic games, along with the user’s ability to roam freely around the world, discovering treasures, fighting enemies, and gaining useful abilities. In addition to these classic elements, “Drago’s Inferno” will feature elements that weren’t made available in games of the SNES console era. For instance, physics will allow the player to fly, slide on ice, and carry momentum.

Since “Drago’s Inferno” is developed in Java, there are no software interfaces other than a machine running Version 1.4 of the Java Virtual Machine (JVM). The application will need to be downloaded and run as a desktop application.

## Product Functions

“Drago’s Inferno” allows the player to gain new abilities as the game progresses, allowing for more user interaction and functions to be implementing as the game wears on. Some of the functionalities provided to the user include:

* Provides a graphical user interface for the game to be played on;
* Actions are taken according to the user controls the user has pressed;
* Allows directional movement of the on-screen character through input from the user;
* Allows the user to attack enemy characters through multiple attack style;
* Allows the player to choose abilities to power-up for their character;
* Player has the ability for limited flight during the game;
* Player has the ability to save and load game states;

These are some of the basic functions the player will be able to execute. For more detail of these functions see Section 4.

## Game Intended Audience

“Drago’s Inferno” is a product that could be enjoyed by anyone. The classic games allowed a wide range of users to enjoy the gameplay through simple gameplay techniques, but also allowed additional features to keep the experienced gamers entertained. “Drago’s Inferno” aims to show younger generations that classic style games can be fun, while allowing older generations to reminisce of the glory days when Nintendo was the only game system that people knew about.

While the interface for gameplay remains simple, “Drago’s Inferno” will best be enjoyed by users who are familiar with platform games and know their way around a keyboard. Some gameplay requires fast-action responses made by the user, allowing for an exciting experience. The most tangible group expected to play “Drago’s Inferno” are the PC gamers who scour the internet in search of a new gaming experience and respect the work put in to achieve a great game. This group of users allow for games to become popular by word of mouth, recommended the ‘must-plays’ to friends, while warning to avoid the ‘rotten tomatoes’ or ‘knock-offs’.

“Drago’s Inferno” aims to show the newer generation of gamers the roots of the present-day games they enjoy. Today’s games revolve around the latest graphics, sophisticated control schemes, 3D layouts, and abilities that the player may never even know exists. By presenting a simple, yet challenging game, this group will be searching for similar games for an enjoyable experience.

## Operating Environment

The software will operate under any framework that supports the Java interpreter. Specifically, Java 7 requires a computer with Windows 7, Windows Vista, Windows Server 2008, or Windows XP 64-bit and 128MB of RAM or Windows XP 32-bit, Mac OS X, or a variety of Linux distributions and 64 MB of RAM.

## Design and Implementation Constraints

The software will be developed in the Java programming language and distributed as freeware. It will function as a stand-alone application requiring no access to external databases or to the Internet. At this time, there are no specific hardware or memory requirements, although we expect the requirements to be similar to the base requirements of Java.

## Available User Documentation

Drago’s Inferno Game Manual

Software Installation process

Game Loading and execution process

Basic game play operations – player movement, attacks, and defenses

User interface description – including screen shots of health meter, score, remaining lives, etc.

Listing of basic enemies

Listing of allies

Listing of levels – screen shot of menu map

Game saving process

Game Hints

Troubleshooting

Drago’s Inferno On-line help

Access to Game Manual

Game tutorial

FAQ

Bug and error reporting

Drago’s User Blog

Game play tips/hints

Suggestions

## Assumptions and Dependencies

The software will be developed on the Windows platform with the Eclipse™ editor and possibly other editors. Software graphics will be developed as \*.GIF images using a variety of image editors including the Photoshop™ editor. Graphics can be drawn from the ground up in the editor but may also be obtained from scanned drawings. Sound effects will be included, possibly as \*.WAV files, and will be partly obtained from public domain Internet sources, partly from sound effect generation utilities, and partly self-recorded by the development team.

# External Interface Requirements

## User Interfaces

The user interfaces of “Drago’s Inferno” will mostly be presented on screen in the form of game environments and menus. This is how the user will interact with the game and know what is happening throughout the game. The environments, or game levels, on screen will vary greatly with different textures, colors, themes, and layouts, as can be expected from an exploration game. The menus will consist of lists that give the user options that change game settings or carry into another menu. The main menu, which pauses the game when brought up, will consist of settings for sound, display, and save states. Each of these menus will feature customization options such as sound level, brightness, and lists of saved games, respectively.

Another important interface is how the user is informed of what is going on in the game. “Drago’s Inferno” will include text dialog on-screen that displays important information about the game’s story or about the player. The user will be notified when abilities are gained, when the character is killed, when items are attained, and other such interactions. The text dialog will also be used to entertain the user with the events of the story in the game. Through this means, the user will understand the player character’s motives, values, and personality. There will also be graphical representations of player data, such as remaining health, included on-screen at all times.

In addition to the main playing screen and the main pause menu, there will also need to be screens displayed before the game has begun (“title screen”) and after the player has died (“game over screen”). The title screen will display a menu that allows the user to start a new game, load a saved game, or customize many of the settings available in the pause menu. The game over screen will display a menu that allows the user to load from the most recently saved game, load any other saved game, or return to the title screen.

Hardware interfaces are another vital part in playing “Drago’s Inferno”. This interaction will be provided by a standard keyboard. The actions of the player in the game will be programmed to respond according to key presses. These actions will come with a default set up that the user will be able to customize to better suit gameplay. The default controls will use the up, left, down, and right keys for movement accordingly. The controls for player actions will default to Z, X, C, V for a standard attack (either melee or shooting a projectile), Q and E for special abilities, and space for jump. The special abilities will change throughout gameplay, depending on what abilities the player has unlocked and chosen. Enter will be used to select choices from menus, while P will be used to pause the game. S will slow down character movement in the game and F will speed up character movement.

## Hardware Interfaces

The hardware interface is the heart of every game. The hardware allows the user to translate thoughts and strategy into actions in the game world. The user communicates through a keyboard to interact with the game. Java is a great language for user interaction. Java has built-in software interfaces to help the programmer with responding the key presses from a keyboard. The way this is done is by implementing Java’s KeyListener Interface. This notifies the program when a key has been pressed. As programmers, it is then our job to decide which key was pressed so the correct action can be taken accordingly. This will be handled using a switch statement since switches are more efficient for this job than if-else statements. Once the key has been determined, the correct action ensues; for example, pressing W will move the character up on the screen.

## Software Interfaces

Java is a versatile language. This allows the game to operate on multiple operating systems. Java is also a great game development platform because of the powerful, built-in graphical libraries. This allows us to reduce outside dependencies resulting in a more robust and modular, and overall more independent, application.

While there is no database interaction, information about the game such as save states will be stored in a flat-file. This allows information to be stored and retrieved at any time by the user. This interaction is achieved by saving the state of each instance at a given time that can be resumed or reset. Information about the player (current weapons, health, level, etc.), enemies, and location are all relevant in this process. This is achieved by storing and accessing the local file system, stored at a location selected by the user.

The levels for “Drago’s Inferno” will be created in a separate level editor, so the game will need to be able to interact with the files written by the editor. In order to facilitate this, we will use a level editor, such as Ogmo, that allows for XML output and the game will be built with the expectation that it will need to interface with XML files.

## Communications Interfaces

Currently, we do not foresee any need for the game to use any communications functions.

# System Features

## Game Start

### Description and Priority

This feature refers to the user’s ability to play a game. Once the game application has been properly loaded and run, the available user options on the Game Start Display will be presented in a menu and selected by the arrow keys and return key. Game Start is a necessary function of the game to allow users to begin a new game or continue an existing game. Game Start will also allow for customization of the game play options in order to provide the maximum user enjoyment. This is a high priority feature.

### Stimulus/Response Sequences

In order to begin a game, the user will have to select the game from their desktop and run it. Once the program is running, the initial menu will be displayed and the user can select to start a new game or to open a saved game. When beginning a new game, the player will be asked to enter a character name so that the progress of this game can be saved under that name. The player will then be asked to select the level of difficulty at which they wish to play the game (easy/difficult). When opening a saved game, the user will be given a list of the character names of the currently saved games. The user will then be able to select a name from the list in order to begin that game.

### Functional Requirements

*Game Start 🡪 GS*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_GS-1 | 2 | The Main Menu shall be presented, once the application is run. |  |
| DI\_GS-2 | 2 | The Main Menus shall include the following available options: |  |
| DI\_GS-3 | 3 | Table DI\_GS-3 –Main Menu Options |  |
|  |  | |  |  | | --- | --- | | Option | Description | | Game Start | Begin a new game | | Load Game | Load an existing game, from moment of last save | | Options | Available game options: Sound, brightness, Show Blood | | Statistics | Previous Game Stats | | Quit | Exits the application | |  |
| DI\_GS-4 | 2 | The Main Menu options shall be selected using the up/down Arrow Keys. |  |
| DI\_GS-5 | 2 | The Main Menu shall default to New Game when the Application is run. |  |
| DI\_GS-6 | 2 | A selection shall be made when the user presses the enter key. |  |
| DI\_GS-7 | 2 | If the user selects *New Game*, the application shall launch into the background story. |  |
| DI\_GS-8 |  | The User shall be prompted for a profile name for the current game session. |  |
| DI\_GS-9 | 2 | The User shall be prompted for the desired difficulty level. |  |
| DI\_GS-10 | 2 | The Background Story shall be completed before the game play is possible. | The Game Intro/Background is not able to be skipped |
| DI\_GS-11 | 2 | If the User selects *Load Game*, the application shall present a list of available saved games. |  |
| DI\_GS-12 | 3 | The list of available games shall be selected using the up/down arrow keys. |  |
| DI\_GS-13 | 3 | The load saved game selection shall be executed when the User presses the enter key. |  |
| DI\_GS-14 | 3 | If the saved game load is successful, the game shall load the player to the exact last saved location. |  |
| DI\_GS-15 | 3 | If the game load is unsuccessful the, the user shall be presented with an error message. *“Sorry the selected game could not be loaded. Please try again or select a different saved game.”* |  |
| DI\_GS-16 | 2 | If the User selects *Options,* the application shall present a toggle able list of *Game Play Options.* |  |
| DI\_GS-17 | 3 | Table DI\_GS- Game Play Options |  |
|  |  | |  |  |  | | --- | --- | --- | | Option | Selections | Description | | Sound | On/Off | Game sounds on or off | | Blood | On/Off | Content control to have damage to enemies player presented with blood or not. | | AI Re-Spawning | On/Off | Enemies are defeated once per area and do not regenerate or enemies can re-spawn infinitely. | |  |
| DI\_GS-18 | 3 | The list of options shall be selected using the up/down arrow keys. |  |
| DI\_GS-19 | 3 | The option shall be toggled on/off using the left/right arrows. |  |
| DI\_GS-20 | 3 | The option selection shall be executed when the User presses the enter key. |  |
| DI\_GS-21 | 3 | All options shall be defaulted to *Yes.* |  |
| DI\_GS-22 | 2 | If the user selects *Statistics,* the user shall be presented with a list of game statistics. |  |
| DI\_GS-23 | 3 | Table DI\_GS- Game Statistics |  |
|  |  | TBD | Future scope |

## Game End

### Description and Priority

This feature refers to the user’s ability to end a game that is currently running. The user shall be instructed (through game manuals, online help, or other game documentation) to properly end the game and to exit the program upon completion. The Game End, shall exit the current session and exit the game application. This will ensure the game has been properly terminated and all application processes have been stopped. This is a high priority feature.

### Stimulus/Response Sequences

The user can end the current game at any point in time or the game will automatically end when the user has either completed the final level by defeating his/her final opponent or lost the game by being killed during game play. If the user wishes to end a game that they have not completed, they will select the option to pause the current game. A menu of options will then be displayed and the user can select to end the game from that menu. The user will then be given a choice of whether to save the current game or to exit without saving. If the option to save the current game is selected, the current progress of the game will be saved under the character name that was selected at the beginning of that game. After either option is selected, the user will be returned to the main game menu where they can select the option to close the program. (When opening a saved game, the user will be returned to the exact spot they left off at when the game had been previously closed out.) The game can also be ended when the player dies (as discussed in Section 4.5). If the player completes the game or dies, a graphic will be displayed showing the players score and the game will be exited (without saving), returning the screen to display the Main Menu.

### Functional Requirements

*Game End 🡪 GE*

*Game Finished 🡪 GF*

*Player Dies 🡪 PD*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_GE-1 | 2 | The Game End shall launch from the Pause Menu. |  |
| DI\_GE-2 | 2 | Game End shall prompt the user to save before closing |  |
| DI\_GE-3 | 3 | If the User chooses to save the current game, then Game End shall prompt the user to confirm the profile name. |  |
| DI\_GE-4 | 2 | If the User chooses to not save the current game, game end shall exit the current game, ending all running game processes. |  |
| DI\_GE-5 | 2 | Game End shall completely close the application. |  |
| DI\_GE-GF1 | 2 | If the user completes the game by defeating final boss, then the Game End shall launch Game Ending Animation/End story. |  |
| DI\_GE-GF2 | 3 | After the Game Ending Animation/End Story, Game End shall display the game credits. |  |
| DI\_GE-GF3 | 3 | After the completion of the game credits Game End shall end the Application. |  |
| DI\_GE-PD1 | 2 | If the Player’s health is zero, the user has lost, Game End shall prompt the user to continue or end the game. |  |
| DI\_GE-PD2 | 3 | If the user selects to continue, the player shall be re-spawned to the last save location. |  |
| DI\_GE-PD3 | 3 | If the user selects to End Game, End Game shall exit the player back to the main menu. |  |

## Playing Game

### Description and Priority

This feature refers to the user’s ability to actually play the game. This includes being able to move through levels, around obstacles, and it makes reference to other features including the ability to attack and die. This is a high priority feature.

### Stimulus/Response Sequences

Once the game is started, an intro graphic will be displayed and then the first level map will be loaded into the system and the starting point of the map and the player will be displayed on the screen. The user will be able to use the keyboard to make the player move in the game. As the player moves through the map the map will automatically scroll on the screen display so that the user will be able to see a certain area of map around the player. The user will be able to see other characters and opponents on screen as the player approaches them. The player will not be able to occupy the same space on the map as other characters and certain obstacles (walls/trees/rivers) on the map. The player can attack characters on the map (as discussed in Section 4.4) and characters and the player can die (as discussed in Section 4.5). As each level is completed and a new level is begun, the map for the previous level will be removed and the map for the new level will be loaded into the system. A level is completed after all opponents on that level have been defeated. The game is completed when the player completes the last level.

### Functional Requirements

*Game Play 🡪 GP*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_GP-1 | 2 | The Level map shall load at the beginning of that level. |  |
| DI\_GP-2 | 2 | User can move the player. |  |
| DI\_GP-3 | 2 | The player shall be controlled by the arrow keys on the key board. |  |
| DI\_GP-4 | 3 | The “left arrow” shall move the player to the left. |  |
| DI\_GP-5 | 3 | The right arrow shall move the player to the right. |  |
| DI\_GP-6 | 3 | The up arrow shall move the player up the map. |  |
| DI\_GP-7 | 3 | The down arrow shall move the player down the map. |  |
| DI\_GP-8 | 2 | The screen shall scroll in the current player’s facing direction, if the player was moved passed the middle of the game play field. | If player is moving right the map will scroll right. If the player is moving up the map will scroll up. |
| DI\_GP-9 | 2 | The Player shall be able to roam the entire level at any given time. | Free roaming. |
| DI\_GP-10 | 2 | Enemies shall be randomly spawned throughout the level. |  |
| DI\_GP-11 | 2 | Level “Bosses” shall be spawned at specified locations in the level. | When a player reaches a set location in the map, the boss shall be spawned. |
| DI\_GP-12 | 2 | Enemies shall seek out the player. |  |
| DI\_GP-13 | 2 | The Space bar shall be used to perform a basic strike. |  |
| DI\_GP-14 | 2 | The “F” key shall be used to shoot a projectile. |  |
| DI\_GP-15 | 2 | The projectile shall fire in the same direction in which the player is currently facing. |  |
| DI\_GP-16 | 2 | The projectile shall remain on the same path in which it was fired from. |  |
| DI\_GP-17 | 2 | The projectile shall remain on the screen in motion until one of the following criteria is met. |  |
|  | 3 | 1. The projectile collides with an enemy. 2. The projectile exits the current screen. 3. The projectile timer expires. |  |
| DI\_GP-18 | 2 | The “D” key shall be used to block or deflect an enemies attack or projectile. |  |
| DI\_GP-19 | 2 | The “S” key shall be used to perform a selected “Special Attack”. |  |
| DI\_GP-20 | 2 | The “A” key shall be used to perform a jump or flight attempt. |  |
| DI\_GP-21 | 2 | The level is complete once the boss has been defeated, and the player has reached the exit portal. |  |

## Attacks and Avoidance

### Description and Priority

This feature refers to the ability of the player and other characters in the game to attack other characters and for them to also be able to avoid attacks made on them by others. This is a high priority feature since the user will need to be able to attack opponents in the game in order to complete a level and win the game. Also the player and characters in the game will need to be able to avoid attacks made on them by others.

### Stimulus/Response Sequences

In the game the player will be able to attack opponents and other characters using a variety of attack methods. Characters will be able to respond to these attacks and attempt to avoid them altogether. Characters will also be able to attack the player and the player will be able to avoid and or respond to these attacks. The user will be able to use certain keys on the keyboard to select the type of attack he/she would like the player to make. The user will also be able to move the player in order to shield the player from attacks being made on him or avoid being hit by a projectile. Some of the attacks that can be made include projectiles that can be thrown (as discussed in Section 4.6). Each character in the game (including the player) will maintain health data that tells how strong/weak that character is depending on the number of attacks it has sustained and the level of difficulty at which the game is being played. Once a characters health level reaches 0, that character will die (including the player).

### Functional Requirements

*Game Attacks 🡪 GA*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_GA-1 | 2 | Drago’s attacks |  |
|  | 3 | |  |  | | --- | --- | | Key | Attack Performed | | Space | Tail Whip | | F | Fire Ball | | D | Duck/Cover up | | S | Special | | A | Jump/Flight | |  |
| DI\_GA-2 | 2 | Enemy Attacks may be avoided or absorbed. |  |
| DI\_GA-3 | 2 | Once Flight is enabled/discovered, attacks the following attacks shall be available. |  |
| DI\_GA-4 | 2 | Drago’s Flight attacks |  |
|  | 3 | |  |  | | --- | --- | | Key Combination | Attack performed | | A + F | Arial Fireball | | A + S | Arial Special | |  |
| DI\_GA-5 | 2 | Drago’s Special attacks |  |
|  | 3 | |  |  | | --- | --- | | Attack | Required item | | Flame | Red Egg | | Freeze | Blue Egg | | Charge/Dive | Brown Egg | | Stomp/Pound Landing | Green Egg | |  |
| DI\_GA-6 | 2 | The Special attack shall be equipped in the pause menu. |  |
| DI\_GA-7 | 2 | The amount of Damage inflicted to enemies by attacks shall increase as game experience increases. |  |
| DI\_GA-8 | 2 | Enemies shall vary in the amount of damage they can with stand. |  |
| DI\_GA-9 | 2 | Enemy attacks |  |
|  | 3 | |  |  | | --- | --- | | Enemy | Attack | | Villager | Pitchfork thrust | | Villager | Pitchfork projectile | | Blacksmith | Hatchet projectile | | Blacksmith | Ax strike | | Archer | Arrow projectile | | Wizard | Freeze | | Wizard | Flame | | Wizard | Fireball projectile | |  |
| DI\_GA-10 | 2 | Enemies will randomly attempt to block attacks. | Not every attack will attempt to be blocked by the CPU. |
| DI\_GA-11 | 2 | All damage shall be determined by collision detection. |  |
| DI\_GA-12 | 2 | An enemy is defeated when the enemy’s health is equal to zero. |  |

## Projectiles

### Description and Priority

Projectiles can be thrown as part of an attack by a character in the game against an opponent. This is a low priority feature since it is just an extra feature that is used as part of an attack by a character. The characters will be able to have other types of attacks without launching a projectile.

### Stimulus/Response Sequences

Certain types of attacks in the game will require the launching of a projectile at an opponent. The dragons will be able to launch fire/ice balls or throw spikes from their tails. Humans will be able to shoot arrows, throw spears and some castles will be equipped with a catapult, which is capable of launching large stones at opponents. Not all of these attacks will be available at the start of the game. Some of the dragon’s abilities will be earned as he progresses through the game. Characters will be able to avoid projectiles thrown at them. If a character is hit with a projectile, their health level will drop.

### Functional Requirements

Projectiles 🡪 P

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_P-1 | 2 | Player can throw projectiles as an attack. |  |
| DI\_P-2 | 2 | A projectile shall cause damage when collide with a Player or Enemy. |  |
| DI\_P-3 | 2 | Projectiles may be avoided. |  |
| DI\_P-4 | 2 | Projectiles may be blocked/deflected or absorbed. |  |
| DI\_P-5 | 2 | If a projectile is blocked/deflected or absorbed the health deduction shall be less than a direct hit. |  |
| DI\_P-6 | 2 | Projectiles shall exist for a limited amount of time. |  |
| DI\_P-7 | 2 | Drago’s projectiles shall not exist longer than 2 seconds. |  |
| DI\_P-8 | 2 | Enemy projectiles shall exist for a randomly assigned time. |  |
| DI\_P-9 | 2 | If a Drago projectile collides with an enemy the health deduction shall be based on the current experience level. |  |
| DI\_P-10 | 2 | If an Enemy projectile collides with Drago the health deduction shall be based on the current experience level. |  |
| DI\_P-11 | 2 | As a Player advances through the game, Drago’s projectiles increase in damage given. |  |
| DI\_P-12 | 2 | As a Player advances through the game, Enemy projectiles increase in damage given. |  |
| DI\_P-13 | 2 | As a Player advances through the game, Drago’s Defense of Projectiles shall increase. | Projectiles Health deduction is lessened. |

## Collision Detection

### Description and Priority

Characters in the game must be kept from occupying the same space on the game map as other characters and certain obstacles in the map. There must be a specific class designed to keep track of all characters and obstacles on the map and be able to evaluate if they are near or are colliding with each other. This is a high priority feature needed to maintain real world physics inside the game.

### Stimulus/Response Sequences

Objects in the game area will be in motion and will interact and collide with each other. Collision will typically result in a rebound of the colliding object at a reflected angle in the case of circle-to-line collisions, and in an orthogonal reflection in the case of circle-to-circle collisions. The collision detection algorithm will make use of selective pruning to decide which objects to test for collision. Collision will happen automatically as objects in the game area move by themselves and collide with other objects. The player may also collide with objects by walking into them. When this occurs, the player will move as far as possible toward the obstructing object without actually overlapping its territory. A collision will also be able to be detected if a projectile is thrown at and collides with a character on the map.

### Functional Requirements

*Collision Detection 🡪 CD*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_CD-1 | 2 | System can detect all characters and where they are located in the map. |  |
| DI\_CD-2 | 2 | Enemies on the board respond to player movement on the map. |  |
| DI\_CD-3 | 2 | Player shall not cross certain walls or boundaries. |  |
| DI\_CD-4 | 2 | System ensures that the characters cannot pass through certain objects on the map. |  |
| DI\_CD-5 | 2 | Characters attack the player when he is within a certain range. |  |
| DI\_CD-6 | 2 | System detects when a projectile has collided with an object/character |  |
| DI\_CD-7 | 2 | System ensures that two characters cannot occupy the same space on the map at the same time. |  |
| DI\_CD-8 | 2 | If Drago and an enemy collide then, Drago will incur damage. |  |
| DI\_CD-9 | 2 | If Drago collides with an enemy as an attack, then the enemy will incur damage. | Charge, Stomp, Pound, or Dive |
| DI\_CD-10 | 2 | If the system detects collision from a projectile to Drago and the “D” key is depressed, the amount of Damage incurred will be 30% less than the original damage amount. |  |
| DI\_CD-11 | 2 | If the system detects collision from a projectile to Drago and the “D” key is not depressed, the amount of Damage incurred will be 100%. |  |
| DI\_CD-12 | 2 | If the system detects collision from a projectile to an Enemy and the Enemy is attempting to block, the amount of Damage incurred will be 30% less than the original damage amount. |  |
| DI\_CD-13 | 2 | If the system detects collision from a projectile to an Enemy and the Enemy is not attempting to block, the amount of Damage incurred will be 100%. |  |

## A\* Pathfinding

### Description and Priority

Certain objects in the game area will search for the shortest path to find and attack the player, or to find other objects. This is a high priority feature that allows the characters to move toward the player in the fastest manner possible.

### Stimulus/Response Sequences

The shortest path will be found using the A\* search algorithm together with a navigation mesh. The algorithm consists in applying a heuristic estimate of the distance of any given location from the target and attempting to minimize sum of the traveled distance and the estimated remaining distance in a priority-based search similar to breadth-first search. Objects using the A\* path-finding procedure will respond automatically to the location of the target object. As the target object moves, the A\* algorithm will recalculate the shortest distance to the object.

### Functional Requirements

*A\* 🡪 A\**

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_A\*-1 | 2 | A suitable navigation mesh be generated for each level. |  |
| DI\_A\*-2 | 2 | The navigation mesh will span the entire explorable area of a level. |  |
| DI\_A\*-3 | 2 | The navigation mesh will be represented by a graph data structure whose nodes correspond to convex partitioned regions of the explorable area. By a ‘convex’ region, we mean that given any two points in the region, an object can travel in a straight line from one point to the other under the normal collision detection scheme, possibly sliding against walls (so long as the target destination can be reached). |  |
| DI\_A\*-4 | 2 | The algorithm generates a suitably short path to the target, given obstacles. |  |

## Heads Up Display

### Description and Priority

The screen must display the current state of the game at all times. There must be built in output to show menu screens as well as the game map and any characters that are on the current portion of it. This is a high priority feature. The user must be able to read menu screens, see the level map and see where the player and characters are on it.

### Stimulus/Response Sequences

The Heads Up Display object will output the information for the current state of the game to the screen. The Heads Up Display object will need to consistently read data from the GameState object in order to know what state the game is in at every second. The Heads Up Display object will need access to all graphics data in order to display characters, level maps, and menus. The Heads Up Display will interface with an overlay so when the map changes or is scrolled the Heads Up Display is still fixed to the screen

### Functional Requirements

*Heads Up Display 🡪 HUD*

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_HUD-1 | 2 | The HUD shall interface with an Overlay. |
| DI\_HUD-2 | 2 | The HUD shall access the GameState for display data. |
| DI\_HUD-3 | 2 | The HUD must display the current state of the game to the user. |
| DI\_HUD-4 | 2 | The HUD shall have access to graphics data in order to display on the screen. |
| DI\_HUD-5 | 2 | The HUD shall be “fixed” to the game screen. |
| DI\_HUD-6 | 2 | The HUD shall not move when a map is scrolled in any direction. |
| DI\_HUD-7 | 2 | The HUD shall display the following information. |
|  | 3 | |  |  |  | | --- | --- | --- | | Attribute | Location | Display type | | Health Bar | Top right corner | Bar | | Health Value | Top right corner | Integer value | | Power Bar | Top right corner | Bar | | Power Value | Top right corner | Integer value | | Equipped Special Attack | Top right corner | Text | | Egg Counter | Bottom right corner | Graphic Display | |
| DI\_HUD-8 | 2 | The HUD shall be updated when the following game actions occur |
|  | 3 | |  |  | | --- | --- | | Action | Update | | Drago incurs damage | Health Bar is decreased  Health Value is decreased | | Drago performs special attack | Power bar is decreased  Power value is decreased | | Drago collides with a Heart object (“Picks Up”) | Health bar is increased  Health value is increased | | Drago collides with a Potion object (“Picks Up”) | Power bar is increased  Power value is increased | | Drago collides with an Egg that has not been previously found | The Egg Counter is updated. | | Drago collides with an Egg that has been previously found | No Update. | | Player changes the equipped special attack. | Text is changed. | | 2 seconds has elapsed and Drago has not incurred any damage or Picked up a Heart. | Health bar is increased  Health value is increased | |

## Audio Effects

### Description and Priority

There will be music and sounds tied into the game. This is a low priority feature and it is just added to enhance the user’s experience.

### Stimulus/Response Sequences

Music will play throughout the game and it will change as different areas in the map are reached by the player. The Sounds will be tied into something happening in the game (i.e. when the dragon throws a fire ball, there will be a sound effect associated with that). The system will automatically play these sounds when an attack is made, a projectile hits an object/character, or when certain goals are reached in the game.

### Functional Requirements

*Audio Effects 🡪 AE*

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_AE-1 | 2 | Sounds are played for certain events in the game. |
| DI\_AE-2 | 2 | Music changes corresponding to the area in the map that the player is in |

## Peripheral Input

### Description and Priority

The program must be able to accept input from the keyboard to allow the user to make selections and move the player through the map. This is a high priority feature since the user will have no way to interact with and play the game if the program cannot accept input from the keyboard.

### Stimulus/Response Sequences

Once the program is started, the user will be able to make menu selections using the characters buttons on the keyboard. Then, once a game is started/restarted the user will be able to use keyboard selections to move the character through the levels and also to use special abilities of the player and launch attacks against opponents. Some of these abilities and attacks will include the dragon’s ability to fly, perform a tail whip attack, launch a fire ball, launch an ice ball, or eat humans. Input from the keyboard will also allow the user to pause the game and then save/exit the game.

### Functional Requirements

Peripheral Input 🡪 PI

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_PI-1 | 2 | Program will accept input from keyboard for user selections |
| DI\_PI-2 | 2 | User can make selections from menus |
| DI\_PI-3 | 2 | User can move player object through the game |

## Binary Space Partitions

### Description and Priority

The Playfield map will be divided into a BSP Tree that will be used in the A\* Search for the Enemies to locate and move to the Player on the Playfield. This is a high priority, essential feature. Without the BSP Tree, the Enemies would be unable to move intelligently around the Playfield.

### Stimulus/Response Sequences

A BSP Tree is created when the level map is loaded and it is maintained as part of the Playfield class. As the Player moves on the Playfield, each of the Enemy characters check the current position of the Player and whether it can be reached from their current position. A Linked List of Portals is passed from the A\* Search class to the Enemy class and is used to move the Enemies through the BSP Leaves on the map to reach the Player.

### Functional Requirements

Binary Space Partitions 🡪 BSP

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_BSP-1 | 2 | BSP Tree is created and maintained when the level map is loaded |
| DI\_BSP-2 | 2 | BSP Tree is used in the A\* Search function |
| DI\_BSP-3 | 2 | BSP Tree consists of all of the BSP Leaf nodes on the map (This covers all areas of the map) |

## Dynamic Portals

### Description and Priority

Portals separate the BSP Leaves on the map. Portals contain an attribute that says whether they are open (can be crossed) or closed (cannot be crossed). Enabling the open/closed attribute to be set or changed allows us to create dynamic areas of the map that can be opened. This is a medium priority feature that allows us to add some extra features to the game.

### Stimulus/Response Sequences

Rock, Tree and Water obstacles are set up with dynamic portals around them. These portals are created in the closed state when the level map is loaded. However, when the Player is moving through the Playfield he can burn down the trees, break down the rocks and freeze the water in order to move through those obstacles. When this is done, the portals are changed to the open state and the Player can then move across those BSP Leaves.

### Functional Requirements

Dynamic Portals 🡪 DP

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_DP-1 | 2 | Portals are created between all BSP Leaves on the map when the level map is loaded |
| DI\_DP-2 | 2 | Portals can be set in either the open or closed state |
| DI\_DP-3 | 2 | Portals around the BSP Leaves containing a rock, tree or water are set to closed at the beginning of the level |
| D DI\_DP-4 | 2 2 | Portals around the BSP Leaves containing a rock, tree or water are changed to open when the Player breaks the rock, burns down the tree, or freezes the water |

# Non-functional Requirements

## Performance Requirements

The performance of this game will only be limited by the hardware it is run on. If the hardware is the minimum as recommended by the Java developers then there should be no problem running this game. However, there may be times when slow machines may not perform optimum, resulting in lower frame rates, or FPS (frames per second), and screen lag. Java can’t use most hardware at full potential since it cannot run as “close to the metal” as languages such as C, assembly, or even C++. While this isn’t much of an issue for today’s machines, this could still be one reason for lack of performance on older machines.

While the garbage collection in Java is something that programmers praise, it also must be accessed as overhead in performance. This is true because, unlike C or C++, Java garbage collection is managed at runtime and out of the programmer’s control. During garbage collection cycles, there could be noticeable lag, especially in action-intense gameplay.

In examining these performance hindrances, it is important to make the game as stable and responsive as possible. The most important performance requirement is that the game responds to user input immediately, failure to do so could result in a character’s death. Maintaining a consistent frame rate and consistent sound is also important, but take a back-seat to responsiveness.

### Hardware

The most important part of a gaming PC is the graphics card. This will have the greatest effect on what games you're able to play as well as the amount of detail that each game will display. After the graphics card, the processor is the next most important component of a gaming computer. The processor functions as the taskmaster of the system and delivers instructions to all the other parts. The faster the processor, the more tasks it can handle at a given time. Often with a slower processor, games crash or stall when other programs are running in the background such as email or web browsers. Using at least a 2.0 GHz processor will keep your system running smoothly during game play.

### Memory

### Any current generation gaming software requires at least 1 gigabyte of RAM (Random Access Memory) to run properly. The higher the RAM, the smoother your games will function. While the minimum requirement is 1 gigabyte, gamers will want to begin with 2 gigs to be ready for upcoming games.

### Space

Games take up a great deal of space on a hard drive. This is why a gaming PC should include at least a 150 GB hard drive. When deciding on a hard drive, take into account the amount of games the computer will be used for. A gamer looking for a computer to run only World of Warcraft will require less space on a hard drive than someone looking to try new games every week.

## Safety Requirements

This game does not present any safety requirements that are not present with other computer games. Computer games should be enjoyed in small intervals to prevent too much inactivity. The user should make note to stand and stretch sporadically. To help avoid epileptic seizures and eye strain, the user should stress the importance of frequent breaks.

## Security Requirements

As operating system developers are cracking down on malware that affects their systems, new types of threats become abundant. Java is a risky language because it is a third-party application that requires downloading onto the host machine. This opens up the user’s system to outside sources, in hopes that the Java developers took security into account while designing the application. Vulnerabilities in a Java application are a risk to the host machine. Luckily, Oracle has been finding and patching security flaws quite often, but they can only fix what they know exists, and they usually don’t know it exists until it is exploited.

Security requirements are narrowed down to the user maintaining up-to-date patches from Oracle. A knowledgeable user can ensure the system is protected. The user should know what versions of Java are installed on the machine and maintain each version. The best security is knowledge.

## Software Quality Attributes

The game software shall adhere to the following Quality attributes to provide the optimal game experience for the user/player.

### Correctness

5.4.1.1

The game shall be correct in terms of its functionality, calculations used internally and the navigation shall be correct.

5.4.1.2

All game play shall follow the storyline presented, and all coding shall follow a logical path to produce a chronological order of the storyline.

5.4.1.3

The game shall adhere to all of the functional requirements described in this document.

### Reliability

5.4.2.1

The game shall give consistently correct results.

5.4.2.3

The game functionality shall operate the same for each user/player.

5.4.2.4

The results shall be consistently expected. (If enemies defeat the player, the user should not win the game).

### Maintainability

5.4.3.1

The game shall be properly versioned using a logical versioning convention.

5.4.3.2

Only authorized project members shall maintain the versions.

5.4.3.3

The system shall be easy to maintain for correcting defects, and software bugs.

5.4.3.4

The game shall be able to be upgraded or improved.

### Usability

5.4.4.1

The game shall be easy to learn.

5.4.4.2

The actions of the game play shall be properly mapped to the keyboard buttons, to the best of the developer’s ability. (Right arrow means go right, left means go left, etc.)

### Testability

5.4.5.1

The game shall be easy to test and find defects.

5.4.5.2

Project team members shall only do testing.

## Business Rules

### Users

5.5.1.1

Users shall be able to operate in all levels of the game once the previous level has been completed.

5.5.1.2

Users shall only be able to use a skill/attack once it has been unlocked.

5.5.1.3

Users shall be able to save games but only restart as a user.

5.5.1.4

Users are free to play any of the released versions they choose.

5.5.1.5

Users shall only have access to GUI game screens. Not any coding.

### Administrators

5.5.2.1

Project administrators shall have access to each game level at any time.

5.5.2.2

Project administrators shall have all available game moves/attacks.

5.5.2.3

Project administrators shall have options to play as any role.

5.5.2.4

Project administrators shall have access to all GUI and coding.

# Future Requirements

## Health, Life and Death

### Description and Priority

This feature refers to the ability of the player and other characters to maintain their health information to tell the system how strong/weak that character is. When that character’s health level reaches 0, that character/player will die. This is a medium priority feature. The game could be set up so that any hit would cause a character/player to die immediately, but that would make for short games that would not be very fun. Having a health measurement allows the player/characters to be somewhat resilient and to withstand multiple attacks.

### Stimulus/Response Sequences

Each character object in the game (including the player) will maintain a health measurement reflecting the current strength of the player. The health level of each character will be used to determine their strength of attacks and how much more damage they can sustain before they die. When a character launches an attack on an opponent in the game the health level of the character will determine how much damage their attack will cause to their opponent. Whenever a character is attacked by an opponent, that character’s health level will be decreased according to the severity of the attack. Each characters health at the start of the game is determined by the level of difficulty set by the user. However, no matter what the level of difficulty is set to in the game, whenever a character’s health level reaches 0 that character will die. The player’s health level will be displayed to the user on screen during the whole time a game is running.

### Functional Requirements

*Health 🡪 H*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Level | Description | Comment |
| DI\_H-1 | 2 | Player’s health level is displayed on screen at all times. |  |
| DI\_H-2 | 2 | Player is dead once his health level reaches zero. |  |
| DI\_H-3 | 2 | A Player’s max health is 100 |  |
| DI\_H-4 | 2 | Player’s health can be increased if the player picks up a Heart to a max of 100. |  |
| DI\_H-5 | 2 | Player’s health shall increase 2 units for every 30 second played to a max of 100. |  |
| DI\_H-6 | 2 | A player’s health shall be initialized to 100 upon the beginning of a level. |  |
| DI\_H-7 | 2 | If a game save is performed, the health level is set to the current player’s health. |  |
| DI\_H-8 | 2 | If a game load is performed, the health level is set to the player’s health from the last save performed. | If the player save when health was 48, then when the game is loaded the health value will be set to 48. |
| DI\_H-9 | 2 | An enemy’s health will be randomly assigned to a value between 1 and 100. |  |
| DI\_H-10 | 2 | An enemy is dead once the health level reaches zero. |  |
| DI\_H-11 | 2 | A boss’s health shall initialize to 100. |  |
| DI\_H-12 | 2 | A boss’s health shall be displayed on the screen when a boss is present. |  |
| DI\_H-13 | 2 | A boss is dead once the health level reaches zero. |  |

## NPCs

### Description and Priority

AI in the game will be set up to allow for Non-Player Characters that can assist the main player with his goals in the game. This is a low priority feature and is simply added to improve the user’s experience.

### Stimulus/Response Sequences

The goal of the player is to rescue stolen dragon eggs from humans. If the eggs are successfully rescued before they are hatched, the dragons become friends of Drago and they assist him on the rest of his journey. Only one rescued dragon can assist Drago at a time and if more than one dragon has been rescued, the user can select from which one he/she would like to assist on that level. Each rescued dragon has a unique ability and attack. AI built into the program will be used to move the NPC and determine which opponents on the board it should attack.

### Functional Requirements

NPC 🡪 NPC

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_NPC-1 | 2 | Rescued dragon eggs become “helpers” for players. |
| DI\_NPC-2 | 2 | AI allows NPCs to assist player in fighting opponents in game |
| DI\_NPC-3 | 2 | User can select which NPC to activate on that level |

## Visual Effects/Graphics

### Description and Priority

Graphics are displayed on the screen to represent the game map and all of the objects in it. Graphics will change based on the current state of the game. This is a high priority feature. We need a graphical representation of the game to be displayed to the user so that the user can interact with the game.

### Stimulus/Response Sequences

When the program is started, an intro screen is displayed showing the name of the game, the company name, and copyright information. Menu screens are displayed when the main menu section of the program is reached. A pause menu is displayed on screen when the game is paused. An intro video is displayed when a new game is started and a final video is shown when the game is won. A special video is also shown if the game is lost. Sprites are also displayed on screen when the game is being played to represent the level map, and any objects/characters on it.

### Functional Requirements

Visual Effects 🡪 VE

|  |  |  |
| --- | --- | --- |
| Requirement ID | Level | Description |
| DI\_VE-1 | 2 | Graphics are displayed corresponding to each character on the board |
| DI\_VE-2 | 2 | Graphics are displayed to coincide with the objects and layout of the map |
| DI\_VE-3 | 2 | Video graphics are displayed as an intro and ending to the game |

Appendix A: Glossary

**Character-** n. An actor in the video game.

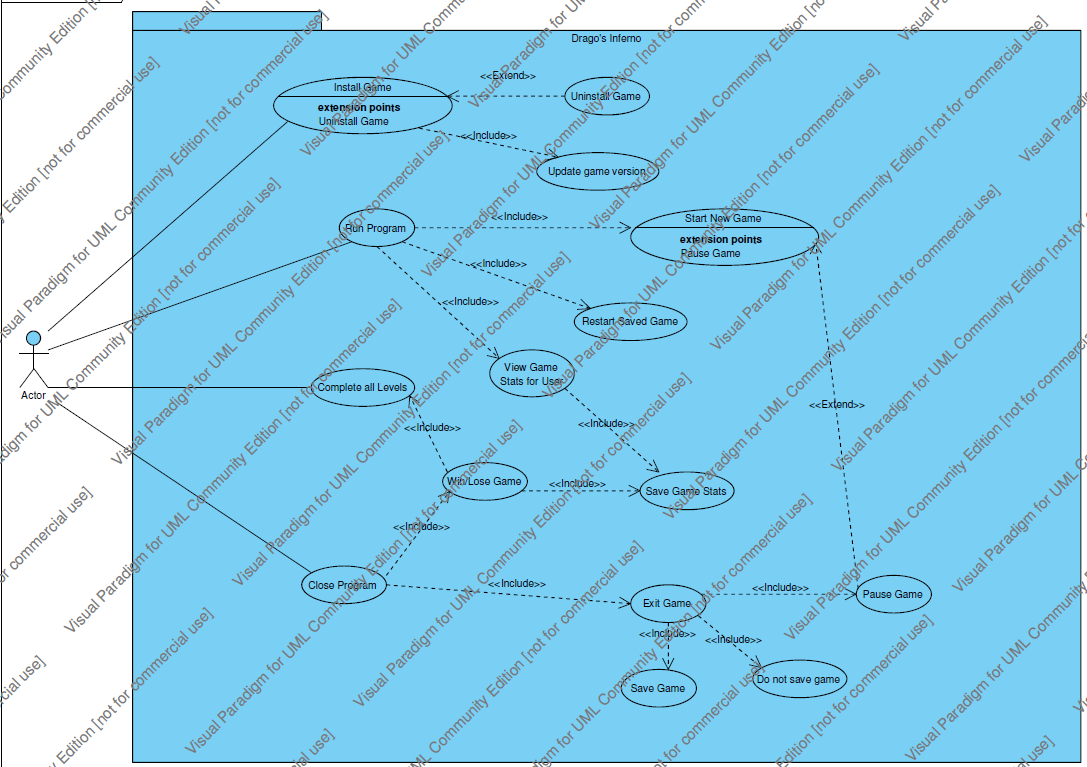
**Genre-** n. In video games, this is a classification for games of a similar type. For example Doom and Halo are part of the First-Person Shooter genre.

**Platformer-** n. A type of game requiring the player to jump to and from suspended platforms and over obstacles to navigate through a level.

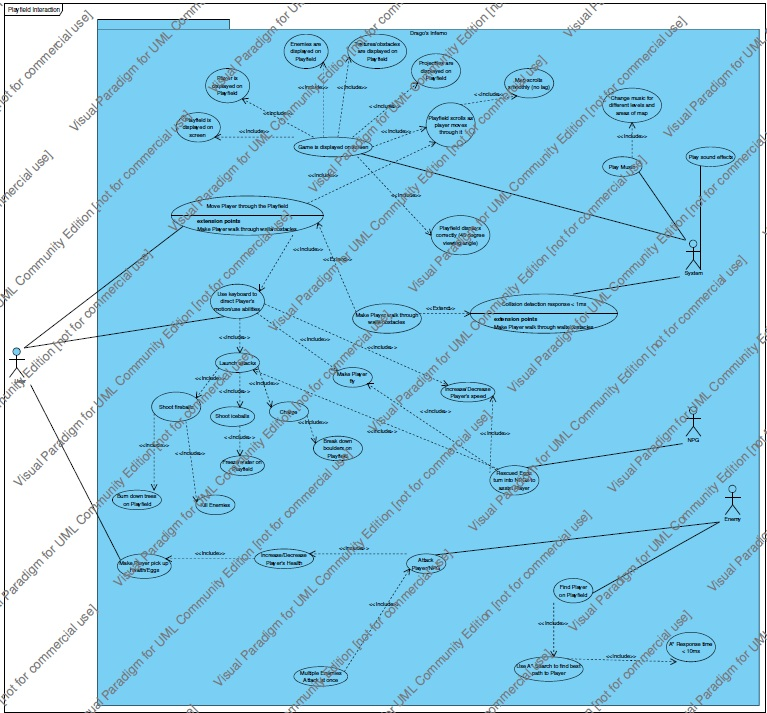
**Player-** n. 1) The person playing the video game. 2) The character represented by the person playing the video game.

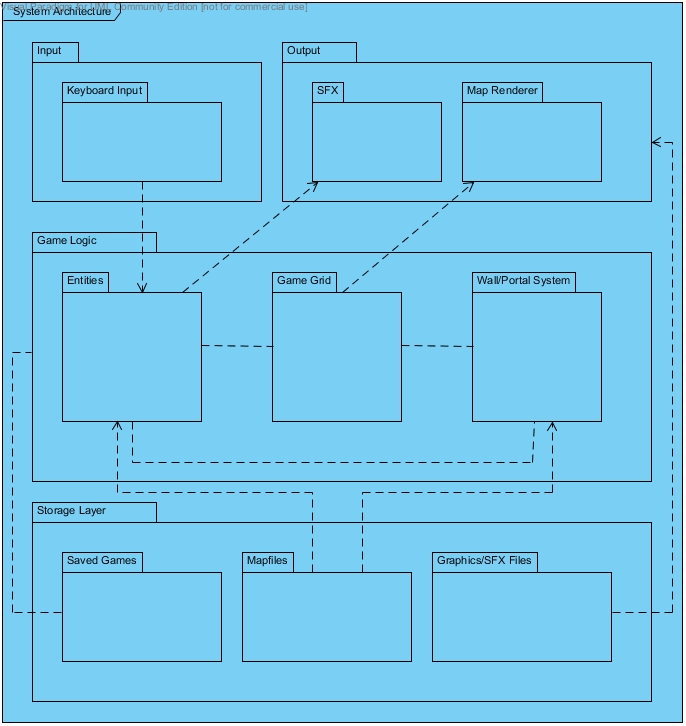
**Projectile-** n. An object in a video game that flies through the air, potentially striking a character and causing damage.

Appendix B: Analysis Models

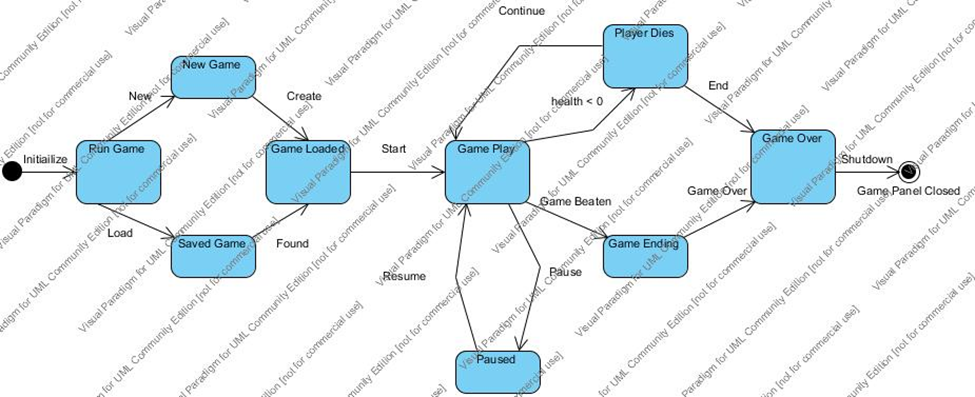


**B.1 Drago’s Inferno Basic Game Operation Use Case Diagram**

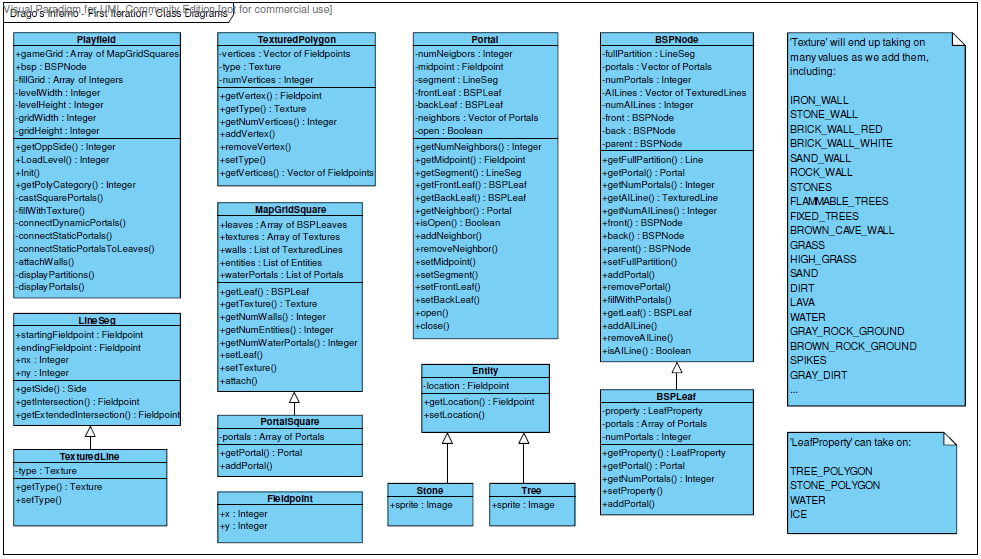
**B.2 Drago’s Inferno Playfield Interaction Use Case Diagram**

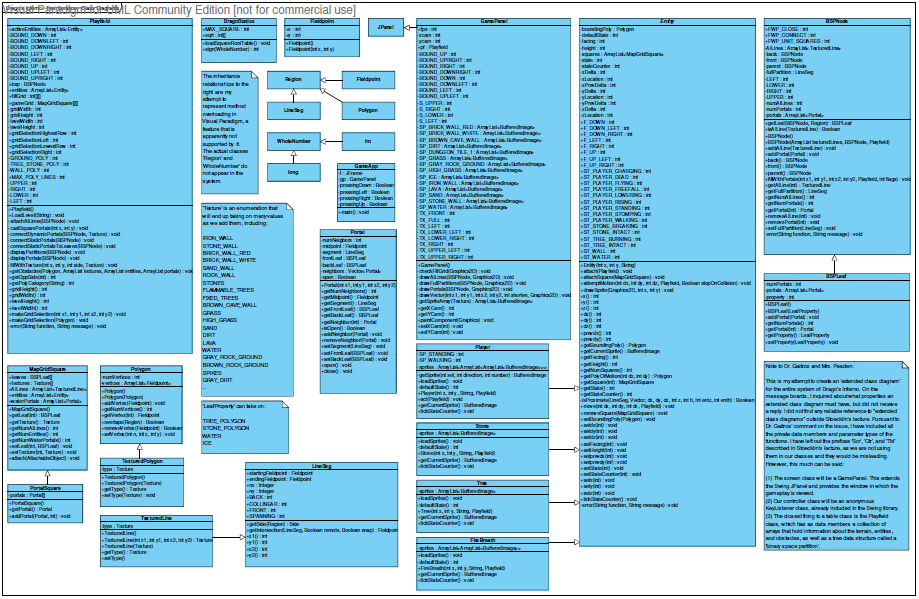


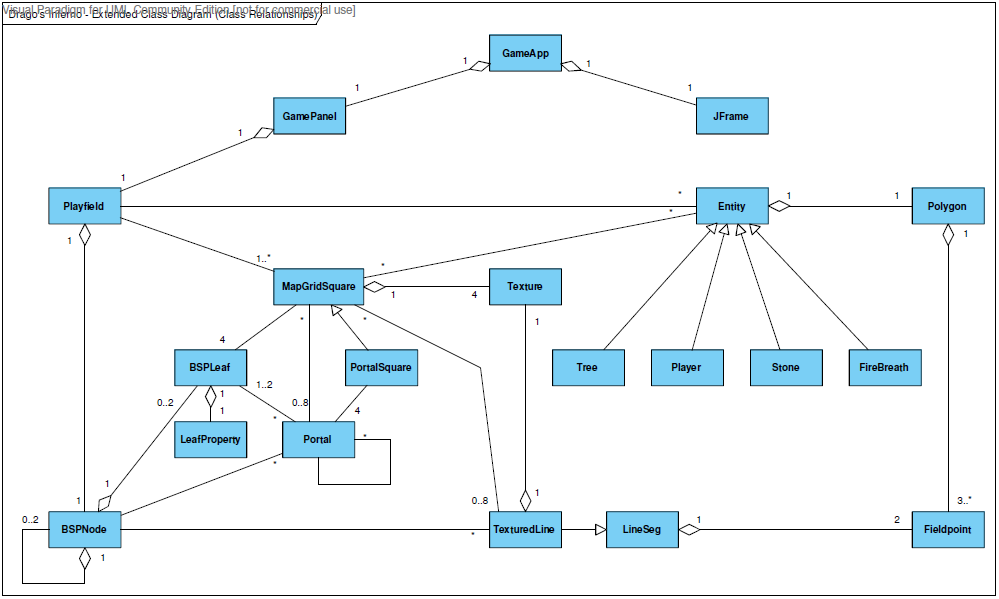
**B.3 Drago’s Inferno System Architecture Diagram**

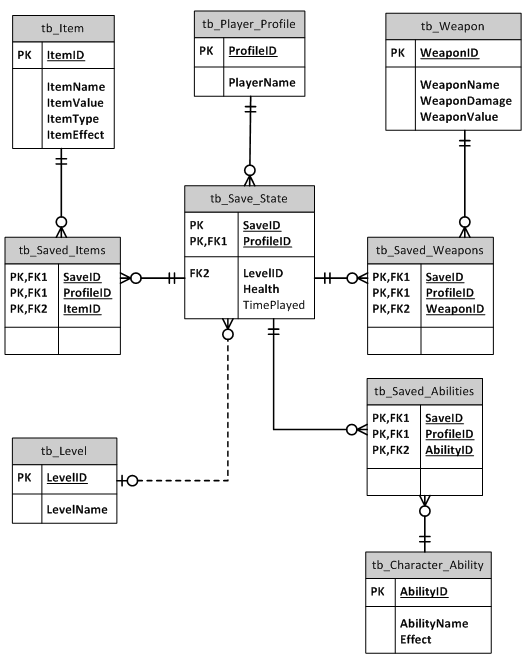
**B.4 Drago’s Inferno General Game Play State Diagram**

**B.5 Drago’s Inferno Playfield Class Diagram**



**B.6 Drago’s Inferno First Iteration Full Class Diagram**

**B.7 Drago’s Inferno Extended Class Relationship Diagram**



`

**B.8 Drago’s Inferno Entity-Relationship Diagram**