**Project:** RPG Battle System

Ali Akid, Adrian Croitoru, Justin Sorensen, Jason Truskowski, Jacob Kucinski

**CIS 350: W21**

Table of Contents

[**Project Information**](#_heading=h.xkj3kh311mt0) **3**

[Features](#_heading=h.30j0zll) 3

[Screenshots](#_heading=h.1fob9te) 4

[**Project Plan**](#_heading=h.3znysh7) **10**

[Requirements & Definition](#_heading=h.2et92p0) 10

[Development](#_heading=h.tyjcwt) 10

[Verification](#_heading=h.3dy6vkm) 10

[Maintenance](#_heading=h.1t3h5sf) 10

[Umbrella Activities](#_heading=h.4d34og8) 11

[**Requirements & Specification**](#_heading=h.17dp8vu) **12**

[Natural Language Requirements](#_heading=h.26in1rg) 12

[Use-Cases](#_heading=h.ltv40fdurjym) 17

[Traceability Matrix](#_heading=h.wmcxu5i594fi) 18

[**Gantt Chart**](#_heading=h.98g0objnsleg) **19**

[**Design**](#_heading=h.dq5138doc0ne) **22**

[**Development**](#_heading=h.1ksv4uv) **28**

Additional Libraries 28

[Code Standards](#_heading=h.44sinio) 28

[Static Analysis](#_heading=h.2jxsxqh) 31

[Code Documentation](#_heading=h.z337ya) 33

[Configuration Management](#_heading=h.3j2qqm3) 33

[**Verification**](#_heading=h.1y810tw) **34**

[Integration Tests](#_heading=h.4i7ojhp) 34

[Unit Tests](#_heading=h.2xcytpi) 34

[Code Coverage](#_heading=h.1ci93xb) 35

[Requirements Coverage](#_heading=h.3whwml4) 36

[**Postmortem**](#_heading=h.2bn6wsx) **37**

[Responsibilities](#_heading=h.2s8eyo1) 37

[Earned Value](#_heading=h.qsh70q) 37

[Variances](#_heading=h.3as4poj) 38

[Lessons Learned](#_heading=h.1pxezwc) 39

# Project Information

Real-time battle system where the player and enemies attack one another in an open overworld.

## Features

Features included in this release are highlighted.

**Player character:**

* Has ~3 different actions, with the first being a basic attack, and the other two being more powerful but having a mana/cooldown system (NOTE: Cooldown system was implemented in basic attack due to battles being changed to real-time)
* Selection from different classes to change actions when picking your player
  + Healer/Warrior/Archer/Mage etc.

**Enemies:**

* Enemy targets and attacks the player
* Multiple different enemies ranged/melee/magic
* Have a chance to drop an item upon defeat to add to the inventory
* Boss enemies appear at certain levels, having stronger attacks and more health

**Level-Up system:**

* Increases player stats such as health/mana/attack
* New abilities and additional characters can be unlocked at specific levels

**Gameplay Flow:**

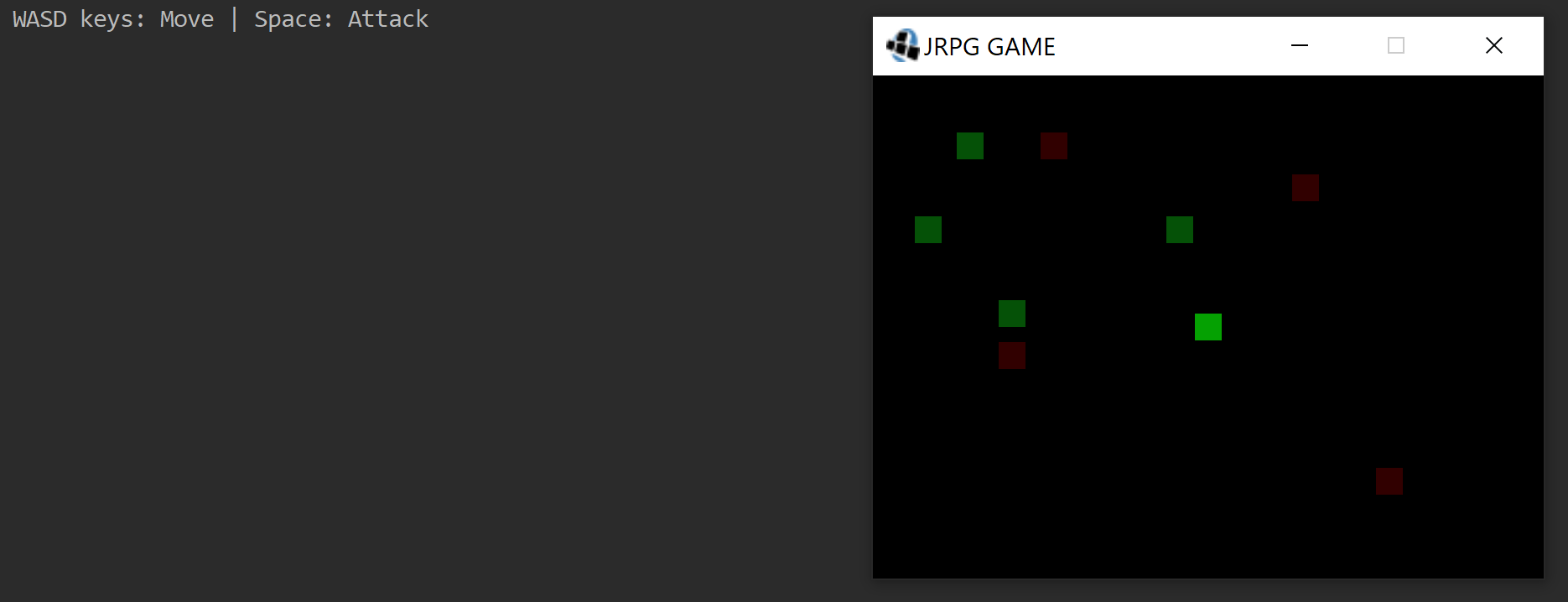
* The player moves the character around a basic overworld with barriers that are tied to the player’s level
* Enemies appear on the overworld, with stronger enemies more likely to appear at higher levels
  + Each enemy has specific actions they will randomly decide to perform, and items that they can randomly drop
* A boss appears once the player reaches a certain level
  + Party members may also be added at certain levels, and their class can be chosen
* Defeating the boss breaks the lowest-level barrier still up

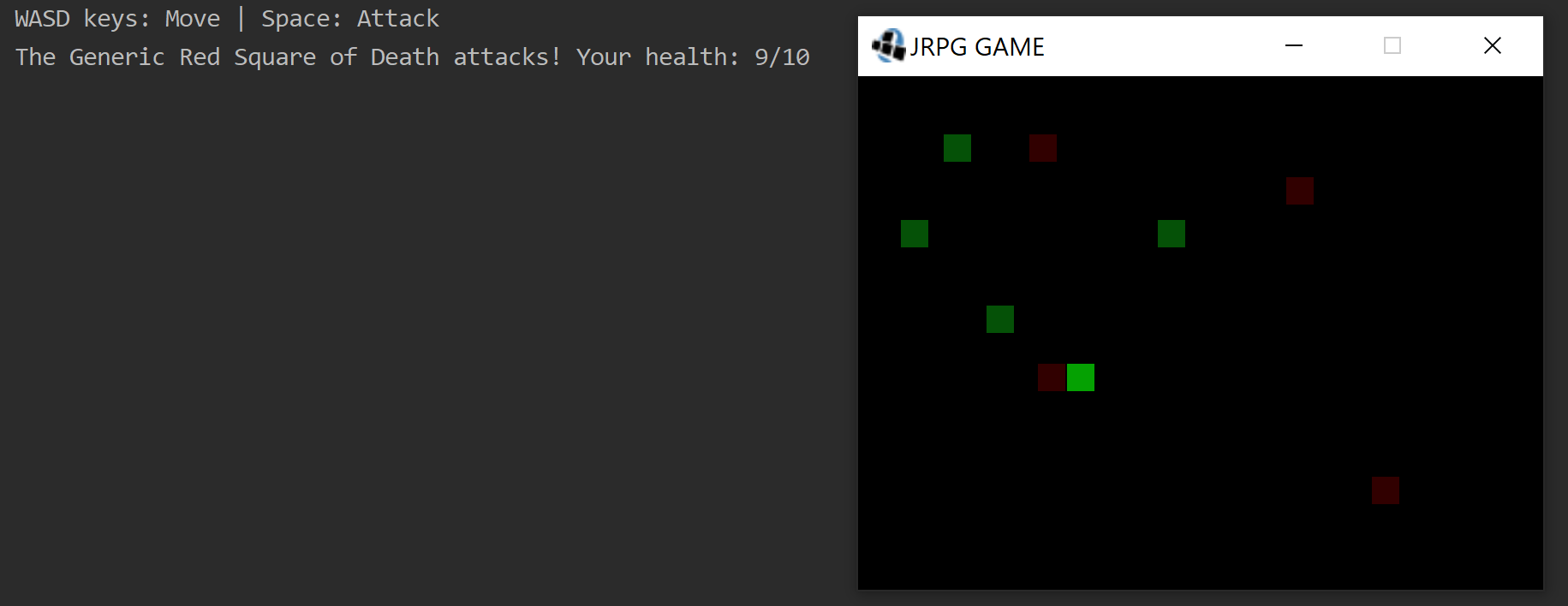
## 

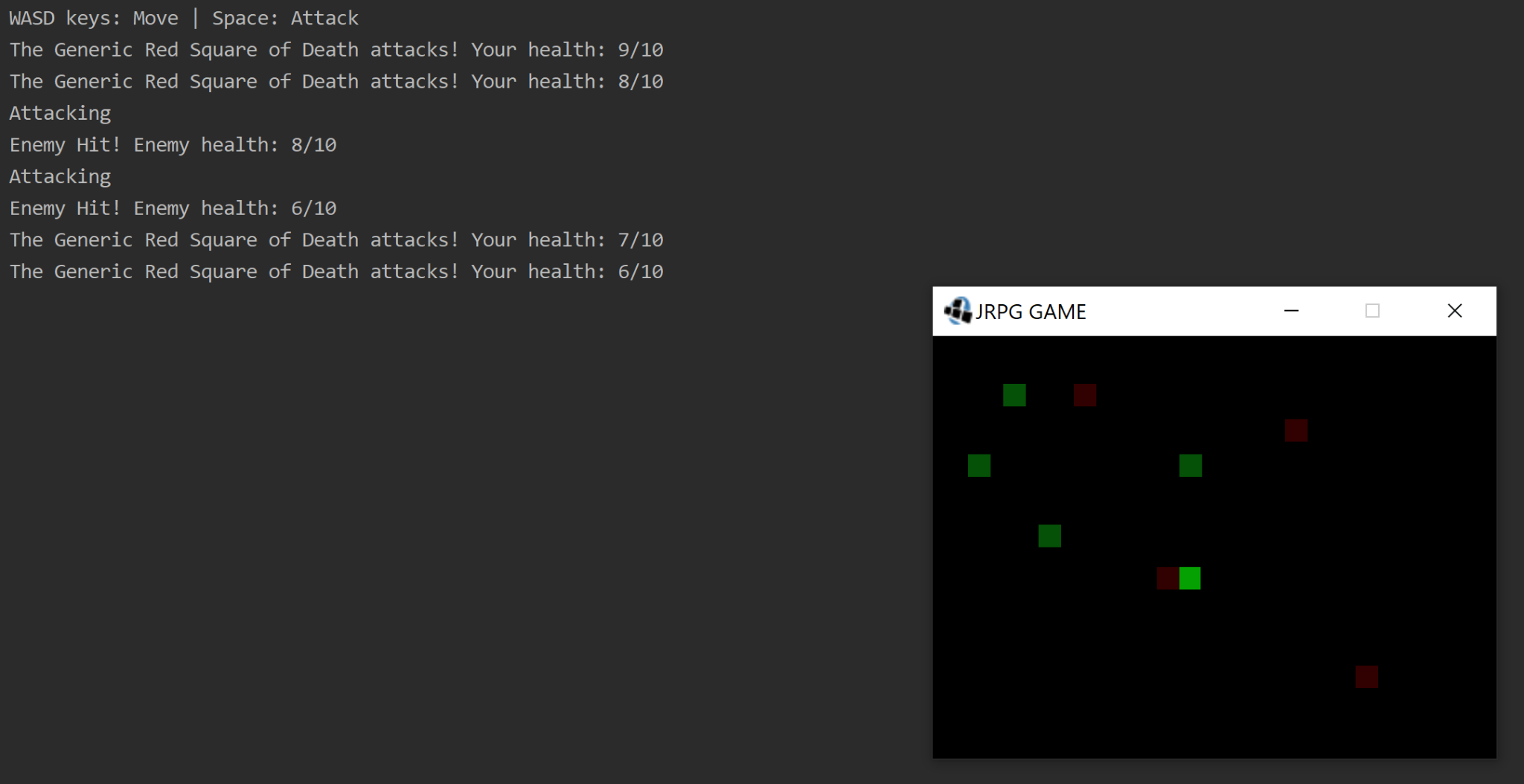
## 

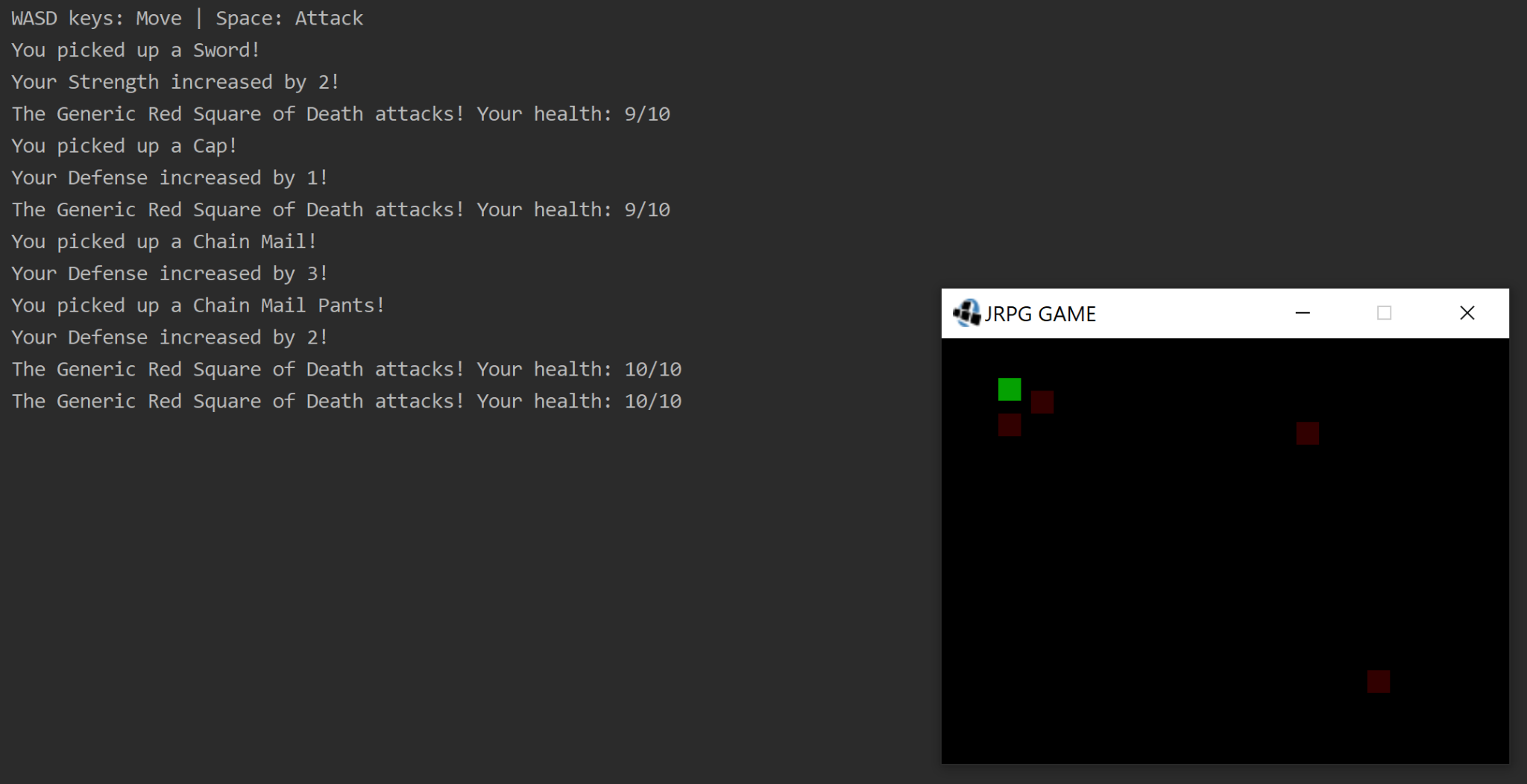
## Screenshots

Initial Game State:

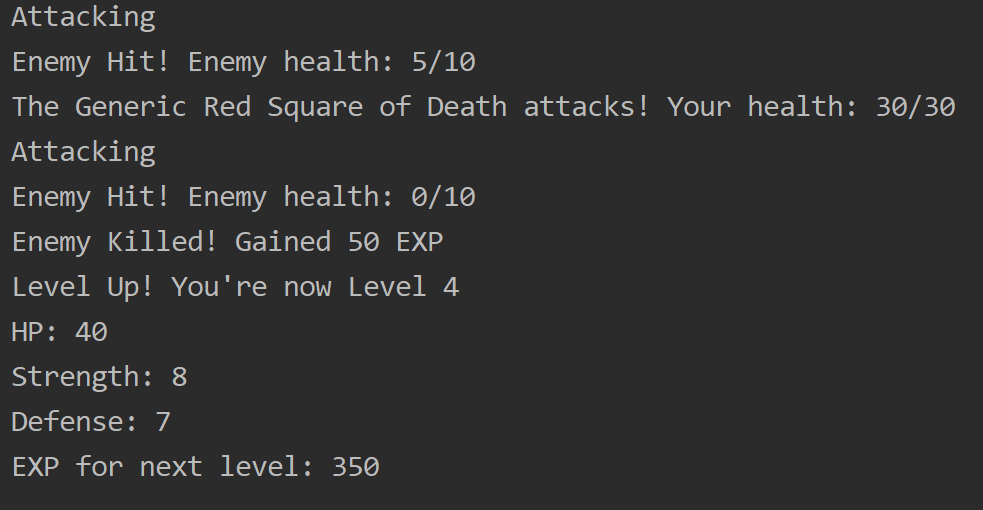
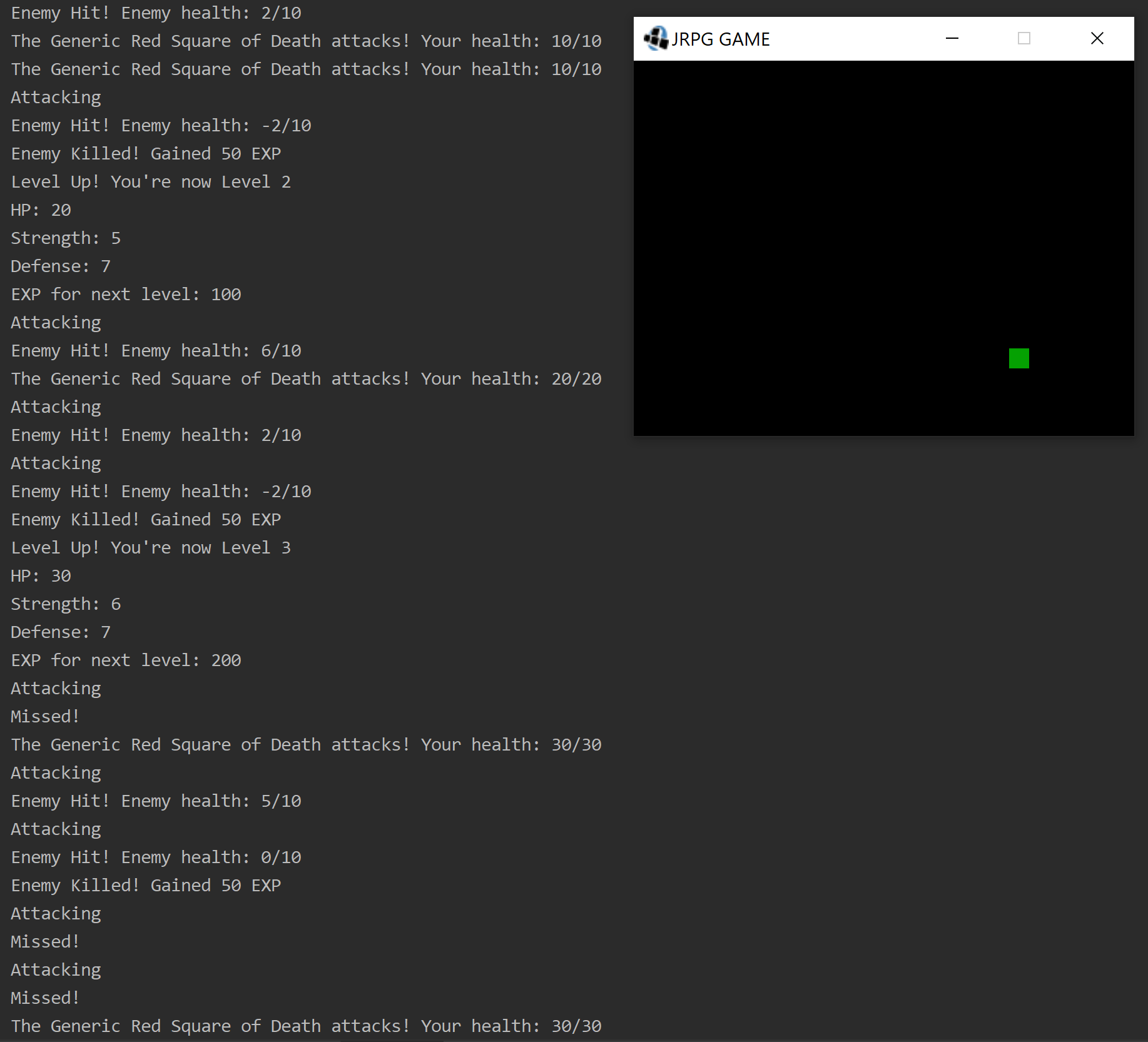


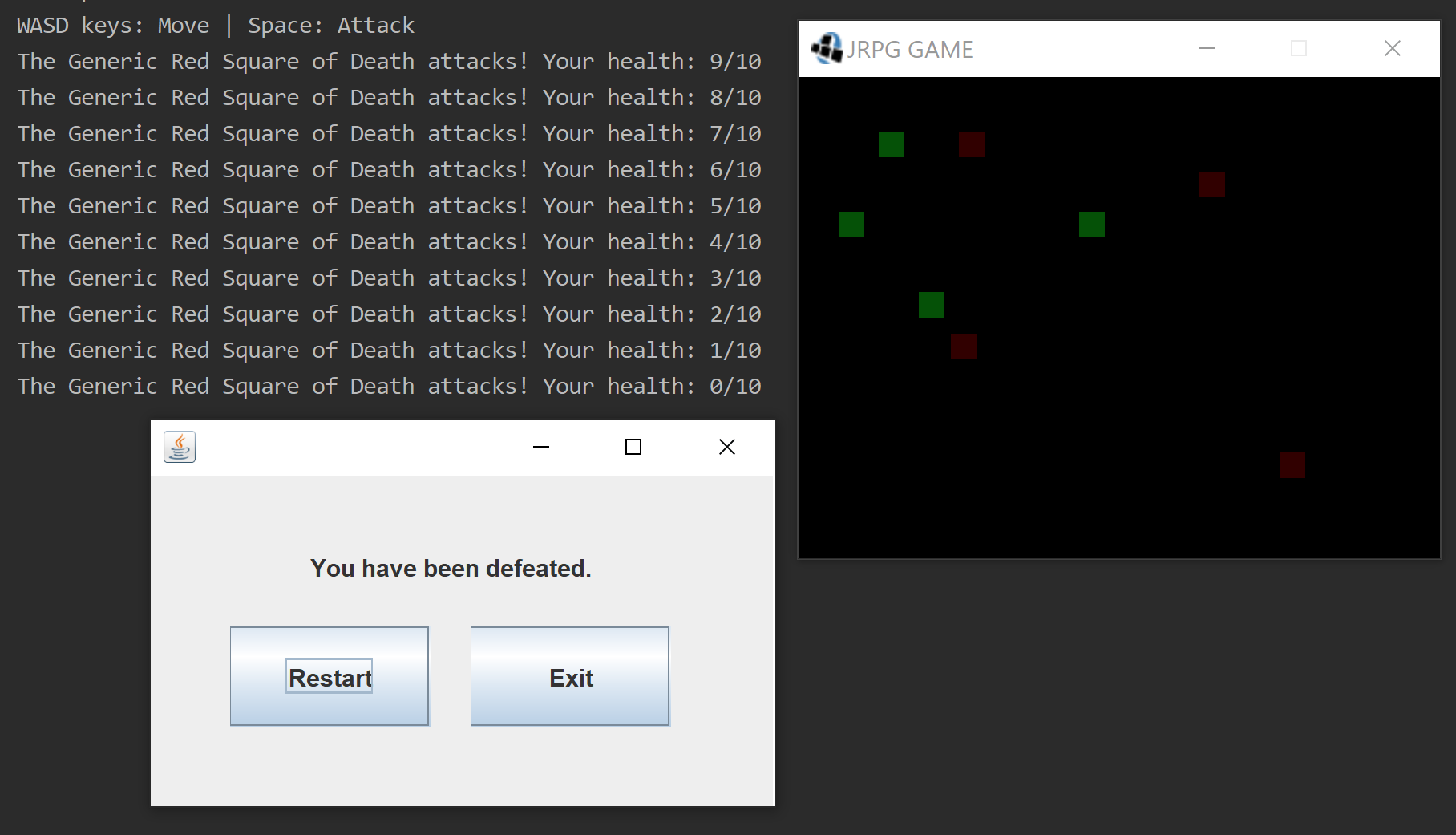
Enemy Attacking:

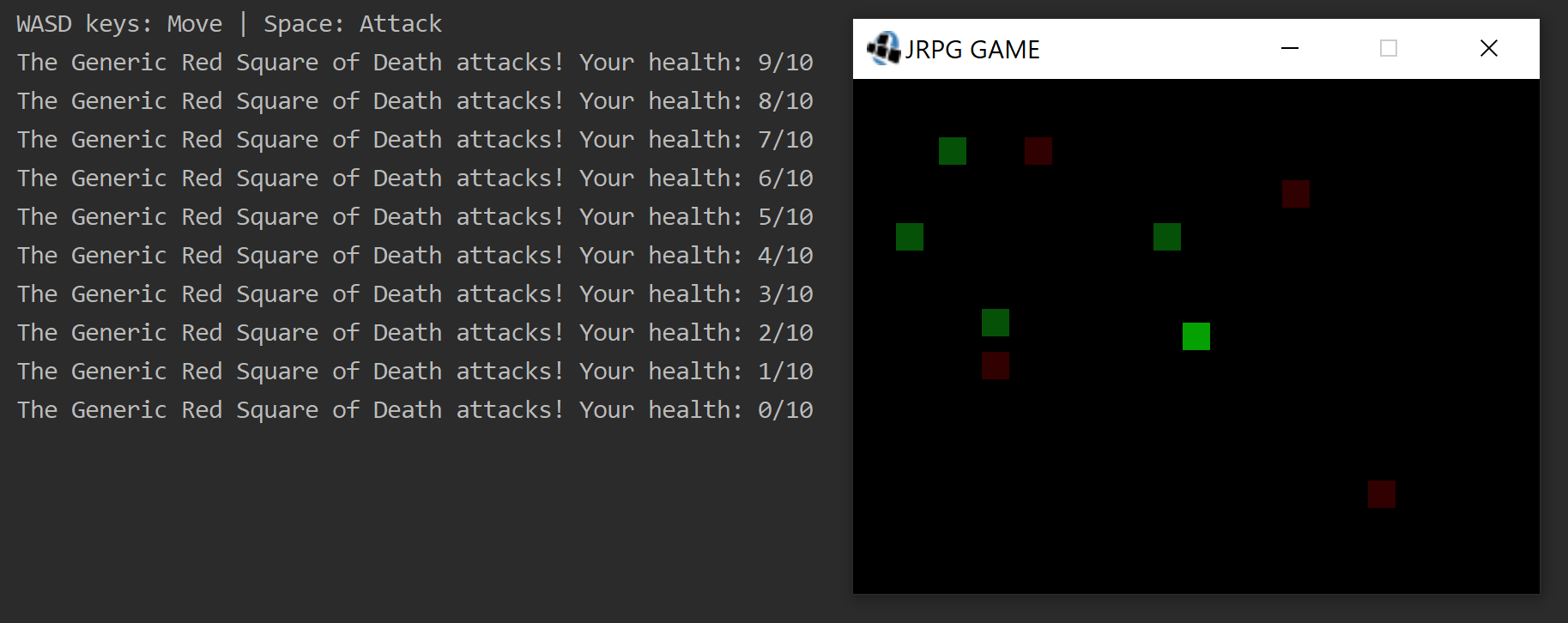
Player and Enemy Both Attacking:

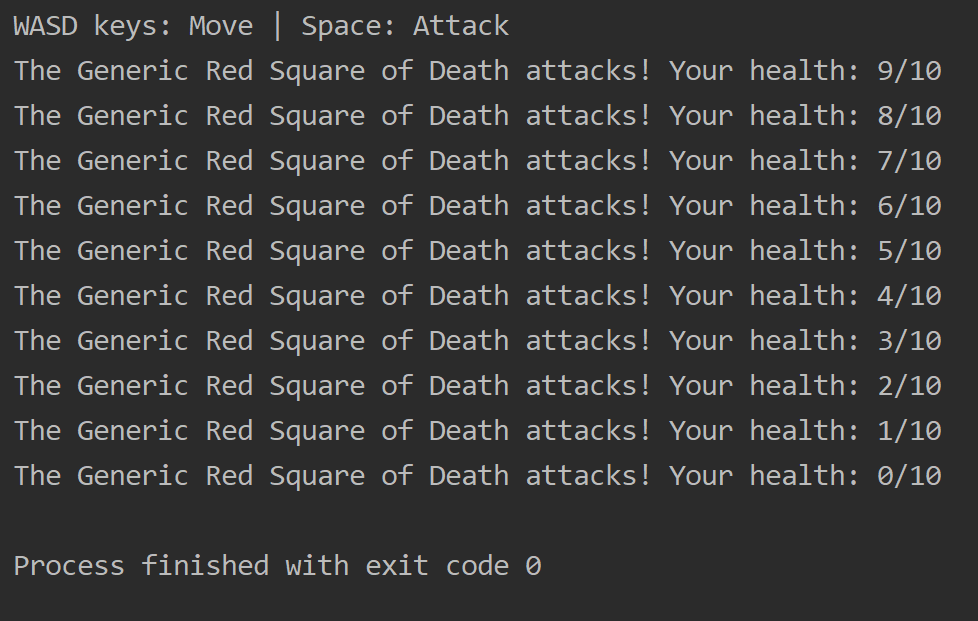
The Player picks up all the equipment to become invincible and overpowered because we didn’t make the armor fair:

Leveling up after defeating the first enemy:

A full playthrough where the player manages to kill all the enemies and level up three times:

The death screen when the player is killed:

The game resetting itself when the “Restart” option is selected on the death screen: 

The game closing properly when the player rage quits and selects “Exit” on the death screen:

# 

# Project Plan

Waterfall with prototype software life cycle.

## Requirements & Definition

We have a list of use cases and natural language requirements that provide an outline of the project’s intended features and functionality. We will use these resources as a guide to build the program in a logical order according to our Gantt chart.

## Development

We will develop our program using Java coding language and use IntelliJ for our IDE.

## Verification

We plan to verify our software at a future point in time when we turn in release 2 and we have our verification steps in place 3/26/21.

UPDATE FOR RELEASE 3: See “Verification” section for information about testing and verification.

## Maintenance

**Documenting Issues**

Any issues found with the program during development should be recorded in the “Maintenance Issues” document in the group Google Drive folder, with a brief description of the issue and where it occurs. When an issue is fixed, it is left on the document with an indication of when and how it was fixed for future reference.

**Bugs**

If bugs or unintended outcomes are found in the program, the group member who originally implemented the corresponding feature should be notified, and the group should work together to fix the problem and ensure that any other connected parts of the project still work properly after doing so.

**Unsatisfactory Task Completion**

If a group member fails to complete a task assigned to them in a timely manner in such a way that it hinders the rest of the group’s ability to make progress, the group should meet with that group member to reiterate their responsibilities and place stricter deadlines on their tasks. If issues persist, other group members may have to take over the responsibilities and keep a record of missed assignments.

## Umbrella Activities

We have a group Discord chat where we will update each other on what we have accomplished on each of our responsibilities. As well as a weekly live meeting to assess what we need to get done or do better next week.

# Requirements & Specification

Refer to the Release 3 version of the Gantt chart for more details on what features were completed and cut.

## Use-Cases

**CASE 1**

**Use case:** Open Program

**Actor(s)**: Player

**Description**: Player starts the game and views the menu to select what choice they want to

make. Players are given the choice to start a new game or exit( later on will also

be able to load games).

**CASE 2**

**Use case**: New Game

**Actor(s):** Player

**Description**: The game loads all the resources and displays a menu to allow the player to

begin a new game.

**Pre-Condition**: Player must have completed **Open Program** use-case.

**CASE 3**

**Use case**: Exit game

**Actor(s)**: Player

**Description**: Player selects exit and the System shuts down the game.

**Pre-Condition**: Player must have completed **Open Program** use-case.

**~~CASE 4~~**

**~~Use Case~~**~~: Save Game~~

**~~Actor(s)~~** ~~: Player~~

**~~Description~~** ~~: The user saves the current game state, so it may be loaded later.~~

**~~Pre-condition~~** ~~: The player must have completed the~~ **~~New Game~~** ~~use-case.~~

**~~CASE 5~~**

**~~Use Case~~**~~: Load Game~~

**~~Actor(s)~~** ~~: Player~~

**~~Description~~** ~~: The user loads a previous game state.~~

**~~Pre-condition~~** ~~: The player must have saved~~~~a previous game using the~~ **~~Save Game~~** ~~use-case.~~

**Cases 4 and 5** for saving and loading games have been dropped due to time constraints

**CASE 6**

**Use case**: Move Character

**Actor(s):** Player

**Description**: The player attempts to move the character on the map. The character can only move in open terrain, but cannot move over unmovable terrain (e.g. mountains, forests, walls, game boundary etc.).

**Pre-Condition:** Players must complete the **New Game** use case.

**CASE 7**

**Use case**: Fight Enemy

**Actor(s):** Player

**Description**: The system will generate an encounter for the player. These encounters consist of

enemies, whose power is based on character level, for the player to fight an try

bringing their health points down to 0. When fighting enemies, the player can

select how to attack.

**~~CASE 8~~**

**~~Use Case~~** ~~: Select Attack~~

**~~Actions~~** ~~: Player~~

**~~Description~~** ~~: The user selects the type of attack to execute against the encountered foe that~~

~~will reduce the enemy's HP encounter’s energy, until it reaches 0.~~

**~~Pre-condition~~** ~~:The system must~~ **~~Fight an Enemy~~** ~~for an attack to be selected.~~

**~~CASE 9~~**

**~~Use Case~~** ~~: Select Item~~

**~~Actions~~** ~~: Player~~

**~~Description~~** ~~: The user selects an item to use. Items can be received from encounters or ground.~~

**~~Pre-condition~~** ~~: The user must have an item in their inventory.~~

**Case 9** for selecting items has been removed as items are automatically equipped.

**~~CASE 10~~**

**~~Use Case~~** ~~: Select target~~

**~~Actions~~** ~~: Player~~

**~~Description~~** ~~: User selects a character to receive the attack or item.~~

**~~Pre-condition~~** ~~: The user must first perform either the~~ **~~Select Attack~~** ~~or~~ **~~Select Item~~** ~~use case~~.

**Case 10** for selecting targets removed as target selection is done automatically.

**~~CASE 11~~**

**~~Use Case~~** ~~: Accept winnings~~

**~~Actors~~** ~~: Player~~

**~~Description~~** ~~: Upon the completion of the encounter, the user is granted experience and the~~

~~items that were held by the encounter.~~

**~~Pre-condition~~** ~~: The encountered foes energy must be reduced to 0, ending the encounter.~~

**Case 10** for selecting targets removed as winning are automatically applied to the player.

**CASE 12**

**Use case**: Gain Experience Points (XP)

**Actor(s):** Player

**Description**: XP is gained through fighting enemies. The number of XP is determined by the

level of the enemy monster and the amount of enemy monsters. XP is only useful

when filled up, causing a Level up.

**Pre-Condition**: The player must first **Fight Enemy** use case.

**~~CASE 13~~**

**~~Use case~~**~~: Open Menu~~

**~~Actor(s):~~** ~~Player~~

**~~Description~~**~~: The player can open an in-game menu containing multiple choices. The player can check character statistics, can save or load a game, or quit the game.~~

**~~Pre-condition:~~** ~~Player must first complete~~ **~~New Game~~** ~~use case.~~

**Case 13** Open menu happens automatically now on player death.

**~~CASE 14~~**

**~~Use Case~~** ~~: Drop item~~

**~~Actors~~** ~~: Player~~

**~~Description~~** ~~: The user is able to remove less desirable items from their inventory to make~~

~~room for newer items.~~

**~~Pre-condition~~** ~~: The user must hold at least one item to drop items.~~

**Case 14** to drop items removed due to time constraints and change in how items are equipped.

**~~CASE 15~~**

**~~Use case~~**~~: Talk to neutral NPC (Non-player character)~~

**~~Actor(s):~~** ~~Player~~

**~~Description~~**~~: The player can talk to non-hostile NPCs. NPCs can give useful information and tips through an in-game text field and can also buy and sell equipment and spells~~

~~for the player’s character.~~

**~~Pre-condition:~~** ~~The player must first complete the~~ **~~New Game~~** ~~use case.~~

**Case 15** for talking to NPCS has been dropped due to time constraints

**CASE 16**

**Use case**: Gain a Level

**Actor(s):** Player

**Description**: Filling the XP levels up the character, increases character statistics making the

character stronger overall. Leveling up can unlock new spells/skills.

**Pre-condition**: The player must first gain XP through the **Gain XP** use case.

**CASE 17**

**Use case**: Pick Up Item

**Actor(s):** Player

**Description**: The player may encounter items on the ground when moving around. These items can range from equipment to potions or even secret hints.

**Pre-Condition:** The Player must use the **Move Character** use case.

**CASE 18**

**Use case**: Character Dies

**Actor(s):** Player

**Description**: If the player’s Health drops to 0 as a result of an enemy action, the player’s

character will die. Character’s death means the player has to either start a new

game or load an older saved game.

**Extends**: **Open Menu** use case.

## Natural Language Requirements

**RQ\_G1: The System** SHALL load all the PC resources necessary WHEN opening program.

**RQ\_G2: The System** SHALL request the Player name their character WHEN starting a new game.

**RQ\_G3: The System** SHALL change the direction and sprite of the character to UP position when the player moves UP WHEN the player tries to move up by pressing the up arrow.

**RQ\_G4: The System** SHALL change the direction and sprite of the character to DOWN position when the player moves DOWN WHEN the player tries to move down by pressing the down arrow.

**RQ\_G5: The System** SHALL change the direction and sprite of the character to LEFT position when player moves LEFT WHEN the player tries to move left by pressing the left arrow.

**RQ\_G6: The System** SHALL change the direction and sprite of the character to RIGHT position when the player moves RIGHT WHEN the player tries to move right by pressing the right arrow.

**RQ\_G7: The** **System** SHALL load a proper enemy or enemies for the player character to fight based on character level WHEN the character engages an enemy.

**RQ\_G8: The System** SHALL calculate the difference between character weapon damage and char level WHEN the player selects to attack with a weapon.

**~~RQ\_G9: The System~~** ~~WILL calculate the difference between character INT statistic and skill used WHEN the player activates an ability~~**~~.~~**

**RQ\_G10: The System** WILL calculate the difference between enemy attack and player ARMOR statistic before reducing player health.

**~~RQ\_G11: The System~~** ~~SHALL calculate and distribute xp points based on the number of enemies defeated during the fight WHEN the player exits a fight screen.~~

**RQ\_G12: The System** SHALL reduce the current XP number to 0 WHEN the character gains a level.

**RQ\_G13: The System** SHALL open a “Defeated” screen WHEN the player health drops to zero.

**RQ\_G14: The System** SHALL increase the character’s statistics WHEN the character gains a level.

**~~RQ\_G15: The System~~** ~~SHALL open the menu and display character stats and items WHEN the player opens the menu window.~~

**RQ\_G16: The System** SHALL display a message indicating what item was received WHEN an item is picked up.

**RQ\_G17: The System** SHALL place the item in the player inventory WHEN the player picks up an item

**~~RQ\_G18: The System~~** ~~SHALL open a dialog box WHEN the character speaks to an NPC.~~

**~~RQ\_G19: The System~~** ~~SHALL display a random hint for the player WHEN the character speaks to an NPC.~~

**RQ\_G18 and RQ\_G19** have been removed due to having the Talk to NPC removed

**RQ\_G20: The System** SHALL open a menu asking the player if they want to start a new game WHEN the character dies.

**RQ\_G21: The System** SHALL exit the program.

**RQ\_G22: The System** SHALL attack the character after the player uses an item, attacks or uses an ability

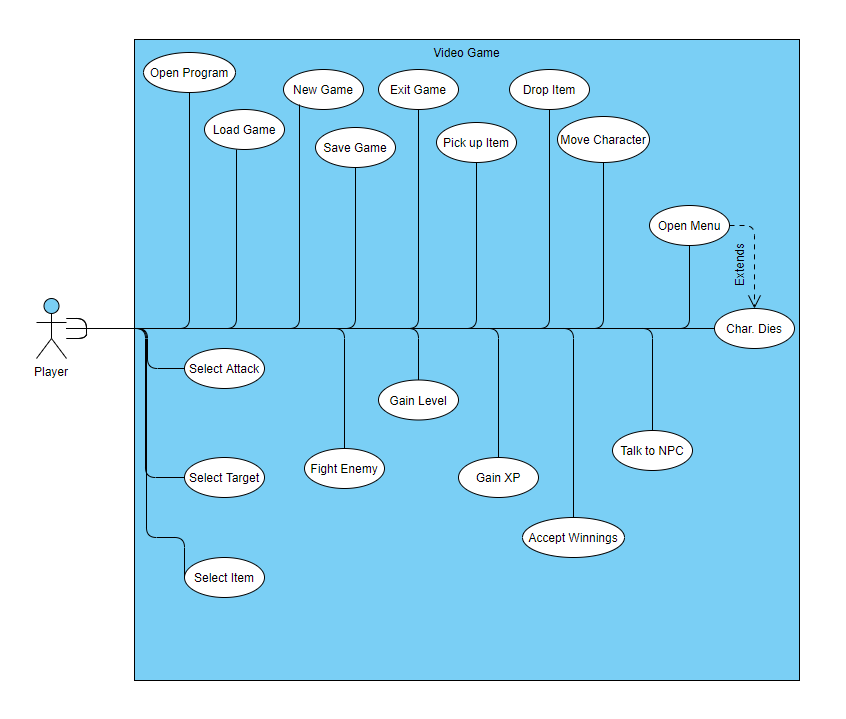
**RQ\_G23: The System** SHALL drop an item in the player’s inventory WHEN they choose ‘drop’.

**~~RQ\_G24: The System~~** ~~SHALL increase character hp WHEN using an item.~~

**~~RQ\_G25: The System~~** ~~SHALL save the current state of the game to a file WHEN the player selects the “Save” option from the menu.~~

**~~RQ\_G26: The System~~** ~~SHALL change the state of the game to the state specified from a file WHEN the player selects the “Load” option from the menu and the file is valid.~~

**RQ\_G24, RQ\_G25 and RQ\_G26** have been removed due to having the Save and Load features removed from the game.



## Traceability Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Open Progr. | New Game | Exit Game | ~~Save Game~~ | ~~Load Game~~ | Move Char. | Engage Enemy | Fight Enemy | ~~Select Attack~~ | ~~Select Item~~ | ~~Select Target~~ | Accept Winnings | Gain XP | Open Menu | ~~Drop Item~~ | ~~Talk to NPC~~ | Gain Lvl | Pick up Item | Char Dies |
| RQ\_G1 | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G2 |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G3 |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G4 |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G5 |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G6 |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G7 |  |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G8 |  |  |  |  |  |  |  | x | x |  | x |  |  |  |  |  |  |  |  |
| RQ\_G9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G11 |  |  |  |  |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |
| RQ\_G12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |
| RQ\_G13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |
| RQ\_G14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |
| RQ\_G15 |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |
| RQ\_G16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |
| RQ\_G17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |
| RQ\_G18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |
| RQ\_G19 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |
| RQ\_G20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |
| RQ\_G21 |  |  | x |  |  |  |  |  |  |  | x |  |  |  |  |  |  |  |  |
| RQ\_G22 |  |  |  |  |  |  |  | x | x | x | x |  |  |  |  |  |  |  |  |
| RQ\_G23 |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x |  |  |  |  |
| RQ\_G24 |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |  |  |  |  |
| RQ\_G25 |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RQ\_G26 |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* Spaces in Red have been removed

# 

# Gantt Chart (OLD)

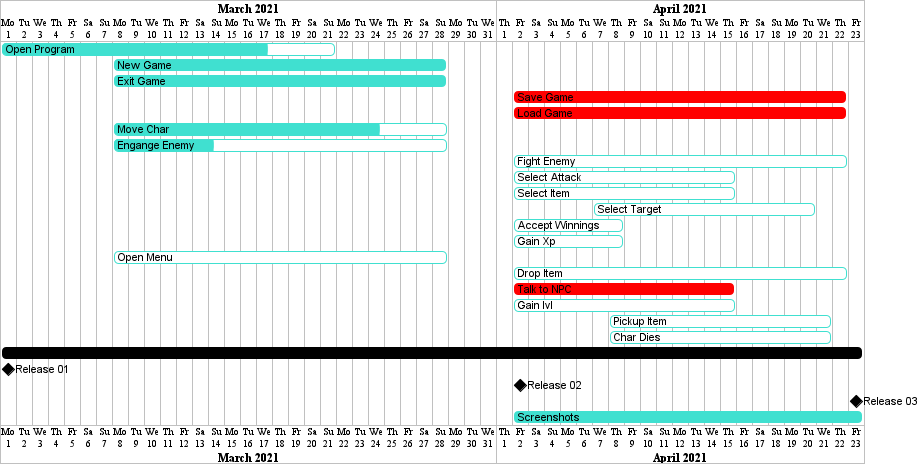
**Blue**: Planned and incomplete **Red**: Incomplete and over deadline **Green**: Complete

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Month | | **MAR** | | | | | **APR** | | | |
|  | **First day of week** | 1 | 8 | 15 | 22 | 29 | 5 | 12 | 19 | 26 |
| **Tasks** |  |  |  |  | **2nd Release 03/26** |  |  |  |  | **Last Release 04/23** |
| **Software Development** | |  |  |  |  |  |  |  |  |  |
| Open Progr. | | **0%** |  |  |  |  |  |  |  |  |
| New Game | |  | **0%** |  |  |  |  |  |  |  |
| Exit Game | |  | **0%** |  |  |  |  |  |  |  |
| Save Game NYI | |  |  |  |  | **0%** |  |  |  |  |
| Load Game NYI | |  |  |  |  | **0%** |  |  |  |  |
| Move Char. | |  | **0%** |  |  |  |  |  |  |  |
| Engage Enemy | |  | **0%** |  |  |  |  |  |  |  |
| Fight Enemy | |  |  |  | **0%** |  |  |  |  |  |
| Select Attack | |  |  |  | **0%** |  |  |  |  |  |
| Select Item | |  |  |  | **0%** |  |  |  |  |  |
| Select Target | |  |  |  |  | **0%** |  |  |  |  |
| Accept Winnings | |  |  |  | **0%** |  |  |  |  |  |
| Gain XP | |  |  |  | **0%** |  |  |  |  |  |
| Open Menu | |  | **0%** |  |  |  |  |  |  |  |
| Drop Item | |  |  |  | **0%** |  |  |  |  |  |
| Talk to NPC | |  |  |  |  | **0%** |  |  |  |  |
| Gain Lvl | |  |  |  |  | **0%** |  |  |  |  |
| Pick up Item | |  |  |  |  |  | **0%** |  |  |  |
| Char Dies | |  |  |  |  |  | **0%** |  |  |  |
| **Documentation** | |  |  |  |  |  |  |  |  |  |
| Release 1 | | **100%** |  |  |  |  |  |  |  |  |
| Release 2 | |  | **0%** |  |  |  |  |  |  |  |
| Release 3 | |  |  |  |  | **0%** |  |  |  |  |
| Screenshots | |  |  |  | **0%** |  |  |  |  |  |

# 

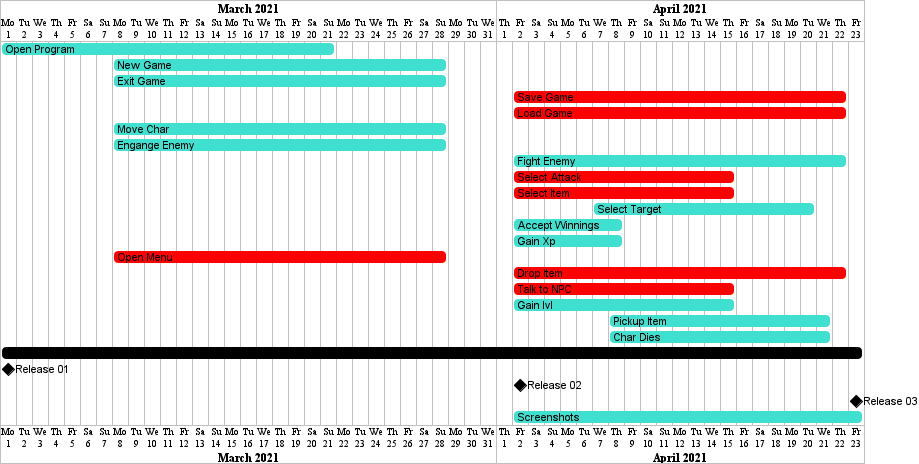
# Gantt Chart (NEW)

Gantt Chart for release 2



* All Tasks in red have been dropped to time constraints

Gantt Chart for release 3



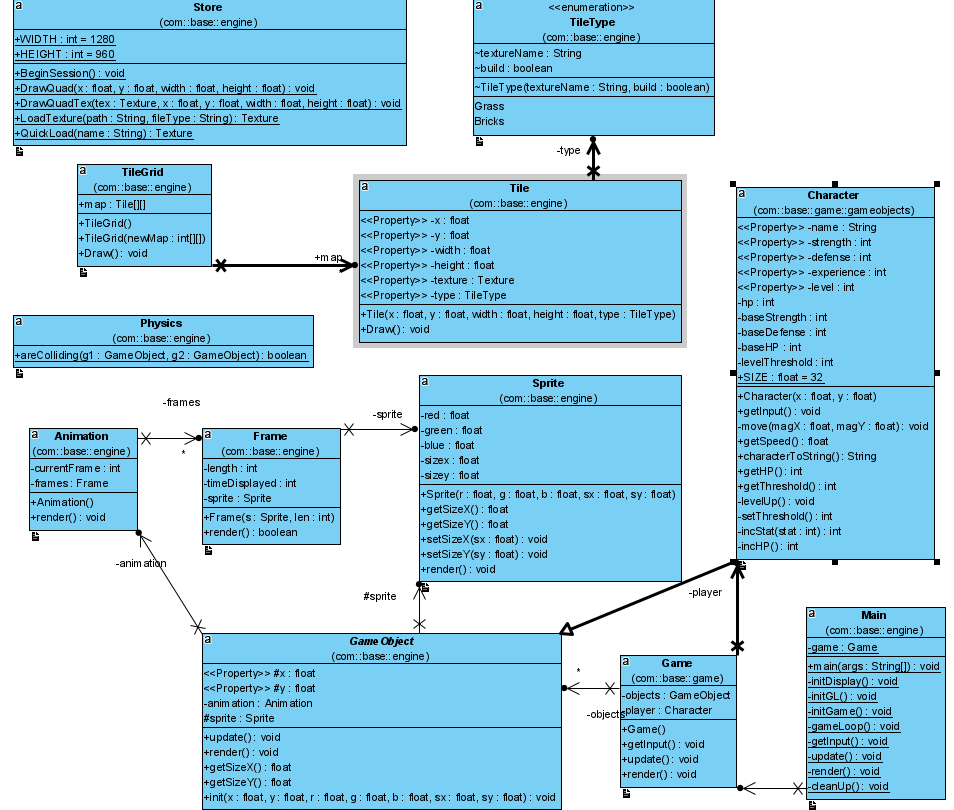
* All Tasks in red have been dropped to time constraints

# 

# Design

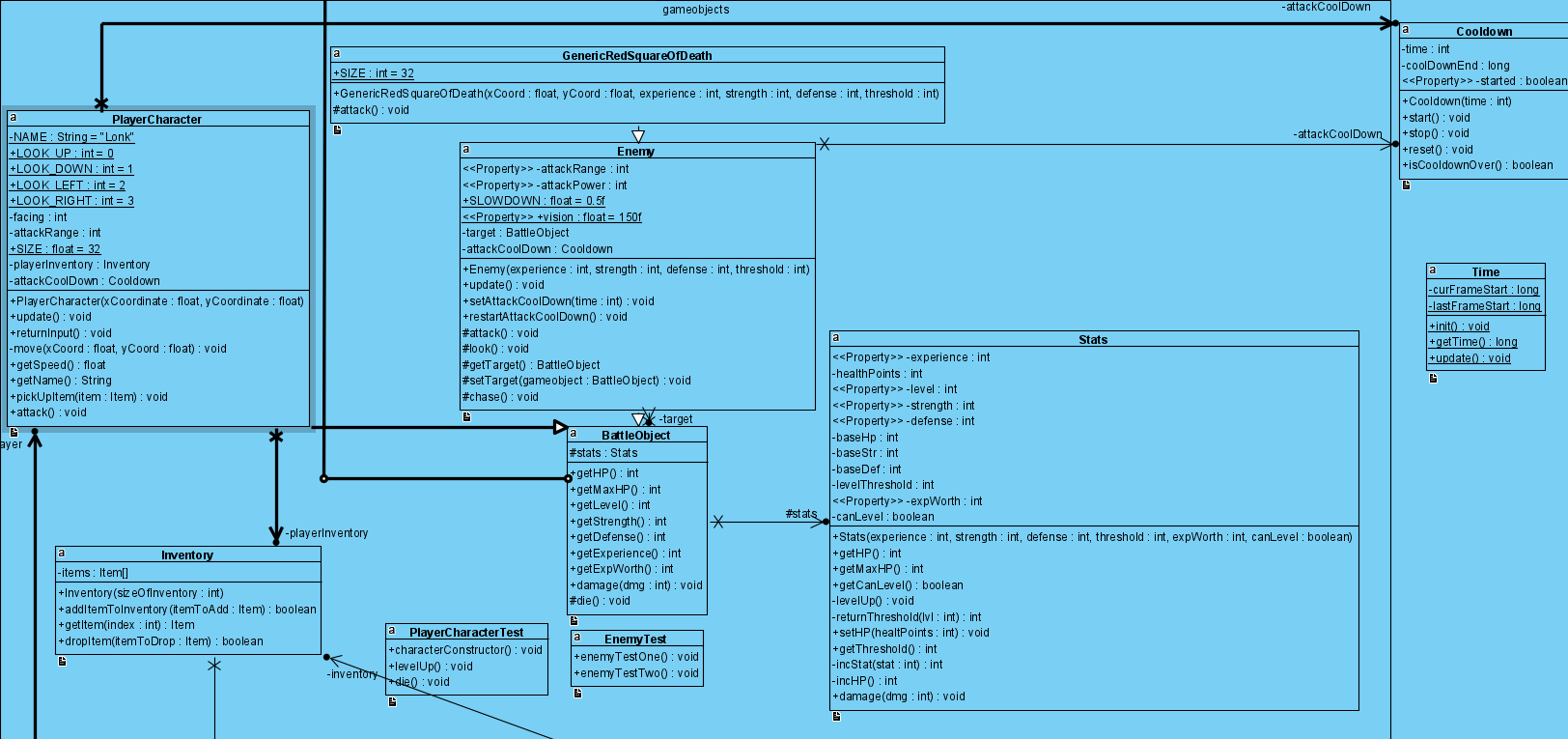
**Refer to Gantt Chart for projected completion of this section.**

**Gantt Chart from 2nd release**

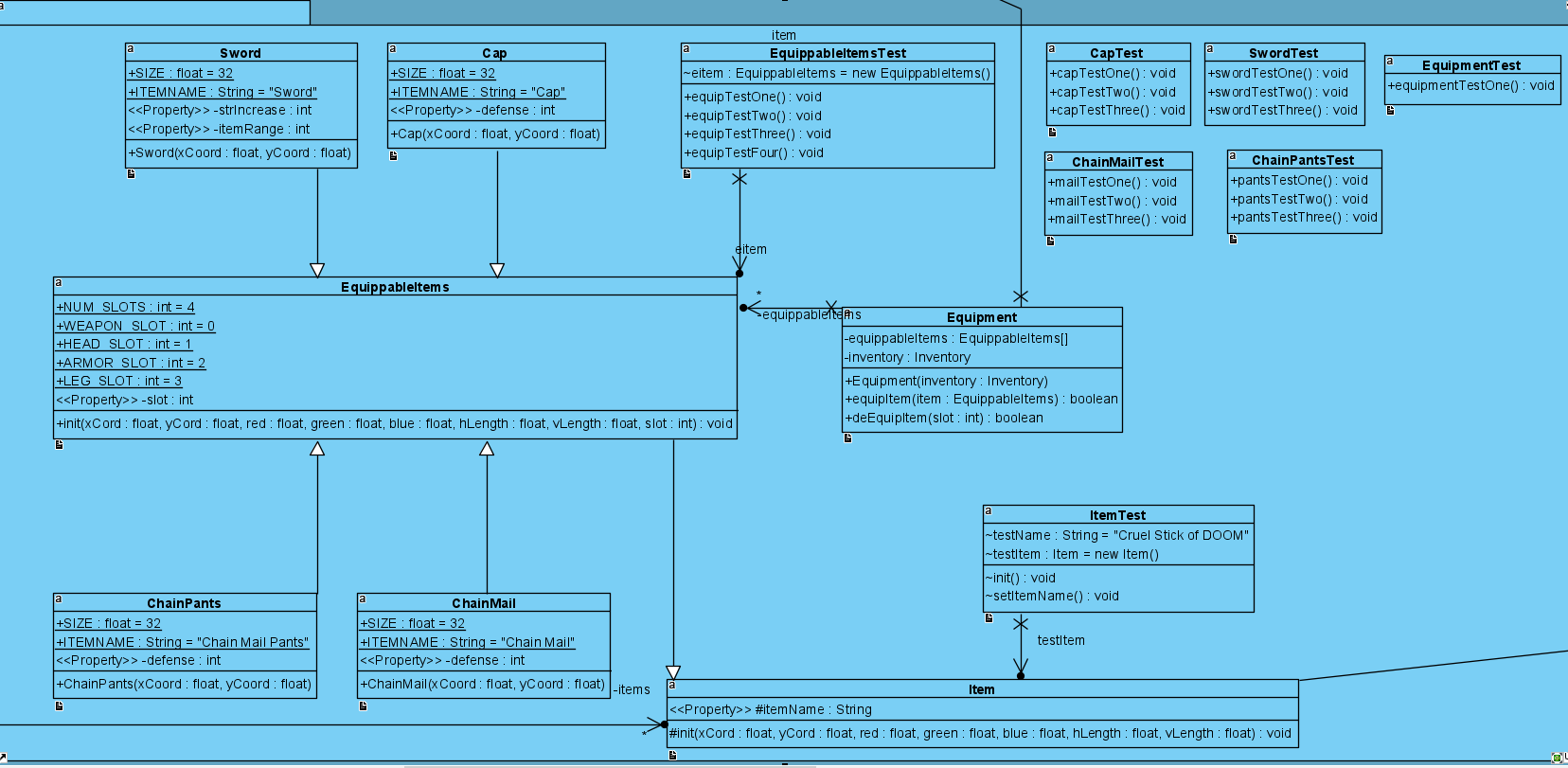


**Gantt Chart from Final Release.**

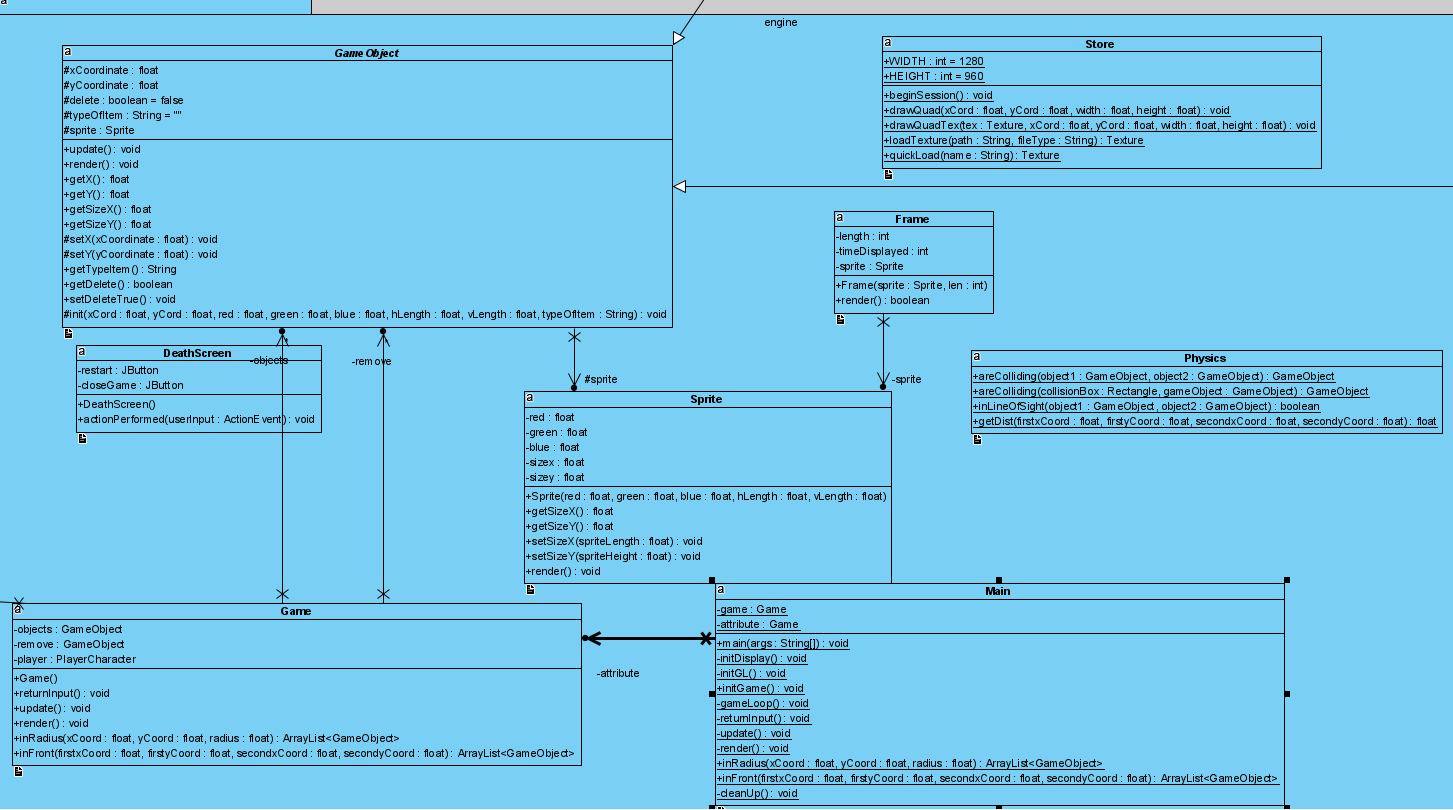
**gameObject Package**



**item Package**

****

**engine Package**

****

* **Because of the size, complexity and number or classes and methods, we divided the class diagram into 3 parts. Each part represents a package.**
* **Some connections are not seen because of the image being split. Bellow is a list of these connections:**
* BattleObject class from gameObject packages extends GameObject class from the engine package.
* Item class from the item package extends GameObject class from the engine package.
* Equipment class from the item package extends Inventory from gameobjects.
* Inventory class from gameObjects packages package uses Item from the item package.
* Equipment class in the item package uses inventory from the Inventory class in gameObjects.
* The GameObject class in the engine package uses the PlayerCharacter class in the gameobject package.
* Both PlayerCharacter and Enemy classes in gameobjects use the Cooldown class.

**DeathScreen**

This class displays a death message if the player dies.

**Frame**

This class determines behavior for rendering and moving between frames of animations.

**GameObject**

Class for all in-game objects with sprites.

**Main**

The main class of the program that opens and closes the game and manages the game loop.

**Physics**

This class handles specific physics of the game.

**Sprite**

This class handles the drawing and size of ingame sprites.

**Store**

This class stores functions to take volume out of other classes in the program.

**Tile**

This class creates a tile based on the parameters and draws it

**TileGrid**

This class creates a grid of tiles based on specifications of the output map.

**TileType**

This class sets a resource texture to a name and makes it easily referenced.

**Game**

Class for functions performed by the game.

**Character**

This class simulates the character and manages the stats.

**Battleobject**

This class allows players and enemies to battle each other.

**Enemy**

This class creates behaviors for Enemies.

**EnemyTest**

This class tests the stats of Enemies.

**GenericRedSquareOfDeath**

This class creates the behavior of the Red Square enemy.

**Inventory**

This class manages the player’s inventory.

**PlayerCharacter**

This class manages stats and behavior of the Player.

**PlayerCharacterTest**

This class tests the stats of the Player.

**Stats**

This class manages the stats of GameObjects.

**Cap**

This class creates a helmet that can be equipped and used.

**ChainMail**

This class creates a chainmail that can be equipped and used.

**ChainPants**

This class creates chainmail pants that can be equipped and used.

**Equipment**

This class manages the current equipped items.

**EquippableItems**

This class Checks to see if an item is equipable .

**Item**

This class manages items that can be picked up in the game.

**Sword**

This class creates a sword that can be equipped and used.

**Cooldown**

This class handles attack cooldown.

**Time**

This class manages time for attack cooldown.

# Development

**Refer to Gantt Chart for projected completion of this section.**

Currently PMD is used on Intellij to produce a CheckStyle list and for static analysis

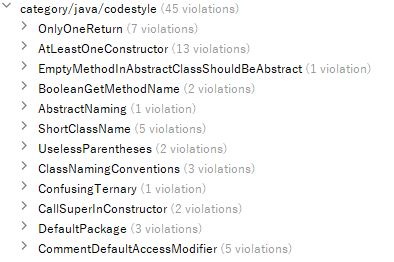
## Additional Libraries

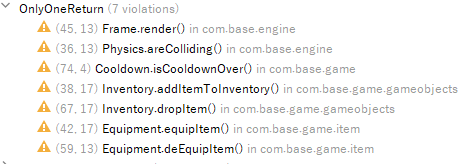
* Lightweight Java Game Library (LWJGL) ver. 2.8.4
  + for simpler game engine setup
  + NOTE: This library is an external library that had to be set up on each group member’s local machines to run the program. The program currently does not run if this library is not installed.
* Slick-Util
  + for overworld tile sprites

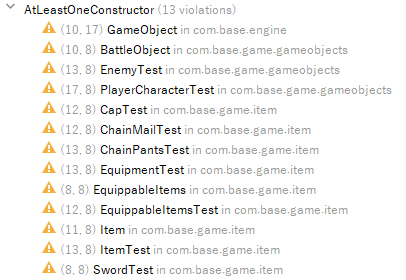
## Code Standards

Code will follow the GVSU Java Style Guide.

The below list shows all the issues reported by PDM



* Only one Return:  
  
  + Good practice to have only one return one variable. In our case we need to return multiple items for different circumstances
* At least one Constructor



* A lot of the Classes are test classes. Other classes could use constructors and cleaner code, but we ran out of time.
* We didn’t get in time to modify this issue:  
  
* Changing the name of these classes:  
  

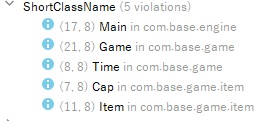
Would not make sense “isDelete and isCanLevel”

* Abstract Class name



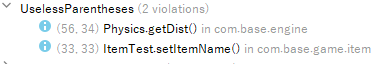
has been created since the beginning of the project and used in most classes. Changing name now would require a lot of time and effort.

* Short Class names:



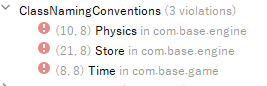
Are easily understandable

* Parenthesis:



While useless, helps with math readability.

* Naming Conventions



Similar to short names are easily understood.

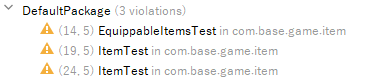
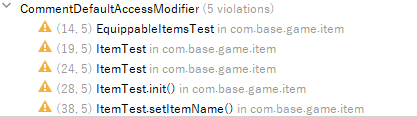
* Confusing statement



if(!something) is not really confusing.

* Call in super   
   

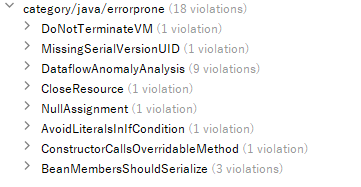
Oversight, doesn’t really affect code

* Default Package  
    
  - Are used for testing purposes only
* Default access modifiers:  
  

Similar to Default packages are only used for inside testing.

## Static Analysis

The below list shows all the issues PMD reported



* Jframe VM termination:



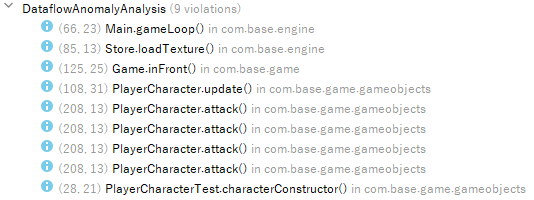
Deprecated system exit. Does not exactly affect anything.

* Missing Serial:



implements and extends while being serialized. Implements last minute. Currently works but could be modified later.

* Data flow anomaly



Most of these initialize variables inside methods. Not necessarily bad, but since most of these are accessed from outside in an update loop, they need to be reinitialized.

* Close resource



resource is actually closed in the try/catch

* Null assignment



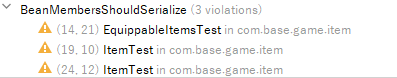
is needed to avoid a nullpointer.

* Literal if



is used to check we encounter an object with actual size above 1

* Serialized Members:

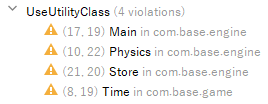


only used in testing.



---- All law of demeter PMD issues were suppressed as they are for security reasons

* Static Variables in



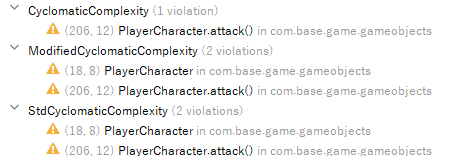
Could be pulled from a utility class. In this case it’s more about preference

* Too many methods in :



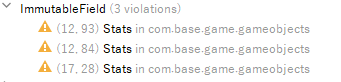
because the it is the main method that all classes connect to

* Complexity in:



Due to the fact that the player has to check each direction to attack.

* Final fields:



Could be made final, but for consistency purposes with other stats are not.

* Local variable:



similar to the above final variable issue, is local due to consistency purposes

* Data Class:



is not DataClass

## 

## 

## 

## Code Documentation

See the JavaDoc.zip folder in the project submission

## Configuration Management

Git repository: **https://github.com/NessXToJason/CIS-350-RPG-Battle-System**

Git Log: **https://github.com/NessXToJason/CIS-350-RPG-Battle-System/commits/main**

The GitHub repository will be used to keep track of all changes made.

Final Release: **https://github.com/NessXToJason/CIS-350-RPG-Battle-System/releases/tag/v2.0-final**

# Verification

Testing and verification was accomplished primarily through playtesting the game for bugs and developing JUnit tests for the GameObjects that can use them.

## Integration Tests / Playtesting

Due to the difficulty of writing proper unit tests for certain aspects of a real-time game, much of our testing involved rigorously playtesting the game and finding unintended behavior to fix.

The following commits are bug fixes that resulted from playtesting (ordered from most to least recent):

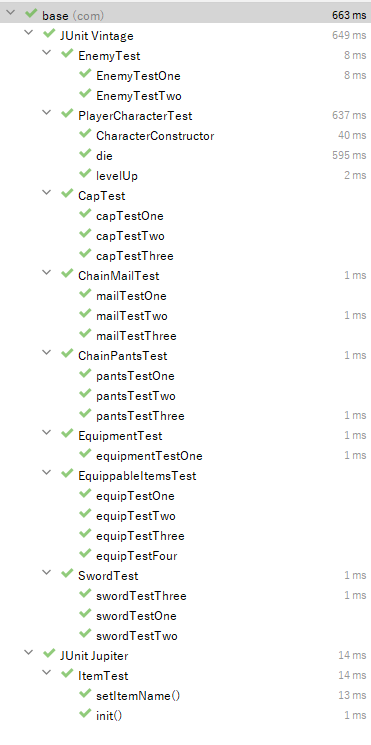
* 514f7e7 - Fixed EXP gain amount from defeating enemies
* 367534d - Fixed bug when attacking enemies
* c7fe099 - Fixed enemy continuing to attack after player death
* 29c15f6 - Fixed crash when attacking before seeing an enemy
* f3160a8 - Fixed issue where overworld and player appeared separately

## Unit Tests

The following classes are used to conduct JUnit tests on their respective classes:

* PlayerCharacterTest
* EnemyTest
* CapTest
* ChainMailTest
* ChainPantsTest
* ItemTest
* EquipmentTest
* EquippableItemsTest
* SwordTest

## Code Coverage



## Requirements Coverage

**Open Progr.**

Just the start of the program. As soon as the program is opened, the game starts with a pre-defined

game state with 4 enemies and 4 items. Tested through play-testing. Playtesting has been done

throughout the life-span of the game.

**New Game**

New Game happens when the player dies. When the player dies a choice between Exiting the game

and starting a New Game can be chosen. Tests done through play-testing (Jframes are not Junit

testable)

**Exit Game**

Similar to New Game, the Exit Game can be tested through playtesting and is a choice when the player

dies. The JFrames used for this and New Game are impossible to test with Junits.

**Move Char.**

Move Char is done through Player input. Testing done through playtesting as movement requires player

input of the W A S and D keys to move, as well as a running Main function.

**Engage Enemy**

Enemy Engagement reacts to MOVE Char. As soon as the Character moves in the enemy’s line of sight,

the enemy engages and fights the player. Testing done through play-testing.

**Fight Enemy**

The player fights the enemy by pressing the SPACE BAR button to attack. It requires player input and is

difficult to translate into Junit testing.

**Accept Winnings**

Winnings are accepted automatically if the player manages to defeat an enemy. Winnings consist of

experience will increase the characters’ level. Junit tests used for this purpose are

“playerCharacterTest” junit.

**Gain XP**

Gaining XP is a byproduct of winning against an enemy. Tested with the Junit test “LevelUp”.

**Gain Lvl**

Leveling up when a certain amount of XP is achieved. Tested with the Juni test “LevelUP”

**Pick up Item**

Item pickup happens when the player’s character touches an item on the ground. Items are

automatically equipped and increase the character’s stats directly. Testing done through the following

Junit tests: “CapTest”, “ChainMailTest”,”ChainPants” test, “SwordTest”, “ItemTest” and “EquippableTest”

**Char Dies**

Character death happens when character HP reaches 0 or below. Junit tests done through the “die” test

in the “playerCharaterTest” Class.

# Postmortem

**Refer to the Gantt Chart for the projected completion of Earned Value and Variances.**

## Responsibilities

* UI Development - Jacob Kucinski
* Overworld Development - Ali Akid
* Player Character System - Justin Sorensen
* Inventory System - Adrian Croitoru
* Encounter System - Jason Truskowski

## 

## Earned Value

On Time

Not included, but not scheduled to be done for this release

Not included (behind schedule)

(Release 2)

**Included**

* The Overworld map has been made in a tileset.
* Ability to swap out textures of the tile set by array.
* Player character stats and leveling system.
* Player Movement.

**Not Included**

* Save/load.
* Items.
* UI Menu.
* Player movement on overworld tileset.
* Encounter System.
* Turn Based Battle system.
* Enemies
* Sprite animation

(Release 3)

**Included**

* Items.
* Player/Enemy Attack.
* Enemy.
* Enemy stats.
* Encounter System.
* Enemy/Player Death

**Not Included**

* Turn Based Battle System.
* UI Menu.
* Save/load.
* Sprite animation
* Our original plan was to have a turn based combat system and an overworld movement system that engaged with static enemies. We would also have an Item system and implement a save/load and a UI menu and a level system for enemies and players. We also discussed a talking to NPC option as a way to use equipment or start battles.
* After release 2 we decided to get rid of Save/load and the talk to NPC option. The group member that was supposed to implement the UI Menu did not end up doing so and we cut that as well at the end of release 2. After a failed attempt of integrating the Tile based overworld system with the movement system we got rid of that in release 3.
* In Release 3 we decided that we did not have enough time to implement the turn based battle system and went towards a hack and slash method. This allowed us to implement enemies and the item system easier and resulted in a playable product at the end of the release.
* We removed the Overworld tile map and used a blank space since flickering was causing issues of integrating the tile map. For release 3 we finished all original ideas besides turn based battle system, UI menu, sprite animation and save/load.

## Justification

* Since the Overworld tile map did not work and was causing flickering bugs with slick2d. Slick2d and lwjgl conflicted with each other as they would alternate usage of the screen on each frame. We removed all of the tile classes and slick2d library since we decided not to use slick anymore and focus only on lwjgl.
* We removed animation class since we did not get to that point and it was wasted space. This saved us a lot of room and increased our code coverage.

## Variances

Time - Finding and implementing LWJGL took significantly longer than expected in release 2, taking us a couple days to fully understand and set up properly. While development became much easier with the tools available in the library, the time loss we suffered from learning to use it set us behind our Gantt chart schedule.

Coverage - Being behind schedule in release 2, we failed to complete many of the features we had hoped to include. We currently just have a simple functioning overworld with a movable player sprite, but we did not get the battle or inventory system in working condition yet due to our early issues implementing LWJGL.

Functionality (Cut Features) - Due to time constraints and the current pacing of the project, these planned features will have to be dropped.

(Release 2)

* Save and Load
* Talk to NPC

(Release 3) - These issues all arose due to unforeseen difficulties with managing LWJGL and Slick that only became significant about a week after the last release, and were cut or simplified for the sake of producing a functional program within the time constraints.

* Overworld with specific image tiles - Unbeknownst to us when choosing external libraries to help develop the project, using LWJGL (for the game itself) and SlickGL (for the overworld tile system) at the same time caused a severe screen flicker issue that would have required a massive overhaul in how the overworld was implemented. We ultimately decided that it was a much higher priority to make sure the game itself functions properly, and removed the tiled overworld while keeping the implementation in the code.
* Turn based battle system - We decided to change the battle system from turn-based to real-time in the overworld.
* UI Menu - Simplified to auto-equip the few items in the game.

## Lessons Learned

* Scheduling group meetings remotely with five people can be difficult, so we’ll have to make sure to plan them well in advance to make sure we can attend most of them. We picked Sunday to work on the Project 1 release and didn’t specify a time, which resulted in one of our members not being able to participate and only listen.
* We should figure out each other's schedule and find out what days and times are going to work best going forward so we don’t run into a scheduling issue again.

***NEW LESSONS FOR RELEASE 2***

* Choose an IDE that we will all be using. This caused some compatibility issues to happen when merging code together.
* In the future, we should do more research regarding the scope of the project and the tools available to us to help in development. While LWJGL has proven useful in developing the game engine and mechanics, it has been difficult to set up and caused errors unfamiliar to all of us, which made certain parts of development harder.

***NEW LESSONS FOR RELEASE 3***

**TEAM**

* Balancing the final weeks of school and a cumulative project is not an easy task. Having the code done around the start of the third stage would have been better timing in retrospect.
* Combining code using LWJGL from different team members portion of the code proved to be very difficult. Building one portion of the game at a time as a team may have resulted in a better finished product.
* At the start of the project it seemed as if it would be simple to implement a lot of features through references we read. We found out that was not the case. We should have planned less features at the start of the project and added on extra ones if time permitted.
* Trying to integrate multiple libraries worked for certain portions of the code, but when more got integrated together it caused problems. We should focus on using one library and then adding in extra libraries once integrated code from a singular library was already functioning. This proved difficult as this is the first time any of us have made a game before.

**INDIVIDUAL**

*Ali Akid*

* I waited till the end of Release 2 to integrate my code with already existing code in the project. We also were using different IDE’s at the time. If I would have implemented my code sooner we would have had more time to resolve the LWJGL, Slick2d integration.
* I tried to solve the flickering issue with the integration of the text tiles as many of us on the project did with no resolution. If I would have finished sooner this would have been easier to deal with and get help from the professor or outside sources.
* I got behind in other classes and negatively affected the team in release 2 and 3. This surely affected the outcome of the project and did/may resulted in a lesser finished project at the end of release 3.
* I would do more research into different IDE’s as some worked better than others for the libraries we were using and also research the library more during the time we had in release 1 that did not involve implementing code.
* I need to work on getting things done in a more timely manner and communicating my issues with the team better.

*Jason Truskowski*

* I spent a lot of time leading up to the first release coming up with ideas for additional functionality for the project before we started coding, which led to a bloated and probably overambitious list of requirements that we were not able to meet within the constraints of the semester. I should have stuck to a more conservative set of requirements that could be expanded upon instead of stripping down an overly large set of requirements.
* I didn’t do enough research into LWJGL before pushing for the idea of using it to help with the project, which led to lost time where we all had to learn it, and issues with importing libraries and running the program on local machines. Development with this tool that was new to all of us would have gone much more smoothly if I had looked into it more and made myself aware of these issues beforehand.
* I tended to get carried away with making long strides of progress on code without communicating with my partners during the fact, which led to several large commits that probably made it difficult for them to get up to speed on all the changes I did. I could have split up my commits more to make it more clear what smaller, specific things I was adding.
* I got into a habit of doing all of my work for other classes before working on the project, and then making up for it by doing a ton of work all at once, which factors into my previous issue of not always communicating effectively with my team about what I was working on. I should have balanced out my efforts among all of my classes to get into a more stable and consistent schedule to work on this project.

*Adiran Croitoru*

* Definitely have to start working much earlier on the project instead of waiting to get more information in class. Even though we were warned about starting late and having our workload increase and having crunch time towards the end, I didn’t expect the workload to be this large.
* Avoid using multiple libraries that I am unfamiliar with. We had issues with flickering while trying to apply a background. This conflict arose from the 2 libraries conflicting with each other.
* Issues arose when using Git with conflicts and merging causing me to lose a good 2 hours of work. Even though I keep telling myself to commit and push more often, I never do it. I will definitely be pushing more often.
* Testing was all done after the project was mostly completed. While in class we did learn about creating tests first and building code around those tests. This might be a way to spread work across more easily and not have a bunch of tests left towards the end.

*Justin Sorensen*

* I had to reorganize or completely redo the code I had written because I did not do enough research into how to implement lwjgl properly before beginning to work on the project. If I had an appropriate understanding of how to use the library beforehand, it would have saved a lot of time reworking the same lines of code.
* I would sprint to get what I could accomplish right away, which later on I encountered issues with because I did not check back in on the git to see where my partners had modified and added classes. This led to issues where I needed to integrate my classes into other’s and I would have to spend ample time getting familiar with the additions and modifications. I have to ensure that I keep up with any ongoing projects.