Unity Project

Design Specification

Members: Anthony Jamora, Jonny Nabors, Thomas Burke

Draft 1

*CSC 4420*

*Unity Group*

*2015*

[1 Game Description 5](#_Toc413699923)

[1.1 Design Goals 5](#_Toc413699924)

[1.2 Influences and Sources 5](#_Toc413699925)

[1.3 Target Audience 5](#_Toc413699926)

[1.4 Target Customer 5](#_Toc413699927)

[2 Functional Specifications 6](#_Toc413699928)

[2.1 Core Game Play 6](#_Toc413699929)

[2.2 Game Flow 6](#_Toc413699930)

[2.3 Game Flow Diagram 7](#_Toc413699931)

[2.4 Characters 7](#_Toc413699932)

[2.5 Game Play Elements 8](#_Toc413699933)

[2.6 Game Physics and Statistics 10](#_Toc413699934)

[2.7 Artificial Intelligence 10](#_Toc413699935)

[3 User Interface and Menus 10](#_Toc413699936)

[3.1 Main Game Menu 10](#_Toc413699937)

[3.2 Splash Screen 12](#_Toc413699938)

[3.3 Game Menus 13](#_Toc413699939)

[3.3.1 New Game Menu 13](#_Toc413699940)

[3.3.2 In-Game Menu 16](#_Toc413699941)

[3.3.3 Exit Confirmation Menu 16](#_Toc413699942)

[3.3.4 Heads-Up Display/User Interface 17](#_Toc413699943)

[4 Art and Video 18](#_Toc413699944)

[4.1 Overall Artistic Goals 18](#_Toc413699945)

[4.2 3D Art and Animations 19](#_Toc413699946)

[4.3 GUI 19](#_Toc413699947)

[4.4 Terrain 23](#_Toc413699948)

[4.5 Game Play Elements 23](#_Toc413699949)

[4.6 Special Effects 23](#_Toc413699950)

[4.7 Marketing and Packaging Art 23](#_Toc413699951)

[4.8 Assets Pipeline & Folder Structure 23](#_Toc413699952)

[5 Sound and Music 24](#_Toc413699953)

[5.1 Overall Goals 25](#_Toc413699954)

[5.2 Audio Sources 25](#_Toc413699955)

[5.3 Audio Diagram 28](#_Toc413699956)

[6 Characters 29](#_Toc413699957)

[6.1 Player Characters 29](#_Toc413699958)

[6.2 Secondary Characters 30](#_Toc413699959)

[6.3 Enemy Characters 31](#_Toc413699960)

[6.4 Story Theme 31](#_Toc413699961)

[6.5 Story Outline 32](#_Toc413699962)

[7 Level Requirements 32](#_Toc413699963)

[7.1 Level Diagram 32](#_Toc413699964)

[7.2 Asset Revelation Schedule 32](#_Toc413699965)

[7.3 Level Design Seeds 32](#_Toc413699966)

[8 Technical Specifications 32](#_Toc413699967)

[8.1 Game Mechanics 32](#_Toc413699968)

[8.2 Game Engine 32](#_Toc413699969)

[8.3 Naming Convention 32](#_Toc413699970)

[8.4 Platform and OS 33](#_Toc413699971)

[8.5 External Code 34](#_Toc413699972)

[8.6 Code Objects 34](#_Toc413699973)

[8.7 Control Loop 34](#_Toc413699974)

[8.8 Game Scripts 34](#_Toc413699975)

[8.9 Game Objects Architecture 36](#_Toc413699976)

[8.10 Data Flow 38](#_Toc413699977)

[8.11 Artificial Intelligence 38](#_Toc413699978)

[8.12 Hardware Architecture 38](#_Toc413699979)

[8.13 Controller Map 39](#_Toc413699980)

[9 Production Schedule 39](#_Toc413699981)

[9.1 Scope 39](#_Toc413699982)

[9.2 Scheduling 39](#_Toc413699983)

[9.3 Dependencies 39](#_Toc413699984)

[9.4 Cost Estimate 39](#_Toc413699985)

[10 Use Cases 39](#_Toc413699986)

[10.1 Actors 39](#_Toc413699987)

[10.2 List of Use Cases 40](#_Toc413699988)

[10.3 Use Case Diagram 41](#_Toc413699989)

[10.4 Use Cases 41](#_Toc413699990)

[10.5 Sequence Diagrams 51](#_Toc413699991)

[10.6 Class Diagrams 52](#_Toc413699992)

[11 References 52](#_Toc413699993)

[12 Document Revision History 53](#_Toc413699994)

[13 Appendix 53](#_Toc413699995)

1. Game Description
   1. Design Goals

The goals this game aims to achieve for its target audience is to provide an interactive, immersive, and entertaining experience for the player to enjoy. This standalone PC game will be available within seconds whenever the customer wants to play it. The main focus is to draw the player in with an awe-inspiring world, stimulating combat, and a rich story. The game will provide for the user a glimpse into a fantasy world where civilization has been established among people who unknowingly live within a massive creature.

* 1. Influences and Sources

A team of interdisciplinary students attending the College of Creative Studies were the original creators of the games artwork and lore. They came up with an idea for an RPG where the main protagonist explores different imaginative environments to discover that he and his people live within this immense organism. They created multitudes of artwork, ranging from sketched concept art, to video and animations, and also 3D models. They created the models using tools such as Zbrush and Maya. These artwork assets were then handed to the programmers to be implemented within Unity.

* 1. Target Audience

Our target audience for the game are gamers who specifically play action-adventure roleplaying games. This audience makes up 13.9% of the entire U.S. gaming market which consists of 211 million people. This becomes 20 million plus customers within the United States alone. While the age of the average gamer lies at 31, the age of the average RPG players lies at 26. (1)

* 1. Target Customer

Our target customer is a male or female whose age fall within the range of 22-30 with the average age being 26. This customer takes many precautions before making the decision to purchase a new game. Being so integrated with the digital age, our target customer has a tendency to retrieve information on potential purchases through blogs and other online media outlets. Much of their decision is centered on reviews of the games by others who have played it. When our customer chooses to purchase a game, they expect three criteria to be met:

1. The ability to customize the character they interact with to some extent.
2. The freedom to explore the setting in which they are placed.
3. Last and most importantly, a strong storyline with characters they can empathize and connect to. This customer enjoys the option of choice and takes pride in the characters they create. The power to choose what their character wears, says, and does, places them in a position where they gain a sense of vested interest in the character. (1)
4. Functional Specifications

This section will go over our game mechanics.

* 1. Core Game Play

Explanation of what our user will experience during the game and how.

* 1. Game Flow

Explanation of how our game will flow from launch 🡪 exit.

* 1. Game Flow Diagram

Figure 7.1: Flow of Game and Data

* 1. Characters

1. TheHero
   1. The main character who is controlled by the player, he is an emissary who was created by the massive creature so that, through you, it could communicate with the cities of people who live now within it.
2. Friendly Old Man NPC
   1. Interacts with the main hero through a short dialogue, providing the player information about the island, how to fight enemies, and how to collect/use potions and essences.
3. Enemy Insect NPC
   1. The main antagonist in the game, a simple minded but fierce insectoid. They are the main combatants in the game for the player to interact and fight with. They will have small patrol routes and react to the player walking within a certain proximity before they aggress onto the player.
   2. Game Play Elements
4. Health
   1. A simple health points counter which will determine whether the player has taken enough damage to reach the threshold for in-game death. This will be displayed as a sliding red bar on the UI, and the player will start with 200 health points. A health count greater than zero will allow the player to maintain control over the hero. Conversely, once it reaches zero, the hero will play a death animation and the player will be directed towards a continue screen.
5. Stamina
   1. Another player counter visually represented by a (green) bar on the UI. The hero will start with a maximum 150 stamina points, and will always recover 10 pts/sec (5 if actively blocking). If stamina reaches zero, then there will be a two second delay before recharging resumes. If the player’s stamina bar is depleted, their movement actions will be limited to the standard walk.

The player’s stamina count will be deprecated by each of the following player actions:

* + - 1. Sprinting: 15 pts/sec
      2. Attacking: 30 pts/ea.
      3. Blocking: Equivalent to incoming attack
      4. Rolling: 30 pts/ea.
      5. Sidestepping: 15 pts/ea.

1. Essence
   1. Small pick-ups which will automatically fly to the player then disappear upon collision when the player is within a short proximity. Each essence will increase the counter on the UI by one, and after a pre-determined threshold is met, a special attack will be unlocked, and available for the player to perform. ¹
2. Potion
   1. Collectable items found throughout the game world which can be consumed by the player to recover health and stamina on-demand. Potions will have a dedicated animation where the player takes a drink whenever the player decides to use one. Using a potion will restore 80 health points, and 70 stamina points. They will not be able to restore points over the pre-defined maximums.
3. TreasureChest
   1. An interactive model in-game which the player can approach and use an action key to open it, and collect five potions from within.
4. Boulder
   1. An interactive game-world object which will be released after solving the puzzle. It can roll around the island, and damage enemies who come into contact with it.
5. Puzzle
   1. There will be a puzzle intended to briefly vary the style of gameplay, requesting the player to step on three different raised tiles in the correct order. When a tile is stepped on in the right order, it will remain lowered. Once the correct order has been inputted, the boulder will be released. Upon stepping on a tile in the wrong order (whether it be the first or last), the tiles will all raise and reset to their original positions. The purpose of which is to instantly tell the player something went wrong due to that specific action, and to encourage them to try again.

¹ This functionality is currently a stretch-goal for the demo

* 1. Game Physics and Statistics

The in-game physics is controlled by Unity’s built-in physics engine. We first create objects (3D model, light source, etc.), then assign components to those objects to add basic functionality and behaviors to that object. Essentially, components are the “nuts and bolts”, or inner workings, which we manipulate to change our objects behavior. The rigidbody component, for example, adds a built-in functionality for an object to react to basic physics (e.g. mass, drag, gravity). These new properties can be adjusted during development, and also harnessed by scripts to be changed during run-time. Along with the rigidbody component, other core components which are built-in to the Unity engine are: colliders, ray casting, joints, force, and torque.

* 1. Artificial Intelligence

Explanation of how our AI will work for all three types of our enemies. Aggro range, deaggro range…

1. User Interface and Menus

Similar to how a vehicle has a dashboard that shows you the critical, need to know information about the status of your vehicle while driving, the game must provide the user with a simple and intuitive user interface and heads-up-display as well as an easy to navigate game menu.

* 1. Main Game Menu

Upon loading the game from the executable, the player will be presented with a screen containing the Capstone Team’s logo. The game will be loading for the duration of this on-screen graphic and will not move to the main menu until this has completed. After successful load of the game data, the player will then be taken to the game’s main menu screen.

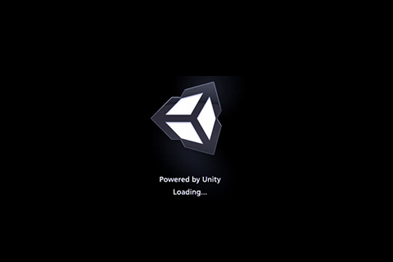
Menu Items:

1. New Game
   1. By selecting this the player will launch a fresh instance of the game and will be brought immediately to the game world.
2. Options
   1. Change Resolution
      1. From this selection the player will be able to change the resolution from what is currently selected to something that better fits their computer monitor. The aspect ratio is limited to widescreen or the 16:9 aspect ratio.
   2. Adjust Brightness
      1. The player will be able to adjust how bright the game is from this option in case of low light or if they are sensitive to the bright lights in the game.
   3. Adjust Graphical Quality
      1. In the case that the player is unhappy with the game’s default video settings, they can select one from here that better fits their computer’s video output capabilities.
3. Control Scheme
   1. By selection this option the player will view an image of a keyboard depicting clearly what each key does in the game and how to control the player character from the game.
4. Credits
   1. From the main menu, the player can choose an option to see the list of names of all the people that have worked on the game.
5. Exit to Desktop
   1. In the case that the user does not want to start a game or change an option, they can choose this selection to return them to their desktop.

  
Figure 12.1: Main Menu Flow Diagram

* 1. Splash Screen

While the game is performing its initial load of data after the executable has been launched, the game player will be presented with a splash screen that displays the logo of the Unity Capstone team. This is to give the game player the impression that the game is still loading and has not hung or frozen at a blank screen.



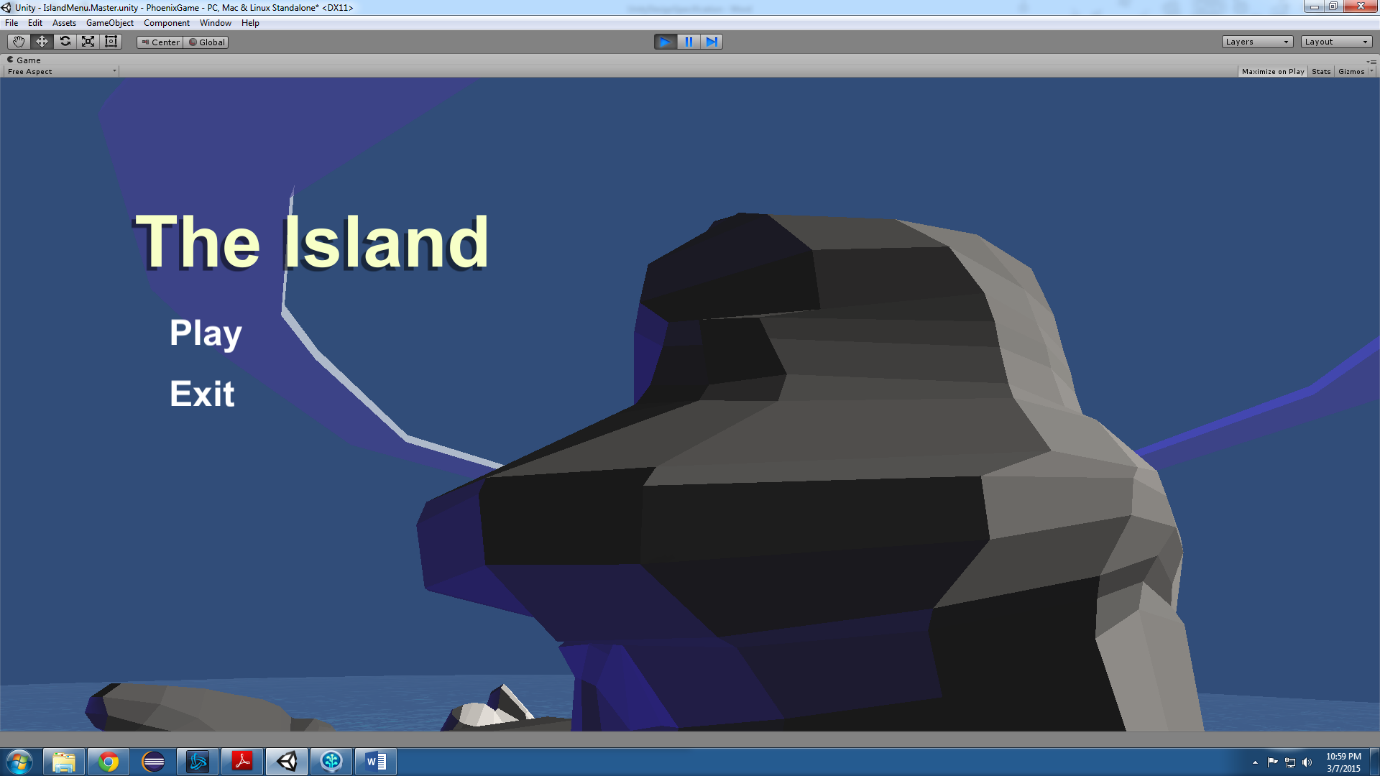
Unity’s Default Splash Screen

* 1. Game Menus

The Game Menus exist to give the game player easy and intuitive navigation through the game’s menu system. The game player will be able to select various options from both the main and in-game menu.

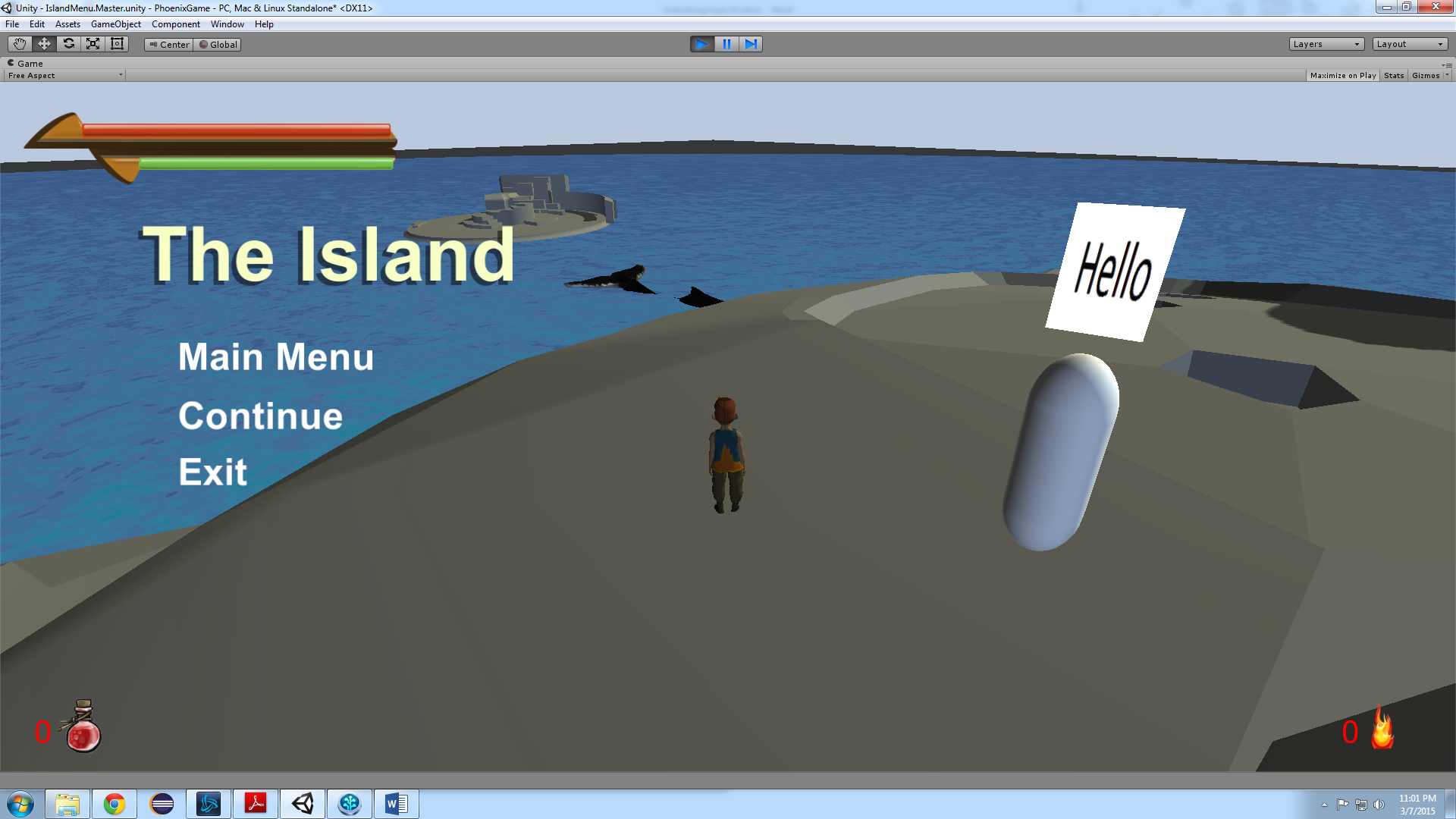
* + 1. New Game Menu

The first menu the player must interact with will preview the island environment in the background, and play dedicated menu music. It will have five buttons: new game, options, controls, credits, and exit. Hovering the mouse over the buttons will play a small sound so the user gets acknowledged feedback from the game, this is important as we begin to immerse the player in the game through interactivity.



*Figure 13.1: Snapshot of the current New Game menu (to be updated as development progresses).*

1. New Game
   1. Instantly begins a new game, changing the screen to the island setting where control of the hero is given to the player.
2. Options
   1. In the event that the player is unhappy with the graphical settings that are preset within the game, the options menu will be able to adjust the game to their preferences. There will be drop-down menus, and sliders to select options for brightness, graphics quality, and screen resolution.
      1. Brightness Options
         * Change ambient light to 70%
         * Change ambient light to 85%
         * Change ambient light to 115%
         * Change ambient light to 130%
      2. Resolution Options
      * 1920x1080 (16:9, default)
      * 1024x768
      1. Graphics Quality Options
      * Low
      * Good
      * Best
3. Controls
   1. Will generate an overlay image displaying a keyboard and mouse with lines connecting certain keys and mouse buttons to player controls. There will be a message noting to press the ESC key to exit to previous screen.
4. Credits
   1. The credits screen will display a canvas overlay showing the names of those involved in the making of this game. This includes the programmers, and all the artists involved in creating game assets (models, sounds, etc.).
5. Exit
   1. Pressing the exit button will close the game application, and return the player to their desktop or previously opened application.
      1. In-Game Menu

The second menu the player will interact with is the in-game pause menu. Activated by pressing the ESC key while in-game, this menu will display the menu’s overlay and freeze all game objects. The in-game menu will have the same menu buttons as the New Game menu. It serves to give the player the option to pause the game for a break and do other things, and be able to resume when they are ready.

*Figure 15.1: Snapshot of the current In-Game menu (to be updated as development progresses).*

* + 1. Exit Confirmation Menu

The in-game menu is nearly identical to the new game menu. The only difference is that the exit menu will have a new sub-menu. This sub-menu is our third, and final menu. Its purpose is to confirm with the player that they wish to exit the application, this will prevent losing progress due to accidentally clicking the exit button.

*Figure 16.1: Snapshot of the current Confirm Exit menu (to be updated as development progresses).*

* + 1. Heads-Up Display/User Interface

The Unity Capstone game utilizes numerous display interfaces such as the Main Menu listed above. The heads-up-display is a persistent representation of the player character’s health, inventory, and other need-to-know details. This is to allow the game player to always have knowledge of the current status of the in-game character.

1. Health and Stamina Bars
   1. The health bar is a visual representation of the player’s remaining health. As the count decreases towards zero, the bar will slide left behind a mask component attached to the background UI asset. When the hero’s health reaches zero, the red bar should be completely hidden from the player.
   2. The stamina bar has the same principles and responsibilities to the player as the health bar indicator. This counter is indicated by a green bar slightly shorter, and just under that of the health bar on the top-left corner of the HUD.
2. Essence Counter
   1. The essence counter is represented by a flame icon in the bottom-right portion of the screen. The number begins at zero and will be incremented by one every time the player comes into contact with a dropped essence from an enemy.
3. Potion Counter
   1. The potion counter is a temporary feature to be replaced by the quick access menu. However, in the future it will still be displayed at the top of the quick access menu. It is represented by a potion icon and a small number to tell the player how many potions are remaining. The counter is decremented by one every time the player presses the use potion key.
4. Quick Access Menu
   1. The quick access menu will display the number of remaining potions, as well as icons representing the currently equipped sword and shield. The bottom slot will be the special attack indicator, which is unlocked from collecting enough essences from enemies.¹²

¹ This functionality is currently a stretch-goal for the demo.

² For the developing prototype, the quick access menu will replace the potion indicator.

1. Art and Video

The artistic styling of any video game is always the first thing that an individual notices about it. In the case that the game performs well technically but fails to succeed from a visual standpoint, the game’s quality is severely reduced. Though the Unity Capstone students are not designers, there are a number of visual and artistic things that we are responsible for implementing and overseeing.

* 1. Overall Artistic Goals

The game will be produced in a “Low Poly” style. This translates to low polygons. These polygons determine how much detail will be sculpted into the 3d models for the game’s assets. The 3d models being referred to will be created using Zbrush and Maya (industry standard development software). Rather than raising the polygon count on these models, our team will focus on making our models interesting through the use of colors and textures. Lowering the amount of these polygons per model will decrease the overall time necessary to develop the game, which in turn cuts down on production costs.

To really make the models stand out, a major aspect of our production will focus on the lighting and coloring of the assets found within the game. This approach to production will take the game towards a stylized direction. This in turn will place more emphasis on the game’s storytelling and interaction. (1)

* 1. 3D Art and Animations

After the three-dimensional models have been textured, they will then be sent over to the animators for rigging and animation. Rigging is the process in which the models are given joints that enable animators to move the models for the task of animation. The rigging and animations will be created within the animation program known as Maya. (1) Some pictures of our raw 3D models and animations can go here as well.

  
3D Render of Player Character Concept

* 1. GUI

The graphical user interface displays core character information that the player needs to be aware of during gameplay. Values such as current health points and stamina, as well as remaining health potions must be taken into account by the player for precise and deliberate game play. Having a user interface which is automated to instantly respond to the current game situation is an extremely useful tool for the player.

1. Health and Stamina Bars
   1. The current health and stamina values for the player’s character will be displayed to the user by a red and green bar, respectively. These bars will be stacked on top of a small background overlay. When the player’s health or stamina value reaches zero, the corresponding bar should not be visible to the user.

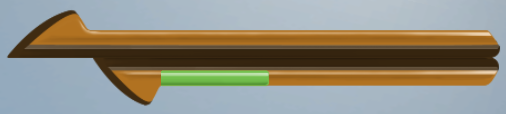
*Figure 17.1: Depiction of health and stamina bars at maximum count.*



*Figure 17.2: Depiction of health and stamina bars somewhat depleted.*

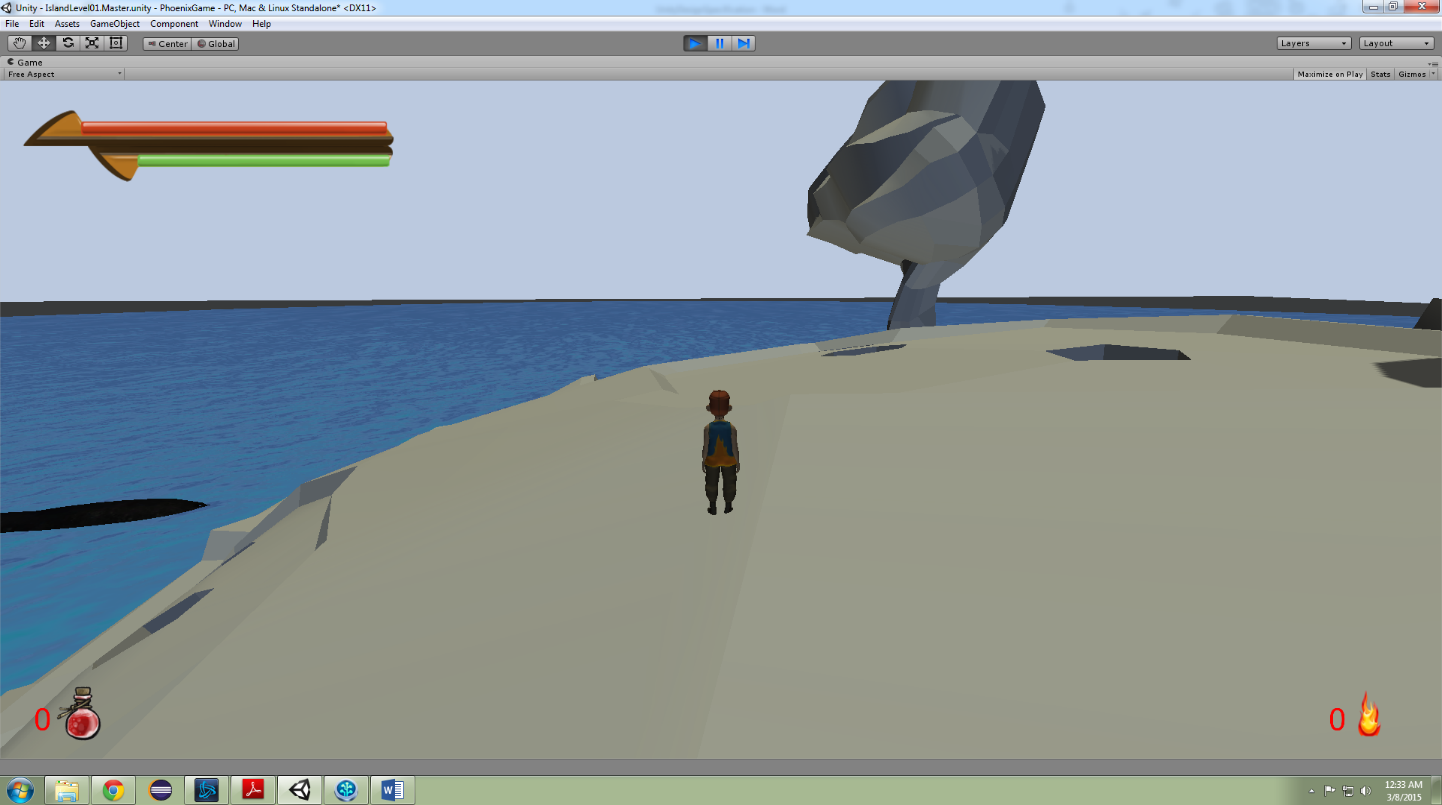
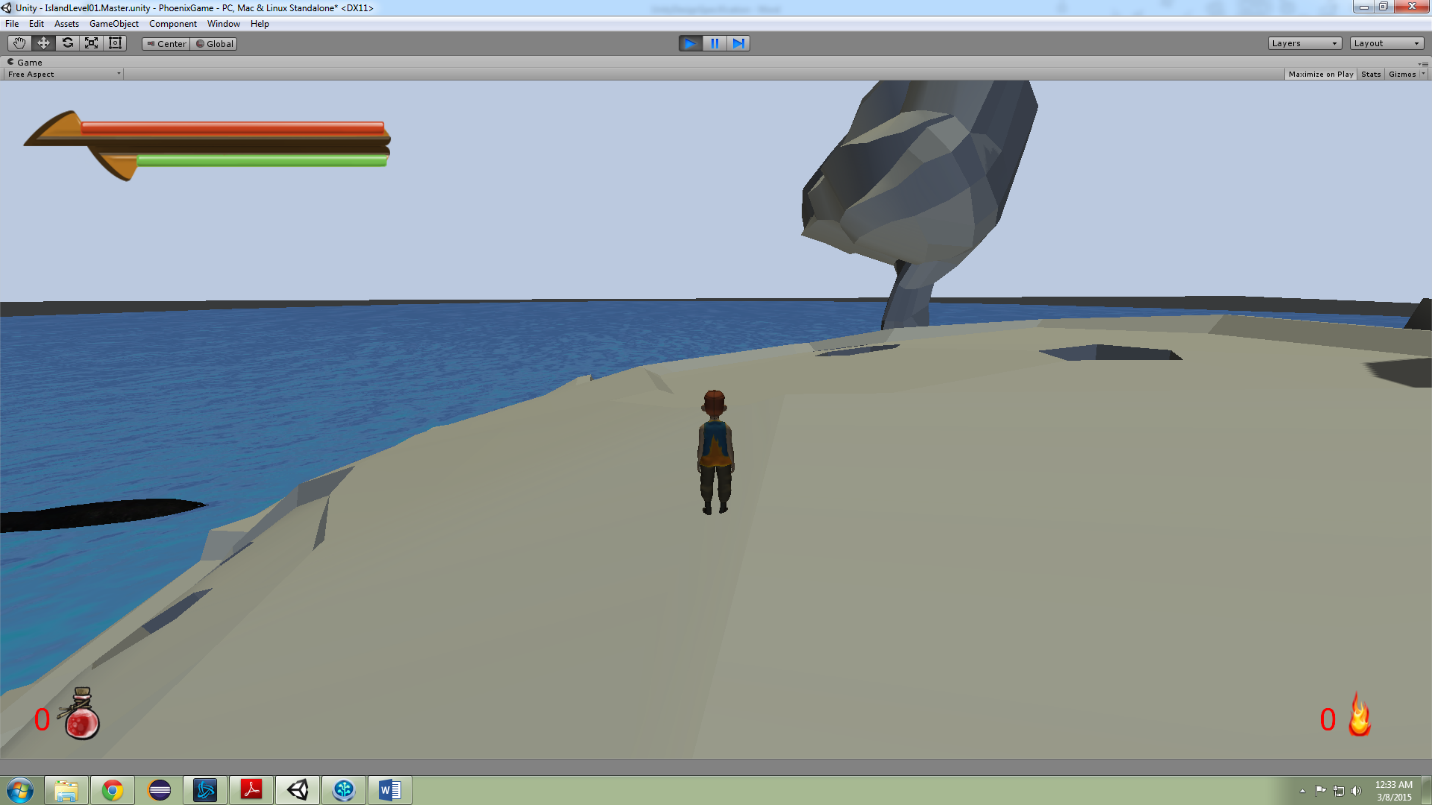


*Figure 17.3 Depiction of health and stamina bars completely empty.*



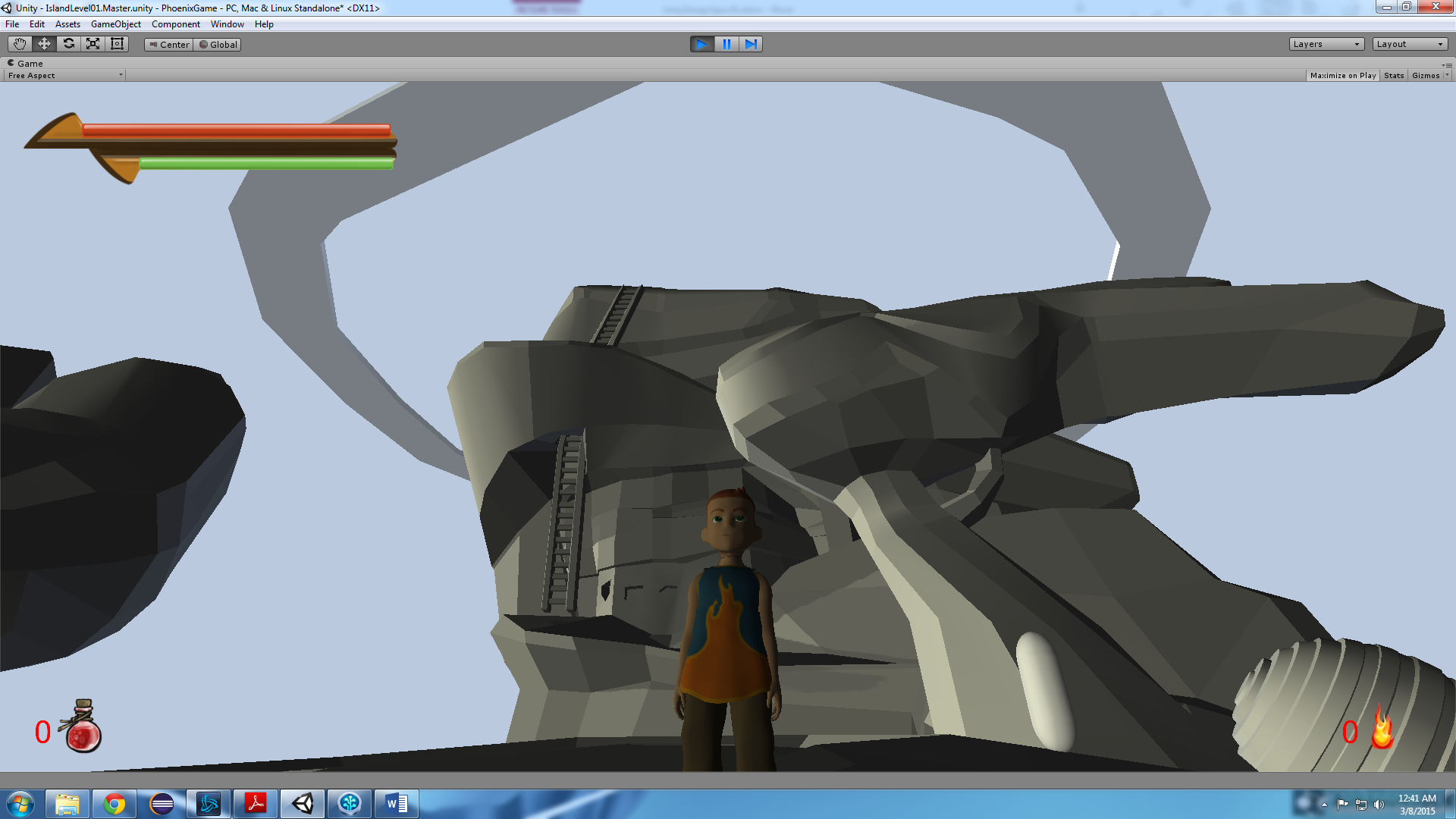
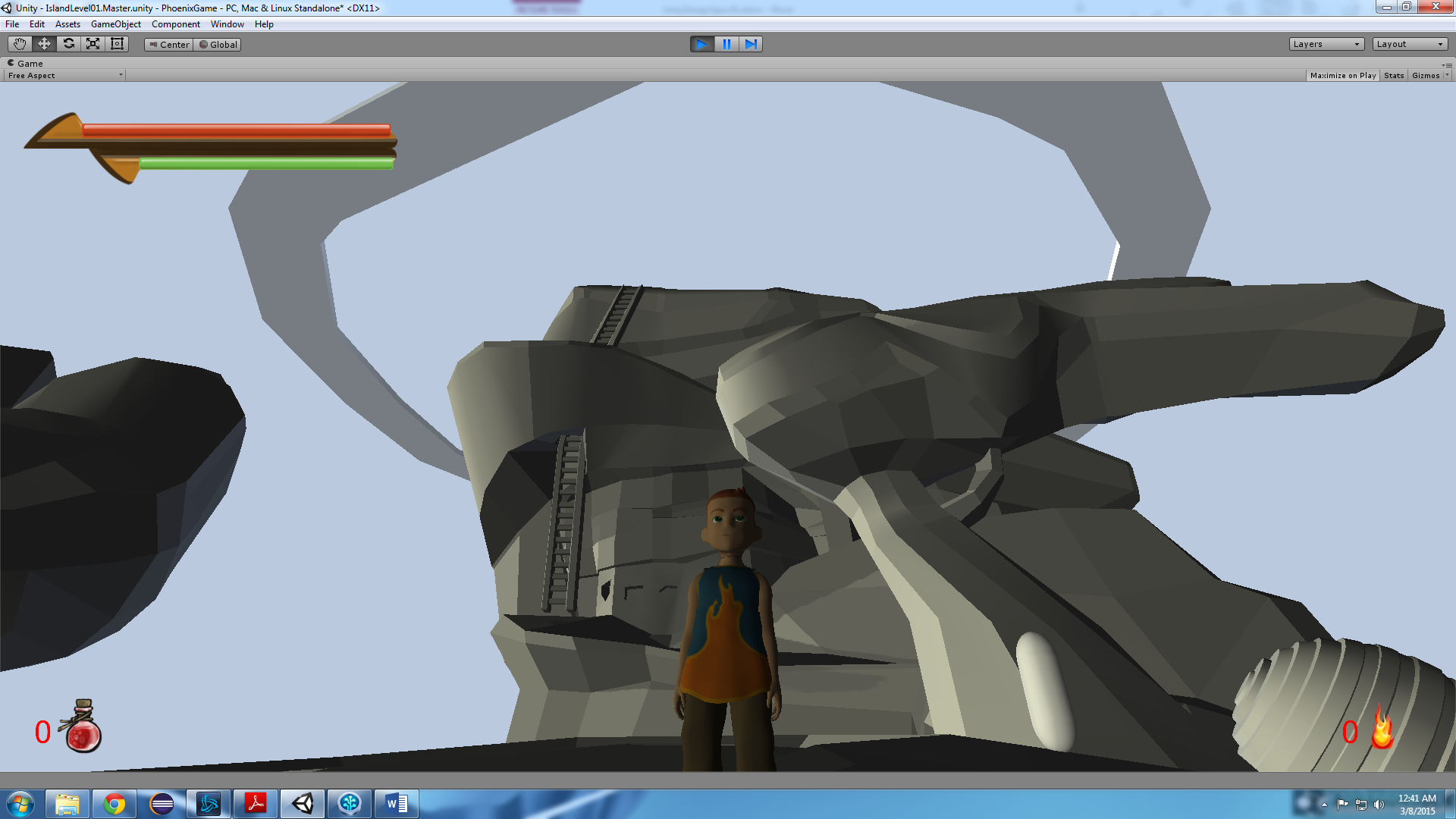
*Figure 17.4 Depiction of health bar at zero, but stamina bar greater than zero.*

1. Essence Counter
   1. The essence counter is represented by the flame icon and number counter in the bottom-right corner of the HUD. The number will update instantaneously when the player comes in contact with an essence.

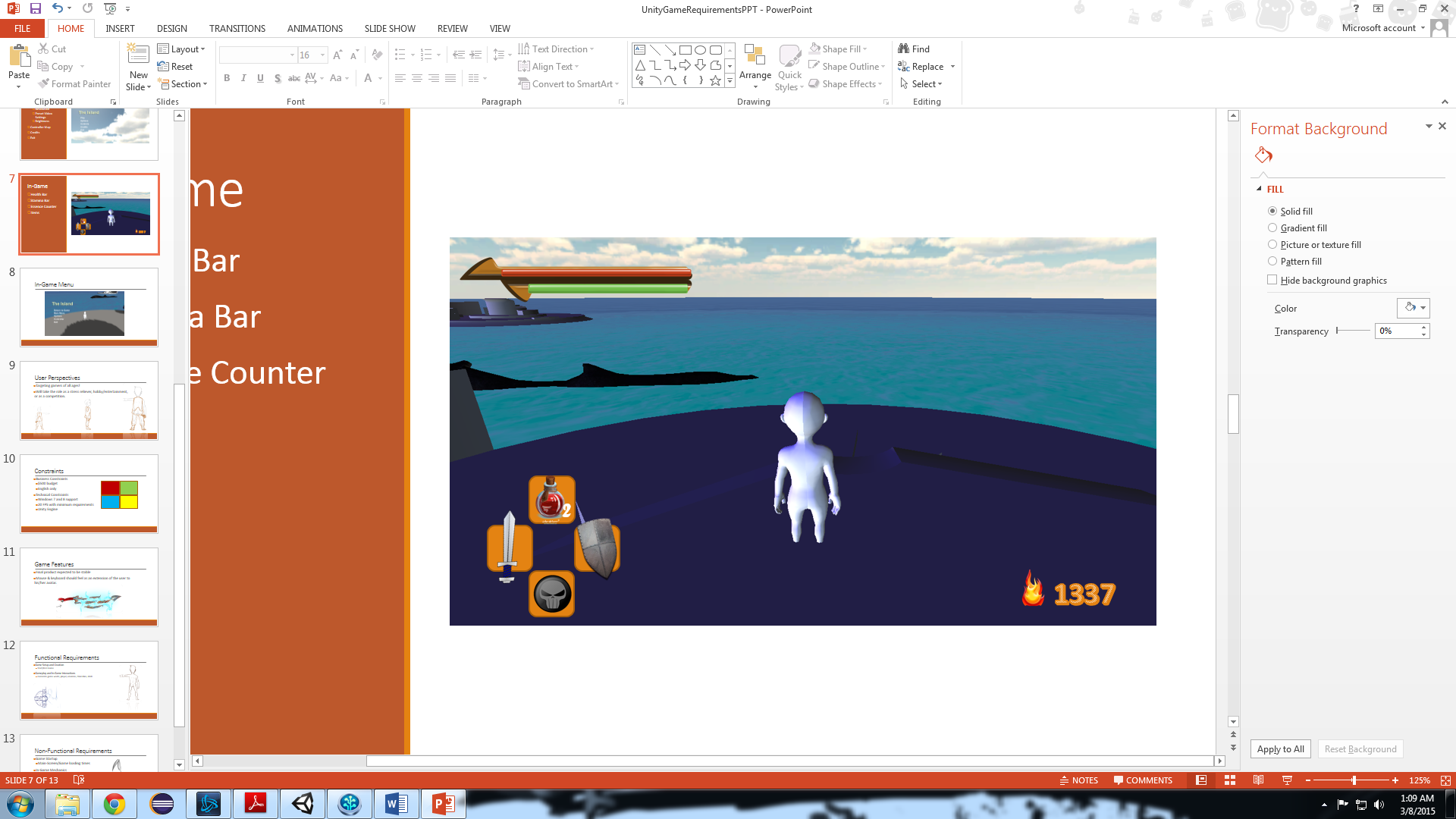


*Figure 18: Depiction of essence counter on the prototype user interface.*

1. Potion Counter
   1. The potion counter is represented by a red potion icon and a number to describe how many potions are remaining for the player to consume. This can be seen in the bottom-left corner of the HUD.



*Figure 19: Depiction of potion counter on the prototype user interface.*

1. Quick Access Menu
   1. The quick access menu is a small cross-shape of boxes which display the current weapon/shield, number of potions, and the availability of the special hero attack. This HUD element will replace the potion counter display once the asset is acquired.
   2. Terrain

List of terrain items categorized by interactive, background(skyboxes), undecided.

  
Figure 20.1: Terrain Concept Art

* 1. Game Play Elements

List Description of different game play elements, health pots, stamina…

* 1. Special Effects

Description of special effects in game, ie when the boulder goes crashing into the ground or character. When the enemy hits the player and vice versa.

* 1. Marketing and Packaging Art

The primary marketing strategy for the game will be centered on the creation of a strong demo. The demo will be a presentational tool, solidifying the core features of the game with consumers.

* 1. Assets Pipeline & Folder Structure

The diagram below is a representation of how the game assets will be organized in the Unity Capstone development environment. Items in yellow are children of the blue folder name above them.

  
Layout of Folder Structure (3)

1. Sound and Music

Music and sound effects for the game will be developed in parallel with the game art and assets. Sounds used within the game will all be custom made for the sole purpose of the game itself. Each sound and music clip’s intended use is for game enhancement and as such the player could still play the game without any audio if they so desired without compromising the quality of the gameplay.

* 1. Overall Goals

The overall goal of the Sound and Music of the game is to add to the level of immersion the player should feel while playing the game. Similar to a good movie, the audio in the game should lend to the sense that the player is actually in a real, living place. Music should swell and change during moments of tension, weapons should make noises when colliding with enemies, and the player should know when they have selected a specific option in the main menu.

* 1. Audio Sources

The below list exists to give a brief overview of the various elements in the game that are related to sound or will trigger the change of music. All audio sources can be attached to a specific game object and may be controlled by triggered events from the game scripts.

|  |  |  |  |
| --- | --- | --- | --- |
| Audio Type | Audio Purpose | Audio Description | File Name |
| Music | Main Menu Music | The looping audio clip to be played at all times when the game player is at the main menu screen. | audio\_MainMenuMusic |
| Music | Ambient Game Music | During the normal flow of the game, this will be the default music heard in the background. It will be an easy to listen to audio clip that loops continuously while the player is not actively involved in combat. | audio\_AmbientGame |
| Music | Tension/Action Music | Looping audio to be played whenever the game player is actively engaged in combat with the enemy character(s). | audio\_FightMusic |
| Sound Effect | Menu Move | Audio clip to be played whenever the game player hovers over a menu item with their mouse or cycles through them using the keyboard. | audio\_MenuMove |
| Sound Effect | Menu Selection | Audio clip to be played whenever the game player selects a menu option using the mouse or keyboard. | audio\_MenuSelection |
| Sound Effect | Player Footsteps | During regular movement, the player will have footstep noises that play during the movement animation. These footsteps can be specific to the terrain type, such as grass or sand. This audio will be changed based off the movement speed of the player character. There are four different footstep sounds. | audio\_FootstepA, audio\_FootstepB, audio\_FootstepC, audio\_FootstepD |
| Sound Effect | Player Attack Action | Audio clip to be played whenever the player character is making an attack with his sword. Will be similar to a grunt. | audio\_PlayerAttack |
| Sound Effect | Player Attack Collision | If the player’s sword comes into combat with an enemy or an in-game object, an audio cue will be played to signify the type of collision made. There will be one generic sound played for the sword colliding with the terrain & assets, and one sound for when the player’s sword successfully makes contact with an enemy. | audio\_CollisionEnemy, audio\_CollisionOther |
| Sound Effect | Enemy Movement | The enemy character in the game will use the same footstep audio as the player character. | audio\_FootstepA, audio\_FootstepB, audio\_FootstepC, audio\_FootstepD |
| Sound Effect | Enemy Activation | When the player character has come close enough to an enemy to trigger combat, an audio cue will play originating from the enemy character. This may loop as a continuous noise as a base representation of the sound the enemy makes at all times such as a buzzing. | audio\_EnemyNoise |
| Sound Effect | Essence Collection | Upon the player picking up an essence token dropped by the enemy character, a sound similar to a chime will be played. | audio\_EssenceCollect |
| Sound Effect | Drink Potion | When the player presses the button to drink a potion, a drinking or swallowing sound will be played. | audio\_PotionDrink |
| Sound Effect | Open Chest | The treasure chest in the game world will have a sound attached to it that will trigger when the chest is opened. It will be similar to a creaking door opening. | audio\_ChestOpen |
| Dialogue | Non-Player Character Speech | When the player comes into contact with the non-player character and interacts with him, an audio cue similar to a grunt will be played that represents what the non-player character may sound like. | audio\_NPCDialogue |
| Additional Audio | Ocean Sounds | When the player is near the water, an audio clip will loop the sound of the ocean. | audio\_OceanMusic |

* 1. Audio Diagram

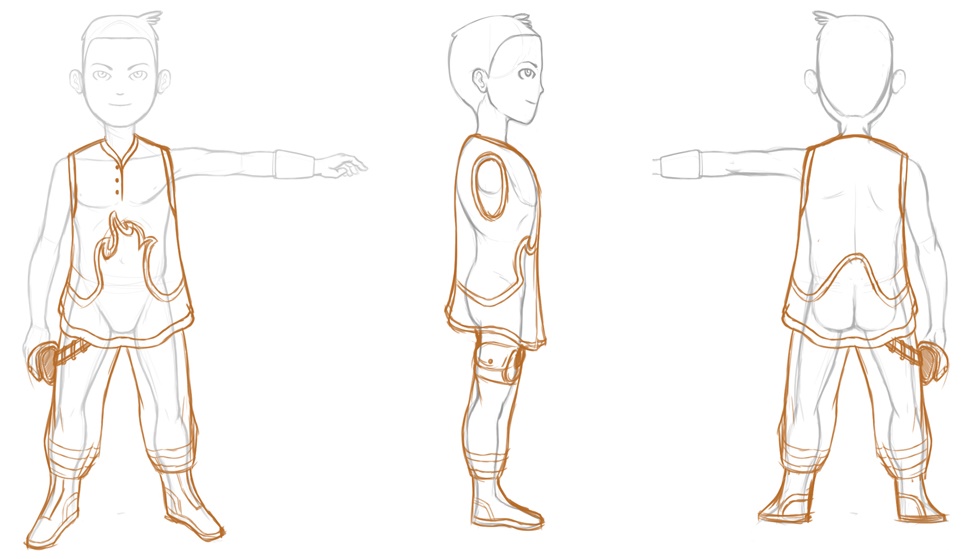
Below is a representation of how the various audio components of the game interact with each other and with the three character types.



1. Characters

The Unity Capstone game has three characters -- one playable and two non-playable. The enemy character will have 6 copies of itself in the game world, but each copy of the enemy character will have the same attributes and artificial intelligence. The game is played from the perspective of the player character who can interact with both the non-player character and the enemy characters in different ways.

* 1. Player Characters

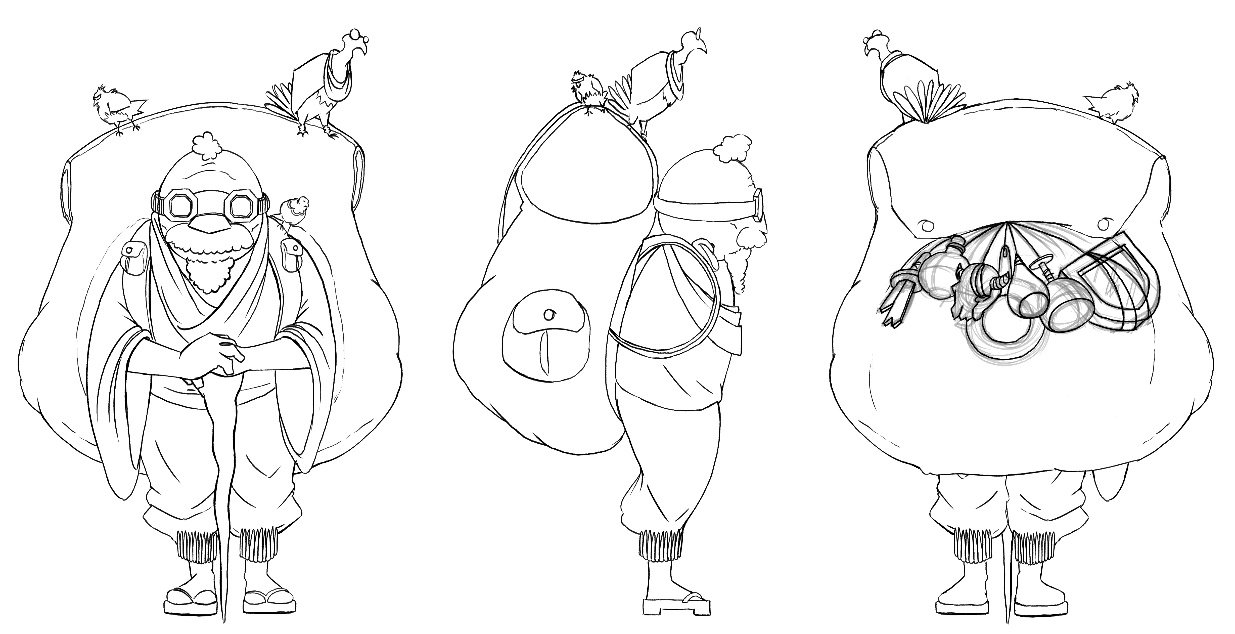
  
Figure 24.1: Player Character Concept Art

The Player Character, Jomz, is the protagonist and only playable character in the Unity Capstone game. Everything that Jomz does in the game is controlled by the game player through the use of the mouse and keyboard. The class diagram below illustrates each thing that the player character is able to do in the game environment.

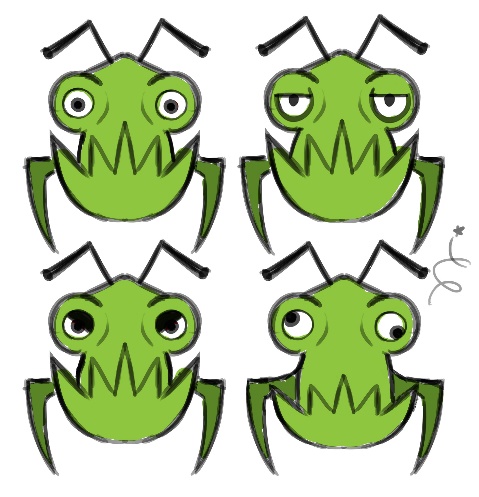
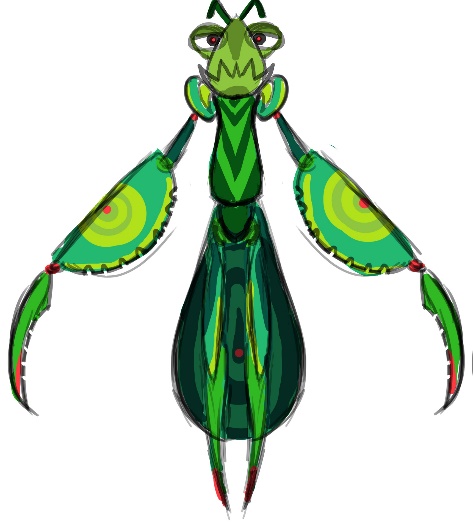


Player Class Diagram

* 1. Secondary Characters

  
Figure 24.2: Non-Player Character Concept Art

* 1. Enemy Characters

  
Figure 25.1: Concept Art of Enemy Character

* 1. Story Theme

Countless years have passed since the first humans took refuge within the creature. Little is known of the outside world, few speak of it, and most ensure that there is no such place. This immense organism, a being, godlike in size, seems transient to the humans. It wishes to connect with those who have taken refuge within its body. It needs a way to personally interact with these humans. It needs you. You, who embodies the essence of this creature. You, who now stands at a crossroads between two worlds. You, who now has the power to change the world. A great struggle lies ahead, but fear not, for you will never walk alone. (1)

* 1. Story Outline

1. Level Requirements
   1. Level Diagram
   2. Asset Revelation Schedule
   3. Level Design Seeds
2. Technical Specifications
   1. Game Mechanics
   2. Game Engine
   3. Naming Convention

The Unity Project must remain consistent with a uniformly used and closely monitored naming convention for all game objects and assets.

|  |  |  |
| --- | --- | --- |
| Identifier Type | Rules for Naming | Examples |
| Player Object | All assets in the game that refer to the player character object must be identified as player\_PlayerDetail, where PlayerDetail explains the purpose of the particular player object. There is typically only one player object per scene. If there are other copies of the player character in one scene, they are to refer to the player object's current location in the game world. | player\_MainPlayer, player\_UnderwaterPlayer |
| Physical Environmental Assets | Objects and assets in the game that are a part of the active game world, meaning that the player can collide with or come into contact and interact with these objects, will be identified by env\_ObjectName. In this case, ObjectName is a succint description of the type of object this asset represents. | env\_Island, env\_RollingBoulder, env\_Water |
| Enemy Characters | Enemies that have been added into the prefab folder will be identified as enemy\_EnemyType\_Number where EnemyType refers to the type of Enemy it represents and Number is identifying which particular enemy this is in the game world. Since there may be multiple copies of a particular enemy type, this will alleviate confusion with multiple enemies being placed on the island. | enemy\_EasyEnemy\_1, enemy\_Boss\_2, enemy\_MediumEnemy\_7 |
| Lighting Elements | Elements and objects that give light to the game world will be named light\_LightType, where LightType refers to the type or light or purpose that the light has in the game world. There are many different light sources in the game that serve various purposes based on time of day, so there will be several different lights placed in the playable game world. | light\_Sun, light\_TreasureChest, light\_BossSpotlight |
| Camera Objects | Instances of the camera in game should be named as camera\_CameraType where CameraType refers to whate the particular camera object does. These camera objects will typically be children of a player object. | camera\_ThirdPerson, camera\_BossFocus, camera\_FirstPerson |
| Audio Objects | Any game audio sources should be named audio\_AudioDescription where AudioDescription is a succinct description of what the audio source represents. | audio\_AmbientMusic, audio\_EnemyActivate,  audio\_FootstepsA |
| Heads Up Display Elements | The static, on-screen heads-up-display elements as well as any pop-up text boxes or messages shall be named as hud\_HUDType where HUDType is a brief description of what this particular heads-up-display element achieves in game. | hud\_HealthBar, hud\_StaminaBar, hud\_EssenceCounter |

* 1. Platform and OS

The Unity Capstone game will be released as a “Standalone” game for the Windows 7 and Windows 8 Operating System. It will be compiled using Unity’s Mono compiler for the script files and source code and presented as an executable file to the game player after installation is complete. The game will have a splash image for the loading screen as mentioned in section 3.1 as well as an icon for the executable shortcut.

Minimum Requirements:

-Intel HD Graphics 4000 Series

-4GB RAM

-Intel i3 Processor

-2 Gigabytes of Hard Disk Storage

* 1. External Code
  2. Code Objects
  3. Game Loop

The Game Loop is where the game player will be spending all of their time outside the menu screens. This loop exists to handle all of the main game logic and display the outcome to the screen for the game player to see. This loop only represents the active game session, not the menu screens.



Main Game Loop Diagram

* 1. Control Loop
  2. Game Scripts

Each player and non-player object will need a script in order for any action to take place. Along with any objects that are interactive in anyway such as, Item Container, Potion, Weapons, Ladder, and even particles.

|  |  |  |  |
| --- | --- | --- | --- |
| Game Object | Attached Scripts | Description | Public variables |
| Player | Potion | Controls the animations and the health bar when a potion is used. | -potion animation  -health gain |
| Container | Controls the animation and item received when a container is opened. | -item inside  -open animation |
| Attack | Controls the player model, particles, stamina bar, and damage dealt when the player attacks. | -attack animation  -speed  -damage  -range |
| Block | Controls the player model, particles, stamina bar, and damage blocked when the player blocks. | -stamina value  -block amount  -block animation |
| Health | Controls the health bar when player is dealt damage an enemy or the environment. | -Total health  -health slider  -player death |
| Movement | Controls the player model and corresponding animations. | -Speed  -turn speed  -mouse turn speed |
| Climb | Controls the player model and animation when climbing objects. | -Speed  -climb animation |
| Interact | Controls the player model and animation when interacting with game objects. | -range  -interaction animations |
| Stamina | Controls the stamina bar when player is using any stamina reducing abilities. | -total stamina  -stamina slider  -exhaustion animation |
| Collect | Controls the essence model and animation when player “collects” essences. | -Speed |
| Enemy | Movement | Controls the enemy model and corresponding animations. | -Speed |
| Aggro | Controls the ‘aggro’ range and ‘deaggro’ range of an enemy. | -Aggro range  -De-aggro range |
| Attack | Controls the enemy model, particles, and damage dealt when the enemy attacks. | -attack animation  -speed  -damage  -range |
| Health | Controls the health of the enemy when damage is dealt by the player or environment. | -total health  -death clip  -essence value |
| Menu | In-Game Menu | Controls the in-game menu and all submenus. | -return  -main  -options  -controls  -exit |
| Main Menu | Controls the main menu and all submenus. | -start  -options  -controls  -credits  -exit |
| HUD | Quick Access | Controls what items are displayed in the quick access menu |  |
| Game Over Manager | Controls the ‘Game Over’ screen as the player dies. | -player health  -restart timer |
| Characters | Enemy Manager | Controls spawn points and spawn rates of enemies. | -enemies  -spawn locations  -spawn speed |
| NPC Manager | Controls the friendly NPC. | -chat bubble time  -chat bubble |
| Environment | Day-Night Manager | Controls the rotating light and skyboxes. | -Skyboxes  -rotation speed  -day length |
| Music Manager | Controls what music to play depending on what is occurring in the game. | -Music tracks  -volume |
| Camera | Close Fade | Eliminate character meshes when camera is too close | -Character Meshes  -fade distance  -hide distance |
| Camera Follow | Will follow the character appropriately | -ground check  -rotation speed  -distance update speed  -max angle  -max distance |

* 1. Game Objects Architecture

Every object in the game is considered to be a GameObject in Unity. GameObjects need special properties before they become characters, environments, or special effects. In short, GameObjects are containers that can hold different pieces called components. (2) All game objects have a Name, Tag, Layer, and Transform Component. The name is simply what we call the GameObject. A tag is a word that is linked to one or more GameObjects to make them easier to sort and find as well as being able to group GameObjects as a certain type. A layer is used by the camera to render specific parts of the GameObject and can also be used in testing for collision. The transform is the physical location of the GameObject in the game world as well as its rotation and scale on the X, Y, and Z axes. Lastly, a GameObject can have any number of attached components, such as an Audio Listener, a Physics RigidBody, a Collider, and a Camera.

  
Figure 31.1: The Sun GameObject, Component, and Script

When a component is created, there may be one or more scripts attached to it. A script is a specific type of Component that the user is creating and giving functionality to. Once a script is attached to a component it will begin working when the game runs. See below a visualization of the GameObject – Component – Script architecture and relationship.

Each GameObject in the game will have these Components and Scripts and can be tweaked and changed during the game’s development.

  
Visualization of the GameObject and its Components

* 1. Data Flow
  2. Artificial Intelligence

Enemies will spawn from set locations across the map. Each enemy will *patrol* a specific area using *waypoints* directing their path. Each enemy has a set vision distance. As a player approaches this distance range, the enemy will then become engaged with the player. At this point the enemy will follow the player until the player is either *dead* or is out of the enemies *return range,* in either case the enemy will return to following the designated waypoints path. Please refer to *Stat Sheet* for updated values.

visionRange: 15m

returnRange: 40m

* 1. Software Architecture

The Unity Capstone game makes use of a modified Model – View – Controller software architecture. In short, there is a separation between the interface and the game logic. Each input controller will only give commands to their appropriate components to let them know that the controller has been invoked.

Each component only maintains data and processes its own state. In short, the graphical user interface should only serve to display the game state data that is managed from the controllers.

Lastly, gameplay objects know virtually nothing about what is happening on the graphical user interface. If we were to delete all of the GUI classes, the game should still be able to compile and we could in turn re-implement the GUI and input without requiring any new game logic.(3)



Software Architecture Diagram

* 1. Hardware Architecture

The Unity Capstone game will be run as an executable (.exe) file from somewhere on the player’s local machine or an external hard drive device. The game must be installed via the game installer or have the files extracted to this location. All files will be stored and hosted locally on the user’s machine as the game requires no online-connectivity and will not be connected to any type of server or online database. All game interactions will be made through the computer or laptop’s available mouse and keyboard.



Figure 32.1: Hardware Architecture Diagram

* 1. Controller Map

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Action | Key | Description |
| Movement | Forward up | W | Player runs north with forward mechanic |
|  | Forward Left | A | Player runs west with forward mechanic |
|  | Forward Right | D | Player runs east with forward mechanic |
|  | Forward Back | S | Player runs south with forward mechanic |
|  | Turn left | \*Mouse | Player turns left |
|  | Turn right | \*Mouse | Player turns right |
|  | Sidestep left | Q | Player dashes to the left (\*Stamina) |
|  | Sidestep right | E | Player dashes to the right (\*Stamina) |
|  | Jump | Spacebar | Player jumps (\*Stamina) |
|  | Roll | R | Player rolls/tumbles (\*Stamina) |
|  | Sprint | Shift | Player movement is increased (\*Stamina) |
| Combat | Attack | Left Click | Player swings sword (\*Stamina) |
|  | Block | Right Click | Player raises shield (\*Stamina) |
| Interaction | Use item | C | Player uses item |
|  | Interact | F | Player interacts with environment or other characters |
| Menu | Pause game | Escape | Pauses game and brings up menu options |

1. Production Schedule
   1. Scope
   2. Scheduling
   3. Dependencies
   4. Cost Estimate
2. Use Cases
   1. Actors

Though there are several possible user types for the Unity game, will can generalize all different user types into one category: the game player.

**Game Player –** any individual that is playing the Unity Capstone game project. The player could be a casual gamer, an older player, or any other combination of attributes. Though there are different reasons to play a game, you can only perform the specified actions in the game.

* 1. List of Use Cases

Case 1: Begin New Game

Case 2: Change the Game Options

Case 3: View the Game’s List of Credits

Case 4: View the Game’s Control Scheme

Case 5: Exit the Game and Return to the Desktop

Case 6: Pause the currently active Game Session

Case 7: Continue the currently paused Game Sessions

Case 8: View the In-Game Menu List

Case 9: Change the In-Game Options from the In-Game Menu

Case 10: Return from the currently active Game Session to the Main Menu

* 1. Use Case Diagram



Figure 34.1: Use-Case Diagram for Game Player

* 1. Use Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 1 | | | |
| **Use Case Name:** | Begin New Game | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of starting a new game session. | | |
| **Description:** | | This use case describes the request to begin a new instance of the Unity game. In this use case the actor’s overall goal is to start a new game from the main menu. | | |
| **Trigger:** | | When the actor selects the ‘New Game’ option from the main menu and begins a new game. | | |
| **Preconditions:** | | 1. The game player must have the game successfully installed. 2. The game must complete its initial load of data. 3. The game player must be at the main menu. 4. The game player must have access to a mouse and/or keyboard | | |
| **Postconditions:** | | The game player is taken from the Main Menu to the game. | | |
| **Normal Flow:** | | 1. Game player selects the ‘New Game’ option from the Main Menu. 2. The Main Menu controller script accepts request from the ‘New Game’ component. 3. Request is processed by the script. 4. The script controller loads the new game scene in place of the Main Menu scene. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if no scene data is found   1. Game logic will not be able to load the main game scene 2. Game player will stay at the Main Menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This use case is the most frequently used menu selection. It is triggered on-demand, not automatically. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player understands that choosing ‘New Game’ will take them to the game. 3. The game player can navigate the Main Menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 2 | | | |
| **Use Case Name:** | Change Game Options | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of changing the game’s available options. | | |
| **Description:** | | This use case describes the request to change the game’s options and settings from the Main Menu screen. The options menu is a sub-menu of the Main Menu where the game player can modify the brightness, screen resolution, and graphics quality of the game. | | |
| **Trigger:** | | When the actor selects the ‘Options’ menu object from the Main Menu. | | |
| **Preconditions:** | | 1. The game must complete its initial load of data. 2. The game player must be at the main menu. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard | | |
| **Postconditions:** | | The game player is taken from the Main Menu screen to the Options sub-menu | | |
| **Normal Flow:** | | 1. Game player selects the ‘Options’ button from the Main Menu. 2. The Main Menu controller script accepts request from the ‘Option’ game component. 3. Request is processed by the script/controller. 4. The script controller loads the Options sub-menu in place of the Main Menu screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if no Options data is found   1. Game logic will not be able to load the Options sub-menu 2. Game player will stay at the Main Menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | The user is expected to only use this case on an as-needed basis. The default screen resolution and graphics settings should be fine for the average player. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player understands that choosing ‘Options’ selection will take them from the Main Menu to the Options sub-menu. 3. The game player can navigate the Main Menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 3 | | | |
| **Use Case Name:** | View Game Credits | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of viewing the game’s credits. | | |
| **Description:** | | This use case describes the request to view the game’s credit screen by selecting the ‘Credits’ option from the Main Menu. This will load in a new screen, replacing the Main Menu with a list of all the individuals who have worked on the game. | | |
| **Trigger:** | | When the actor selects the ‘Credits’ menu object from the Main Menu. | | |
| **Preconditions:** | | 1. The game player must be at the main menu. 2. The game must complete its initial load of data. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard | | |
| **Postconditions:** | | The game player is taken from the Main Menu screen to the Credits screen. | | |
| **Normal Flow:** | | 1. The ‘Credits’ option is selected from the Main Menu 2. The Main Menu controller script accepts request from the ‘Credits’ game component. 3. Request is processed by the script/controller. 4. The script controller loads the credits screen in place of the Main Menu screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if no Credits data is found   1. Game logic will not be able to load the Credits scene 2. Game player will stay at the Main Menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This is likely going to be the least accessed use case as the game player will usually only choose this selection once. It is a very infrequently changing list that will be finalized at the end of the game’s development. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player understands that choosing ‘Credits’ selection will take them from the Main Menu to the Credits screen. 3. The game player can navigate the Main Menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 4 | | | |
| **Use Case Name:** | View Game Controls | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of viewing the game’s control scheme. | | |
| **Description:** | | This use case describes the request to view the game’s control scheme from the Main Menu. This is so the player knows exactly which buttons correspond with their in-game actions. | | |
| **Trigger:** | | When the actor selects the ‘View Controls’ menu object from the Main Menu. | | |
| **Preconditions:** | | 1. The game player must be at the main menu 2. The game must complete its initial load of data. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard | | |
| **Postconditions:** | | The game player is taken from the Main Menu screen to the Controls screen. | | |
| **Normal Flow:** | | 1. The ‘View Controls’ option is selected from the Main Menu 2. The Main Menu controller script accepts request from the ‘View Controls’ game component. 3. Request is processed by the script/controller. 4. The script controller loads the controls screen in place of the Main Menu screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if no Controls data is found   1. Game logic will not be able to load the Controls scene 2. Game player will stay at the Main Menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | Similar to the ‘Credits’ option, the player will likely only access this screen a handful of times at maximum. The game’s controls are similar to the controls found on other games of this type, so the player should already know how to play this game if they have played other games before. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player understands that choosing ‘View Controls’ selection will take them from the Main Menu to the controls screen. 3. The game player can navigate the Main Menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 5 | | | |
| **Use Case Name:** | Exit to Desktop | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of exiting the game from the Main Menu screen. | | |
| **Description:** | | This use case describes the request to exit the current game from the Main Menu screen and be returned to the desktop of their computer. | | |
| **Trigger:** | | When the actor selects the ‘Exit to Desktop’ menu object from the Main Menu. | | |
| **Preconditions:** | | 1. The game player must be at the main menu 2. The game must complete its initial load of data. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard | | |
| **Postconditions:** | | The game player is taken from the Main Menu screen to desktop of their computer. All game processes will be terminated. | | |
| **Normal Flow:** | | 1. The ‘Exit to Desktop’ option is selected from the Main Menu 2. The Main Menu controller script accepts request from the ‘Exit to Desktop’ game component. 3. Request is processed by the script/controller. 4. The script controller kills the current game session and all game-related processes are ended. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if the script does not execute the game’s termination protocol   1. Game logic will not be able to exit the current game 2. Game player will stay at the Main Menu 3. Game player will have to manually kill the game process | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This will be used every time that the game player wants to end their game session. So it will be used once per load of the game. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player understands that choosing ‘Exit to Desktop’ selection will take them from the Main Menu to desktop of their computer. 3. The game player can navigate the Main Menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 6 | | | |
| **Use Case Name:** | Pause Game | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of pausing the current game session. | | |
| **Description:** | | This use case describes the request to stop or pause the running game session. This could be to enter the in-game menu system, or if the game player needs to take a break from the active gameplay for any reason. | | |
| **Trigger:** | | When the actor presses the Escape key on their keyboard while the game is currently playing. | | |
| **Preconditions:** | | 1. The game player must be in an active game session. 2. The game must complete its initial load of data. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard. 5. The game player must not be in any menu screen. | | |
| **Postconditions:** | | The game player’s game session is frozen and the actor can now navigate through the in-game menu. | | |
| **Normal Flow:** | | 1. The Escape key on the keyboard is pressed sometime during the active game session. 2. The game controller script accepts request from the keyboard to halt the game. 3. Request is processed by the script/controller. 4. The script controller pauses the current game session and directs game player to in-game menu screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if the script does not execute the game’s pause method   1. Game logic will not be able to pause the current game 2. Game player will stay in the active game session 3. Game player will have to manually kill the game process to exit | | |
| **Exceptions:** | | 1. In step 1 of the Normal Flow, if the game player presses the Escape key from the menu it will just return them to the previous menu item they were at. | | |
| **Frequency of Use:** | | This will be used every time that the game player wants to pause their game session or be returned to the main menu to exit successfully. It will be a moderately frequently used case during the normal game play. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player can navigate a standard US keyboard. 3. The game player understands that pressing the Escape key will pause the game 4. The game player can navigate the in-game menu | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 7 | | | |
| **Use Case Name:** | Continue Game | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of unpausing or resuming the current game session. | | |
| **Description:** | | This use case describes the request to continue or resume the running game session. This operation will take the game player out of the in-game menu and back to the running game. | | |
| **Trigger:** | | When the actor presses the Escape key on their keyboard while the game is currently paused. | | |
| **Preconditions:** | | 1. The game must complete its initial load of data. 2. The game player must be in a paused state. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard. | | |
| **Postconditions:** | | The game player’s game session is unfrozen and resumed and the actor can now play the game. | | |
| **Normal Flow:** | | 1. The Escape key on the keyboard is pressed sometime during the paused game session. 2. The game controller script accepts request from the keyboard to resume the game. 3. Request is processed by the script/controller. 4. The script controller resumes the current game session and directs game player out of the in-game menu screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if the script does not execute the game’s resume function   1. Game logic will not be able to exit the in-game menu 2. Game player will stay on the in-game menu 3. Game player will have to manually kill the game process or return to the Main Menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This will be used every time that the game player wants to continue their currently paused or halted game. It will be used almost as many times as the pause use case. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player can navigate a standard US keyboard. 3. The game player understands that pressing the Escape key will continue the game if it is currently pause. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 8 | | | |
| **Use Case Name:** | View In-Game Menu | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of navigating through the in-game menu. | | |
| **Description:** | | This use case describes the request to navigate or interact with the in-game menu. The in-game menu is only available when the game is paused per Use Case 6. It is a reduced version of the Main Menu screen. | | |
| **Trigger:** | | When the actor presses the Escape key on their keyboard while the game is currently active. | | |
| **Preconditions:** | | 1. The game must complete its initial load of data. 2. The game player must be in a paused state. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard. | | |
| **Postconditions:** | | The game player can now visualize and interact with the in-game menu and its options. | | |
| **Normal Flow:** | | 1. The Escape key on the keyboard is pressed sometime during the active game session. 2. The game controller script accepts request from the keyboard to bring up the in-game menu. 3. Request is processed by the script/controller. 4. The script controller shows the game player the in-game menu and its items. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if the script does not execute the game’s in-game menu method   1. Game logic will not be able to display the in-game menu 2. Game player will stay in an active game session 3. Game player will have to manually kill the game process | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This will be used every time that the game player wants to pause or halt the current game. It will happen exactly as many times as Use Case 6. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player can navigate a standard US keyboard. 3. The game player understands that pressing an option from the in-game menu will execute the advertised operation. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 9 | | | |
| **Use Case Name:** | Change In-Game Options | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of changing the game options from the in-game menu. | | |
| **Description:** | | This use case describes the request to enter and interact with the in-game options menu. These options are the same as Use Case 2. | | |
| **Trigger:** | | When the actor selects the ‘Options’ menu button from the in-game menu. | | |
| **Preconditions:** | | 1. The game must complete its initial load of data 2. The game player must be in a paused state. 3. The game player must be at the in-game menu. 4. The game player must have the game successfully installed. 5. The game player must have access to a mouse and/or keyboard. | | |
| **Postconditions:** | | The game player can now change the options exactly as they could from Use Case 2. | | |
| **Normal Flow:** | | 1. The Escape key on the keyboard is pressed sometime during the active game session. 2. The game controller script accepts request from the keyboard to bring up the in-game menu. 3. Request is processed by the script/controller. 4. The script controller shows the game player the in-game menu and its items. 5. The game player selects the ‘Options’ button from the in-game menu | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 5a. In step 4 of the normal flow, ,if no Options data is found  1. Game logic will not be able to load the Options sub-menu  2. Game player will stay at the in-game menu | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This will be used every time that the game player wants to change the Options from the in-game menu instead of exiting out of the game to do it from the Main Menu. It will likely only be used at most several times during one game play session.. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player can navigate a standard US keyboard. 3. The game player understands that pressing an option from the in-game menu will execute the advertised operation. | | |

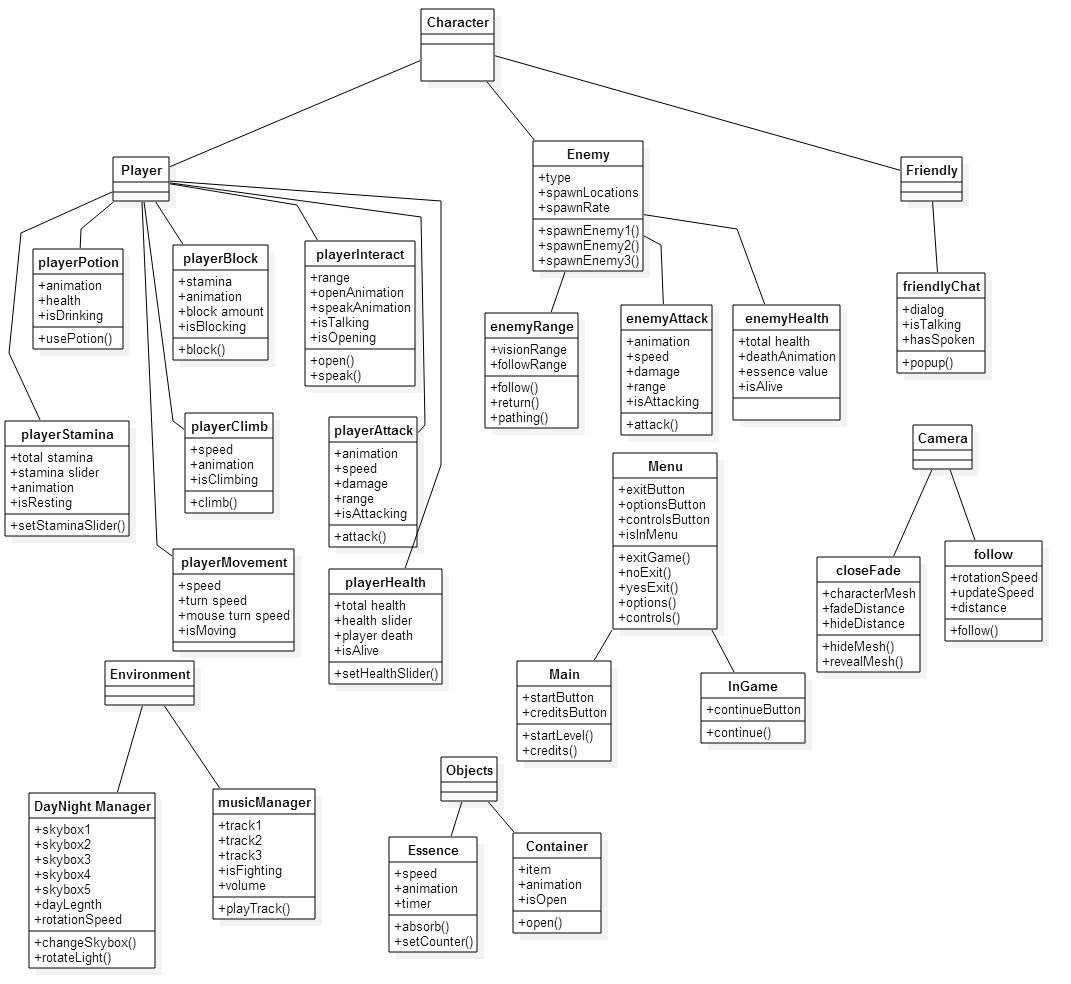
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Case 10 | | | |
| **Use Case Name:** | Return to Main Menu | | | |
| **Created By:** | Jonathan Nabors | | **Last Updated By:** |  |
| **Date Created:** | 3/8/2015 | | **Last Revision Date:** |  |
| **Actors:** | | The actor will be the game player with the current intention of navigating returning to the Main Menu from the in-game menu. | | |
| **Description:** | | This use case describes the request to be returned to the Main Menu from the in-game menu. From here, the game player will be able to execute any of the previously listed Use Cases that originate from the Main Menu. | | |
| **Trigger:** | | When the actor presses the ‘Main Menu’ button from the in-game menu. | | |
| **Preconditions:** | | 1. The game must complete its initial load of data. 2. The game player must be in a paused state. 3. The game player must have the game successfully installed. 4. The game player must have access to a mouse and/or keyboard. 5. The game player must be in the in-game menu | | |
| **Postconditions:** | | The game player can now execute above Use Cases from the Main Menu | | |
| **Normal Flow:** | | 1. The ‘Main Menu’ button is selected from the in-game menu. 2. The game controller script accepts request from the keyboard to end the session and return to the Main Menu scene. 3. Request is processed by the script/controller. 4. The script controller shows the game player Main Menu screen instead of the active game screen. | | |
| **Alternative Flows:**  **[Alternative Flow 1 – No Data Found]** | | 4a. In step 4 of the normal flow, if the script does not execute the game’s Main Menu scene   1. Game logic will not be able to display the Main Menu 2. Game player will stay in an active game session 3. Game player will have to manually kill the game process | | |
| **Exceptions:** | |  | | |
| **Frequency of Use:** | | This will likely be used once per game session as a means of quitting the game. | | |
| **Special Requirements:** | |  | | |
| **Assumptions:** | | 1. The game player can read and understand English. 2. The game player can navigate a standard US keyboard. 3. The game player understands choosing the ‘Main Menu’ option will return them to the Main Menu. | | |

* 1. Sequence Diagrams



Figure 44.1: Sequence Diagram

* 1. Class Diagrams



1. References

|  |  |  |  |
| --- | --- | --- | --- |
| Doc Number |  | Doc Version | Doc Name & Location |
|  |  |  |  |
|  |  |  |  |
| 3 |  | 1 | [Unity Best Practices](http://devmag.org.za/2012/07/12/50-tips-for-working-with-unity-best-practices/) |

1. Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Changes |
|  |  |  |  |
|  |  |  |  |

1. Appendix

Material including referenced documentation the web or elsewhere, as well as alternative designs or items/ideas for future improvements.