CS 141 Final Project Report

Table of Contents

[Introduction 1](#_Toc468352646)

[Approach 1](#_Toc468352647)

[Design 2](#_Toc468352648)

[Discussion of Implementation 4](#_Toc468352649)

[Testing data 5](#_Toc468352650)

[Conclusions 15](#_Toc468352651)

[Suggestions for Improvements 15](#_Toc468352652)

# 1.0 Introduction

Our team, the Constructors, consists of James McCarthy, Rigoberto Canales Maldonado, Owen Dugmore, and Yash Bhure. James was the group leader. Our task was to implement a turn-based game where the player would navigate through a pitch-black building, trying to find a briefcase containing classified enemy documents, and avoiding ninja-assassins. We began the project on November 8, presented it to the class on November 29, and submitted the completed project on December 2.

# 2.0 Approach

Upon receiving the details of this assignment, we immediately got together and began to discuss how we would tackle this challenge. Our two main goals for this first meeting were brainstorming a design for the program and establishing a means of collaboration to be used throughout this lengthy assignment. With pen and paper, we began to hash out a rough layout of our design and the classed we would need. With a rough layout of the program in hand, we came to the conclusion that GitHub would be the best way to collaborate on this project going forward. We then established internal deadlines for ourselves and began to divvy up individual work according with an eye towards each team member's strengths.

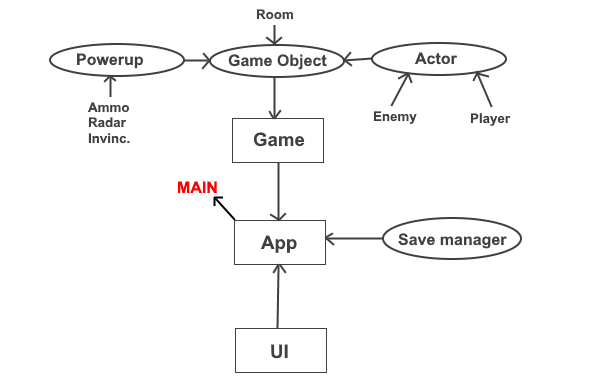
To stay in touch regarding bugs, issues, and deadlines, we also decided to establish a line of communication between group members via group text messaging. Due to our effective early planning, subsequent group meetings found all of us in high spirits, as both internal and class milestone deadlines were met.

# 3.0 Design

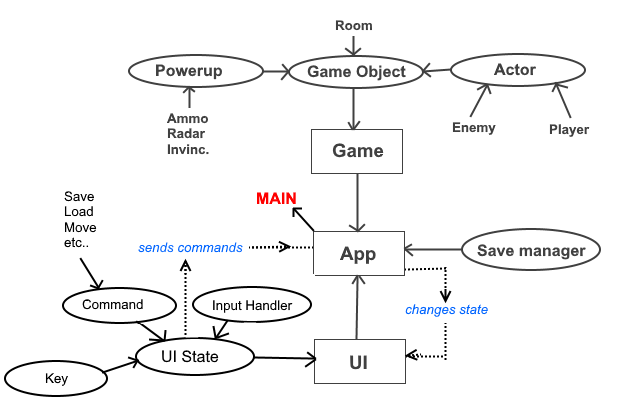
During the first group meeting, we created a rough outline of how we thought the project should be structured. In essence, there were to be three primary components to the program:

* The game component: A completely standalone, platform independent implementation of game logic.
* The UI component: Simply displays data and retrieves user inputs.
* The application component: Processes inputs and uses game object methods, updates UI display.

In addition, there was a multitude of other objects involved in the initial design, but in general they fall within the domain of one of the above components. Here is a diagram representing our initial design as it appeared when we sketched it out during our first group meeting:



This design proved useful through the first milestone, however, changes had to be made particularly to the UI class. It was bulky, handling too many things on its own and breaking the design pattern. It was also never going to easily swap in and out with a GUI when we started adding that in for extra credit. In addition, we decided we wanted a change to the way inputs were handled so that UI didn't have to be so heavily bogged down with logic determining what inputs to take. We added different states to the UI so that altering the valid inputs based on game logic would simply be a matter of changing the state of the UI. A system of commands was also implemented so that different actions could be bound to any key press depending on the state of the game. Eventually, our revised design looked like this:



This design is basically what we ended up finishing the project with.

*(Note: The arrows in these diagrams don't strictly mean anything like "is a child of" or "runs inside", but rather are indicators of some general relationship)*

# 4.0 Discussion of Implementation

Our design held up fine for the standard text based version of the project, but things began to break down a bit on the UI side when we began to implement the GUI. With the drop dead date fast approaching, we decided to go the quick and dirty route and just slap on a rather unthoughtfully designed, but functional GUI. In addition, a decision was made to attempt to have the GUI run in real time. This lead to the GUI class being rather bloated with repetitive logic which could have been avoided through better planning. That being said, our early design did allow us an easy way to swap the UI between text and GUI and allowed us to run the game in two different styles (turn based and real time) without having to do a massive code overhaul. Nevertheless, the game had to be patched all over with quick and dirty fixes to remain consistently playable between the two modes.

# 5.0 Testing Data

## 5.1 General Interaction Testing

### 5.1.1 Player Movement Testing

#### 5.1.1.1 Test Case 1001

**Test**  **Case:** This will test if the player can move out of bounds.

**Test Procedure:** Attempt to move down, when the player is already all the way at the bottom; attempt with top, left and right wall too.

**Expected Result:** Player shouldn’t be able to move out of bounds.

**Actual Result:** Player was unable to move out of bounds.

#### 5.1.1.2 Test Case 1002

**Test**  **Case:** This will test if the enemy can move out of bounds.

**Test Procedure:** Move around repeatedly in debug mode, while monitoring enemies and making sure there are six enemies at all times.

**Expected Result:** There will be six enemies on the grid at all times.

**Actual Result:** There were six enemies on the grid at all times.

#### 5.1.1.3 Test Case 1003

**Test Case:** This will test if the player can walk into the room with or without intelligence.

**Test Procedure:** Attempt to walk into any room.

**Expected Result:** Player shouldn’t be able to walk into a room.

**Actual Result:** Player was unable to walk into a room.

#### 5.1.1.4 Test Case 1004

**Test**  **Case:** This will test if the player can walk over a power up.

**Test Procedure:** Move the player into a spot that has a power up.

**Expected Result:** Player will move into the spot with the power up.

**Actual Result:** Player moved into the spot with the power up.

#### 5.1.1.5 Test Case 1005

**Test**  **Case:** This will test if the player can walk into a power up, and pick it up.

**Test Procedure:** Move the player into a spot that has a power up.

**Expected Result:** Player will move into the spot with the power up, and message will be displayed saying the player received a power up.

**Actual Result:** Player moved into the spot with the power up, and message was displayed saying the player received a power up.

#### 5.1.1.6 Test Case 1006

**Test**  **Case:** This will test if a player can walk into an enemy and get killed.

**Test Procedure:** Move player around the grid until a ninja is adjacent to the player.

**Expected Result:** Ninja will kill the player.

**Actual Result:** Ninja killed the player.

#### 5.1.1.7 Test Case 1007

**Test**  **Case:** This will test if the player can move multiple times.

**Test Procedure:** Attempt to move in different directions for 20 turns.

**Expected Result:** Player should be able to move around the grid without nothing happening.

**Actual Result:** Game crashed when ninjas were trying to move to the same position.

#### 5.1.1.8 Test Case 1008

**Test Case:** This tests if the enemies move around when debug mode is toggled

**Test Procedure:** Repeatedly toggle debug mode, and check enemy positions

**Expected Result:** The enemy positions should remain the same

**Actual Result:** The enemies moved around

**Comment:** The enemies shouldn’t move around as the debug mode is toggled

#### 5.1.1.9 Test Case 1009

**Test Case:** This tests if the enemies move around when debug mode is toggled(v2)

**Test Procedure:** Repeatedly toggle debug mode, and check enemy positions

**Expected Result:** The enemy positions should remain the same

**Actual Result:** The enemies didn’t move around, and remained in the same positions

### 5.1.2 Player Visibility Testing

#### 5.1.2.1 Test Case 1201

**Test Case:** This tests if all spaces around the player in a one space radius are visible.

**Test Procedure:** Walk around the grid, entities within one space radius should be visible

**Expected Result:** Entities within one space radius should be visible

**Actual Result:** Entities within one space radius were visible

#### 5.1.2.2 Test Case 1202

**Test Case:** This tests if a player can look beyond one space, without the look feature.

**Test Procedure:** Look at the grid symbols.

**Expected Result:** Player will not be able to see beyond one space.

**Actual Result:** Player did not see beyond one space.

#### 5.1.2.3 Test Case 1203

**Test Case:** This tests if the look feature allows you to see two spaces ahead in one direction

**Test Procedure:** Use the look feature in one direction

**Expected Result:** Player should be able to see two spaces ahead in one direction

**Actual Result:** Player was able to see two spaces ahead in one direction

#### 5.1.2.4 Test Case 1204

**Test Case:** This tests if the look feature can be used to look beyond two spaces

**Test Procedure:** Use the look feature to look more than two spaces ahead

**Expected Result:** Player shouldn’t be able to look more than two spaces ahead

**Actual Result:** Player wasn’t able to look more than two spaces ahead

#### 5.1.2.5 Test Case 1205

**Test Case:** This test case will check to see if the player can check the room for intel without being in the room**.**

**Test Procedure:** Attempt tocheck the room for intelligence without being in the room.

**Expected Result:** Player should be unable to see the rooms contents without being on top of the room.

**Actual Result:** Player was unable to view the rooms contents without being on top of the room.

#### 5.1.2.6 Test Case 1206

**Test Case:** This tests if a player can look for intel from a position he should not be able to, with the look feature.

**Test Procedure:** Go on the left, right, and under the room and attempt to look inside the room

**Expected Result:** Player shouldn’t be able to look inside the room

**Actual Result:** Player wasn’t able to look inside the room

#### 5.1.2.7 Test Case 1207

**Test Case:** This tests if a player can check a room for intel, when positioned on top of the room.

**Test Procedure:** Move player to the top of a room, attempt to look into room.

**Expected Result:** Message will appear stating whether the intel was there or not.

**Actual Result:** Message appeared stating whether the intel was there or not.

#### 5.1.2.8 Test Case 1208

**Test Case:** This tests if a player is able to view power ups that are one space away, without the look feature.

**Test Procedure:** Player stands one space away from a power up

**Expected Result:** Player should be able to see the power up from one space away

**Actual Result:** Player was able to see the power up from one space away

#### 5.1.2.9 Test Case 1209

**Test Case:** This tests if a player is able to view power ups that are two spaces away, with the look feature.

**Test Procedure:** Player stands two spaces away from a power up, and uses the look feature

**Expected Result:** Player should be able to see the power up from two spaces away, while using the look feature

**Actual Result:** Player was able to see the power up from two spaces away, while using the look feature

#### 5.1.2.10 Test Case 1210

**Test Case:** This tests if a player is able to view enemies that are one space away, without the look feature.

**Test Procedure:** Player stands one space away from an enemy

**Expected Result:** Player should be able to see the enemy from one space away

**Actual Result:** Player was able to see the enemy from one space away

#### 5.1.2.11 Test Case 1211

**Test Case:** This tests if a player is able to view enemies that are two spaces away, with the look feature.

**Test Procedure:** Player stands two spaces away from an enemy, and uses the look feature

**Expected Result:** Player should be able to see the enemy from two spaces away, using the look feature

**Actual Result:** Player was able to see the enemy from two spaces away, using the look feature

### 5.1.3 Player Life Testing

#### 5.1.3.1 Test Case 1301

**Test Case:** Tests if player properly loses lives.

**Test Procedure:** Try to get killed by an enemy.

**Expected Result:** Lives should go down.

**Actual Result:** Lives went down.

#### 5.1.3.2 Test Case 1302

**Test**  **Case:** This will test if the player dies once he has 0 lives.

**Test Procedure:** Attempt to get killed by ninjas 3 times.

**Expected Result:** Player should die after the third time.

**Actual Result:** Player continued to live with negative lives.

#### 5.1.3.3 Test Case 1303

**Test Case:** Tests if the game properly ends once a player has zero lives.

**Test Procedure:** Try to get killed by ninjas.

**Expected Result:** Game will end once player has zero lives.

**Actual Result:** Game ended when player had zero lives.

## 5.2 Game Termination Testing

### 5.2.1 Game Tests After Expected Board Reset

#### 5.2.1.1 Game Tests After New Game Is Started

##### 5.2.1.1.1 Test Case 2001

**Test Case:** This tests if there are six enemies, all randomized around the board, after a new game is started

**Test Procedure:** Start a new game, enter debug mode, check enemies

**Expected Result:** There should be six enemies, scattered around the board

**Actual Result:** There were six enemies, scattered around the board

##### 5.2.1.1.2 Test Case 2002

**Test Case:** This tests if there are three power ups, all randomized around the board, after a new game is started

**Test Procedure:** Start a new game, enter debug mode, check power ups

**Expected Result:** There should be three power ups, scattered around the board

**Actual Result:** There were three power ups, scattered around the board

##### 5.2.1.1.3 Test Case 2003

**Test Case:** This tests if only one room contains the intelligence, after a new game is started

**Test Procedure:** Start a new game, enter debug mode, check for intelligence

**Expected Result:** There should be only one room with intelligence

**Actual Result:** There was only one room with intelligence

##### 5.2.1.1.4 Test Case 2004

**Test Case:** This tests if the visibility for all hidden entities are actually hidden, after a new game is started

**Test Procedure:** Start a new game, check the visibility of all entities

**Expected Result:** Everything should be hidden, unless an entity is in a one spacy proximity of the player

**Actual Result:** Everything was hidden

#### 5.2.1.2 Game Tests After Game Is Loaded

##### 5.2.1.2.1 Test Case 2005

**Test Case:** This tests if there are six enemies, all randomized around the board, after a game is loaded

**Test Procedure:** Load a game, enter debug mode, check enemies

**Expected Result:** There should be six enemies, scattered around the board

**Actual Result:** There were six enemies, scattered around the board

##### 5.2.1.2.2 Test Case 2006

**Test Case:** This tests if there are three power ups, all randomized around the board, after a game is loaded

**Test Procedure:** Load a game, enter debug mode, check power ups

**Expected Result:** There should be three power ups, scattered around the board

**Actual Result:** There were three power ups, scattered around the board

##### 5.2.1.2.3 Test Case 2007

**Test Case:** This tests if only one room contains the intelligence, after a game is loaded

**Test Procedure:** Load a game, enter debug mode, check for intelligence

**Expected Result:** There should be only one room with intelligence

**Actual Result:** There was only one room with intelligence

##### 5.2.1.2.4 Test Case 2008

**Test Case:** This tests if the visibility for all hidden entities are actually hidden, after a game is loaded

**Test Procedure:** Load a game, check the visibility of all entities

**Expected Result:** Everything should be hidden, unless an entity is in a one spacy proximity of the player

**Actual Result:** Everything was hidden

#### 5.2.1.3 Game Tests After a New Game from a Player Death

##### 5.2.1.3.1 Test Case 2009

**Test Case:** This tests if there are six enemies, all randomized around the board, a new game is started from a player death

**Test Procedure:** Start a new game, from a player death, enter debug mode, check enemies

**Expected Result:** There should be six enemies, scattered around the board

**Actual Result:** There were six enemies, scattered around the board

##### 5.2.1.3.2 Test Case 2010

**Test Case:** This tests if there are three power ups, all randomized around the board, a new game is started from a player death

**Test Procedure:** Start a new game, from a player death, enter debug mode, check power ups

**Expected Result:** There should be three power ups, scattered around the board

**Actual Result:** There were three power ups, scattered around the board

##### 5.2.1.3.3 Test Case 2011

**Test Case:** This tests if only one room contains the intelligence, a new game is started from a player death

**Test Procedure:** Start a new game, from a player death, enter debug mode, check for intelligence

**Expected Result:** There should be only one room with intelligence

**Actual Result:** There was only one room with intelligence

##### 5.2.1.3.4 Test Case 2012

**Test Case:** This tests if the visibility for all hidden entities are actually hidden, a new game is started from a player death

**Test Procedure:** Start a new game, from a player death, check the visibility of all entities

**Expected Result:** Everything should be hidden, unless an entity is in a one spacy proximity of the player

**Actual Result:** Everything was hidden

### 5.2.2 Loading & Saving Testing

#### 5.2.2.1 Test Case 2201

**Test Case:** This tests if the game can be saved properly.

**Test Procedure:** Start a new game, save it, and load it in a new game.

**Expected Result:** When the old game is loaded, all the game entities should be in the same position.

**Actual Result:** Typing “3” to save the game didn’t even save it.

**Comment:** The game saving command “3” needs to be implemented/fixed.

#### 5.2.2.2 Test Case 2202

**Test Case:** This tests if a game can be loaded after a new game is started.

**Test Procedure:**  Start a new game, load a previous game.

**Expected Result:** The previous game should successfully load.

**Actual Result:**  The previous game successfully loaded**.**

#### 5.2.2.3 Test Case 2203

**Test Case:** This tests if a game can be loaded properly after a game is won.

**Test Procedure:** Start a new game, win the game, attempt to load a previous game.

**Expected Result:** The previous game should successfully load.

**Actual Result:** The previous game didn’t load.

**Comment:** Game loading has to be fixed, after a game is won.

#### 5.2.2.4 Test Case 2204

**Test Case:** This tests if a game can be loaded properly after a game is lost.

**Test Procedure:** Start a new game, lose the game, attempt to load a previous game.

**Expected Result:** The previous game should successfully load.

**Actual Result:** The previous game loaded.

### 5.2.3 Test Case 2205

**Test Case:** This tests if player can quit the game.

**Test Procedure:** Start the game, quit the game.

**Expected Result:** Game will end.

**Actual Result:** Game ended.

## 5.3 GUI Testing

### 5.3.1 Test Case 3001

**Test**  **Case:** This will test if the room icons in the GUI change properly.

**Test Procedure:** Look into a room, die three times, start a new game, see if the room’s icon changed from open to closed.

**Expected Result:** Room’s icon should change to closed after starting a new game.

**Actual Result:** Room’s icon did not change.

### 5.3.2 Test Case 3002

**Test Case:** This will test if the player is able to shoot ninjas after running out of bullets in the GUI.

**Test Procedure:** Shoot one time, have no bullets left, attempt to shoot.

**Expected Result:** Message saying you can’t shoot will be displayed.

**Actual Result:** Message saying you shot a ninja was displayed.

# 6.0 Conclusions

This project allowed us to realize things about ourselves and our experience. For some of us, this had been the first real coding project that we had done. We had only previously done small programs. Because of this, the project seemed impossible to do at the beginning. We had members that had a lot of experience coding so they helped the others that did not have much. We realized that with enough effort everything is possible regardless of how hard you think it is. Even though we started with six people in our group and ended with four, we still managed to do the GUI for extra credit. We found the project to be fun and interesting because it was challenging, and fun to play at the end.

# 7.0 Suggestions for Improvements

The concept of having two vastly different types of UIs, to us, was quite a difficult one to understand. No one in our group had any experience with something like that before. Due to the unusual nature of this assignment, our final project submission contains many more bad programming practices than we would have liked to avoid. That being said, we are proud of our accomplishments here, but if we had a chance to do things differently, better planning would be a top priority.

Additionally, no member of our group truly considers themselves an expert in Java nor did we properly gauge the size and scope of this project. On the surface, it seemed like a small, one day project, but in reality, the amount of time spent testing and fine tuning was enormous. Also, in an attempt to step outside of our zone of comfort, we endeavored to construct this project using unfamiliar design patterns and coding methods. This only added to the difficulty of bug fixes and slowed the whole programming process down. Although we all agree that it was beneficial from an educational standpoint to go about things this way, we also understand that the drawbacks are quite evident.