CMPE 485
TERM PROJECT - PHASE II
BULL RUSH
OPTION B - REPORT

Gökhan Uysal - 201740024
Ufuk Arslan - 2017400217
04.06.2022

Game Description

The player controls a mage warrior that can attack the Bull Boss via magical projectiles, and as a special ability, they can send an orb to which they can later teleport its position. Minions spawn in random locations in a fixed period, and upon defeating, they might drop heath collectibles. The goal is to defeat the bull before being defeated. Players can select difficulty as well.

Detailed Game Information

- **Environment:** The battle takes place in a 3D circle arena with boundaries. The arena has visuals of a gladiator arena, and it takes place in the space.
- **Camera:** The camera is placed high from the ground and it automatically aligns the player and the boss so that players won't have to worry about the camera.
- **UI:** We have a UI at the start of the game for selecting difficulty (Easy, Medium, Hard). Both hero's and the boss's health bars are visible inside the game. There is an ability cooldown indicator next to the hero's health bar. Upon pausing the game, a pause screen appears, where players can resume the game, play again, or quit the game. Upon winning or losing, an end screen appears, where you can select to play again or quit.
- **Hero:** Players control the hero. The hero can **move** and **aim** with **WASD** or **left Joystick**. Hero has two types of skills. The first skill is a simple attack projectile that hits once and won't pierce through the boss, which can be used with **left-click** or **joystick circle**. The second skill is what makes this hero unique, the teleport ability. A teleport attack is a projectile to enemies, that pierces through them and the hero can teleport to the projectile's concurrent position. With the teleport ability, the hero will be able to dodge some attacks of the boss that are otherwise nearly impossible to do. Both sending the teleport orb and teleport action can be done with **right-click** or **joystick rightsholder**. Upon teleporting, the

hero creates a blue trail back to him. Upon teleporting, a cooldown starts, so that player can't spam the teleport ability. The hero has a low hit point, but it has an invincibility frame with prevents him from getting hit rapidly. Upon getting damaged, a red particle shows up indicating the damage. Unity animation FSM was used for hero animations, and we used some parts of the warrior input and animation controller scripts that came with the asset.

- **Boss:** The Bull Boss has two different attacks and a lot of hit points. In the basic attack, if the boss comes close to the hero, it swings his weapons through the hero, dealing damage. In the rush attack, the boss first jumps, giving the player a visual queue and it rushes to the last location of the player, creating a red trail at his back. Unity animation FSM was used for boss animations.
- **Minions:** Minions are low health enemies, which upon defeating might drop health collectibles. Minions will spawn in random locations inside the arena and they follow the hero. If the minions touch the hero, they deal damage. Unity animation FSM is for the minion animations. The number of minions in the scene is used for the stress-test measurement of the game.
- **Game Manager:** The game manager controls the rate minions spawn and the pause screen.
- **Difficulty Selection:** The difficulty affects the hit points of the hero, the hit points of the boss, and the probability of health collectible drops from minions.
- **Sound:** Although we wanted to implement sound in the game, we think that with the current visual aspects of the game, it quite is enjoyable.

Challenges Faced

1. **Arena Boundaries:** The hero asset we used came with a demo environment that used mesh colliders as its walls. We used this environment in the early stages of development. However, it visual theme of that environment did not fit our game and the mesh collider was not the type of wall we wanted to use. Since in attack animation the tip of the staff of the mage warrior is ahead of his collider, we have found bugs where the hero could send the teleport ability outside the mash collider, and teleport to that orb to escape the stage. To overcome this we have decided to use two layers of invisible box colliders as walls. The first box collider prevents the hero from moving outside the arena, and the second box collider prevents the orb from going outside the stage. An alternative solution could be before instantiating the orb object, checking the position of instantiation, and if that position is outside the arena then preventing the orb from spawning. However, defining the safe area of spawn was not easy so we chose to use double-layer invisible box colliders as walls.

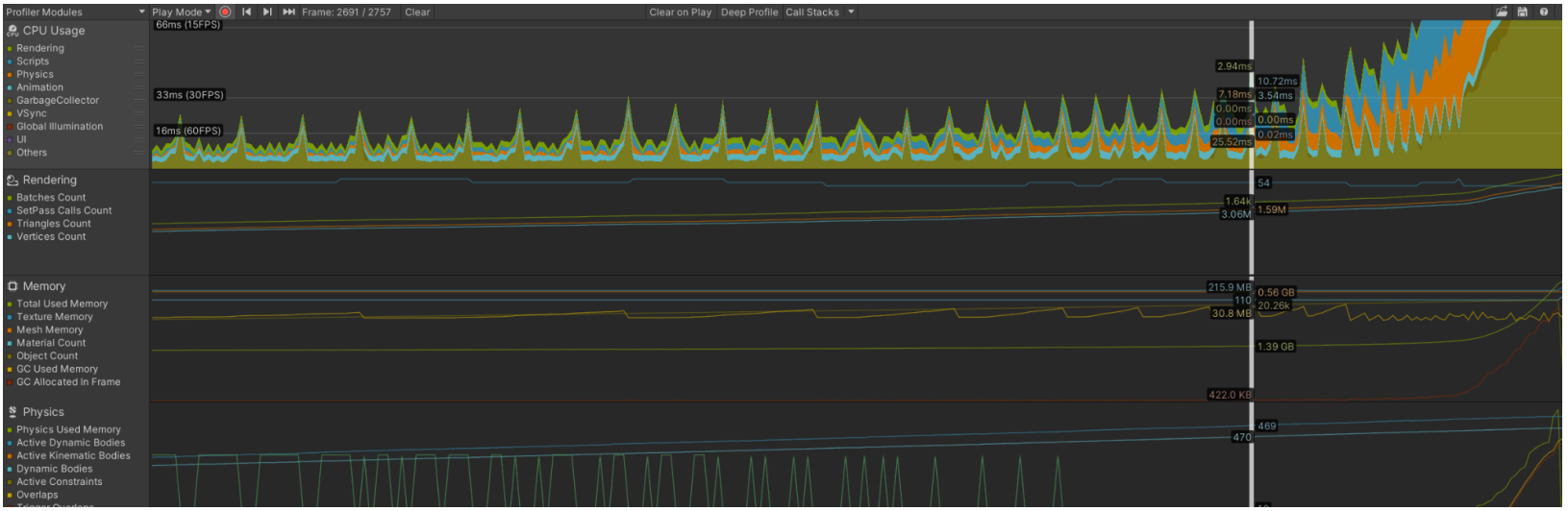
2. **Camera:** We first decided to create our own camera with the default camera object in Unity. However, it was not easy to create one that follows the hero and aligns with the boss which moves smoothly. So we then decided to use the Cinemachine library from Unity. Cinemachine made our life a lot easier. We created an empty object that is at the top of the hero and chose it as the following target and chose the boss as the second target. We chose the FOV, the smoothness, and the distance of the camera with Cinemachine. We had our desired camera. However, when the player was behind the walls of the arena the camera would show the outside of the arena walls and it would obstruct the inside of the arena. To overcome this, we decided to find the raycasthit objects that obstruct the vision between the hero and the camera. The camera script worked when only one layer of wall is rendered between the camera and the hero. However, the arena that we chose had multiple layers of rendered surfaces, causing the obstruction problem to remain. We did not want to change the arena asset we used, so we decided to choose to lower the height of the walls in the arena, which would not intersect the camera view. A better solution would be finding a raycasthit object disable script that disables multiple layers of renderers upon obstruction.
3. **Boss Manager:** The boss has many states and animations making it hard to control and travel between states. We used different states like isAttacking, isAttackEffective, isRunning, isWalking, runToWalk, isDead, isJumping and checked those states in the fixed update function of Unity. The problem with this structure was the number of conditions required. Whenever we wanted to implement a new pattern to the boss, let's say running, we had to add a new flag that represents the occurrence of the running action, and we had to handle all the possible transitions to the other states, like run jump, run to walk, run to attack, which made it very difficult to implement new actions to the boss. Although boolean states get the job done for this project, the best way to manage the action states of the boss would be to use a state machine for the sake of scalability.

Profiling

We decided to use the number of minions in the scene as the stress-test measurement of the game. In the fixed update function of each minion script, we calculate the distance vector between the minions and the hero, then move the minion in that direction. Also, each minion has a collider that gets triggered when a projectile hits them, or they touch the hero.

Inside the game manager, we find a random point inside the boundaries of the area and instantiate a minion object. In the default version of the game, minions spawn every 5 seconds, but in theory, if the player won't destroy the minions, there can be an infinite amount of minions on the scene. To decide on the minion count limit, and to test how massive numbers of minions affect the game, we reduced the spawning period down to 0.01 seconds and waited until 500 minions occur in the scene. You can check out the screenshot below to see how the number of minions affects different profiler modules. Here we can see that the scripts used the most capacity of the CPU with 749ms, followed by other (mostly due to player loop) with 713ms, followed by the physics with 468ms. This result is expected since there are 500 scripts of minions running with update loops, and 500 different collision trigger sources that move with physics calculations. Since we used the low-poly version of minions in our game, and there is only one light source, the rendering did not affect the CPU usage much. We can see that the total memory used is 3GB since there are a lot of objects that allocate memory in the scene. You can see in the physics tab there are 501 active dynamic bodies in the scene.

As you can see in the screenshot below, on our PC we have seen a rapid increase in CPU usage at around 470 minions. However, there could be users with lower CPU capacities, so we decided to put a minion limit at 250, which is extremely uncommon to reach considering minions spawn rate and game mechanics.



Free Unity Assets Used

- Warrior/Hero Animations & Controller
  - Warrior Pack Bundle 2 FREE
- Bull/Boss Animations
  - BOSS Class - Bull
- Minions/Slimes Animations
  - RPG Monster Duo PBR Polyart
- Orb/Projectile
  - Magic Particle Systems Lite
- Environment
  - Low Polly Gladiators Arena
- Skybox
  - Skybox Volume 2 (Nebula)
- UI Background
  - Space Star Field Backgrounds

Unity Libraries Used

- Camera
  - Cinemachine
- UI Text
  - Text Mesh Pro

Tutorial References

https://www.youtube.com/watch?v=ruufqlXrCzU&ab_channel=ExplosiveLLC
https://www.youtube.com/watch?v=BLfNP4Sc_iA&ab_channel=Brackeys
https://www.youtube.com/watch?v=zc8ac_qUXQY&ab_channel=Brackeys
https://www.youtube.com/watch?v=wtrkrsJfz_4&ab_channel=5HeadGames