

DESCEND

By: Adele Wu

Truman Tang, Jesus Correa, Yarkin Gazi

Instructor: Dr. Ilmi Yoon
Spring 2022
CSC 631/831

Table of Contents

- 01. Abstract
- 02. Introduction
- 03. Game Components
 - a. Characters
 - i. Art
 - ii. Rigging
 - iii. Animation
 - b. Assets
 - c. Overworld
 - i. Scene Development
 - ii. NPCs
 - iii. Portals
 - d. Combat
 - i. Attacks
 - 1. Players
 - 2. Enemies
 - ii. HP / Health Points
 - e. User Interface
 - i. Screens and Menus
 - ii. Player Special Attack Cooldown Bar
 - f. Audio
- 04. My Contribution
- 05. Git Commits
- 06. Challenges and Lessons Learned
- 07. Second Chance

Abstract

Descend is a cartoony, 2D, Greek mythology-based, semi-open world, action-adventure game that game focuses on exploration, real-time combat, and strategy. Players can play as a warrior or an archer to explore the 2 worlds (overworld and underworld), meet NPCs, and defeat enemies, all while helping Zeus prevent Hades from taking over the overworld. The players operate within a large map, using the arrow keys and WASD keys to meet NPCs to drive the story and lead the user to combat. In the combat stages, there are a variety of enemies that try to prevent the user from reaching the final boss, and a level boss in each level. Defeating a level boss would open a portal, allowing the player to exit combat, as well as move on to the next level. In order to descend into the underworld to fight hades, players need to defeat the level boss to unlock portals. To successfully beat the game, the player must be able to dodge enemy attacks and strategically plan when to use their special attack.

As Descend's Visual Lead, I am responsible for all things visual, including designing and creating all of the game's characters (playable characters, NPCs, enemies, and bosses). This involved the process of designing, hand drawing, rigging, and creating the prefabs for the characters. Additionally, I handled creating and organizing all the assets (bushes, trees, volcanos, backgrounds, etc) and animations (every character's idle, attack, walk, death, etc) for our world scenes and combat stages. Furthermore, I took on most of the user interface components (title, screen, game over screen, congratulations screen, main menu, lobby menu, pause menu, and player's special attack cooldown bar).

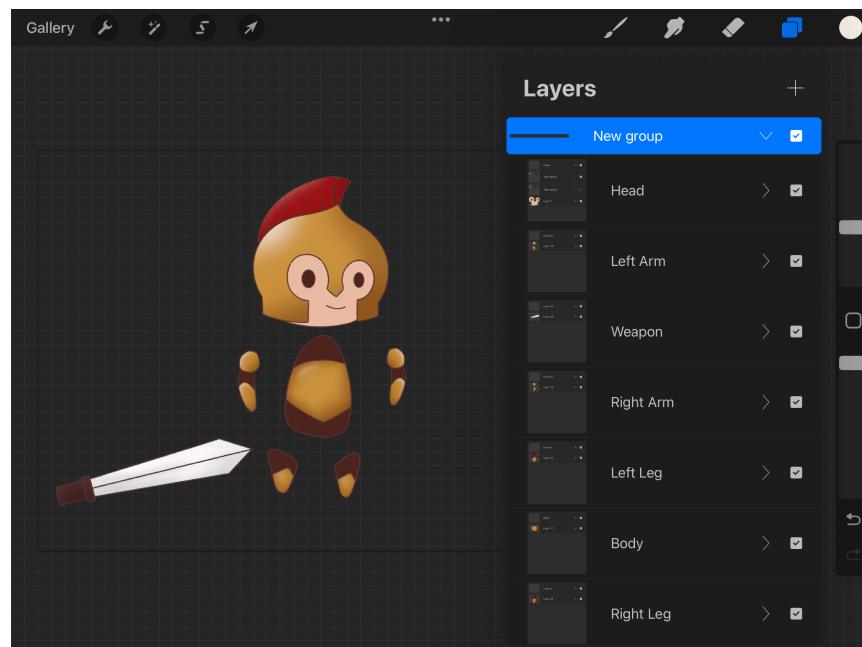
Introduction

The entirety of Descend is created from scratch using the many features within Unity as well as additional tools. Although the game exists as a single entity, Descend can be broken up into several components. The visuals, game logic, and audio all work together to create one cohesive game. This report will discuss the different components of the game and how the project is developed, as well as my contributions and thoughts on the process of creating Descend over the semester.

Game Components: Characters

Art:

All the characters were hand-drawn using Procreate on the iPad. The process includes each body limb (head, body, arms, legs, and weapon) being drawn separately as well as in separate layers. Having each limb separate would allow us to easily move each body part independently to animate the character with ease after we piece the body parts together. Utilizing the separate layers allow for more control of the art. Once the character is finished being drawn, it is saved as a PNG with a transparent background, and the file is sent to a computer.



Rigging:

After the character is drawn, the character needs to be pieced back together through the rigging process. The rigging is done using the Skinning Editor within Unity's Sprite Editor. The rigging process includes adding a digital skeleton and weights to the character sprite. The combination of the skeleton and weights is what will allow us to animate the characters. The skeleton is what allows us to control how to move the character; however, the sprite will only follow if the weights are added.

Then, for some of the characters, inverse kinematics is used for the arms and the weapon. Inverse kinematics is a component in the Unity Library, which makes animating easier and looks more natural. After the character is finished being rigged, it gets some basic components, such as a rigid body, added and then converted into a prefab to be used by the world and combat teams. The same process is done for every character.



Animation:

After each character is rigged, animations can be created for each of the characters. Each animation for each character is

created individually, even though many are similar and follow similar logic. The playable characters have an idle, walking, and attack animation, while the enemies have an idle, attack, and death animation. By default, all characters will be in their idle animations, which constantly loop.

For the playable characters, we have a function that toggles between the idle and walking animation states. In the `FixedUpdate()` method, we constantly check the player's movement to determine when to toggle the animation.

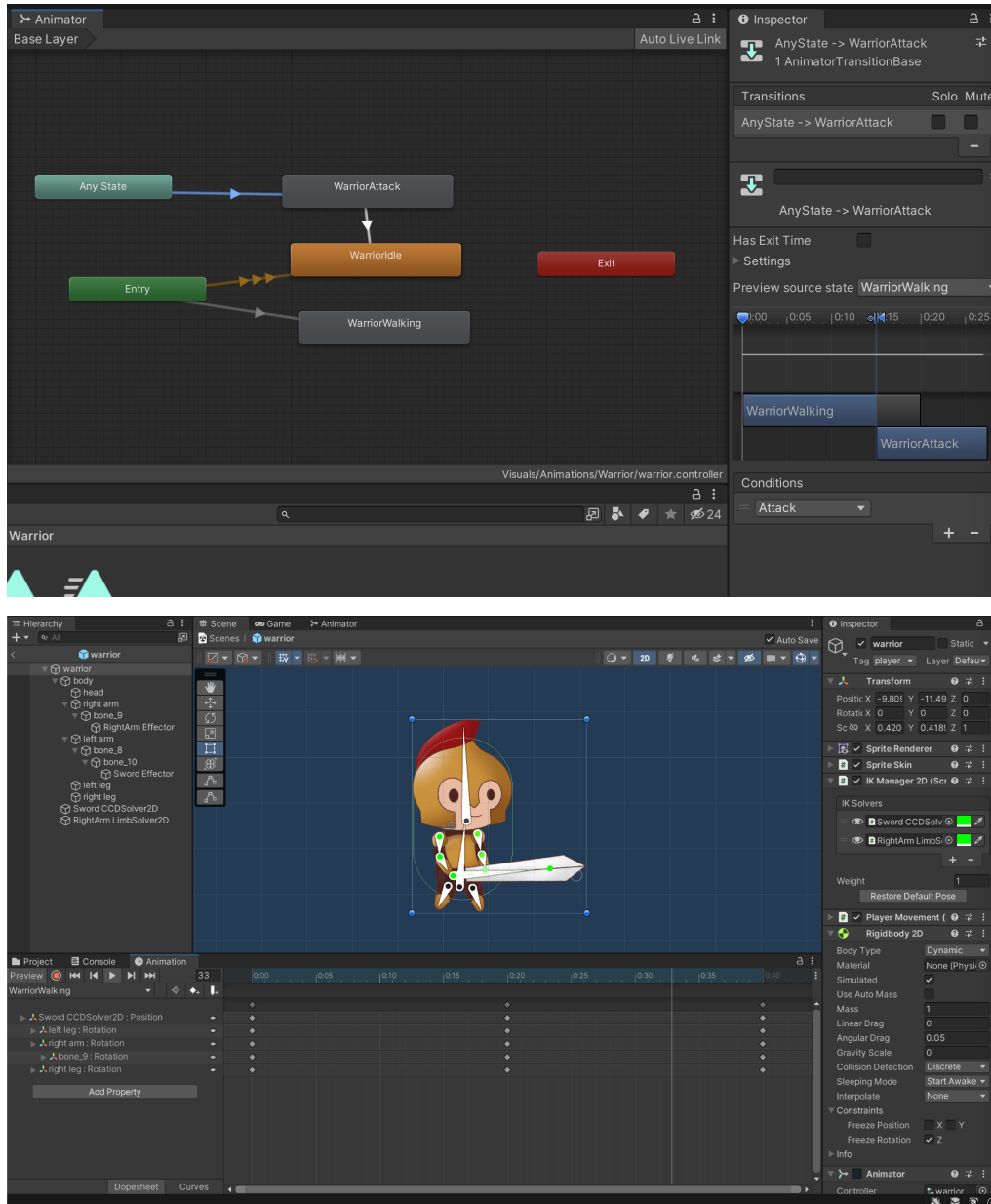
```
// Animation state changer
void ChangeAnimationstate(string newState)
{
    // Stop animation from interrupting itself
    if (currentAnimState == newState) return;

    // Play new animation
    animator.Play(newState);

    // Update current state
    currentAnimState = newState;
}
```

We have an `Attack()` function, which will trigger the `Attack` parameter that we set to the character's attack animation in the animator controller. In the `Update()` function, we check for when the player presses on the keys for the normal or special attack, as we want both attacks to trigger the attack animation, before calling the `Attack()` function.

```
void Attack()
{
    animator.SetTrigger("Attack");
    Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position,
        attackRange, enemyLayers);
    foreach(Collider2D enemy in hitEnemies)
    {
        enemy.GetComponent<EnemyBehaviour>().TakeHit(attackDamage);
    }
}
```



All the enemies' attack animations are triggered using a similar method. However, instead of checking for user input, the attack is called every some set time, depending on the enemy. The most complex enemy to develop is Cerberus, as he has 3 sets of attacks (one for each head), which we handle the same as any other attack. His idle animation is particularly difficult to create due to the speed of the tail wag vs the speed of head movement requiring a change at such drastically different rates.

To trigger the death animation, we track hitpoints. And if the enemy's hitpoint reaches 0, the death animation is triggered.

The image displays three examples of state machine diagrams and their corresponding animation rigs in a game engine.

Example 1: Cyclops

- State Machine Diagram:** Shows states: Any State, CyclopsAttack, CyclopsIdle, CyclopsDeath, and CyclopsWalking. Transitions: Any State to CyclopsAttack and CyclopsDeath; CyclopsAttack to CyclopsDeath; Entry to CyclopsIdle; CyclopsIdle to CyclopsWalking.
- Rig View:** Shows a Cyclops character rig with bones (bone_1 to bone_7), Canvas, Slider, Background, Fill Area, and CyclopsAttackPoint. The timeline for CyclopsAttack shows keyframes for bone rotations.

Example 2: Apollo

- State Machine Diagram:** Shows states: Any State, ApolloAttack, ApolloDeath, and ApolloIdle. Transitions: Any State to ApolloAttack and ApolloDeath; ApolloAttack to ApolloDeath; Entry to ApolloIdle.
- Rig View:** Shows an Apollo character rig with bones (bone_8 to bone_10), LeftArm Effector, left leg, right arm, LeftArm LimbSolver2D, LeftArm LimbSolver2D_Tar, LeftArm LimbSolver2D_Tar, Weapon CCDSolver2D, and Canvas. The timeline for Apollo_Death shows keyframes for various bone positions and rotations.

Example 3: Cerberus

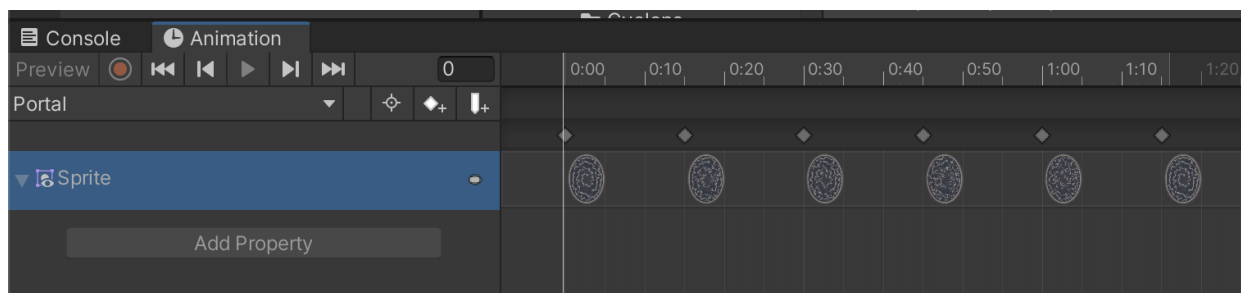
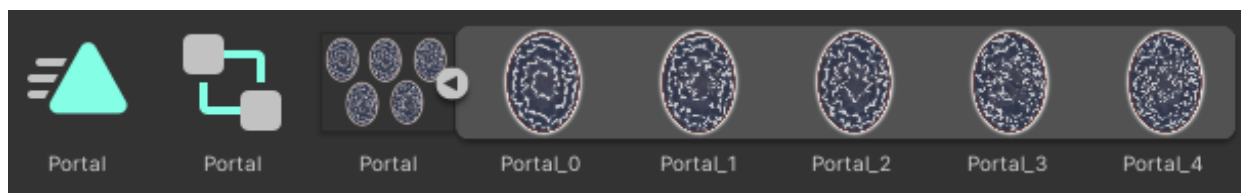
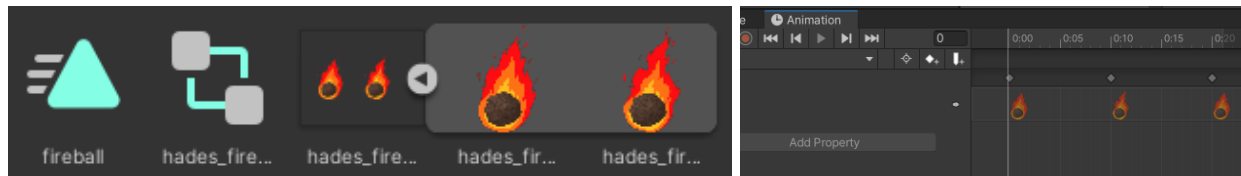
- State Machine Diagram:** Shows states: Any State, CerberusAttack_1, CerberusAttack_2, CerberusAttack_3, CerberusDeath, and CerberusIdle. Transitions: Any State to CerberusAttack_1, CerberusAttack_2, and CerberusAttack_3; CerberusAttack_1, CerberusAttack_2, and CerberusAttack_3 to CerberusDeath; Entry to CerberusIdle; Exit to CerberusIdle.
- Rig View:** Shows a Cerberus character rig with bones (bone_1 to bone_9), Canvas, Slider, Background, Fill Area, and CerberusAttackPoint. The timeline for CerberusIdle shows keyframes for bone rotations.

Game Components: Assets

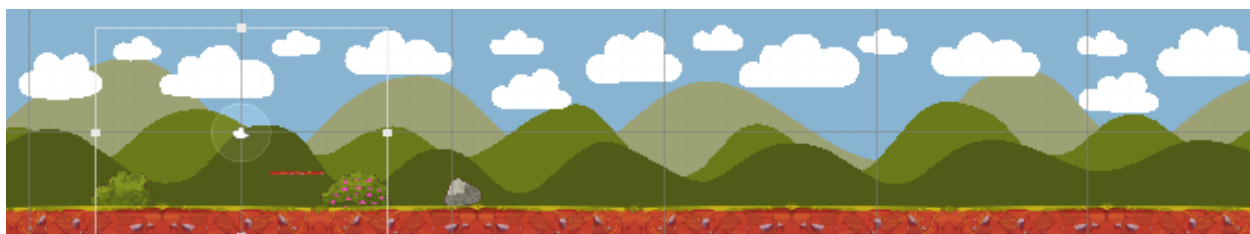
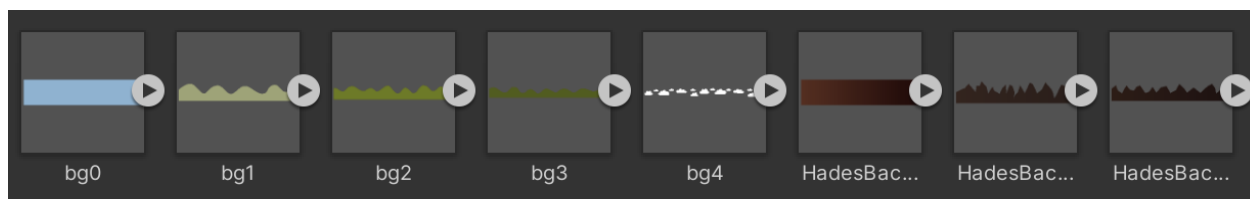
Similar to the characters, the entirety of the game's assets, aside from the ground type dirt in world level 1, which is a tilemap from the asset store, and the ground in combat stage level 1, which is from a free to use image found online that we had to clean up using photoshop. Most of the assets, including all the scenery pieces (bushes, trees, rocks, fences, volcanos, etc) are hand-drawn using Procreate, the same way that is used to create the characters. Several pieces are drawn on the same sheet. After being drawn, they were exported to Unity. Within the Inspector window for the image, the Sprite Mode selected would be "Multiple," which would allow us to slice each entity automatically using Unity's Sprite Editor tool. Oftentimes, we have to clean up slices using Custom Outline.



Some sprites were more complex because they were animated. To do this, the same piece would be drawn multiple times with some small differences. Then, the sprites would be sliced and added to an animation.



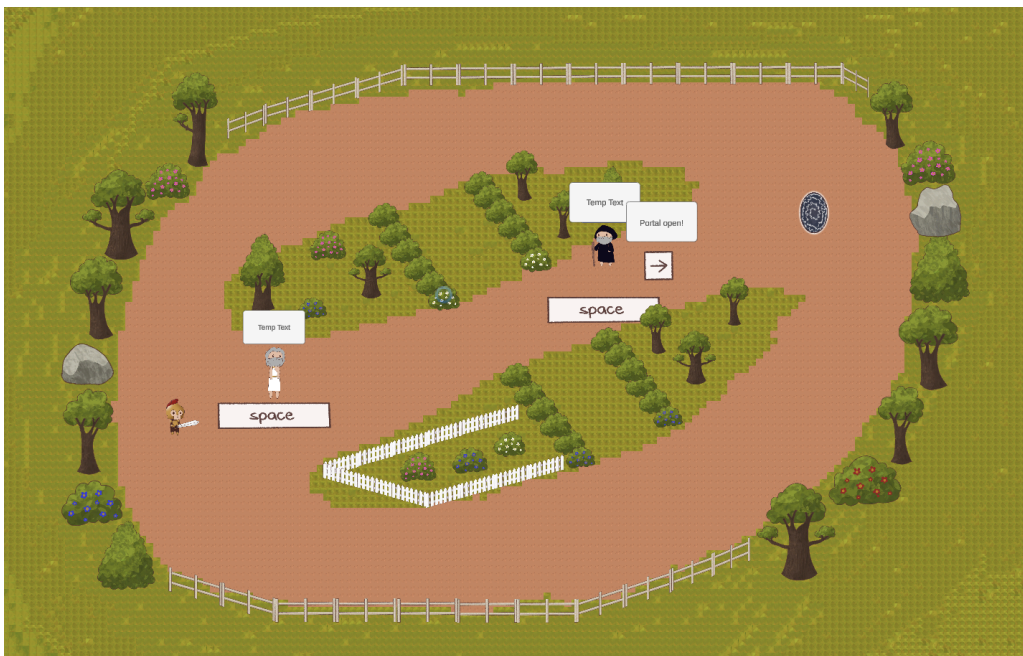
The background assets for the combat scenes were created using vector graphics in order to ensure clarity of these assets in-game and to accommodate being stretched and manipulated as needed to fit the scene.



Game Components: Overworld

Scene Development

The flooring of the scenes is created using the tilemap feature of Unity. Then, the scenery pieces are placed by dragging sprite assets onto the scene and arranged to our liking. Collisions are implemented using the Polygon Collider 2D component, which is attached to all the scenery pieces in the scene. Then, the NPC characters, which are prefabricated, get added the same way as the scenery pieces.



NPCs

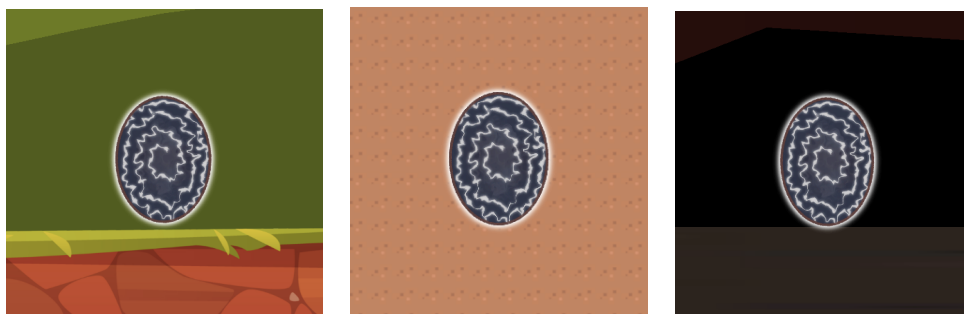
The story is driven by the NPCs in the world. The NPCs are prefabs that contain the rigged character and a canvas where the dialogue is displayed. The script includes a string array for each NPC, which we have populated with the lines that we want the character to say.

When the player collides with an NPC, the script attached to the NPC triggers a button that the user must click on to initiate the dialogue, which will display the first string from the array. To continue the dialogue, the space bar needs to be pressed, which will trigger the next element to be displayed. After the last string is displayed, the SceneManager is invoked, loading the scene that is attached for that NPC. This is how users are taken to combat stages.



Portals

Portals play a vital role in our game. Portals are used to transport the player between different world levels and between combat and world. By default, the portal will not be active. The portal has a script attached to it that checks if the level boss has been killed by checking that it is equal to null. If that condition is met, then the portal's visibility is enabled. When a user collides with the portal, a function is called to check for the current scene. Then, the scene manager is triggered and loads the next scene that we set.

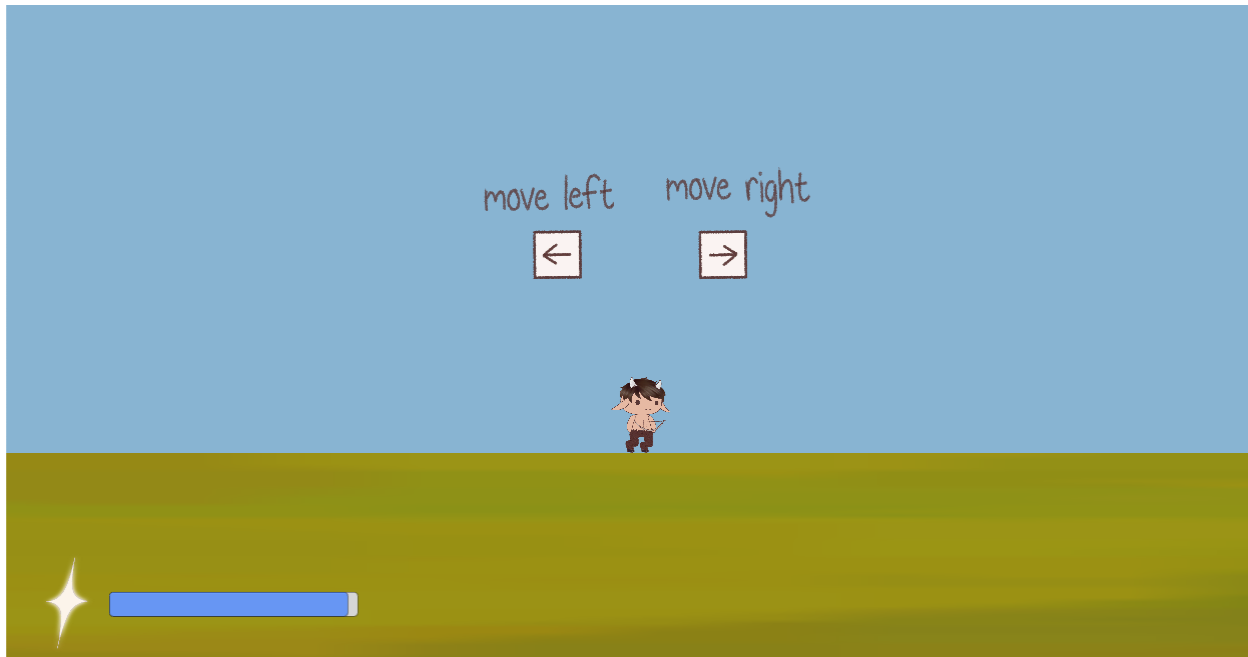


Game Components: Combat

The combat system is a vital component of the game. Players enter a different view that only allows x-axis movement, compared to the world view which can allow characters to move in the y-direction. In this world, the characters are now able to use their abilities and attack. There will be a UI for the enemies' health at the top of the gameplay, as well as a UI health bar for our characters. When players defeat the boss of the level, they will transition back into the World Map.



An important component that we have in our game is a tutorial stage, which aims to teach players how to navigate the combat stages and how to attack.



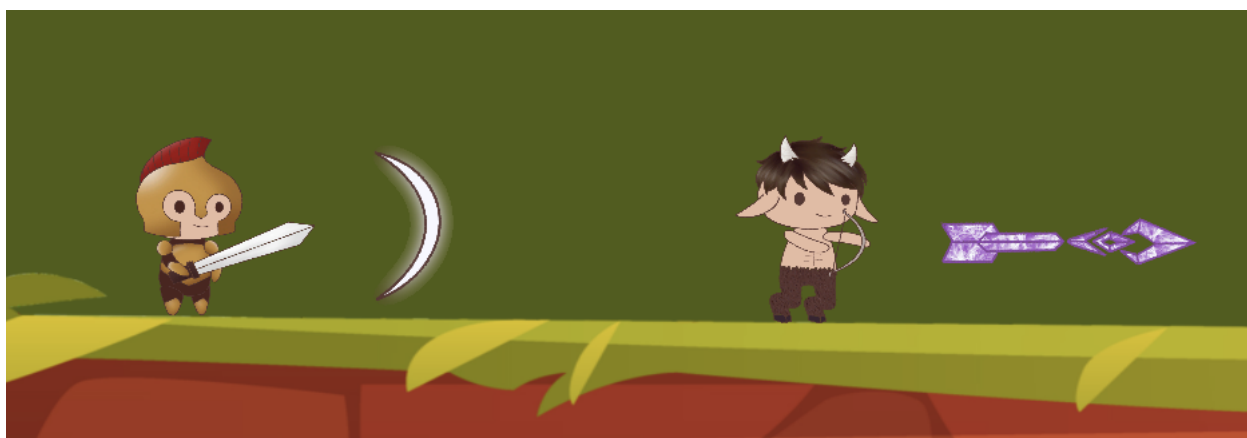
Attacks

The game features 2 playable characters, a human warrior, who has a sword for striking (close range), and a satyr archer, who has a bow and arrow for shooting (far range). Each character has 2 attacks - a normal attack and a super attack.

Player attacks are triggered by user input. If the user presses the spacebar, that will trigger the character's normal attack. As part of the prefab, the characters have an "attack point" object attached. For the warrior, it is located at the end of the sword. If the attack point collides with an enemy, that invokes the TakeHit function on the enemy. For the archer's normal attack, the archer has a script component called "Archer Basic," which will instantiate an arrow, which is a prefab that we have, from the attack point. The arrow will travel in one direction on the horizontal axis until it collides with an enemy or until it hits the ground, as it is bound to gravity.



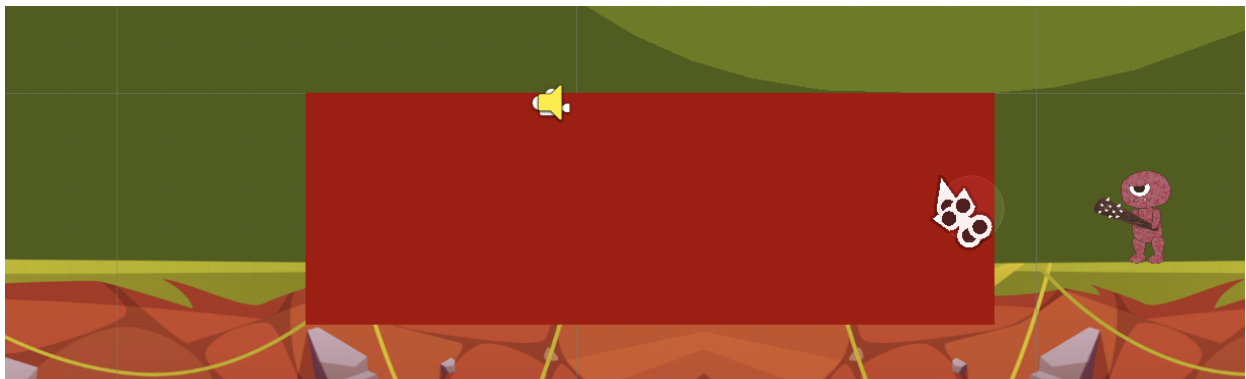
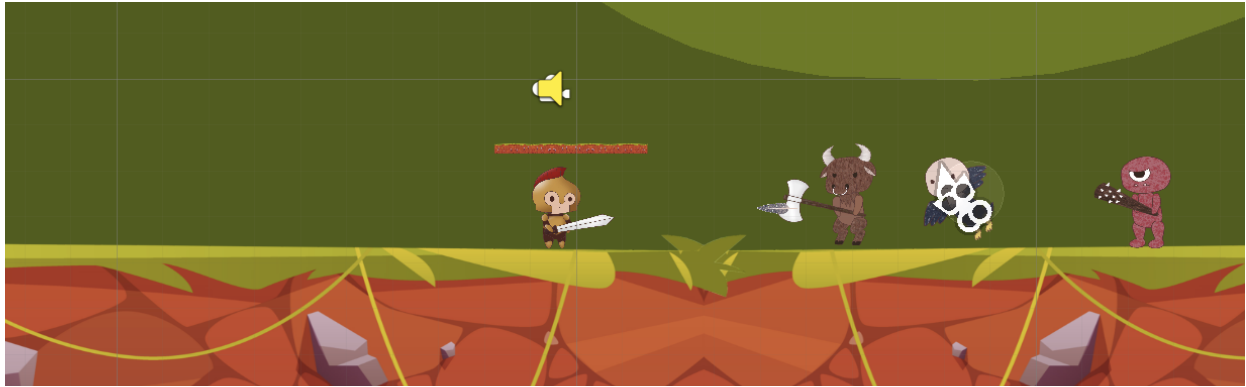
Both the warrior and archer's special attacks are created similar to the archer's normal attack of instantiating a prefab from the character's attack point, which will travel in one direction on the horizontal axis until it collides with an enemy, as it holds no gravity. The difference is that these special attack prefabs hold a higher number for attack damage. Additionally, there is a cooldown set in place in order to balance the character's strength and ability. The special attack can only be invoked once every 10 seconds. Both the normal and special attacks trigger the attack animation. If the "x" key is pressed, that will try to trigger the character's special attack. If the user attempts to use the special attack before the cooldown, we simply trigger the character's normal attack.



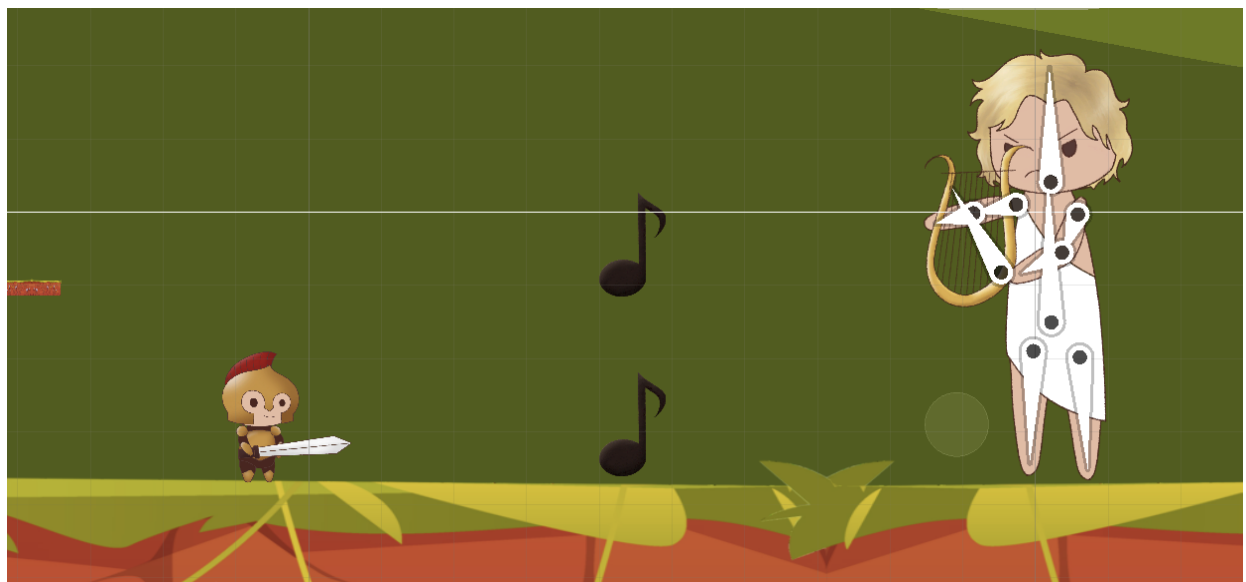
On the UI, there is a bar to indicate to the player when they are able to use their special attack. During the entire time of the charge up, the fill of the bar will be pink. When the special attack is ready, i.e. when the 10 seconds cooldown is over, the bar turns blue, as an indication to the player that they are now allowed to use their special attack.



Enemies only have a normal attack, which is triggered every some preset time. This interval is different for each enemy. Enemies that are considered grunts possess familiar attack styles. The cyclops and minotaur's attack is similar to that of the warrior's normal attack. When a user is colliding with an enemy, the enemy's attack is invoked, and the enemy will strike again the cooldown is over. Meanwhile, the harpy's attack is similar to that of the player's special attacks, as it instantiates prefabs that travel horizontally until a player is hit. The harpy does not require collision between the character but instead has a detector. A detector is a separate game object that is attached to the character. It is located before the character. The detector contains a script component that checks that a character has collided with it. If a player is colliding with the detector, the enemy's attack will be invoked.



Like the harpy, Apollo's attack also instantiates prefabs that travel horizontally to hit a player. The difference with Apollo is that Apollo bears two attack points that are vertically stacked - one at the player's standing height and the other at the character's jump height from the stage's ground. Both attack points invoke an attack at different speed rates, resulting in a unique pattern for Apollo's attack.



Cerberus's attack is also similar. However, Cerberus has three attack points, one for each of his heads. The additional attack point allows for an even more complex pattern of attack.



Amongst all the enemies in the game, the character that holds the most unique attack is Hades, who unleashes fireballs from the sky. The implementation is similar to the previous characters' attacks that involve shooting a prefab of some sort. However, Hades has 16 attack points that are located across the top of the scene within Hades' detector, and the prefabs of his attacks travel vertically.



HP / Health Points

Health points reset at every stage. Each character has a unique number which is their amount of health points. In combat, all players and enemies have a hp bar attached to the prefab of the character, in which the visibility is toggled on upon being attacked. As previously stated, each attack is attributed with a number referred to as "attack damage." As a result of being collided with an attack, the TakeHit() function of the character is called, reducing the hp of the character equal to the number of "attack damage" from the attack. At the character's max hp, the hp bar will be green, then gradually becomes red as their hp number decreases. When a character's hp reached 0, the game object is destroyed. For enemies, the death animation is invoked. Whereas for the player, the "Game Over" scene will be loaded.



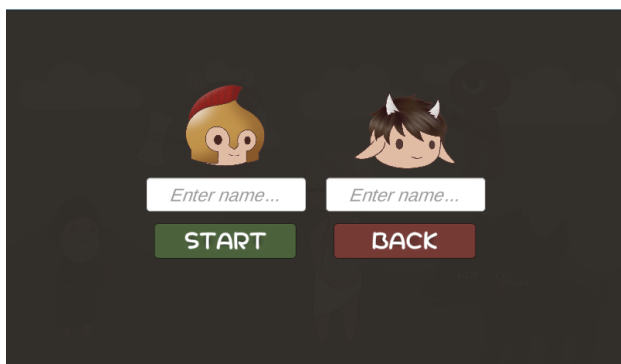
Game Components: User Interface

Screens and Menus

The game begins with the title screen, which consists of two panels on a canvas and a moving background. The first panel contains the title of the game and a “play” button. Selecting the play button will invoke a script that will activate the next panel and deactivate the current panel, which are both manually set. The following panel allows the user to indicate whether they want to play by themselves or with someone.



Selecting “solo” will call the scene manager to load the game’s World: Level One. Selecting “Multiplayer” will load up a Lobby Screen, which allows the players to pair up.



The pause menu is a panel made into a prefab that is added to every game scene. By default, it is not active. When a player hits the esc key, the pause menu will be set active on the current scene, and the game’s timescale will be set to 0f. Clicking on the continue button or hitting the esc key again will resume the game by deactivating the panel and setting the game’s timescale to 1f.

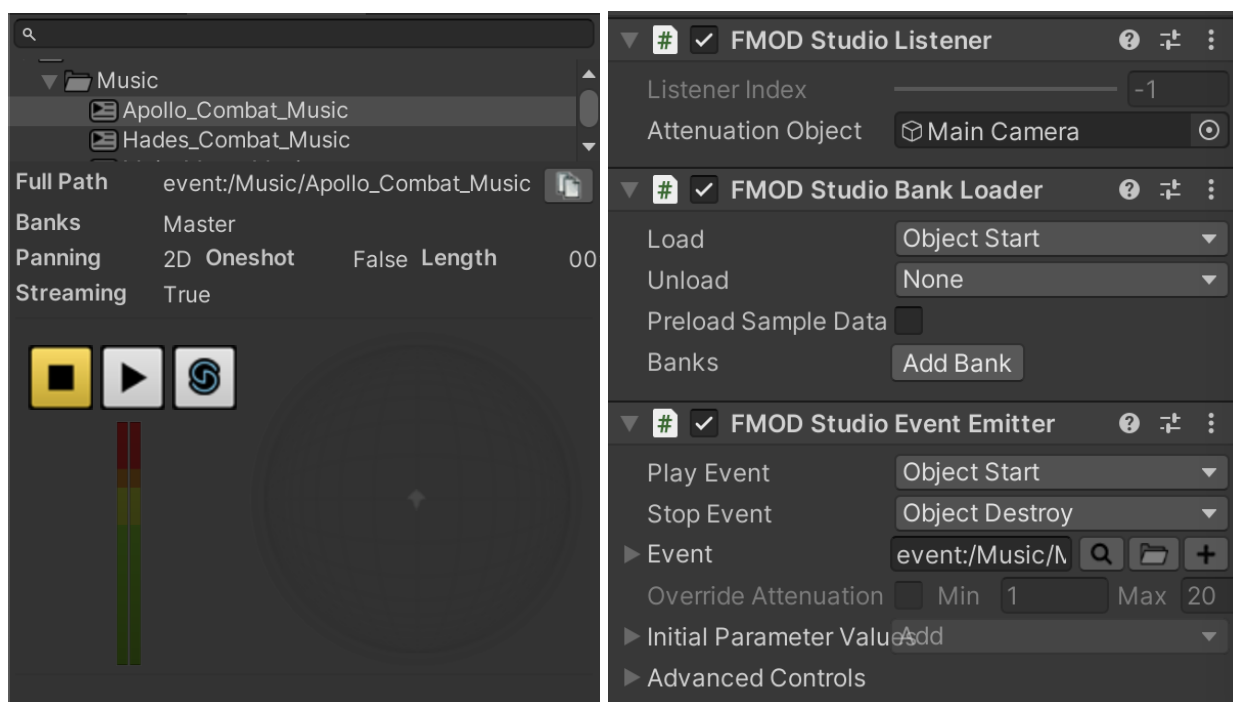


The Game Over screen will be loaded if the user dies, which is the game's losing condition. The Congrats scene will be loaded if the player defeats Hades, which is the game's winning condition. Both scenes consist of the option to play again, which loads the title screen. The player's current progress in the world is stored. This means that if the player defeats the level one boss, the portal that is active on world level one will remain active; the player does not have to fight the level one boss again to activate the portal.



Game Components: Audio

All of the game's audio (music, ambiance, sound effects, etc.) are produced by the audio team. The audio clips are loaded into an audio bank, which gets integrated with Unity through the FMOD tool. To add music to a scene, the components FMOD Studio Listener, FMOD Studio Bank Loader, and FMOD Studio Event Emitter are added to the camera. Selecting the camera as the source of music allows the music to be consistent everywhere throughout the scene.



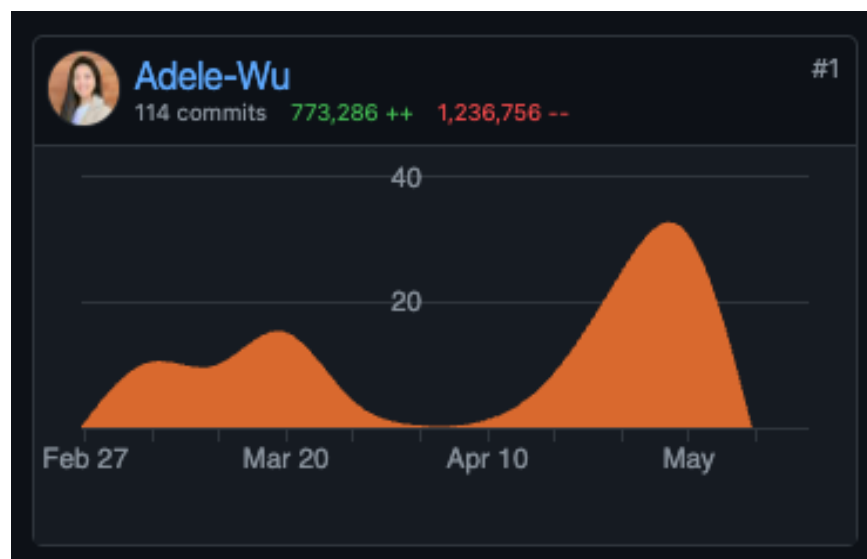
Most sound effects are dependent on game logic, and so, they are integrated within scripts. To invoke a sound effect, simply use the `PlayOneShot()` function of `FMODUnity` and insert the location of the desired audio file from the FMOD audio bank. For example, the following code is used to prompt the audio for the warrior's super attack. This line of code is embedded into game logic to determine when the warrior's super attack is used.

```
FMODUnity.RuntimeManager.PlayOneShot (
    "event:/SFX/Player/Big_Slash",
    GetComponent<Transform>().position
);
```

My Contribution

As prior mentioned, I am the team's Visual Leading, making me responsible for all of the game's assets, animations, and UI components, as well as providing assistance wherever needed. I created the characters used in the game by hand drawing, rigging, and creating animations for all the characters. Details for the process can be found under "Game Components: Characters." Additionally, I also hand drew almost all of the game's assets, sliced them, and created animations for them as necessary. Details for the process can be found under "Game Components: Assets." Furthermore, I created many of the game's UI components, such as all the menus and the player special attack cooldown bar. Details for the process can be found under "Game Components: User Interface" and "Game Components: Combat" respectively. In creating the menus, I also created the flow of the game as I wired the scenes and panels.

Git Commits



main		
Commits on May 9, 2022		
fix esc menu's exit button Adele-Wu committed 4 days ago	49c52fe	<>
make scenes for networking, login/reg and multiplayer lobby Adele-Wu committed 4 days ago	96c3667	<>
add countdown timer bar for user's special attacks, and did stuff to ... Adele-Wu committed 4 days ago	f791c85	<>
pull main Adele-Wu committed 4 days ago	93d82b9	<>
user special attack timer is implemented Adele-Wu committed 4 days ago	4a43e98	<>
Commits on May 8, 2022		
fix more idle animations Adele-Wu committed 5 days ago	a72af14	<>
fix grunt idle animations Adele-Wu committed 5 days ago	9af6685	<>
fix ground Adele-Wu committed 5 days ago	bffc2ec	<>
broken code, don't pull Adele-Wu committed 5 days ago	d68513	<>
pulled, trying to switch to my branch Adele-Wu committed 5 days ago	9d53c8a	<>
fix minotaur and harpy death positions Adele-Wu committed 5 days ago	61555da	<>
replace 'ground' with 'real ground', which is clearer Adele-Wu committed 5 days ago	8098ebc	<>
Commits on May 6, 2022		
make 'solo/multiplayer' panel on the same scene as the title panel Adele-Wu committed 7 days ago	f493e52	<>
fix ui screens Adele-Wu committed 8 days ago	341285e	<>
Commits on May 5, 2022		
merge my branch into main Adele-Wu committed 8 days ago	1c8872b	<>
create harpy and minotaur death animations Adele-Wu committed 8 days ago	2aee54a	<>
implemented game over screen to combat scenes Adele-Wu committed 8 days ago	6e42322	<>
pull main Adele-Wu committed 8 days ago	f3dd1c5	<>
fix ui screens Adele-Wu committed 8 days ago	8b68536	<>
create minotaur animations Adele-Wu committed 8 days ago	8e681c2	<>
merge main to my branch Adele-Wu committed 8 days ago	d8e827b	<>
add harpy animations Adele-Wu committed 8 days ago	2824deb	<>
recreate cyclops walking animation Adele-Wu committed 8 days ago	51bce8c	<>
pull main Adele-Wu committed 8 days ago	9b495c2	<>
create cyclops walking animation Adele-Wu committed 8 days ago	cafe94a	<>
fix hades death animation Adele-Wu committed 8 days ago	2ef8523	<>
fix hades death animation Adele-Wu committed 8 days ago	53145ed	<>

pull main Adele-Wu committed 8 days ago	52a8e28	<>
fix hades death Adele-Wu committed 8 days ago	6653c2e	<>
create apollo death Adele-Wu committed 8 days ago	ed87c8e	<>
fix merge conflict Adele-Wu committed 8 days ago	b74da17	<>
fix merge conflict Adele-Wu committed 8 days ago	d172a46	<>
pull main Adele-Wu committed 8 days ago	27b5ed4	<>
apollo death Adele-Wu committed 8 days ago	a27bf2b	<>
create hades death animation Adele-Wu committed 8 days ago	8348c2d	<>

Commits on May 5, 2022		
create cyclops attack animation, fix warrior animations Adele-Wu committed 8 days ago	b34838e	<>
figure out bug that was causing the esc menu to not work properly in ... Adele-Wu committed 9 days ago	c27ca47	<>
create cerberus death animation Adele-Wu committed 9 days ago	842e135	<>
speed up fireball animation Adele-Wu committed 9 days ago	7b6d248	<>
create cerberus attack animations. organize animations folder Adele-Wu committed 9 days ago	8358395	<>
create cerberus attack animations. organize animations folder Adele-Wu committed 9 days ago	b36a34f	<>
add rigid body to cerberus. do not remove. will break code Adele-Wu committed 9 days ago	8e53c23	<>
fix warrior's 'animation event has no function name specified' error Adele-Wu committed 9 days ago	1d19090	<>
merge main into my branch Adele-Wu committed 9 days ago	5faa488	<>
create archer attack animation and incorporated in gameplay Adele-Wu committed 9 days ago	2e3d6f3	<>
Commits on May 4, 2022		
create idle and walking animation for archer. modify PlayerController... Adele-Wu committed 10 days ago	a98292e	<>
create idle and walking animation for archer. modify PlayerController... Adele-Wu committed 10 days ago	45d6e3e	<>
Commits on May 3, 2022		
create scene and background for enter game and login/registration panel Adele-Wu committed 11 days ago	838aeb3	<>
create scene and background for enter game and login/registration panel Adele-Wu committed 11 days ago	5f678b6	<>

Commits on May 2, 2022

- add esc menu to all points in game
Adele-Wu committed 11 days ago [10e9f6d](#) <>
- create cerberus idle animation
Adele-Wu committed 11 days ago [3057732](#) <>
- create cerberus idle animation
Adele-Wu committed 11 days ago [ba31faa](#) <>
- create hades animation for idle and attack
Adele-Wu committed 11 days ago [0236c6d](#) <>
- apollo animations idle and attack
Adele-Wu committed 11 days ago [2fda08e](#) <>
- add cerberus howl sound waves
Adele-Wu committed 11 days ago [096a438](#) <>
- add hades fireball
Adele-Wu committed 11 days ago [567443c](#) <>
- add archer super arrow
Adele-Wu committed 11 days ago [3f8c087](#) <>
- create zeus, made as prefab
Adele-Wu committed 11 days ago [3023093](#) <>
- turn arrow 90 degrees
Adele-Wu committed 11 days ago [bb1d47a](#) <>

Commits on May 1, 2022

- unity stuff. trying to merge main in my branch
Adele-Wu committed 12 days ago [0a44b90](#) <>
- unity stuff happening
Adele-Wu committed 12 days ago [6afabfc](#) <>
- create idle animation for cyclops
Adele-Wu committed 12 days ago [94bd7e](#) <>
- create idle animation for cyclops
Adele-Wu committed 12 days ago [6b41921](#) <>

Commits on Apr 29, 2022









- fixed esc screen. not sure why it stopped working. it was working bef... [ed8d88f](#) <>
- clean up
Adele-Wu committed 14 days ago [d42721d](#) <>
- create lobby menu. it is named main menu
Adele-Wu committed 14 days ago [3bc2b89](#) <>
- made pause menu as prefab. working on lobby from hw4 provided code
Adele-Wu committed 14 days ago [63d16c3](#) <>
- made pause menu as prefab. wokring on lobby
Adele-Wu committed 14 days ago [151774a](#) <>
- fix merge conflict
Adele-Wu committed 14 days ago [232ba12](#) <>

Commits on Apr 28, 2022





- create congrats and game over screens
Adele-Wu committed 15 days ago [41e85fc](#) <>

Commits on Apr 28, 2022		
add font and created font assets Adele-Wu committed 15 days ago	fec888	<>
clean up Adele-Wu committed 15 days ago	1888b5b	<>
connect 'solo or multi' screen into game Adele-Wu committed 15 days ago	3fe68fa	<>
create 'solo or multiplayer' screen and 'esc menu' screen Adele-Wu committed 15 days ago	39ef276	<>
add background for hades combat Adele-Wu committed 15 days ago	bdb7114	<>
made portal as prefab Adele-Wu committed 15 days ago	84381a9	<>
add asset: background for hades Adele-Wu committed 15 days ago	e33fe38	<>
Commits on Apr 27, 2022		
add sword slash Adele-Wu committed 16 days ago	18c92ef	<>
trying to switch to my branch. adding this file Adele-Wu committed 16 days ago	4195a16	<>
add words for keys Adele-Wu committed 16 days ago	9887c9f	<>
add keys Adele-Wu committed 17 days ago	81cfe14	<>
Commits on Apr 26, 2022		
fix bug Adele-Wu committed 17 days ago	e9ce7cb	<>
fix bug Adele-Wu committed 17 days ago	c96d57e	<>
add title to menu Adele-Wu committed 17 days ago	b21db1f	<>
create starting menu Adele-Wu committed 18 days ago	46ca327	<>
Commits on Apr 25, 2022		
add dead plants Adele-Wu committed 19 days ago	4ced7ec	<>
add dark dirt ground for world Adele-Wu committed 19 days ago	4da5a38	<>
Commits on Apr 21, 2022		
clean up Adele-Wu committed 22 days ago	f5892c6	<>
create cerebus and hades, made as prefabs Adele-Wu committed 22 days ago	9ce76bf	<>
Commits on Apr 20, 2022		
add volcano + smoke + more sharp rocks Adele-Wu committed 23 days ago	581cc71	<>
create old man npc for story, make him as prefab Adele-Wu committed 23 days ago	1bdabb7	<>
add feather for grunt-harpy's attack Adele-Wu committed 23 days ago	a157886	<>
make existing characters as prefabs Adele-Wu committed 23 days ago	7823957	<>













Commits on Mar 31, 2022

- Merge branch 'main' of github.com:doomphd/Descend  b382e1f 
- Merge branch 'adele_branch'  d454eac 
- add question mark emblem for tutorial stage  92a4c5e 
- add question mark emblem for tutorial stage  85cdbe1 















Commits on Mar 24, 2022

- update and rig archer  4fa6ef1 
- add archer files  e47563f 













Commits on Mar 23, 2022









































- rig minotaur  b4c45c8 
- rig minotaur  2075b2c 
- rig harpy grunt  47f1225 
- rig cyclops grunt  ef17e86 
- add grunt images  8394498 
- update warrior walking animation  f12d112 

Commits on Mar 23, 2022

- organized visuals directory  03a9064 
- rename background gameobjects for combat to make more easily understand...  d9a5416 
- add smaller ground, could be used as platform for combat scene  c338834 
- add background to combat  1644a3d 
- add background files  a3d1526 
- add package for svg  01d92ba 
- add some background files  e263fb4 

Commits on Mar 21, 2022

- make main camera follow player  c28581a 
- add user control to warrior + idle and walking animations in world  3c9b789 
- add user control to warrior + idle and walking animations  395353e 
- add user control to warrior + idle and walking animations  1f18c9c 
- added idle animation and walking animation to warrior  da84cf5 
- add fences  02076e0 

Commits on Mar 16, 2022		
add grass for world Adele-Wu committed on Mar 16	 1f66b32	
redo bushes with outline Adele-Wu committed on Mar 16	 8fec818	
make new version of ground less long Adele-Wu committed on Mar 16	 588ab17	
add ground Adele-Wu committed on Mar 16	 d63a844	
add bushes and 8th note Adele-Wu committed on Mar 16	 a6740bb	
Commits on Mar 15, 2022		
add apollo Adele-Wu committed on Mar 15	 cf63ae9	
Commits on Mar 11, 2022		
centered warrior more Adele-Wu committed on Mar 11	 a2753cc	
centered warrior more Adele-Wu committed on Mar 11	 4144518	
remove unused scene Adele-Wu committed on Mar 11	 b3882fa	
remove unused scene Adele-Wu committed on Mar 11	 1462e45	
removed unused scene to clean up Adele-Wu committed on Mar 11	 1dbae36	
seperated trees. trees are ready for use Adele-Wu committed on Mar 11	 28c28c7	
imported trees Adele-Wu committed on Mar 11	 8cba8a9	
made rocks individual Adele-Wu committed on Mar 11	 5246611	
added rock drawing Adele-Wu committed on Mar 11	 593cf68	
Commits on Mar 10, 2022		
added version 1 of warrior, just incase we prefer that method later on Adele-Wu committed on Mar 10	 9ae9651	
cleaned up warrior rig Adele-Wu committed on Mar 10	 244a6df	
rigged warrior Adele-Wu committed on Mar 10	 acd5188	
created warrior Adele-Wu committed on Mar 10	 b945dbd	
created adele branch Adele-Wu committed on Mar 10	 83f96bd	

Challenges and Lessons Learned

The greatest challenge of working on this project, for both my team and me, is using Git. We would always get merge conflicts in addition to other errors whenever we pull, push, or merge. It was a process, but by the end of the semester, we were able to figure out the best solution to these errors. We initially tried to add some of these files to the gitignore page; however, that did not fix the problem. For a while, the git stash command was able to get us by. Then, upon approaching the TA, we were advised to simply delete the conflicting files before we pull. That came to be tedious and was not always effective. Our go-to method now is to simply add and commit those files. Through the process of dealing with endless git conflicts and errors, we better understand the different files that Unity creates and modifies.

A lesson that I learned is to not be over-optimistic. My team initially wanted to create 5 levels and 3 stages within each level, making 15 combat stages in total. We even devised a plan and set milestones that allowed us to believe it was a possible task. Upon every member having to spend an unexpected amount of time to create the first level and stages, we learned that it was not possible for us to create 5 levels and 15 stages that would be decent given the short amount of time, that is a single semester. As a result, we decided on quality over quantity, and we scaled down so that we would be able to create components that we would be proud of. This meant that some work and plans had to be scrapped entirely. However, I believe that this was the right decision.

Second Chance

If given a second chance to develop this game from scratch, I would like to set up a more rigid working structure. I believe communication and ensuring that everyone is aware of the project's current state and trajectory is extremely important.

My team did not have consistent meetings; we simply arranged meetings whenever we feel that a meeting is needed. For the most part, that was fine for the majority of the team, as we were able to hold ourselves accountable for our tasks, even without the consistent meetings. However, I acknowledge that not everyone is like that, and I think having more consistent meetings would keep everyone on their toes and engaged with the project/team.

Additionally, I believe another method to ensure that all members are aware of their tasks is to have a place for the team to display each member's responsibilities and expected due dates, along with the progress of each task. This could be as simple as setting up a Discord channel that contains messages about each member's tasks and their respective due date. Additionally, the member could react or respond with a message to indicate that the task is completed as well as maintain an organized history of the project's progress. An alternative is using a resource like Trello to assign and track each member's tasks. Some of our members already inform the rest of the team of when they have pushed into the main branch and describe the changes that they have made. However, not everyone does this, so not everyone is always aware of the state of the project. To add, sometimes our general channel gets flooded with messages, so people would miss the message on project updates.