**Brock Barlow Anh Tran**

**ID #1113 Member ID**

**Anthony Touchet Brandon Pugh**

**ID #1085 Member ID**

**Benjamin Odom Jonah Muguira**

**Member ID Member ID**

**AIE CIP\_CODE\_11.0804**

**ADGP225\_Game Pitch and Prototype**

**I.0 Requirements Documentation**

**I.1: Description of the Problem**

**Name:** Game Pitch and Prototype (ADGP225).

**Problem Statement:** Create a game while simulating a major corporate development/production team.

**Problem Specification:** Students will form a team of three to four programmers and artists. The artists will provide the team with assets to use for the game, while the programmers will code and test the inner workings of the game. The project will need to be versioned using github. At the end of the project, the final product will be compared to the original pitch.

**I.2: Input Information**

**Name:** Touch Controls

**Description:** The user will use their fingers to navigate the various areas/points of the game and will use their fingers to manipulate the combat play field area of the game to make gem/token matches.

**Type:** Touch Controls

**Range of Acceptable Values:** The user’s finger tips.

**I.3: Output Information**

**Results that are expected from “Touch Controls”:** The user should be brought to different scenes of the game by pressing ui buttons, the gems/tokens should move in the direction the user is sweeping, the target indicator should highlight the desired enemy when pressed.

**I.4: User Interface Information**

The main/title scene displays the game’s title and two buttons: play and credits. The play button will load the stage selection scene while the credits button will load the credits scene.

The combat scene contains a grid. Inside of the grid are gems which can be moved around within the grid horizontally and vertically as a unit. When hitting the end of the screen, it wraps around to the other side. Above this grid is the player’s health bar/shield bar. To the right is the combat mode button. This swaps between defense and attack mode. Matching at least 3 gems in a row or column will give either damage, or give the player shield depending on the mode. The game can be paused by pressing the menu button. When the menu is active, sounds can be turned off by pressing the icons correlating to the sound type. The power icon will exit the combat scene.

The credits scene displays a return button ad who made the game as well as the names of the individuals who did the sounds and music. The return button reloads the main/title scene.

The Stage Selection Scene will visually display nodes and the paths connecting them. The scene will also contain a panel that displays the information on the currently selected node. Players can select a node by tapping on the node’s visual representation. This scene also has a button that can bring them back to the Title scene.

**I.5: Code Descriptions**

**LevelSystem.cs (created by Brock Barlow):** This script is used to level up the user after winning combat. The script takes the user’s base xp and multiplies it by the level aiming for times the increase factor. When the user levels up, their health, attack, defense and stamina stats will go up by a preset amount.

**BaseItem.cs (created by Brock Barlow):** This script is used to create items. These items are instant items, turn buff items and time buff items. Instant items have an instant use for the user. Once used, they will get destroyed. Turn buff items will stay active until a certain amount of turns are performed by the user. Time buff items will stay active until a certain amount of time passes during combat.

**ItemManager.cs (created by Brock Barlow):** This script is used to manage the items the user has. This script manages the saving and loading of items. This script manages the main inventory, combat inventory and active inventory. The main inventory is the overall inventory that holds all the items, the combat inventory is the user’s inventory that’s available during combat, and the active inventory is for the turn and time buffs when they are active on the user.

**GameManager.cs (created by Anthony Touchet):** This script is used save the state of the game and handle the transitioning between scene.

**Attributes.cs (created by Anthony Touchet):** This script is used in the enemies and player to represent certain stats about the entity while allowing for modifiers and coefficients to be used while not altering the base value.

**AudioManager.cs (created by Anthony Touchet):** This script controls the sounds within the game. Placed on the same Gameobject as the GameManager, this script grabs information from the GameManager and plays the appropriate music and sound effects for each level.

**PlayerData.cs (created by Anthony Touchet):** This script is placed within the GameManager and is used to represent the current state of the player and the game-world.

**StaminaManager.cs (created by Anthony Touchet):** This is a system designed to regularly increment a resource called Stamina. This resource is used to enter combat. This resource comes back every so many seconds and even works when the ap0p is closed by storing the time of when the application is closed.

**CombatStatsBar.cs (created by Anthony Touchet):** This code is in charge of displaying the player’s Attributes to the user.

**Enemy.cs/EnemyMono.cs (created by Anthony Touchet):** These two scripts are used to represent the enemies within the game. Enemy.cs is the technical side of the enemy. Controlling when the Enemy attacks, updates, and takes damage, as well as keeping track of its Attributes. EnemyMono.cs is handling the visual representation of the Enemy.

**EnemyManager.cs (created by Anthony Touchet):** This script manages all the enemies within the scene as well as keeping track on which enemy is the current target of the player. The majority of this script takes place at the beginning of the scene for it has to create and place each enemy on the field.

**TargetingEnemy.cs (created by Anthony Touchet):** This script is allowing the player to choose a current target and allowing the player to switch the current enemy that the player would like to attack.

**MonoNode.cs (created by Anthony Touchet):** This script was used as a visual representation of the StageSelection tree of nodes, primarily displaying information and having a reference to the node it represents.

**StageSelectionManager.cs (created by Anthony Touchet):** This system is used to manage, create, hold, and display information about the game-world. This system is based of of a tree-node system where each node is connected to another node, with each system of connections being grouped into trees. The system statically defines all of the trees and nodes and places the connections inside them. This manager also creates and managers visually displaying the system and managing how it moves and the information about the current node selected.

**StaminaDisplay.cs (created by Anthony Touchet):** This script is used to visually display the information about the StaminaManager within the stage selection menu.

**Gem.cs/GemMono.cs (created by Benjamin Odom):** These scripts contain all logic for gems within the board. This involves positional updates, offset updates, and any member variables and fields which are needed for a ‘gem’.

**GemImage.cs (created by Benjamin Odom):** This script contains logic used to sync the visual representation of a gem to its information. This involves keeping its image, and colors proper for each image that is part of the visual representation.

**GemMonoDuplicate.cs (created by Benjamin Odom):** This script is used to make a copy of a gem visually without corrupting its information. This is useful when the gem wraps around the screen. It is an exact copy of its GemMono counterpart visually.

**GridCollection.cs/GridCollectionMono.cs (created by Benjamin Odom):** These scripts contain logic regarding the rows and columns of the grid. This includes handling position and offsets for each the row or column. Inside the script are two separate classes which inherit from GridCollection called Row and Column. These classes handle the separate functions related to either a row or a column since logic for movement is different.

**Grid.cs/GridMono.cs (created by Benjamin Odom):** These scripts contain the logic for the grid itself. This includes creating gems and gridcollections and storing a reference to them for future use. Anytime something needs to be changed or moved around within the grid it goes through these scripts.

**CombatCamera.cs (created by Benjamin Odom):** This script contains the logic for the camera when in the combat scene. This includes the animation behaviour, and shake behaviour.

**InputManager.cs (created by Benjamin Odom):** This script contains custom logic for input. It determines what kind of input was given; press, hold, release, beginDrag, drag, endDrag. It invokes UnityEvents when a certain type of input is detected.

**Particle2D.cs (created by Benjamin Odom):**  This script is used to handle a custom particle on the canvas. Since unity cannot use particle system on a canvas, this alternative class was made. After a certain period of time, they destroy themselves.

**ParticleSystem2D (created by Benjamin Odom):** This script is used to create particles from the particle2D class and attach them to the proper canvas.

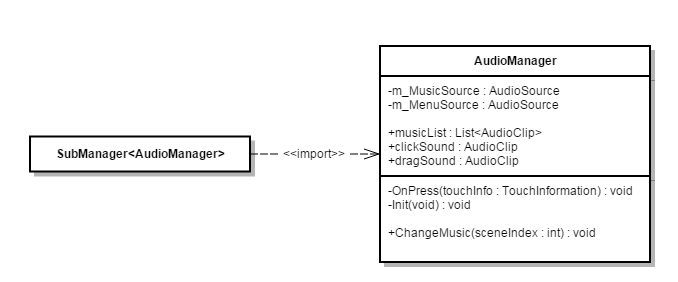
**RandomManager.cs (created by Benjamin Odom):** This script contains a manager for random number generators. Any system of object that needs a random number generator gets it from this script. This way it can be serialized into a .json and replicated later after the game ends. It is useful when attempting to replay a gaming session.

**InputRecorder.cs (created by Benjamin Odom):** This script records input from the input manager and serializes it into a .json file for replaying later.

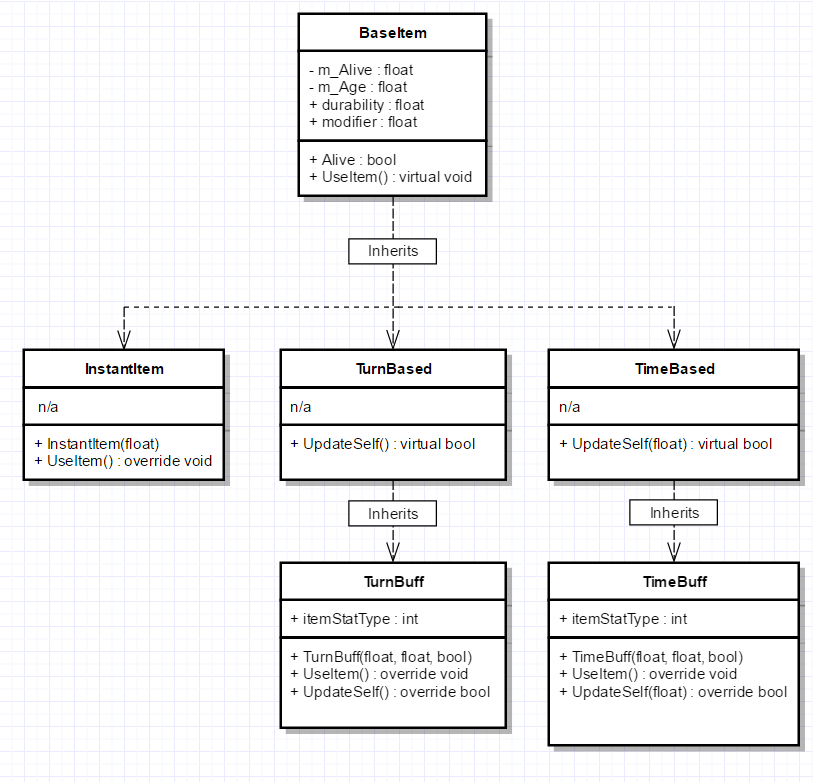
**InputReplay.cs (created by Benjamin Odom):** This script takes .json information from the InputRecorder.cs and recreates a gaming session’s input.

**II.0 Design Documentation**

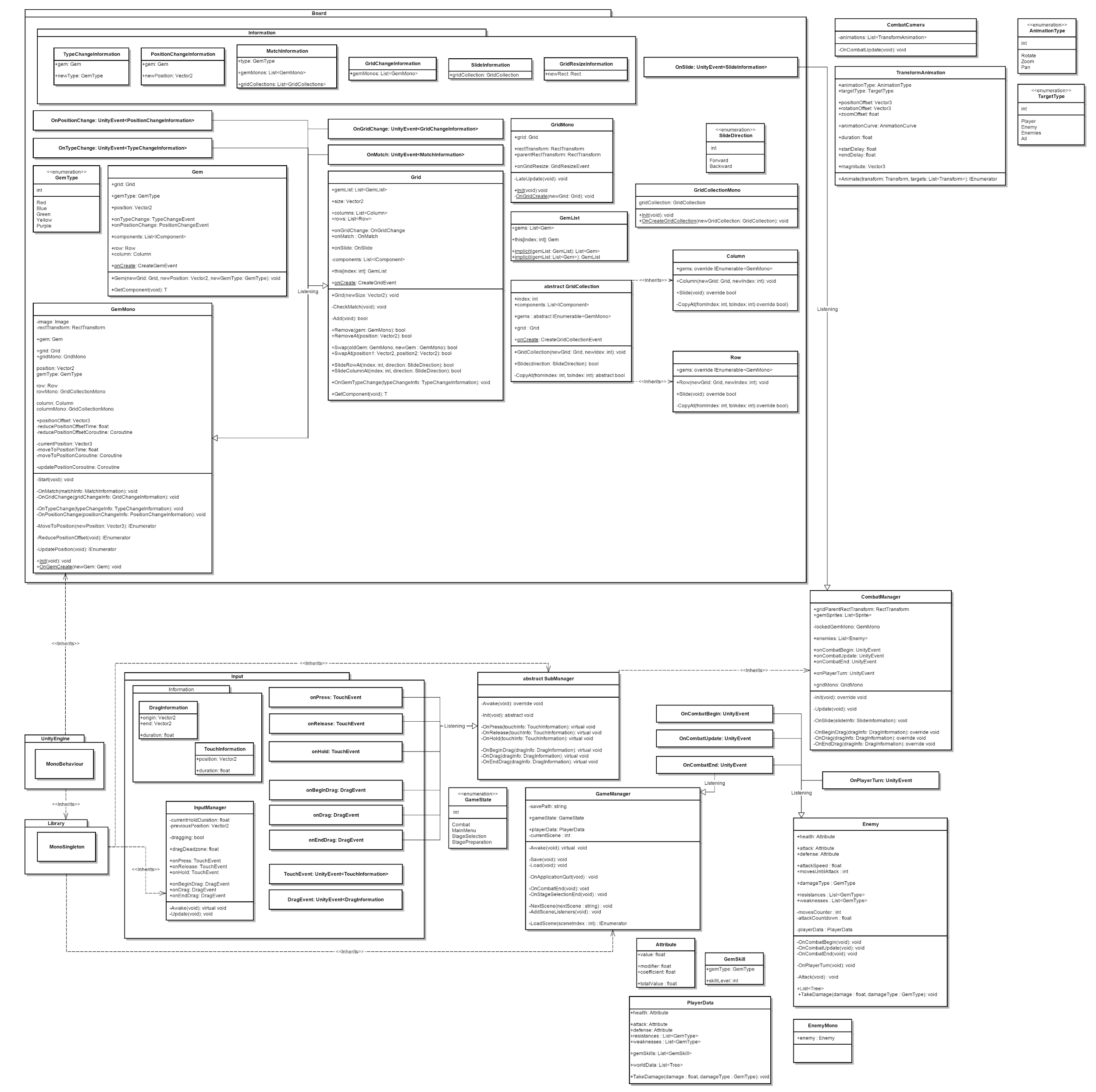
**II.1: Diagrams:**

**Initial AudioManager UML**

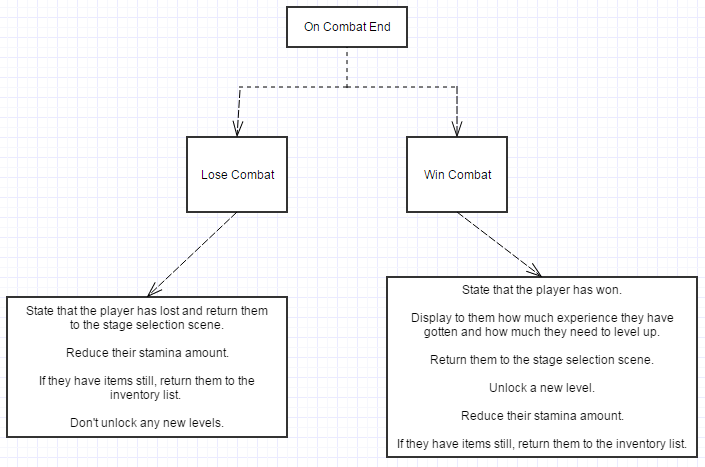
**Initial Item System UML**



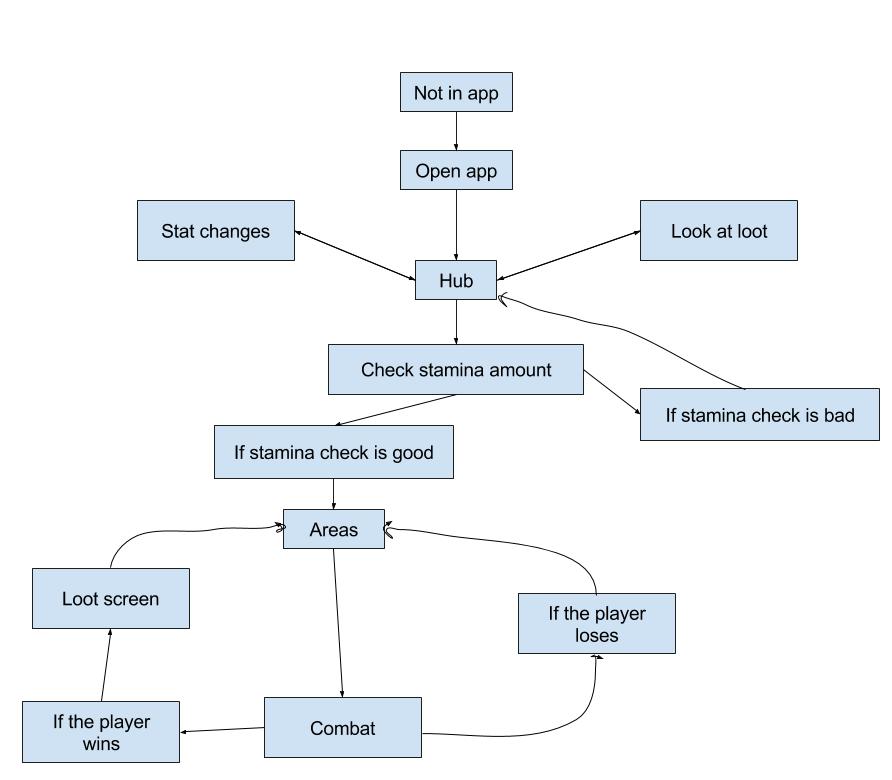
**Initial Board and Manager UML**



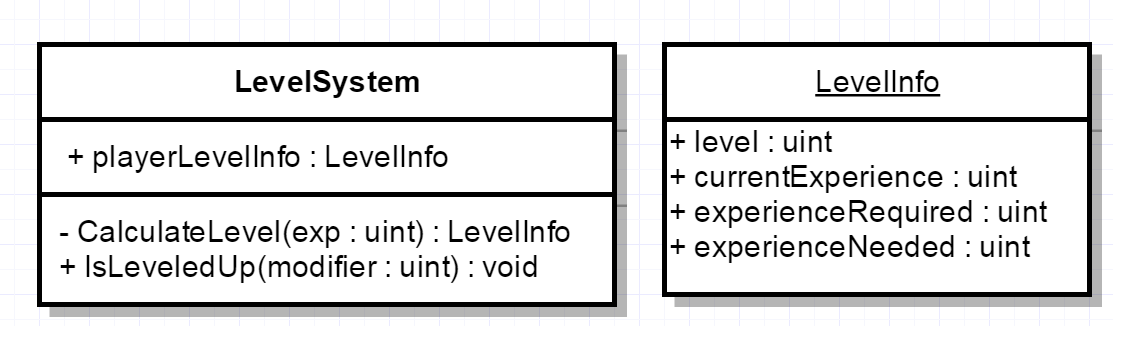
**Defining the End of Combat**



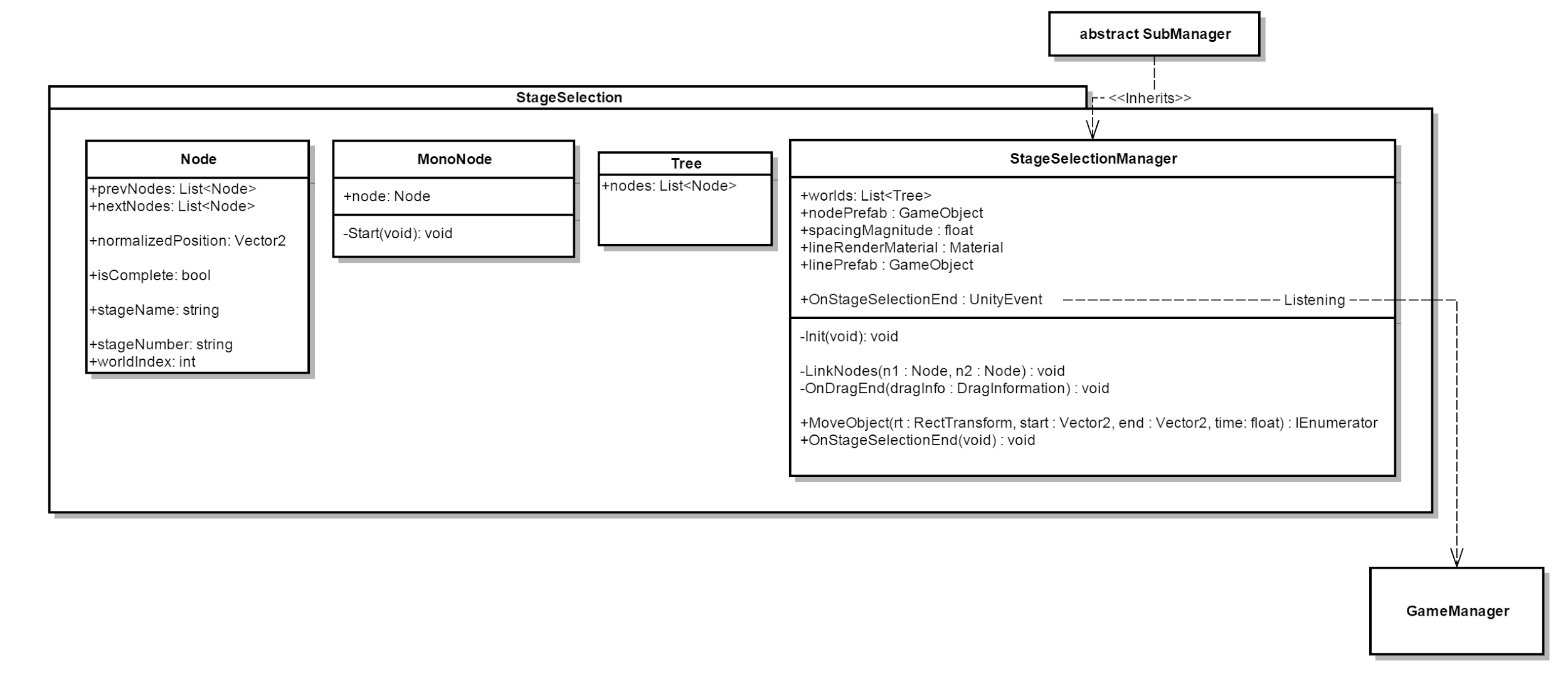
**Game-Flow Chart**



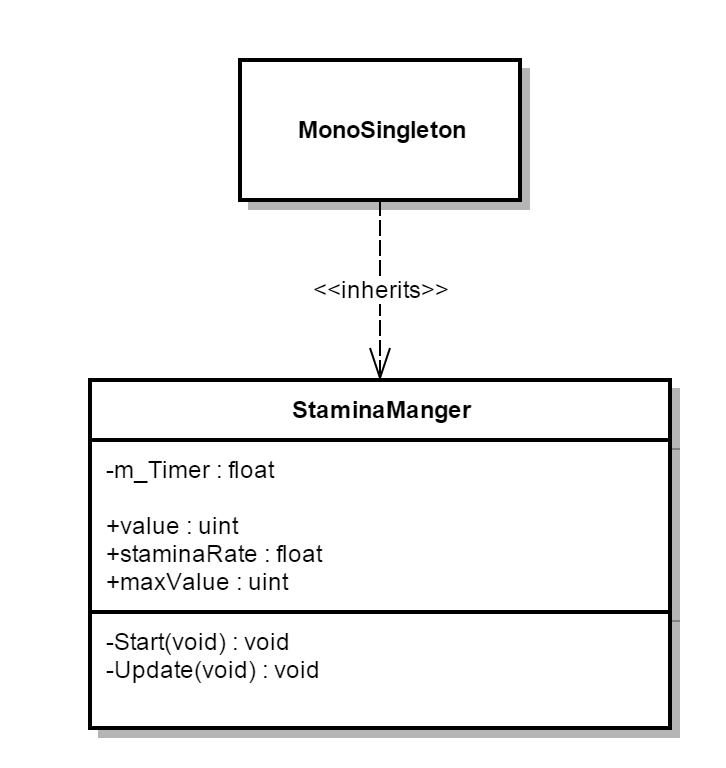
**Initial Level System**



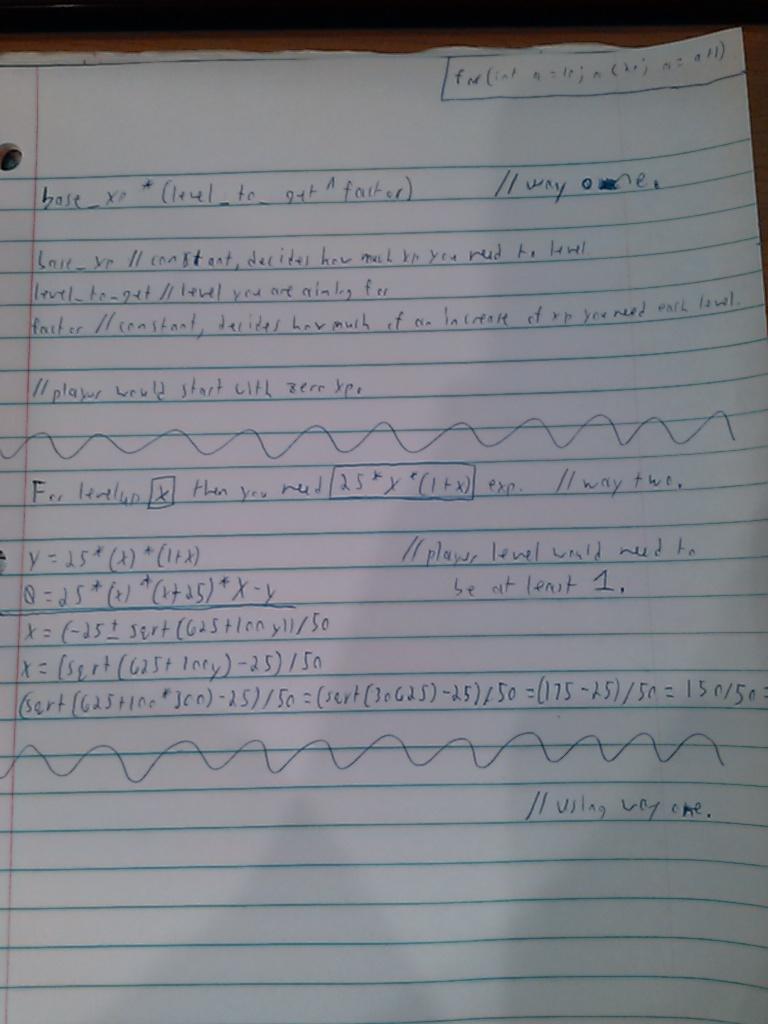
**Initial Stage Selection**



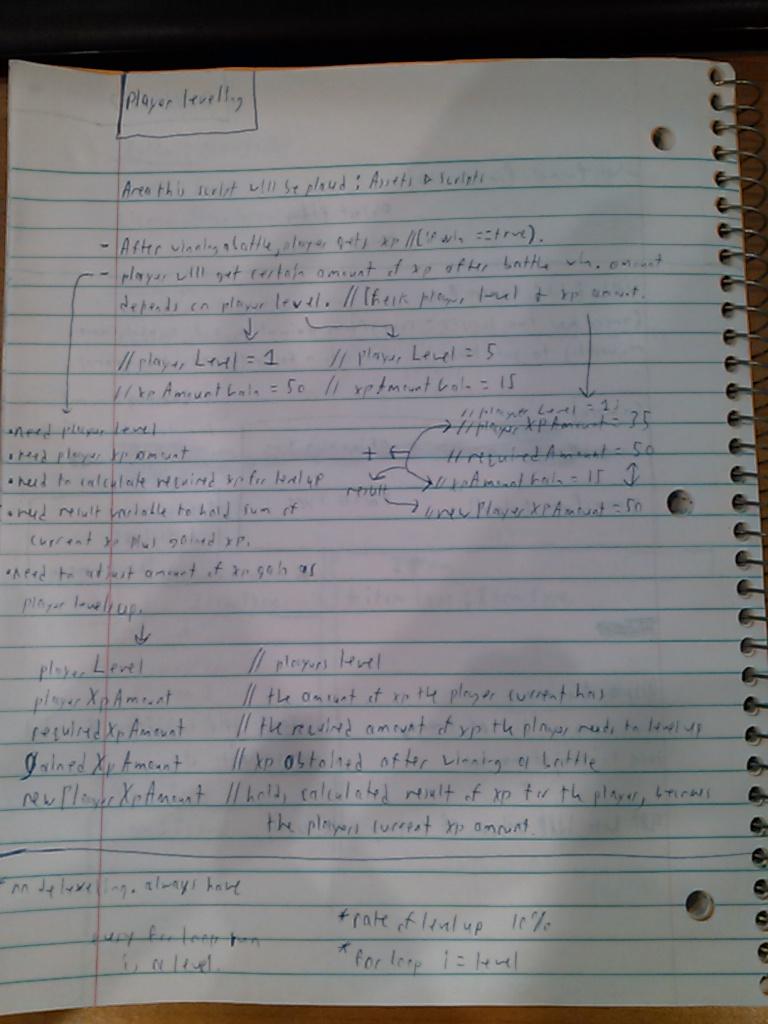
**Initial Stamina System**

**II.2: Drafts and Math:**

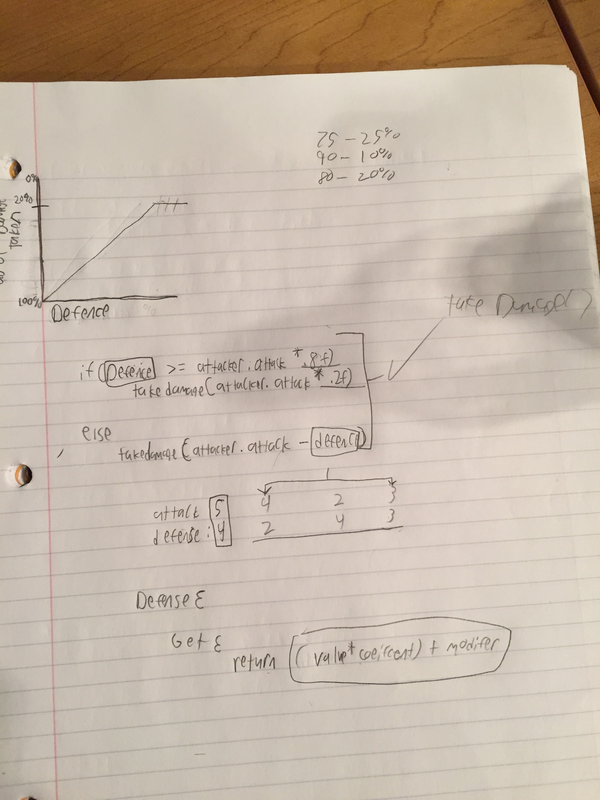
**Algorithm Research**



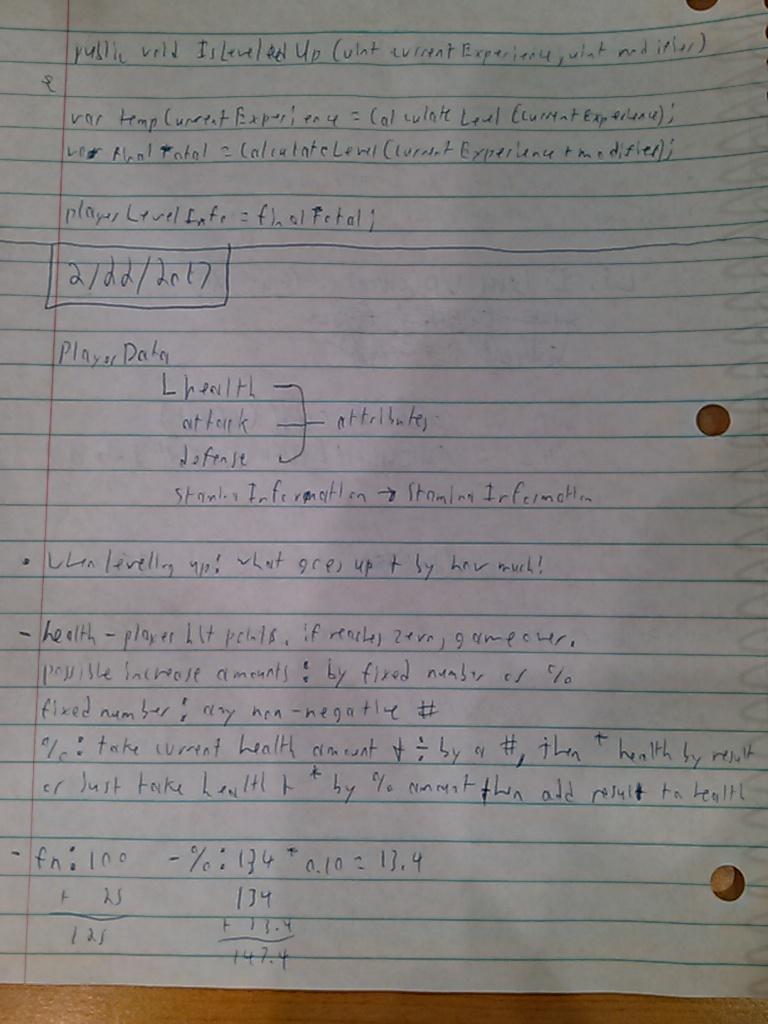
**Initial Level System Drafting**



**Damage Calculations Draft**



**Level Up Draft**



**III.0 Implementation Documentation**

**\*See “Metallic Clashers Source Code” document.**

**VI.0 Verification & Validation Documentation**

**VI.1: Test Data**

**Test #1 - Test Title Scene:** Test the play and credits buttons. Button should load other scenes.

**Test #2 - Test Credits Scene:** Test the return button. Button should load title scene.

**Test #3 - Test Stage Selection Scene:** Test the stage selection scene. Test title button, test screen sweeping, test engage button, test stamina, test nodes.

**Test #4 - Test Combat Scene:** Test combat scene. Test the menu button, test the mode switch button, test the pause menu buttons, test the gem broad, test the enemy lock-on, test the defense mode, and test the attack mode.

**Test #5 - Test Level System:** Test level up system. When player wins combat, should get xp and/or level up. Stats should change when leveled up.

**VI.2: Test Results**

**Test #1 - Test Title Scene:** Tested buttons. Play button loads stage selection scene and credits button loads credits scene.

**Test #2 - Test Credits Scene:** Tested button. Return button loads the title scene.

**Test #3 - Test Stage Selection Scene:** Tested title button. This button loads the title scene. Tested screen sweeping. When sweeping, second world appears and sweeping back shows first world. Tested engage button. Engage button loads the combat scene. Tested stamina. When the player has little or no stamina, levels cannot be played. Tested nodes. Gery nodes remain locked until unlocked, red nodes are playable, and green notes appear after winning combat scene.

**Test #4 - Test Combat Scene:** Tested combat scene. The menu button pauses the game. The mode switch button will switch the player from attack mode to defense mode and vice versa. The pause menu buttons work. Sound button will enable/disable the sound effects. The music button will enable/disable the music effects. The help button will display help text to the player about the broad. The quit button will bring the player back to the stage selection scene. The gem/token broad takes in player sweeping input and performs matches once the requirements are meant. The enemies can be locked on to and can be switched when touching them. Defense mode will change broad color and gems/tokens and after matches will build up defense value. Attack mode will change broad color and gems/tokens and after matches will make enemies take damage.

**Test #5 - Test Level System:** Tested level system. When combat is won, player receives an amount of xp and might perform a level up. When leveled up, the player’s health, attack, defense, and stamina will increase by a flat amount (health goes up by ten, attack goes up by two, defense goes up by two, and stamina goes up by two).

**VI.3: Operating Directions**

**Method #1 - Android Version:** Go to <https://github.com/brockbarlow/Metallic_Clashers/releases> and download the “Metallic\_Clasher.apk” file. Connect your android device to the pc and go to the “Downloads” folder on your pc. Copy and paste or move the “Metallic\_Clashers.apk” file to your android’s “Downloads” folder. Select the “Metallic\_Clashers.apk” file and install it. Once installed, open the app and wait for it to load. Once loaded, the game can be played. The game is controlled with touch inputs. Any action the game requires from the player needs to be a touch input. To play the game, press the “Play” button. At the stage selection screen, select a “red node” and press the “engage” button to being combat. “Green nodes” indicates that the stage was cleared/won. “Grey nodes” indicates that a level is still locked/inaccessible until the previous level is cleared. During combat, the player needs to match at least three gems/tokens together (match needs to be three gems/tokens of the same color) to attack the enemy(s). Once all enemies are defeated, the player will receive exp and unlock the next level. If the player loses, they will need to replay the level. During combat, pressing the “MENU” button will pause the game. From here, the player can enable/disable game sound, enable/disable music, receive info about the user interface, and/or quit the current battle they are in and return to the stage selection. Levels can only be played if the player has enough stamina. If the player does not have enough stamina, a certain amount of time needs to pass to refill their stamina amount.

**Range of Acceptable Values (Touch Input):**

Touch Presses - used for buttons. User must “touch” a button to perform action.

Touch Sweeping - used to manipulate the gem/token broad during combat. Sweeping up, down, left and right will manipulate the broad.

**VI.4: Bug Reports**

**Bug #1 - “EXP Bar Fill”:** This bug occurs right after combat is finished and the results panel is displayed. The EXP bar is supposed to fill with the experience the player has, fill in the experience the player has earned, and the dark blue bar is supposed to fill the rest of the way to the end of the lighter bar. Instead, the bar fills only most of the way and there is a sliver of the mid bar still present.

**Bug #2 - “Chugging Along”:** When the game is played for long periods there seems to be a slowness to it. This slowness is only apparent in the loading of the Combat Scene and when a match is made within that scene. This slowness only appears if the player has already completed multiple battles. After a number of battles, the game starts to increasingly slow, becoming apparent when Combat first loads and when a match is made.

**Bug #3 - “Gatling Decapitation”:** Sometimes in combat, the camera will be in a position where if the Gatling Gal enemy is on the end of the line, only the bottom half of the model will be displayed. This could be a result of the camera being too low or the animation still being too kinematic, moving into the camera and not being properly displayed.

**Bug #4 - “Level Up?”:** When the Player levels up after combat, the bar does not fill all the way to the end of the screen. The foreground of the bar stays stationary.