CEN 4021 – Spring 2013

Little Endian Entertainment, LLC.

**Drago’s Inferno**

Bryan Tanner, Scott Hoge, Ryan Alain, Elizabeth Collins & Michael Enriquez

**Software Design Specification Document**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisions** | | | |
|  |  |  |  |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| 1.0 | Elizabeth Collins | Initial Draft – Sections 1 & 5 | April 23, 2013 |
| 1.0.1 | Scott Hoge | Initial Draft – Section 3 | April 24, 2013 |
| 1.0.2 | Ryan Alain | Initial Draft – Sections 4 & 6 | April 25, 2013 |
| 1.0.3 | Bryan Tanner | Initial Draft – Section 2 | April 29, 2013 |
| 1.1 | Mike Enriquez | Initial Complete Draft (all sections) | April 30, 2013 |
| 1.2 | Elizabeth Collins | Final Draft | May 1, 2013 |

Contents

**1 Introduction**…………………….………………………………………………………………4

* 1. *System Overview*……………………………………………………………………………4
  2. *Supporting Materials*…………………………………………………………………….....5
  3. *Definitions, Acronyms, and Abreviations*…………………………………………………..5

**2 Design Considerations**…………………………………………………………………………7

*2.1 Assumptions*…………………………………………….……………….…………………7

*2.2 Constraints*…………………………………………………………………………………7

*2.3 System Environment*………………………………………………………………………10

**3 Architecture**…………………………………………………………………………………...11

*3.1 Overview*………………………………………………………………………………….11

*3.2 Rationale*………………………………………………………………………………….12

**4 High Level Design**…………………………………………………………………………….14

*4.1 Conceptual View*…………………………………………………………………………14

*4.2 Physical View*…………………………………………………………………………….14

**5 Low Level Design**….…………………………………………………………………………15

*5.1 Module*……………………………………………………………………………………15

*5.2 Sequence Diagrams*………………………………………………………………………21

*5.3 State Diagram*…………………………………………………………………………….26

*5.4 Database Design Diagram*……………………………………………………………….27

**6 User Interface Design**...............................................................................................................30

*6.1 Application Control*……………………………………………………………………...30

*6.2 Screenshots*………………………………………………………………………………30

|  |
| --- |
| **1 Introduction** |

**1.1 System Overview**

This SDS will cover the software for Drago’s Inferno. DI is a single player, 2-D, medieval adventure style game for personal computers.

DI will be a recreational game and will not offer any educational value. It will be suitable for all ages and will require only the skills necessary to launch the program and use the keyboard to move and launch attacks.

DI will be a classic-style 2-D game that will be reminiscent of Zelda. We feel that this game will appeal to a large range of gamers that will enjoy the classic style as well as the storyline. The storyline of DI is as follows:

In an age of castles and kingdoms, lords and peasants, knights and knaves, men and beasts are at war. The war has raged between humans and dragons for nearly a century. A dark lord has assumed power of the kingdom and vowed to end the war. King Malevolent has sent his wizard to steal the last dragon eggs in the realm. With these dragons he hopes to raise an army capable of destroying the last of his enemies and enslave the remaining dragon race.

Drago must save his species. His kind are too few and too weak to stand against King Malevolent any longer. The only way to win is to rescue the stolen eggs before they hatch.

You must search for the eggs that have been hidden throughout the kingdom. But watch out for the king’s men who are guarding them. Remember to be quick on your journey for you must rescue the eggs before they hatch and are turned against you. Collect new powers along the way and add to your army as you collect eggs and hatch the dragons to fight with you. The wizards will be your toughest challengers and you must destroy King Malevolent in order to secure peace for all dragons.

The software will allow the user to track their play time, including their high scores and number of games played. The users will also be able to save games that have not been completed and re-open saved games when they are ready to resume play.

This gaming software is targeted towards older gamers who are nostalgic for some of the earlier adventure games. However, this game will be suitable for gamers of all ages. The user need only need access to a personal computer and they will not require internet access to play the game.

The objective of this software is to entertain. There are no underlying learning objectives or skills that will be specifically taught by this software. However, studies have shown an increase in hand-eye coordination and response times for video game users.

The ultimate goal of the software is to give gamers another option for interesting gameplay. The game is meant to be nostalgic, intriguing, and challenging enough to keep users entertained. The final iteration of the game will come with a level map generator to make additional levels easy to add on in order to keep the game interesting.

**1.2 Supporting Material**

Article reviewing the positive aspects of gaming and citing multiple articles is available at

<http://www.contentedwriter.com/games-hand-eye-coordination/>

The general team webpage is located at <https://campus.fsu.edu/webapps/portal/frameset.jsp?url=/webapps/blackboard/execute/courseMain?course_id=_6376031_1>

The Design of Playfield overview document is located at

<https://campus.fsu.edu/courses/1/CEN4021.sp13.web_cohort2/groups/_183212_1//_6130913_1/Drago%27s%20Inferno%20-%20Design%20of%20Playfield%20Object.pdf>

Article reviewing how the Collision Response for the software should operate is available at <https://campus.fsu.edu/courses/1/CEN4021.sp13.web_cohort2/groups/_183212_1//_6189069_1/Drago%27s%20Inferno%20-%20Collision%20Response.pdf>

**1.3 Definitions, Acronyms, and Abreviations**

**DI** – Drago’s Inferno, the software game that this SDS describes, a single-player, 2-D, medieval adventure game for users of all ages.

**SRS** – Software Requirements Specification, the document which outlines the requirements that the software must fulfill independent from the actual design of the software.

**SDS** – Software Design Specification, this document which outlines the specific design requirements of this software.

**User** – any person who will use the program, being of any age or gender.

**Player** – the main character, Drago, as shown and played by the user in the game.

**Enemies** – any character object in the game that is not the player. They will include archers, knights, villagers, blacksmiths, wizards and King Malevolent.

**GUI** – Graphical User Interface.

**Playfield** – the game map as displayed for a particular level. Portions of this map will be displayed on screen as the player moves through it.

**Main Menu Page** – the main program page that is displayed when the game is first launched and can be returned to when the user leaves the game play screen.

**Game Play Screen** – view of the portion of the playfield that the player is currently in. This screen will scroll as the player moves through the playfield.

**UML** – Unified Modeling Language.

**OS** – Operating System.

|  |
| --- |
| **2 Design Considerations** |

**2.1 Assumptions**

* It is assumed that a keyboard is used for input and game controls
* It is assumed that the game will be run on Java 1.7 or above

**2.2 Constraints**

* Operating Systems

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solaris | | | | | |
| **Platform** | **CPU Architecture** | **Version** | **Introduced In** | **Notes** |
| Solaris | x86 (32-bit) | 10 Update 9+ | 1.7.0 | With OFED 1.5.1+. |
| Solaris | x64 (64-bit) | 10 Update 9+ | 1.7.0 |  |
| Solaris | SPARC (64-bit) | 10 Update 9+ | 1.7.0 | With OFED 1.5.1+. |
| Solaris | x86 (32-bit) | 11 Express | 1.7.0 | With OFED 1.5.1+. |
| Solaris | x64 (64-bit) | 11 Express | 1.7.0 |  |
| Solaris | SPARC (64-bit) | 11 Express | 1.7.0 | With OFED 1.5.1+. |
| Solaris | x64 (64-bit) | 11 | 1.7.0 |  |
| Solaris | SPARC (64-bit) | 11 | 1.7.0 | With OFED 1.5.1+. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Windows | | | | |
| **Platform** | **CPU Architecture** | **Version** | **Introduced In** | **Notes** |
| Windows Server 2008 | x86 (32-bit) | SP2 | 1.7.0 |  |
| Windows Server 2008 | x64 (64-bit) | SP2 | 1.7.0 |  |
| Windows Server 2008 R2 | x64 (64-bit) | SP1 | 1.7.0 |  |
| Windows Server 2012 | x64 (64-bit) |  | 1.7.0\_10 |  |
| Windows Vista | x86 (32-bit) | SP2 | 1.7.0 |  |
| Windows Vista | x64 (64-bit) | SP2 | 1.7.0 |  |
| Windows XP | x86 (32-bit) | SP3 | 1.7.0 |  |
| Windows XP | x64 (64-bit) | SP2 | 1.7.0 |  |
| Windows 7 | x86 (32-bit) | SP1 | 1.7.0 |  |
| Windows 7 | x64 (64-bit) | SP1 | 1.7.0 |  |
| Windows 8 | x86 (32-bit) |  | 1.7.0\_10 | Modern UI (i.e. Metro Mode) is not supported |
| Windows 8 | x64 (64-bit) |  | 1.7.0\_10 | Modern UI (i.e. Metro Mode) is not supported |
| Linux on ARM | | | | |
| **Platform** | **CPU Architecture** | **Version** | **Introduced In** | **Notes** |
| Ubuntu Linux  (Soft-Float ABI) | ARMv7 VFP | 11.10 | 1.7.0\_06 | Not certified on Oracle VM. No support for Plugin or WebStart Requires GNOME Desktop Environment version 1:2.30+7 |
| Debian Linux  (Soft-Float ABI) | ARMv6 VFP | 6.0.5 | 1.7.0\_06 | Not certified on Oracle VM. No support for Plugin or WebStart Requires GNOME Desktop Environment version 1:2.30+7 Java SE Support not available for this platform |
| Mac OS X | | | | |
| **Platform** | **CPU Architecture** | **Version** | **Introduced In** | **Notes** |
| Mac OS X | x64 | Mac OS Lion(10.7.3) or greater | 1.7.0\_04 | Only 64-bit JVM is certified. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Linux | | | | |
| **Platform** | **CPU Architecture** | **Version** | **Introduced In** | **Notes** |
| Oracle Linux | x86 (32-bit) | 5.5+ | 1.7.0 | With OFED 1.5.1+. |
| Oracle Linux | x64 (64-bit) | 5.5+ | 1.7.0 |  |
| Oracle Linux | x86 (32-bit) | 6.x | 1.7.0 | Not certified on Oracle VM. |
| Oracle Linux | x64 (64-bit) | 6.x | 1.7.0 | Not certified on Oracle VM. Only 64-bit JVM is certified. |
| Suse Linux Enterprise Server | x86 (32-bit) | 10 SP2 | 1.7.0 | Not certified on Oracle VM. |
| Suse Linux Enterprise Server | x64 (64-bit) | 10 SP2 | 1.7.0 | Not certified on Oracle VM. |
| Suse Linux Enterprise Server | x86 (32-bit) | 11.x | 1.7.0 | Not certified on Oracle VM. |
| Suse Linux Enterprise Server | x64 (64-bit) | 11.x | 1.7.0 | Not certified on Oracle VM. |
| Red Hat Enterprise Linux | x86 (32-bit) | 5.5+ | 1.7.0 |  |
| Red Hat Enterprise Linux | x64 (64-bit) | 5.5+ | 1.7.0 |  |
| Red Hat Enterprise Linux | x86 (32-bit) | 6.x | 1.7.0 | Not certified on Oracle VM. |
| Red Hat Enterprise Linux | x64 (64-bit) | 6.x | 1.7.0 | Not certified on Oracle VM. Only 64-bit JVM is certified. |
| Ubuntu Linux | x86 (32-bit) | 10.04 - LTS | 1.7.0 | Not certified on Oracle VM. |
| Ubuntu Linux | x64 (64-bit) | 10.04 - LTS | 1.7.0 | Not certified on Oracle VM. |
| Ubuntu Linux | x86 (32-bit) | 11.04 | 1.7.0 | Not certified on Oracle VM. |
| Ubuntu Linux | x64 (64-bit) | 11.04 | 1.7.0 | Not certified on Oracle VM. |
| Ubuntu Linux | x86 (32-bit) | 12.04 | 1.7.0\_06 | Not certified on Oracle VM. |
| Ubuntu Linux | x64 (64-bit) | 12.04 | 1.7.0\_06 | Not certified on Oracle VM. |

* On 64-bit operating systems (OSs), both 32- and 64-bit Java virtual machines (JVMs) are certified, except where noted.
* All default desktop managers are certified, except where noted

**Virtualization**:

* All certified platforms are also certified when virtualized in a certified hypervisor, except where noted. Certified hypervisors are: Oracle VM 2.2, VirtualBox 3.x, 4.x, Solaris Containers, and Solaris LDOMs.
* VMware and Microsoft Hypervisor are not certified.

(Oracle JDK 7 and JRE 7 Certified System Configurations, 2013)

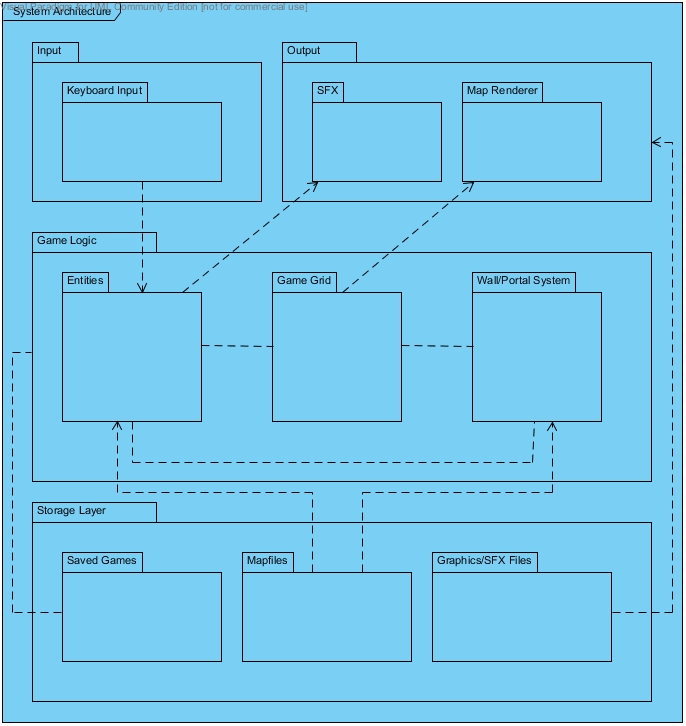
* Hardware:
  + The computer should have an audio output system and speakers or headphones for best experience
  + At least 128 MB of RAM
  + A QWERTY keyboard with arrow keys for control of the game

**2.3 System Environment**

The system runs a Java Virtual Machine on any operating system with Java installed.

There are no specific hardware requirements for the system, other than the assumptions made in section 2.1 and the constraints made in section 2.2.

|  |
| --- |
| **3 Architecture** |



**Figure 3.1 The Architectural design for Drago’s Inferno**

**3.1 Overview**

Our system will implement a three-tier architectural style as follows: input and output on the top layer, game logic on the middle layer, and storage of game data on the bottom layer. The only input required in the game will be keyboard input, implemented with a Java keyboard listener. The output of the game will consist of graphics and sound effects. Graphics will be implemented primarily using Java’s BufferedImage class. Sound effects will be in \*.WAV or a suitable format and implemented using the AudioClip class.

The game logic will consist in the main of three subsystems: one for terrain objects such as projectiles, obstacles, and enemy characters, collectively called “entities,” one for walls, ground textures, and the portal system, and one for a central “game grid” which unites both. The entity subsystem will contain the game physics and the collision response mechanism by which separate entities may collide and interact. The wall and portal system will make use of a data structure called a binary space partition and allow enemy characters to navigate intelligently through the playfield by means of stepping stones called “portals.” The game grid will be used to provide a quick reference to any data elements belonging to, or overlapping, any specific region on the playfield.

The storage layer will make use of three types of files: so-called “mapfiles,” which contain data in the most basic form needed to construct a level, \*.GIF and \*.WAV files for graphics and sound effects, and savegame files. A mapfile will be organized into two sections: “polygons” and “objects.” The polygon section will describe where the walls and ground textures are to be placed (walls can be diagonal). The object section will describe the starting position of each entity on the map, with the exception of tree and stone entities as they are generated automatically from suitable outlined polygons in the polygon section. We have not yet determined the precise format of the savegame file, but it will contain all the information necessary to load a saved game and thus include: (1) player attributes at the time of save, (2) entity locations, states, and directions of motion at the time of save, and (3) miscellaneous environmental data, such as whether a portal in the portal system is in an “open” or “closed” state.

**3.2 Rationale**

Although the three subsystems in the game logic layer will be moderately coupled, the coupling between this system and the output system will be loose. This will allow us to revise the way a gameplay scenario is drawn without having to update the game logic itself. Because we will make use of binary space partitions, it is even feasible in theory to rework Drago’s Inferno into a first-person game using modern 3D rendering techniques. The storage layer, too, is loosely coupled with the game logic layer. By providing only the most basic information needed to generate a map, a mapfile leaves the responsibility of constructing the more sophisticated data structures to the game logic itself. This will allow another developer, for example, to create a level editor without having to worry in detail about the format that a resulting mapfile should have.

The function of the game grid is to optimize gameplay in favor of speed by providing redundant references to entities in the playfield, as well as to textures, walls, and portals. This allows the rendering engine to quickly loop through an array of grid squares to determine what to display on the screen, and also the collision response engine to perform culling tests in determining what obstacles a given entity might collide with. By using a game grid, we utilize a slightly larger memory footprint, but the benefit of this is that we achieve faster gameplay by eliminating slower algorithms (as slow as *O*(*n*2), in some cases).

|  |
| --- |
| **4 High Level Design** |

**4.1 Conceptual View**

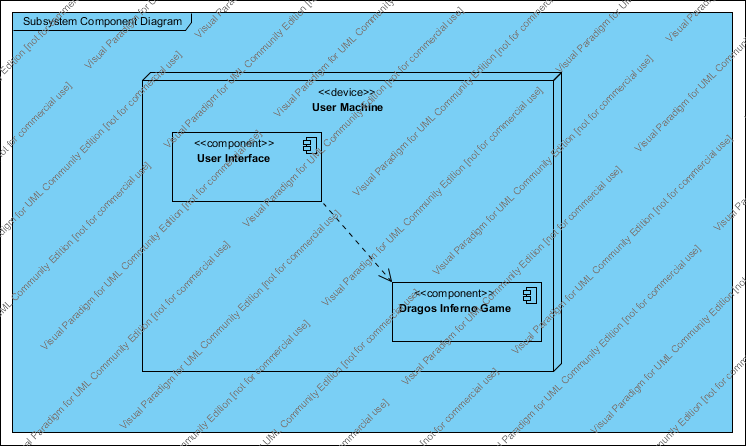


Figure 4.1.1 The Conceptual Diagram for Drago’s Inferno

**4.2 Physical View**

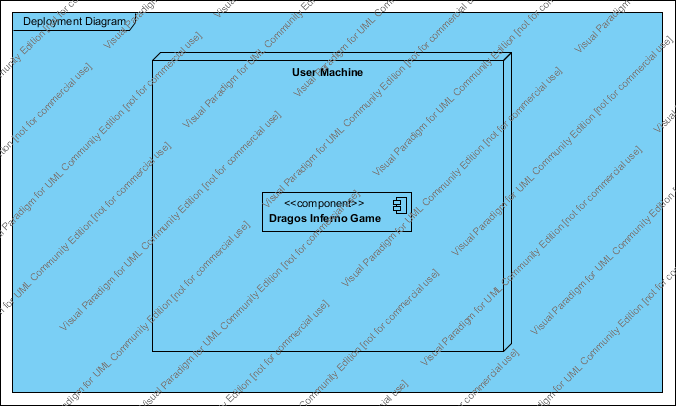
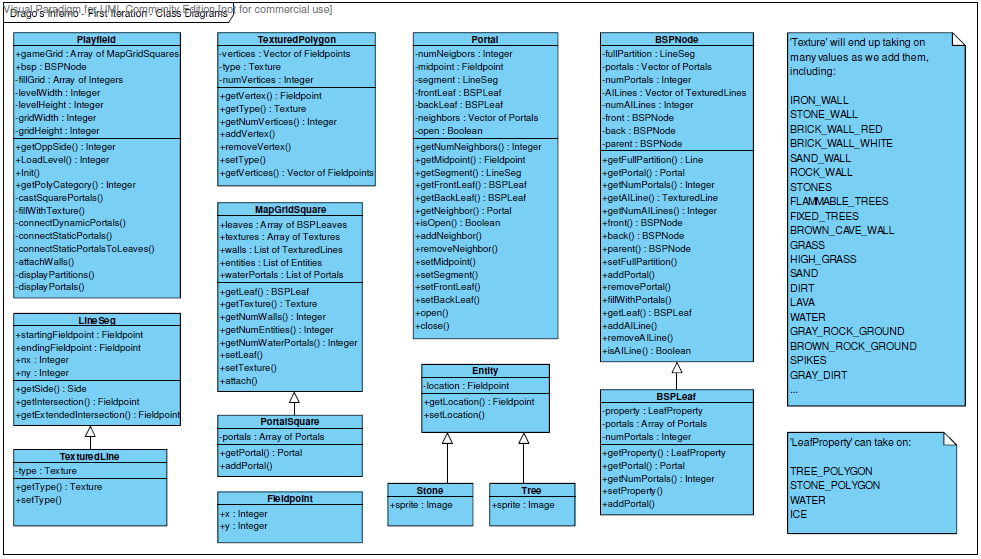


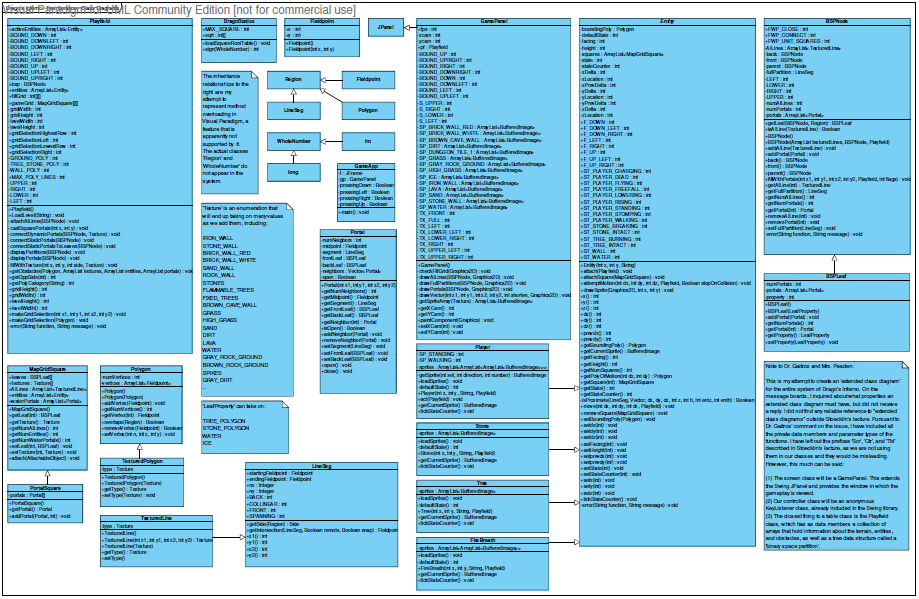
Figure 4.2.1 The Deployment Diagram for Drago’s Inferno

|  |
| --- |
| **5 Low Level Design** |

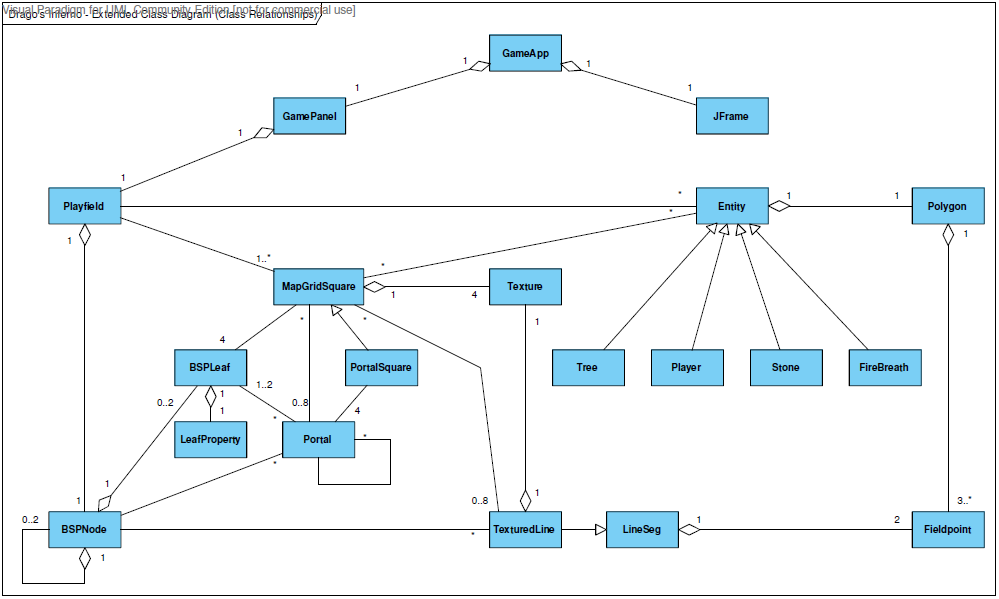
**5.1 Module**



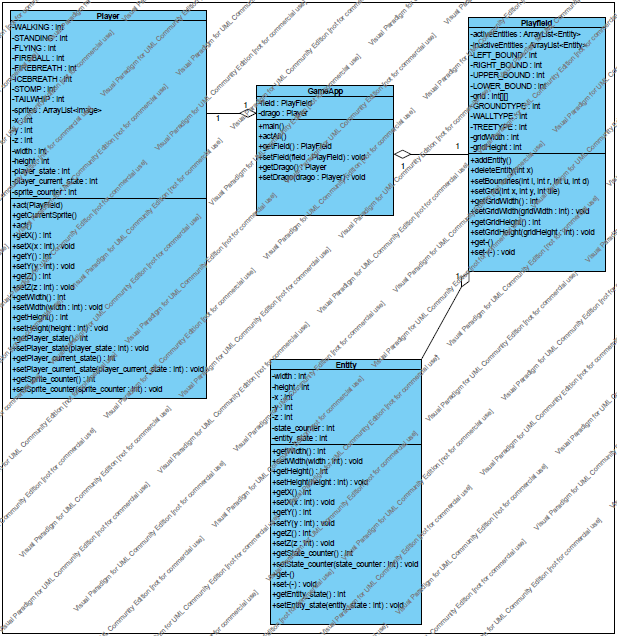
**Figure 5.1.1 The Playfield Class Diagram**



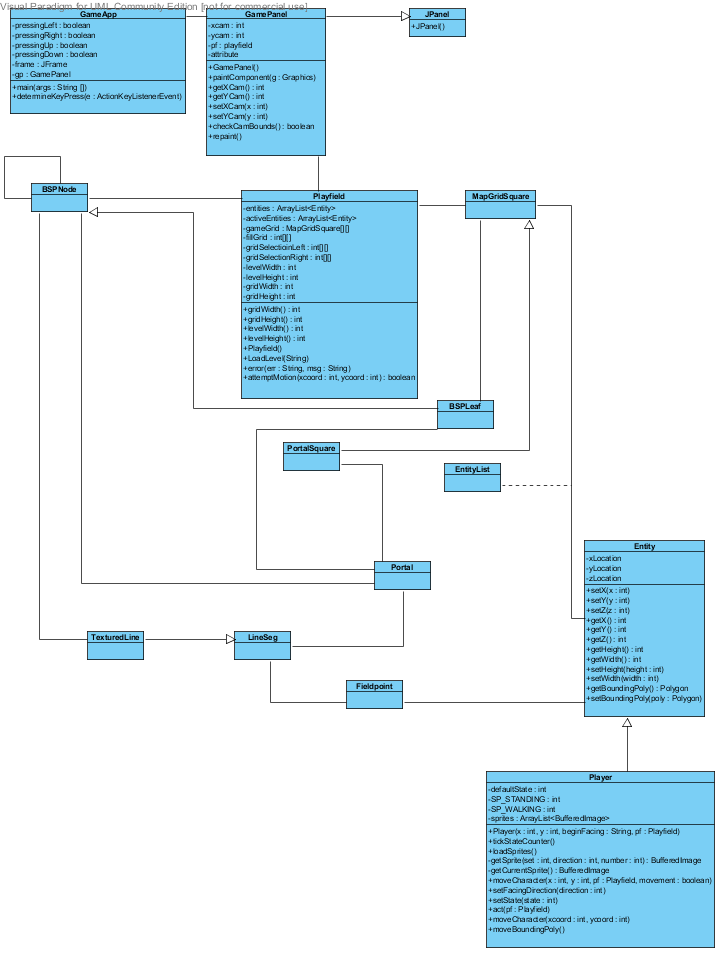
**Figure 5.1.2 The Full Game First Iteration Class Diagram**



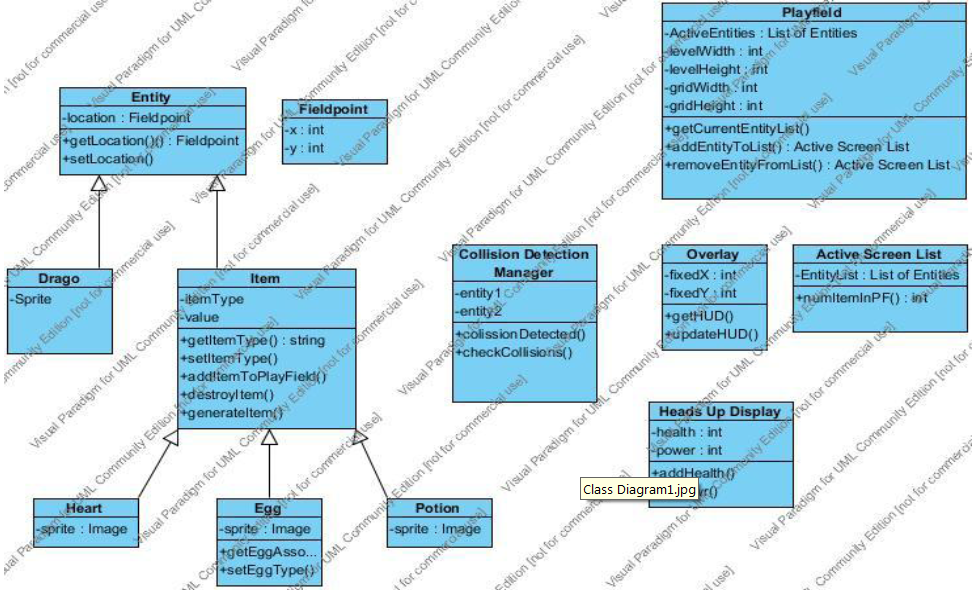
**Figure 5.1.3 The Extended Class Relationship Diagram**



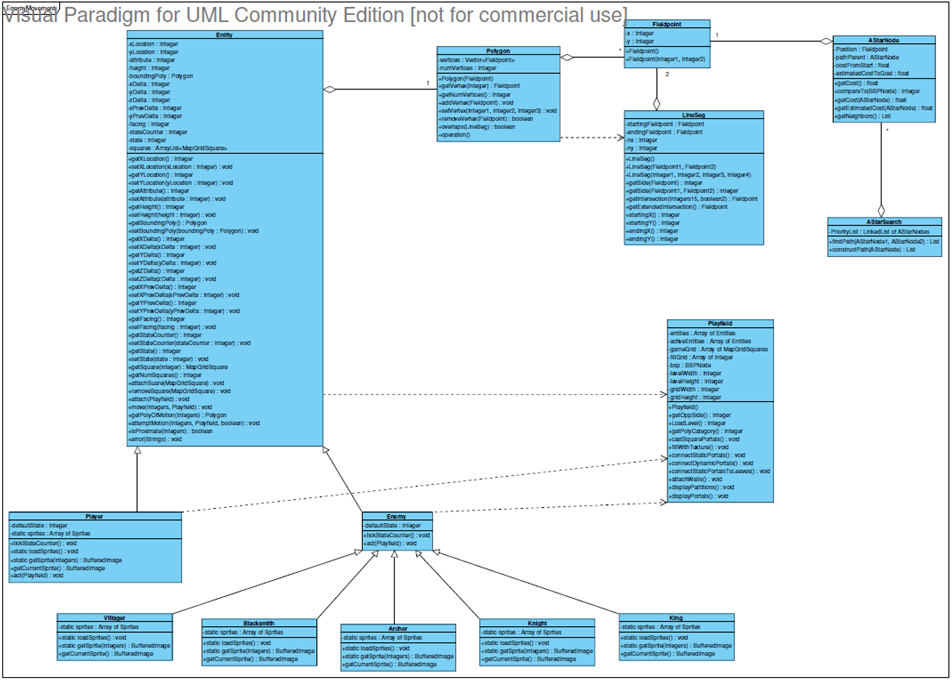
**Figure 5.1.4 Player Flying Use Case Class Diagram**



**Figure 5.1.5 Player Move Use Case Class Diagram**



**Figure 5.1.6 Pickup Item Use Case Class Diagram**

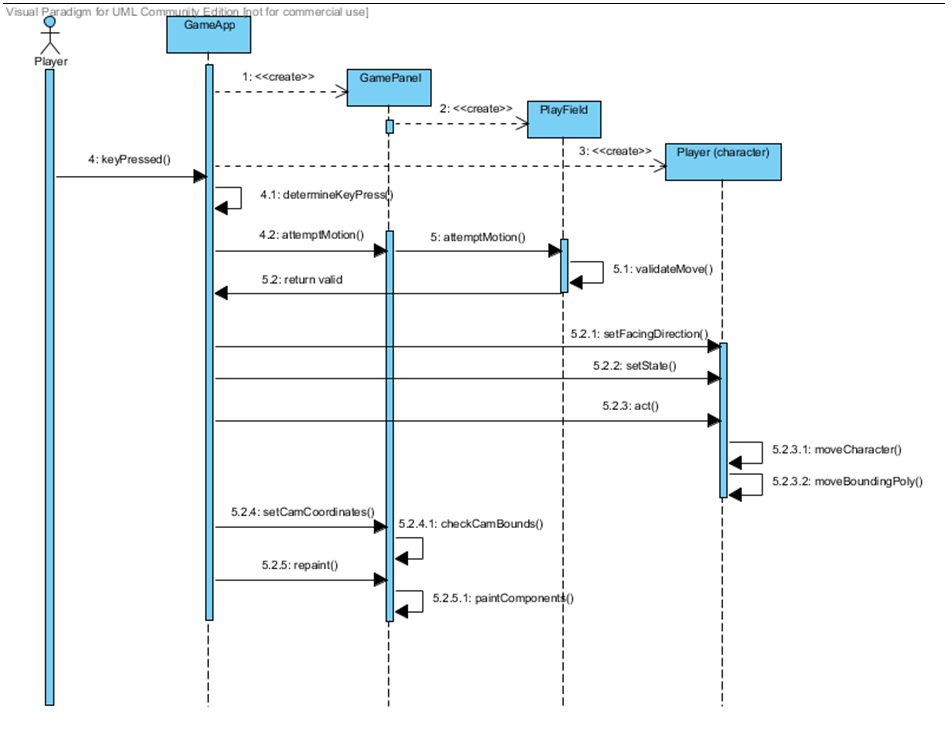


**Figure 5.1.7 A\* Search Use Case Class Diagram**

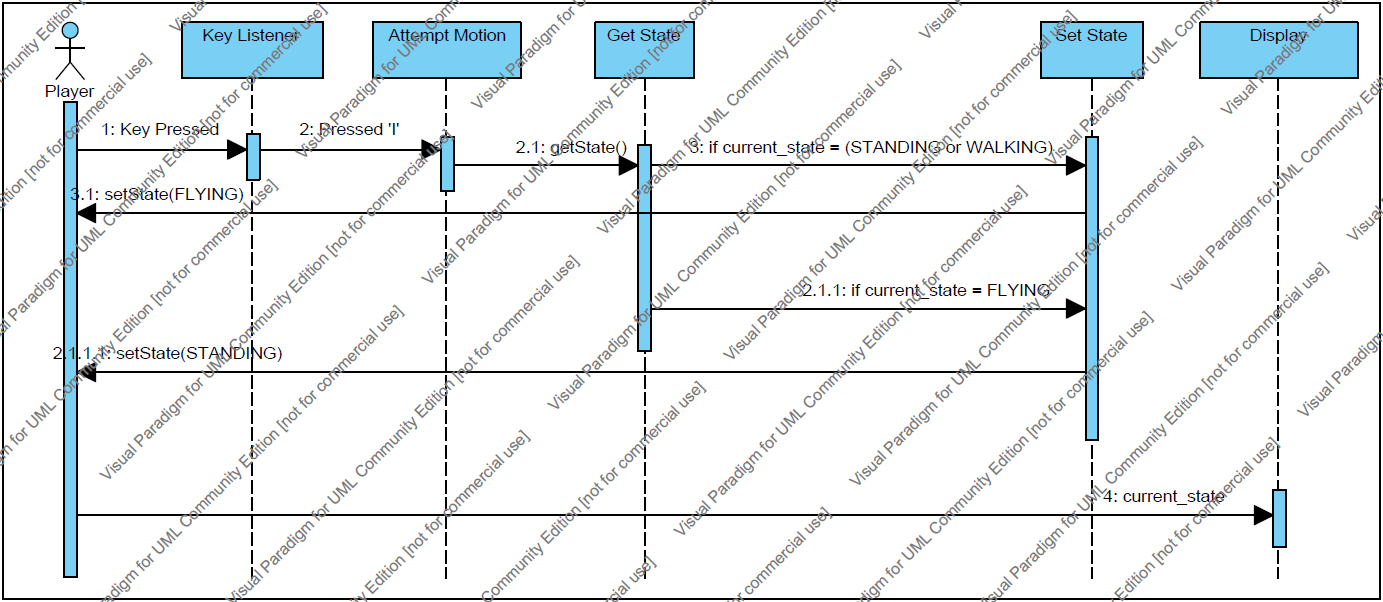
**5.2 Sequence Diagrams**

**5.2.1 Player Motion**

*5.2.1.1 The Player should move when user presses the appropriate key*

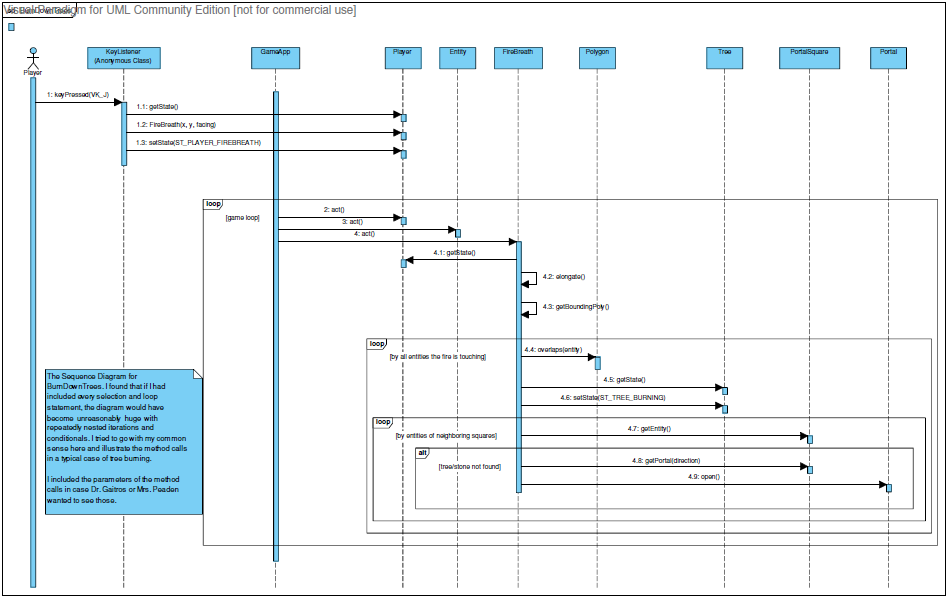
**

*5.2.1.2 The Player should fly when user presses the ‘l’ key*



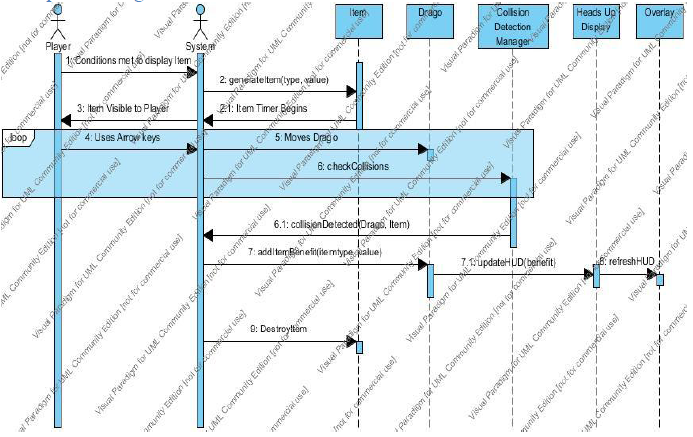
**5.2.2 Playfield Response**

*5.2.2.1 The Tree objects in the Playfield shall burn down and open the Portal if the user launches a fireball attack at them*

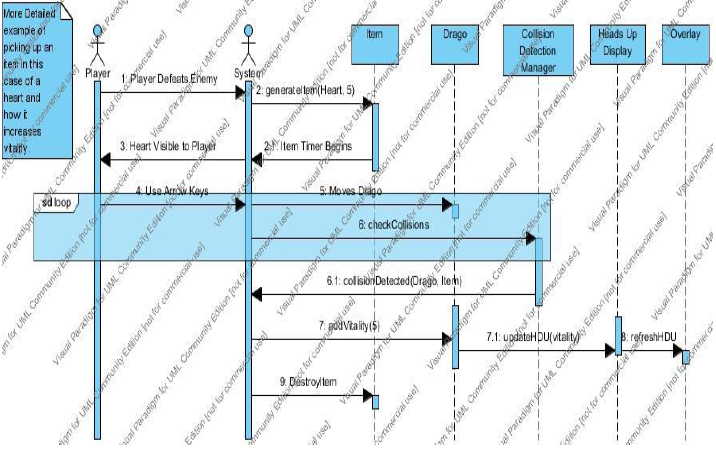
**

**5.2.3 System Response**

*5.2.3.1 The System shall allow the Player to pick up Items and add them to his Inventory*

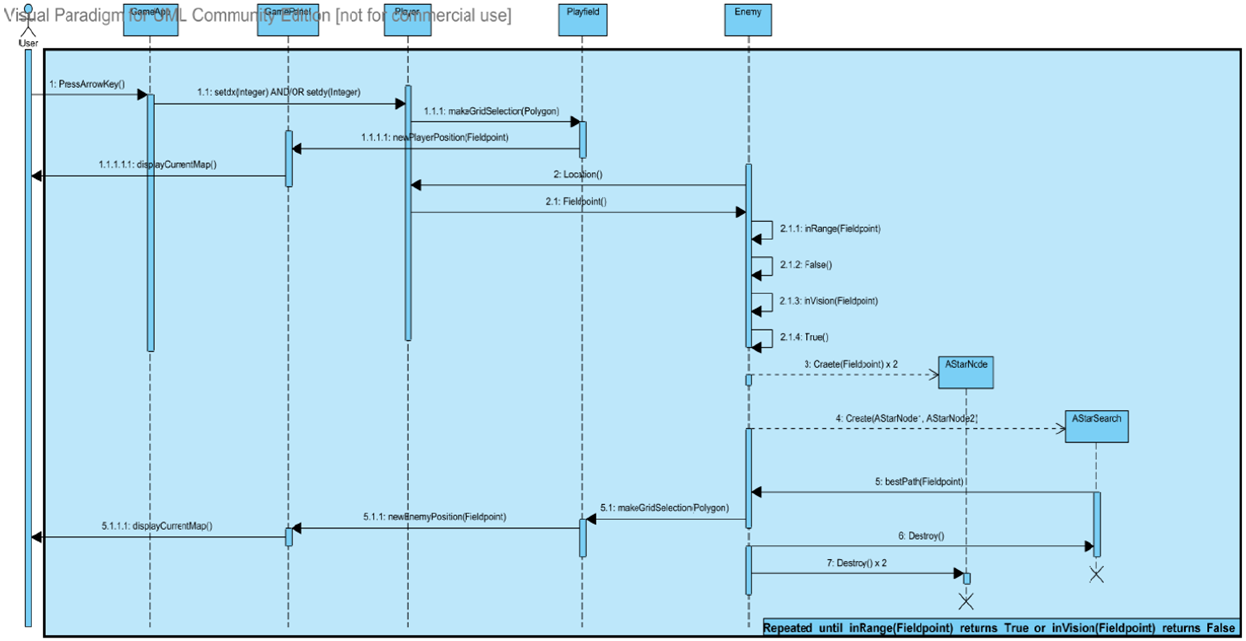
**

*5.2.3.2 The System shall allow the Player to pick up a heart and increase his Vitality*

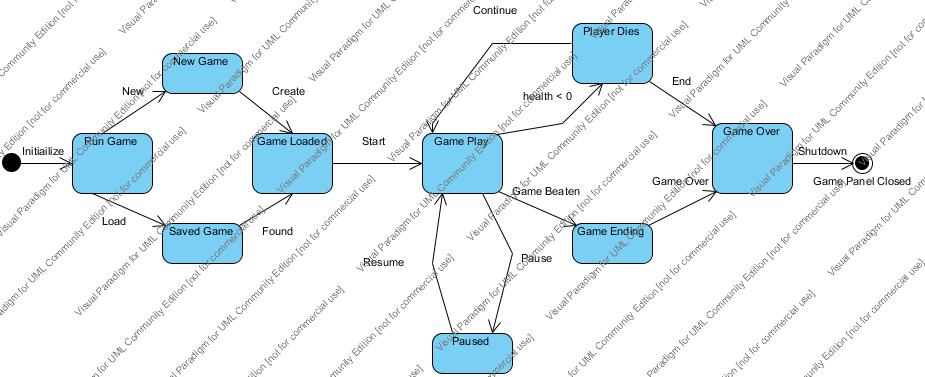
**

**5.2.4 Enemy Response**

*5.2.4.1 Enemies on the Playfield shall use A\* Search to find the best path to the Player*

****

**5.3 State Diagram – General Game Play**

****

**Figure 5.3.1 General Game Play State Diagram**

**5.4 Database Design Diagram**



**Figure 5.4.1 Entity Relationship Diagram**

Shown is the Entity Relationship Model for Drago’s Inferno’s database development. The database only needs to store information about save states made in the game. Each entity in the database, other than the association tables that receive a composite primary key, is given a unique identifier of type *int*. This ensures a unique value for each instance in the table.

#### tb\_Player\_Profile

First of all, the player will create a profile and give that profile a name. That profile may contain multiple save points if the player chooses.

#### tb\_Save\_State

The tb\_Save\_State entity is the central part of data and the relationships in the database. The tb\_Save\_State entity will store information about the player’s current skill level, character health, and total game time played up to the point of that save state. Because the relationship to the profile is an identifying relationship then the foreign key of ProfileID part of a composity key for the save state.

#### tb\_Level

This entity contains the level identifier so the correct level can be loaded to be played. The LevelName attribute gives the player a verbal reference to the level, for example ‘The Cave’ may be the level name but the ID may be 12.

#### tb\_Weapon

This entity contains information about every type of weapon in the game. Each weapon has a name, damage value to determine damage inflicted, and monetary value used to buy and sell the weapon in the game.

#### tb\_Saved\_Weapons

Since a save state may contain many weapons and a single weapon may also be associated to many save states then there is a many-to-many relationship between these entities. In SQL this is performed using an *association table* or *junction table.* The association table simply contains references to the primary keys of each table.

#### tb\_Item

Similar to weapons, items can be carried and saved in the game. Each item has a common name, a monetary value, a type that tells us the general purpose or category of the item (defense, offense, dexterity, protection, etc.) for inventory sorting purposes, and an affect which tells us the specific effects of the item (restore 10 health, reduce damage for 30 seconds, etc.).

#### tb\_Saved\_Items

Similar to the tb\_Saved\_Weapons association table, this entity accomplishes the same task of allowing a many-to-many relationship between the tb\_Save\_State and tb\_Item tables.

#### tb\_Character\_Ability

The character in the game also gains abilities as the game progresses. Some of the abilities include the ability to fly, to tail whip, to breathe fire, to breathe ice, and several other abilities. The ability is given a common name and an effect that tells the game engine what this ability does.

#### tb\_Saved\_Abilities

Another association table is used to save the character abilities in the game.

To retrieve the values for a save state then the game will do a join of the tables on the primary keys to gather all information about the save state to start the game at the beginning of the saved level.

|  |
| --- |
| **6 User Interface Design** |

**6.1 Application Control**

**Visual Aspects**

The display window for the game will be the Java applet style window with no additional feautres besides minimize, maximize, and close. The game will not feature any toolbars.

**Minimum System Requirements**

Idealy, the use is running the application on Windows 98 or higher or Macintosh OS X.4. The application can be run if the users machine is running the latest version of Adobe Flash Player.

**6.2 Screenshots**



Figure 6.2.1 shows the general look and feel of the game



Figure 6.2.2 shows the layering effect to get sprites drawn in the correct order



Figure 6.2.3 shows the destructible terrain

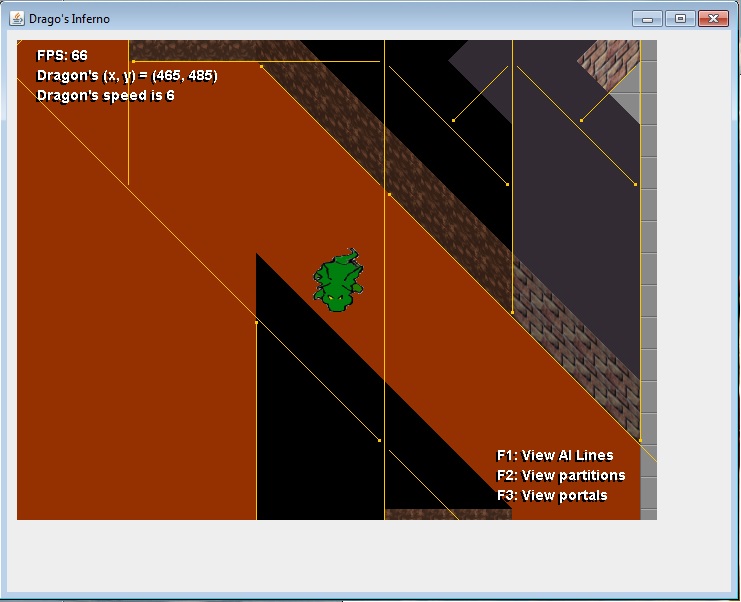


Figure 6.2.4 shows the binary space partition of the polygons that make up the map

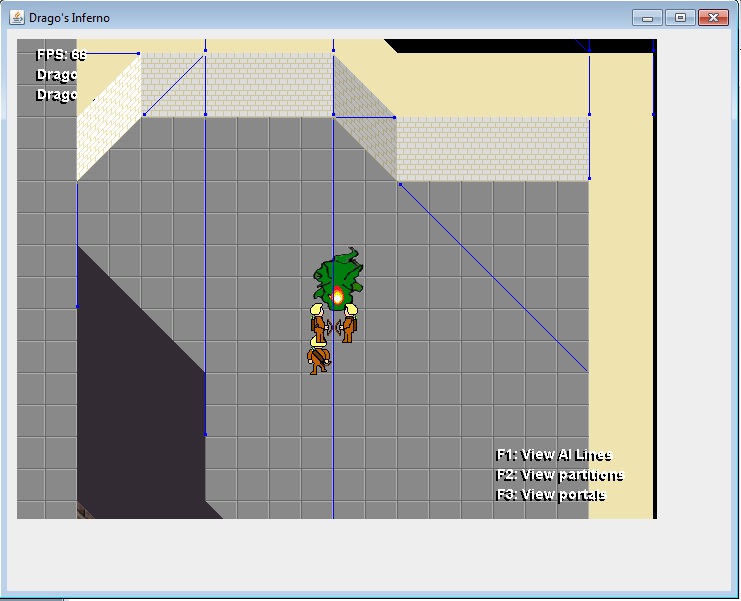


Figure 6.2.5 shows the invisible portals on the map that aid enemy AI movement



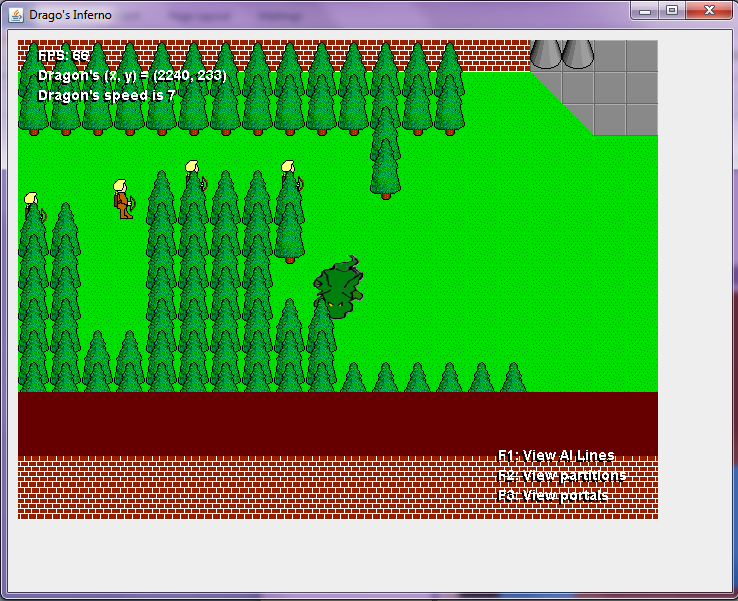
**Figure 6.2.6 shows an egg that Drago must rescue**

****

**Figure 6.2.7 shows Drago’s fireball striking the Enemies (Archers)**

****

**Figure 6.2.8 shows the water obstacle that Drago cannot cross**

****

**Figure 6.2.9 shows the Enemies navigating obstacles to reach the Player (using A\*)**

****

**Figure 6.2.10 shows the Dynamic Portals surrounding obstacles**