Keith Francalangia

Imagination

Project Documentation



# Executive Summary

This project is a Single User Dungeon crawler game called *Imagination*. It utilizes PNG images to generate 2D graphics, as well as a supplementary combat dialog. The player interacts with the environment by typing action commands at an interpreter (i.e. “attack”). The game combines the nostalgic feel of a text-based dungeon crawler, with more modern graphics, to create a unique gaming experience. The in-game user interface includes a dungeon map showing where the player has been, a list of *Insights* known and a tutorial on how to play the game.

My nostalgia for this style of game comes from text based Multi-User Dungeon (MUD) games like Gemstone IV. The series began as Gemstone in 1988, and it is one of the longest running online games still active (Gemstone IV). The advantage of a text-based game is it allows for nearly unlimited creative possibilities. *Imagination* will appeal to those that value creativity and freedom over fixed plotlines and one dimensional combat consisting of endlessly clicking the mouse button. It will also appeal to those that play table top Role Playing Games like Dungeons and Dragons, which is now in its fifth edition (Dare to Descend Rage of Demons).

The game will initially be released for the desktop operating system. I am choosing to revitalize an older style of game because I don’t want to compete with big budget 3D games like Diablo III and Borderlands 2. I want to set myself apart by making a retro game with unique features I desire in a game experience, but have never seen before.

The alpha version of the game will consist of you, the Hero, navigating an endless dungeon containing one enemy type, the giant rat. The goal of the game is simple: get at far from the starting room as possible. Beware! The greater the distance, the stronger the enemies become!

Health and Imagination are the primary Hero statistics. When the Hero gets to zero health, the game ends. Imagination grants the Hero bonuses to attack and defense. It is gained by having insights or killing enemies, and it is lost when the Hero takes damage. Insights are a take-off on traditional fantasy game spells. The initial version of the game will have three action commands: *attack* (swing sword), *second wind* (heal over time) and mage armor (bonus to defense).

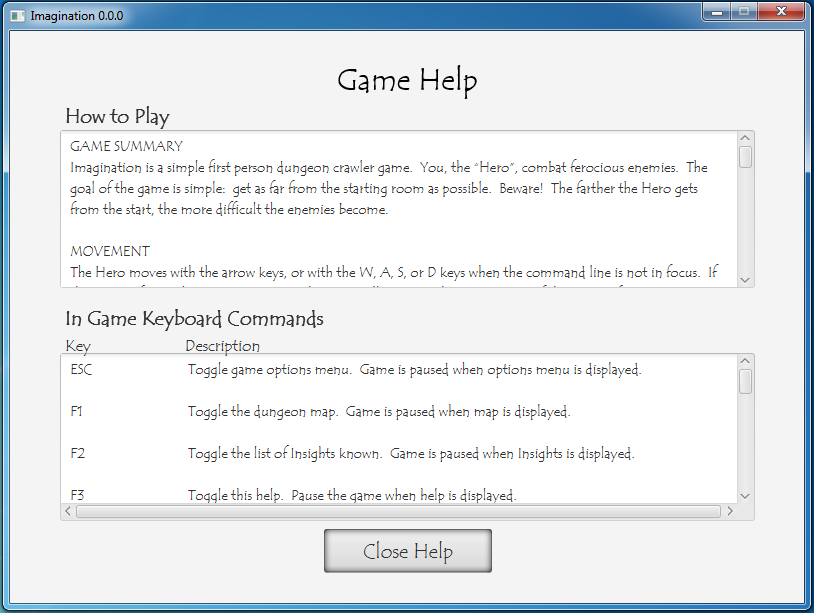
# Imagination Title Screen

This is the title screen which is displayed when the application starts. It has three buttons: *Start Game*, to start the game; *Quit* to exit the application; and *Help* to display the help screen.



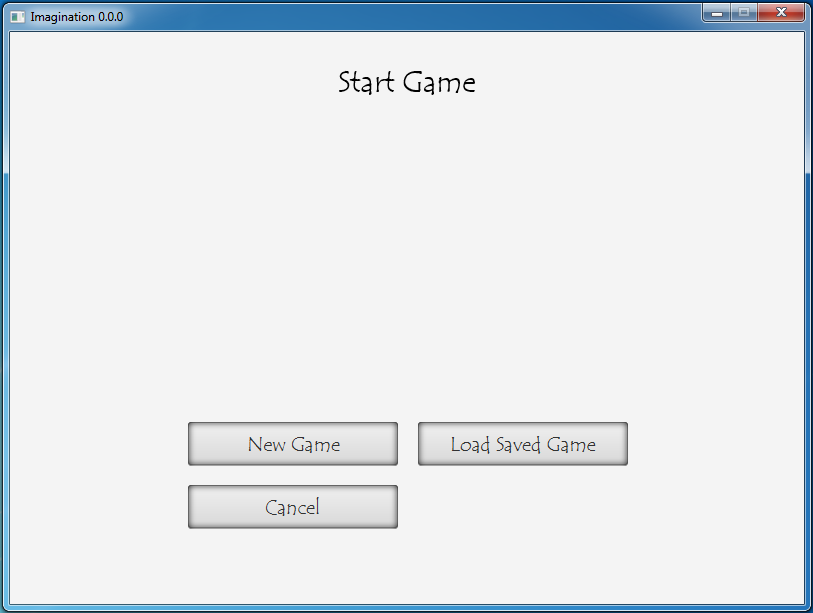
# Help Screen

This is the Help Screen which provides instruction on how to play the game. It is accessible from the splash screen or by pressing F3 in game. The game is paused if accessed from in game.



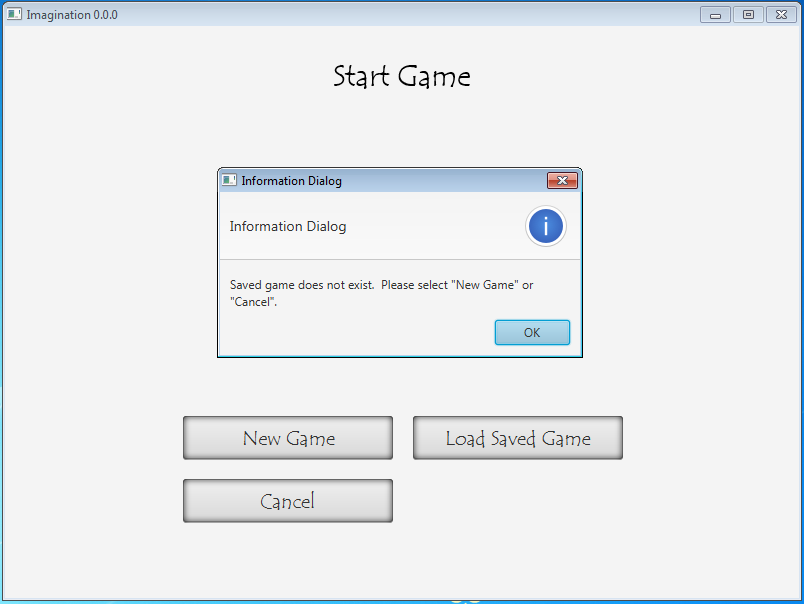
# Start Game Screen

This screen is displayed when the *Start Game* button is pressed from the Title Screen. It has three buttons, *New Game*, to start a new game; *Load Saved Game*, to load a saved game; and *Cancel*, to go back to the Title Screen.



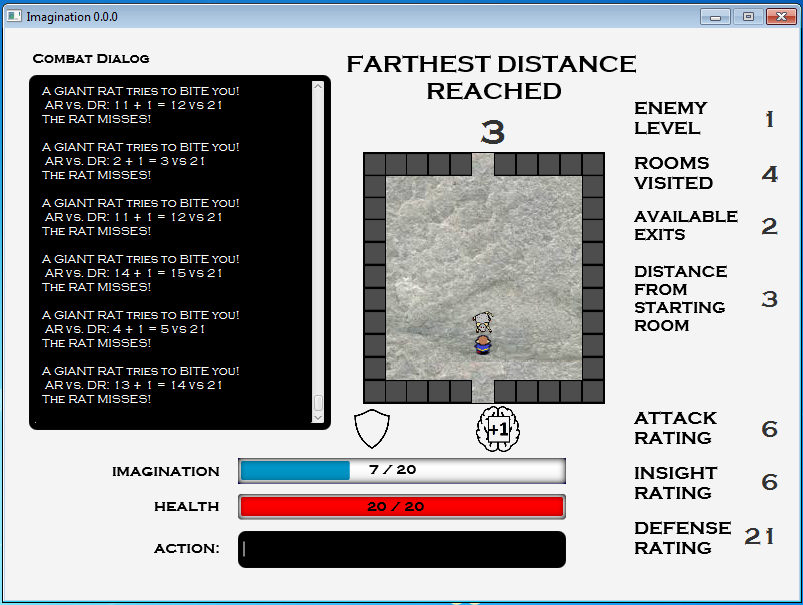
# Load Game Error

If the *Load Saved Game* button is pressed and a saved game doesn’t exist, the Start Game Screen will display this error message.



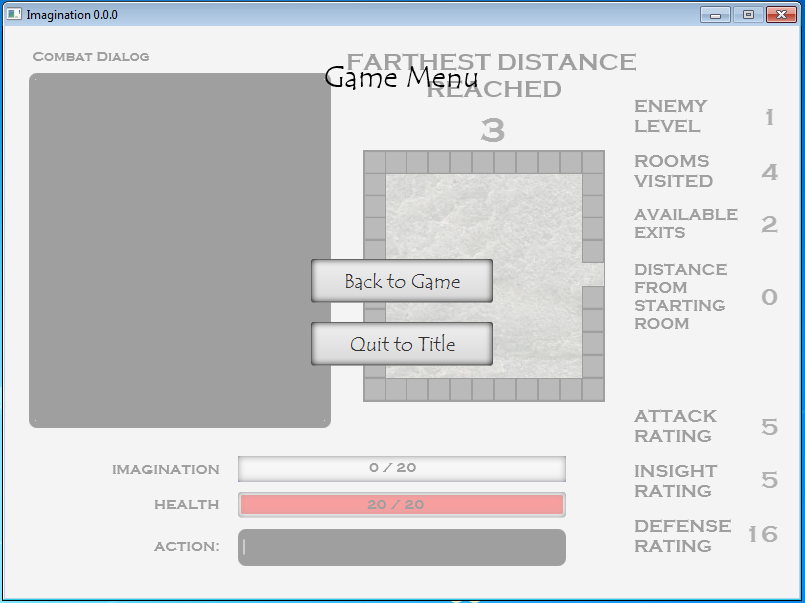
# In Game Screen

This is the in game screen. This is what is displayed when the game is played.



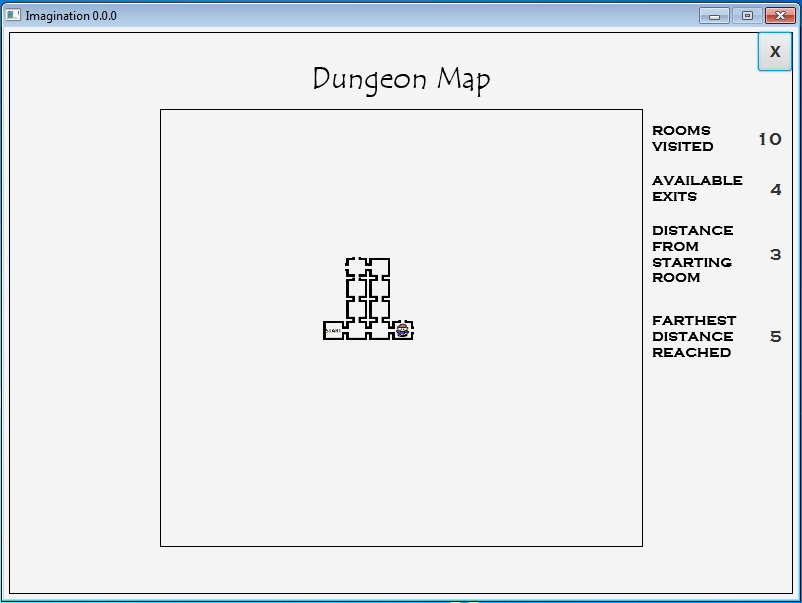
# In Game Menu Screen

This is displayed when ESC is pressed from the In Game Screen. It contains two buttons: *Back to Game* resumes the game and *Quit to Title* saves the save and brings the user back to the Title Screen. The game is paused when this screen is accessed. Pressings ESC again will also resume the game.



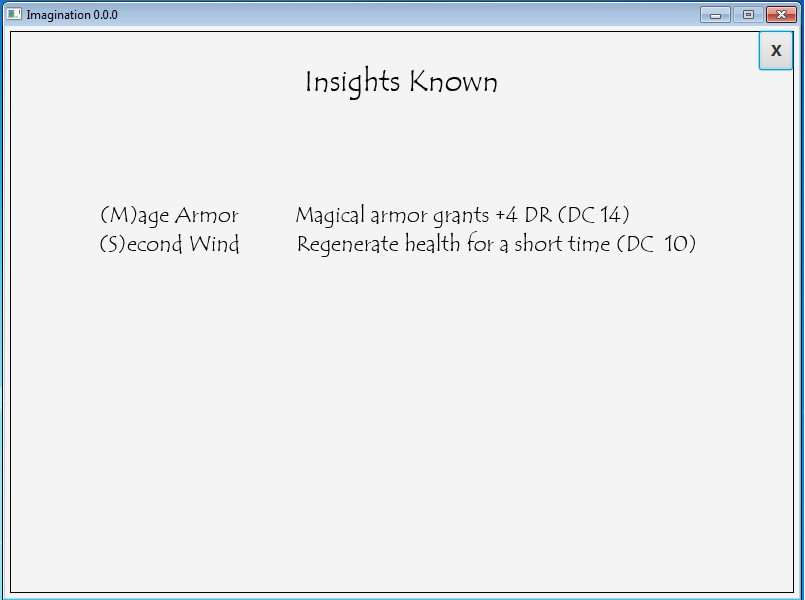
# Map Screen

This screen is accessed by pressing F1 from the In Game Screen. This pauses the game. This screen displays the rooms visited in the dungeon, including the starting room. Pressing F1 again resumes the game. Pressing the “X” button also exits the map and resumes the game.



# Insights Known Screen

This screen is accessed by pressing F2 from the In Game Screen. It displays the insight action commands the Hero knows. The letter in parenthesis ( ) is a shortcut for the command.



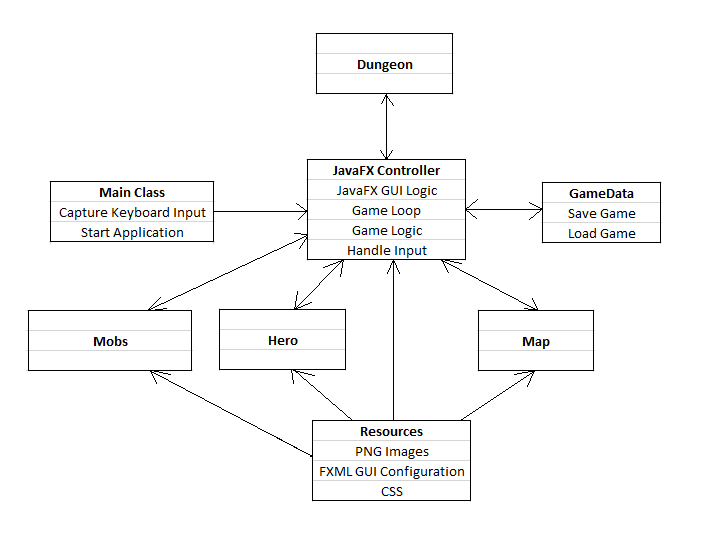
# Game Over Screen

This screen is displayed when the Hero gets to zero hit points and dies. It has two buttons: *Restart Game* restarts the game and *Quit to Title* brings the user to the Title Screen.



# System Architecture

The heart of the game is the JavaFX Controller class. It contains GUI logic, the game loop, game logic and input handlers. The *ImaginationMain* class starts the application and hands off control to the controller. It also captures keyboard input. System resources include images, the FXML GUI configuration file and a CSS file. The images are used by Mobs, *Hero* and *Map*. The FXML and CSS pages are used by the controller class. The *Dungeon*, *GameData*, Mobs, *Hero* and *Map* all communicate to and from the JavaFX controller class. Below is a diagram depicting the system architecture.



**Imagination System Architecture Diagram**

## 

## Source Code Structure

The source code is contained in a NetBeans IDE project title *Imagination*. The following is a summary of the source code directories and their contents:

|  |  |
| --- | --- |
| **Code Directory** | |
| **Directory** | **Usage** |
| src/main/javafx | Contains all Java source code classes. |
| src/main/javafx/resources | Contains a FXML GUI configuration file and a CSS file. |
| src/main/javafx/resources/images | Contains all PNG images used in the game. |
| src/executable/ | Contains the game executable *Imagination.exe* located in Imagination\_x64.zip file. To run the application, this file needs to be downloaded and unzipped. The executable can then be run. |
| *Highlighted rows indicate directories containing source code.* | |

# Executables

The game executable is located in the src/executable/Imagination\_x64.zip file on GitHub in the imagination-sud repository. To run the game, this file needs to be downloaded and unzipped. The executable *Imagination.exe* can then be run.

The game was tested using the 64 bit Windows 7 operating system. The Java Runtime Environment is bundled with the game executable, so a Java installation is not required.

### Imagination.exe

This is executed to run the game.

# Code Architecture

The game is played by the *Hero* moving through the dungeon, beginning with a starting room. The *Hero* attacks giant rats and invokes insights from a command line interface. Rats move and attack the *Hero*. Combat consists of a die-twenty roll plus a modifier. The goal of the game is to get as far from the starting room as possible. The farther the *Hero* gets, the higher the attack rating of the rats, and the greater number of rats the *Hero* will encounter. When the *Hero* gets to zero hit points the game ends.

The basic building blocks of the program are as follows. *ImaginationMain* initializes the JavaFX application. It captures keyboard input and passes it to the *ImaginationController* input handlers. *ImaginationController* is the heart of the game, containing input handlers, a *GameLoop* internal class, a *Hero* player class, a *Dungeon* class, a *Map* class and a *GameData* class.

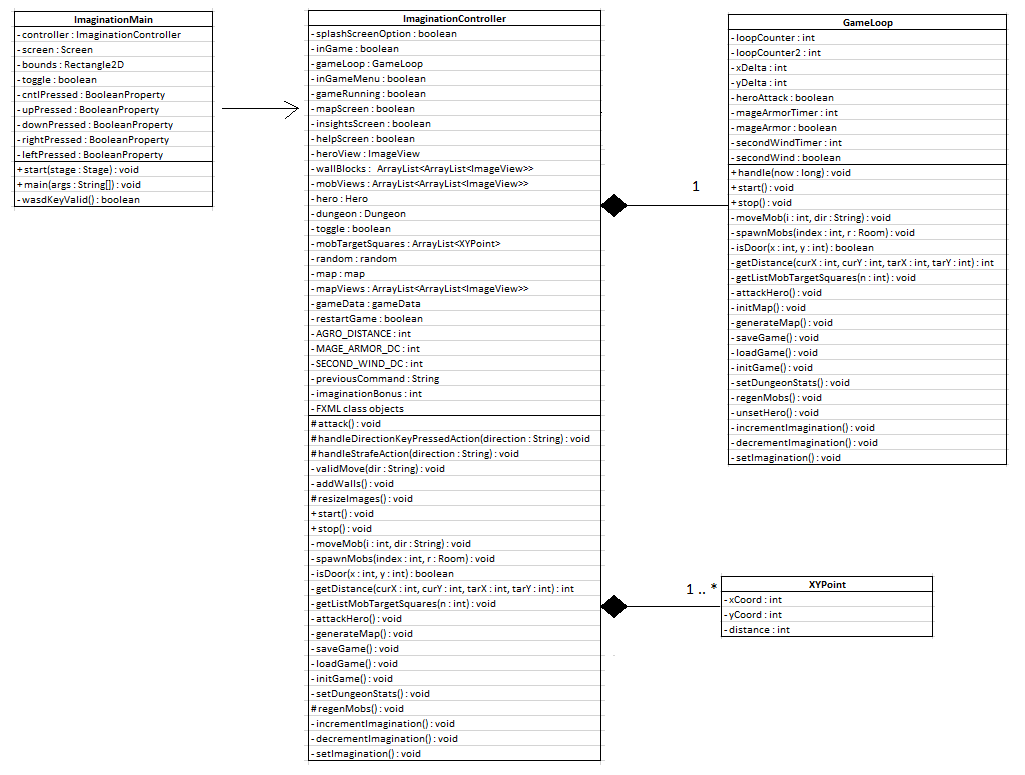
*Hero* contains player data and methods. *Dungeon* contains the game dungeon. It is composed of *Room* class objects, and each *Room* is composed of 11 x 11 S*quare* class objects. The *Room* class is composed of a *GiantRat* mob class, which extends abstract class *MobTemplate*. *MobTemplate* contains data and methods common to all mobs, and *GiantRat* rat contains code unique to giant rats. *Map* contains dungeon map data and methods.

*GameData* contains data and methods used to save and load the dungeon. Object serialization is used to save a dungeon object to a file when the game ends. To load a saved game, the file is then read and the object is retrieved. Note that this requires *Dungeon, Room* and S*quare* to implement the *Serializable* interface.

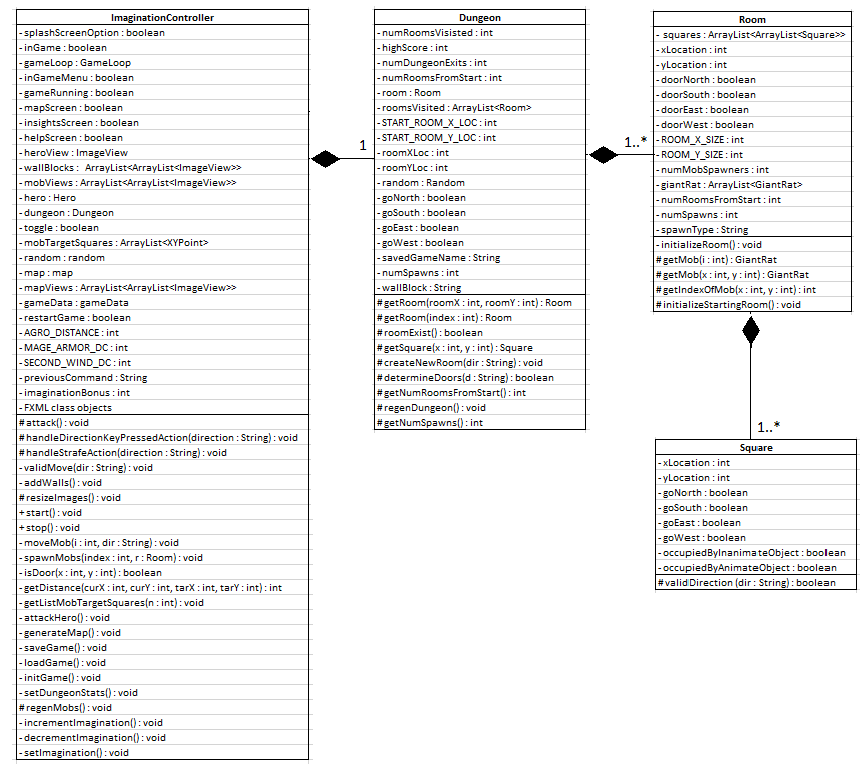
The dungeon is procedurally generated, with a random number and location of doors for each *Room*. This means that the dungeon layout will be unique for each new game. Note that it is possible, especially at the start of a new game, for the number of available dungeon exits to equal zero. This results in a closed dungeon, and the game is effectively over. Therefore, it is beneficial to move as far from the starting room as possible, especially at first, to prevent closing the dungeon.

*ImaginationController* contains the *GiantRat* AI logic, as well as the *GameLoop,* which extends *AnimationTimer*. The class contains a *handle* method that executes once per frame. The game is configured for rats to move or attack once per 15 frames. If the rats are greater than 5 squares from the *Hero*, they will move in a random direction. If they are less than 5 squares, they will move to an adjacent cardinal direction square next to the *Hero* and attack.

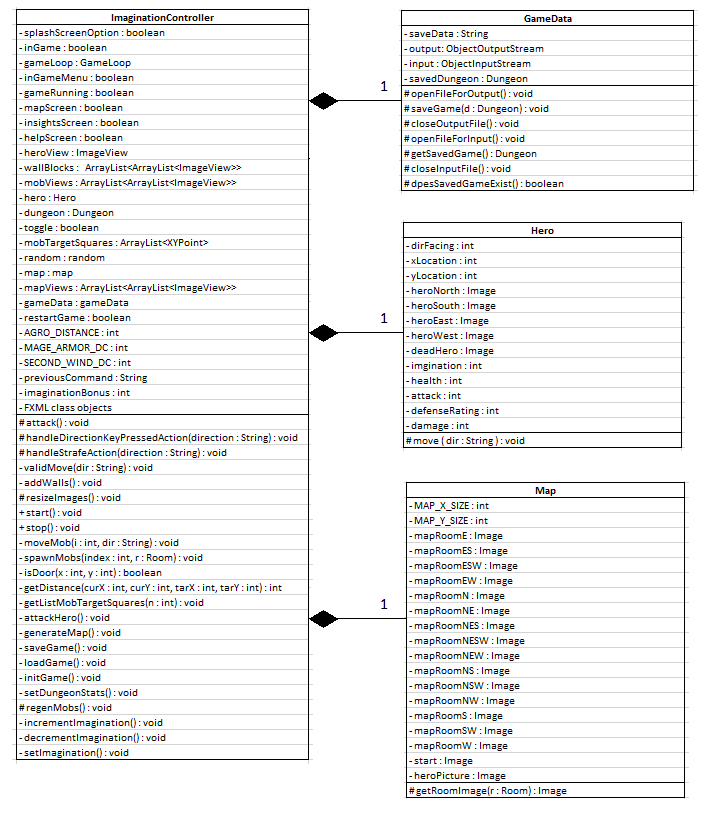
Finally, the *XYPoint* class is used by *ImaginationController* as part of a rat move algorithm that calculates a list of target squares next to the Hero to move to. Below is a series of 4 UML Class Diagrams describing the code architecture.



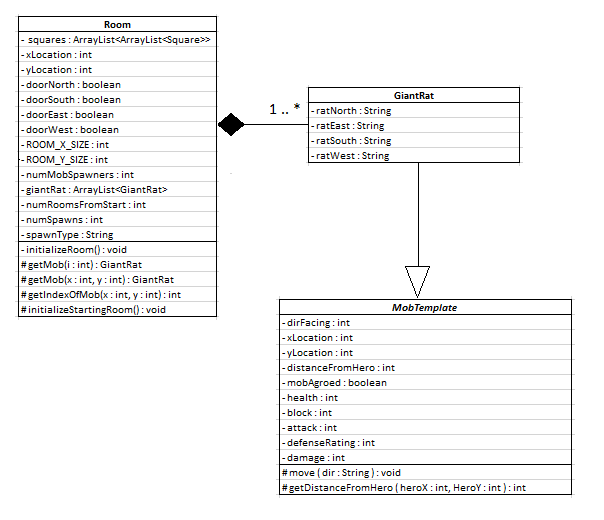
**UML Class Diagram 1**



**UML Class Diagram 2**



**UML Class Diagram 3**



**UML Class Diagram 4**

## Database or Data Store

The game saves or loads a *Dungeon* object using Object Serialization. This will save the state of the object as a sequence of bytes. The object can then be loaded and the game resumed at a later time. The saved file name is *dungeon.ser* and will be located in the same directory as the *Imagination.exe* executable file. Note that exception handling and try/catch code is used to detect file input/output errors.

External Files & Data

The Imagination\_x64.zip file contains the Imagination.exe executable. This is bundled with all files required to run the executable, including the Java Runtime Environment. Note that Java does not need to be installed to run the game.

Programming Language

The program is written in Java, using JDK 1.8.0\_60 (64-Bit) and the NetBeans 8.0.2 IDE. It uses the JavaFX 8 platform. There is a small amount of CSS and FXML used to configure the JavaFX GUI.

Project Classes

Classes within the project are used to abstract re-usable pieces of code. Classes are also used to group related values, known as properties. The project utilizes these classes:

### Game Dungeon | Dungeon.java

Class contains data and methods related to the game dungeon. It is composed of *Room* and *Square* objects. The dungeon object is saved when the game ends, and is retrieved when the game is loaded.

### Manage Game Data | GameData.java

Class contains data and methods required for saving and loading the game.

### Giant Rat Mob | GiantRat.java

Class contains data and methods related to giant rat mobs.

### Hero | Hero.java

Class contains data and methods related to the *Hero*.

### Controller class | ImaginationController.java

Class contains input event handlers. It is composed of the following classes: *Dungeon*, *GameData*, *Hero* and *Map*. It contains the *GameLoop* which updates mob movement and combat.

### Main Class | ImaginationMain.java

Launches the application and captures keyboard input which is passed to *ImaginationController*.

### Map | Map.java

Contains data and methods used to generate the dungeon map.

### Mob Template | MobTemplate.java

Abstract class inherited by *GiantRat*. It contains data and methods common to all mobs.

### Room | Room.java

Contains dungeon *Room* data and methods, as well as an *ArrayList* of *GiantRat*(s) for the Room

### Square | Square.java

Contains dungeon *Square* data and methods.

### XYPoint | XYPoint.java

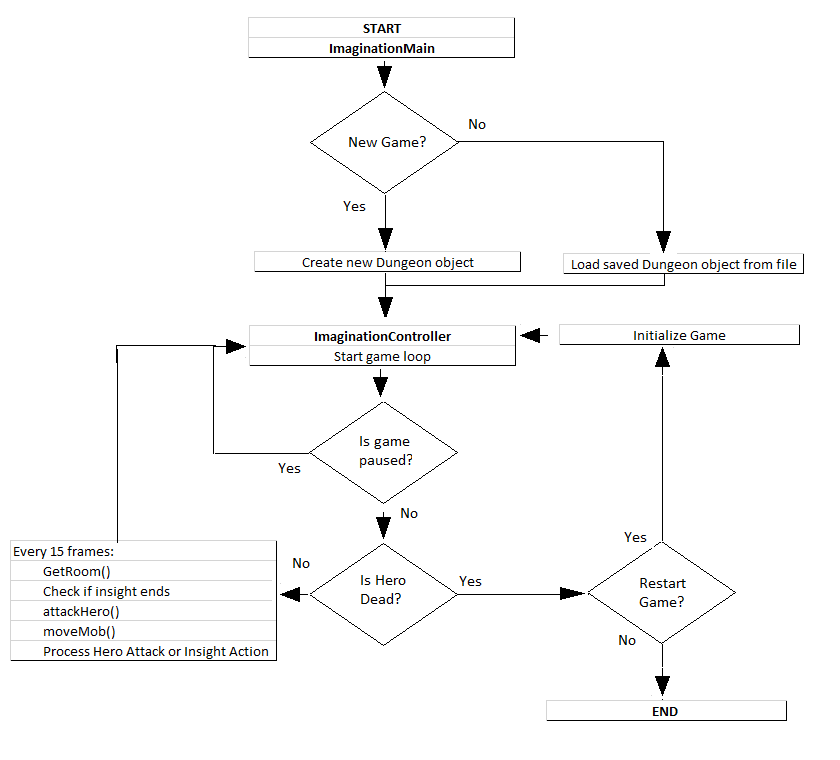
Used by *ImaginationController* for the algorithm to determine a list of ordered target squares for mobs.

Program Start and End Flow

The program begins by executing *ImaginationMain*, which contains the *main* method. If a new game is selected, a new dungeon object is created. Otherwise, the saved dungeon object is loaded from a file.

Next, the game loop is started. If the game is not paused, the program will then check if the *Hero* is dead. If the *Hero* is still alive, the following is performed every 15 frames: the dungeon room is retrieved, a check if the insight has ended is performed, the mobs will either move or attack the Hero, and a Hero action is processed.

If the *Hero* is dead, check to see if the game is restarted. If yes, initialize the game and restart the game loop. If no, the game ends.



**Imagination Flow Chart**

Summary

This system documentation describes the Imagination Game. The game is a dungeon crawler game with a simple goal: get as far from the starting room as possible. The game takes inspiration from old school text-based games, along with more modern techniques, to create a unique gaming experience.

The system documentation begins with the Imagination Project Logo. It includes an executive summary describing the motivation to create the project. Usage scenarios are documented with captures of all game displays. The system architecture is described, along with a diagram. The source code structure is documented, along with the game executable.

Next, the code architecture with UML Class Diagrams is presented. A description of how the game is saved or loaded using object serialization is provided. External file requirements are presented, with no external files required to run the game. The programming language, Java, is then presented. Project classes are listed with a brief description of each. Finally, the program flow is described, along with a flow chart diagram.

# APPENDIX A (BUILD AND RELEASE PROCESS)

This project was created using the NetBeans IDE 8.0.2, and Git was used for version control within the IDE. The build and release process works as follows. After sufficient testing is performed to verify the update, the code is committed to Git. A Clean and Build is issued in NetBeans, which creates the required *dist* distributable directory containing an executable *Imagination.jar* file. The file is copied into a folder named *Imagination\_x64* under a *bin* directory. The Java Runtime Environment 1.8.0\_60 (64-Bit) executable *jre* directory is also copied to the *bin* directory.

[Launch4j](http://launch4j.sourceforge.net/) is then used to create the *Imagination.exe* executable file under the *Imagination\_x64* directory. This effectively bundles the executable with the JRE and jar file. Finally, *Imagination\_x64* is zipped and deployed.

# 

# APPENDIX B (CLIENT INSTALLATION INSTRUCTIONS)

The program is deployed for the 64-Bit Windows 7 Operating System only. The client needs to download the *Imagination\_x64.zip* file and unzip it. To run the program, the client needs to execute the *Imagination.exe* executable file located under the *Imagination\_x64* directory. Note that the executable is bundled with the Java Runtime Environment, so a Java installation is not required.

# APPENDIX C (DEVELOPER SETUP INSTRUCTIONS)

This project was written on the 64-Bit Windows 7 Operating System using the 64-Bit NetBeans 8.0.2 IDE. It requires JDK 1.8.0\_60 (64-Bit). The project is a NetBeans JavaFX FXML Application. Gluon SceneBuilder 8.0.0 was used to edit the *Imagination.fxml* GUI layout file.