Shattered

December 10, 2019

FINAL PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIRMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

COMPUTER GAME DESIGN AND PROGRAMMING (CSCI 5920)

Shattered

Author:
Seth MIERS
Bachelors of Science in Electrical and Computer Engineering (Dec. 2015)

Professor:
Dr. Min-Hyung CHOI

December 10, 2019

Contents

List of Figures

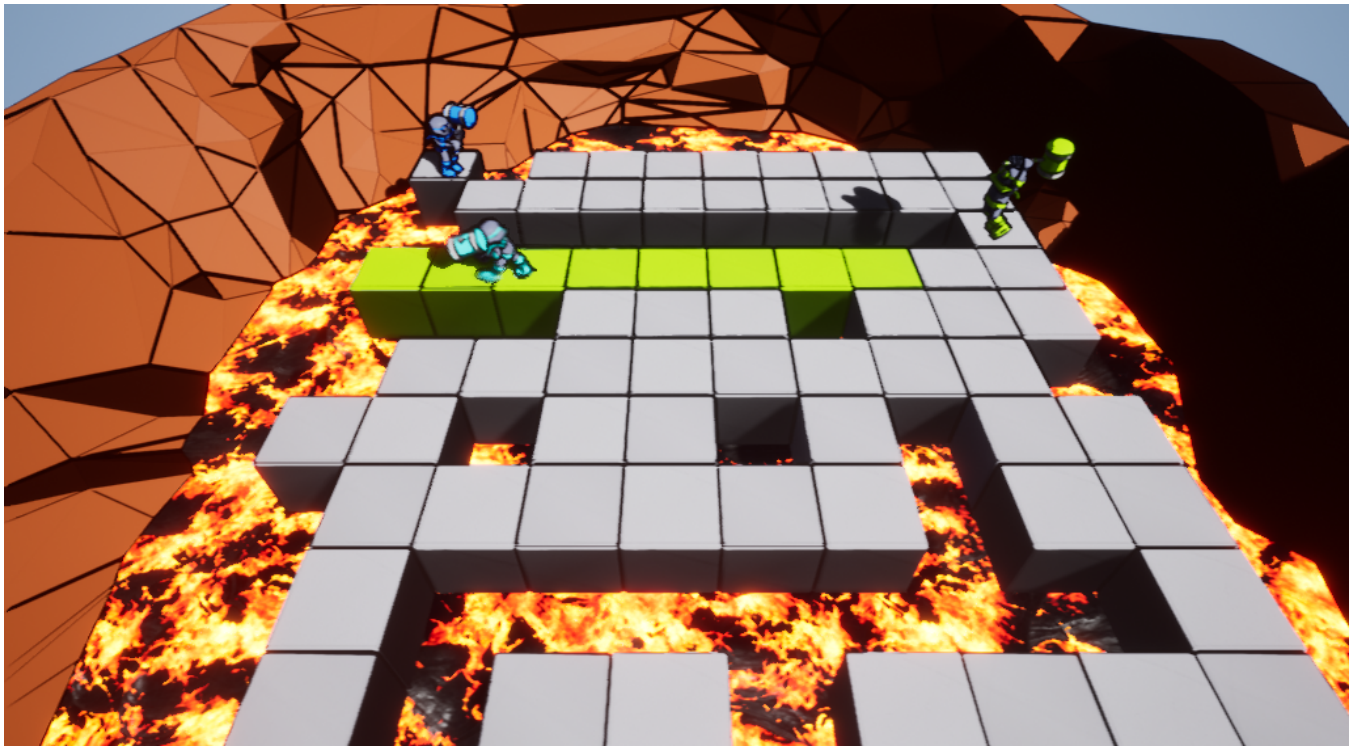

Figure 1: 3 Player Game-Play

***Abstract*—Shattered is a couch co-op fighting game. The goal is to break the blocks underneath your opponents to get them to fall off the map. The last person left standing on the map wins the round. The players can jump to get over gaps and they can slam their hammer into the blocks to break them.**

1. Project Webpage

1.1. Main Code

https://github.com/superzanti/Shattered

1.2. Compiled Release

https://github.com/superzanti/Shattered/releases/download/v0.5/Shattered_Winx64_Alpha0.5.zip

2. Core Strengths

The game's core idea was to be a fun couch arena game with realistic physics. I believe it succeeded in both of these areas and anyone can pick up a controller and quickly understand the basic concept of the game.

2.1. Physics

Physics was the main research component for this project. It ended up being implemented in a very nice way. The character has predictable movement speeds and realistic acceleration. All of the player controls are very tight, with controllable jump heights and animations that match what the character is doing.

Physics goes far beyond just a 'physics engine'. While that still has to be fought with to achieve desirable results, much of physics is simply in how the player views the character and not in what the character actually does.

This process started with the 3D model that would be used as the character. Realistic physics would not have been able to be achieved as well if it wasn't an object the player was familiar with. A humanoid was chosen for these reasons. The humanoid was then rigged such that realistic body physics could be achieved. This meant not skimping out on some bones that help the realism, while removing bones that are not necessary to achieve better optimization.

The animation then had to be done in a way that looked physically reasonable. This took reading, studying, and watching many slow-motion videos of walk and run cycles to make sure the character looked physically 'normal'. This whole process took much longer than expected.

The character was then textured and moved into unreal engine, where one of the next most important features was implemented: the player's physics asset. The balance between physically accurate and number of collision areas had to be compared and balanced. Constraints on every joint

had to be set up so that the body would fall in a realistic manner as well. The result was a beautiful physics asset that would rag-doll properly and really enhanced the feel of realistic physics.

Beside the physics engine configuration, the animation, model, and rigging, the external environment also had to be configured properly. Correct physics simulations of blocks breaking can also be seen in this game. Even players interacting with each other had to be done in a realistic manner. Instead of a hard stop, like one would expect with most primitive unreal engine collision box, a custom collision box had to be made to account for 'soft' or 'bouncy' collisions.

2.2. Fully Playable Game

For the time allocated for this game, it is amazing that there is a fully playable release that can be enjoyed by a group of friends (or even strangers).

The game loops infinitely and it's always hard to say no to another round since they're so quick and easy to play. I found myself simply enjoying the character and running around every time I tested a new feature.

The game is elegant enough with the physics, that there are multiple levels of mastery. Due to the variable jump height, a character can jump a different length of blocks,

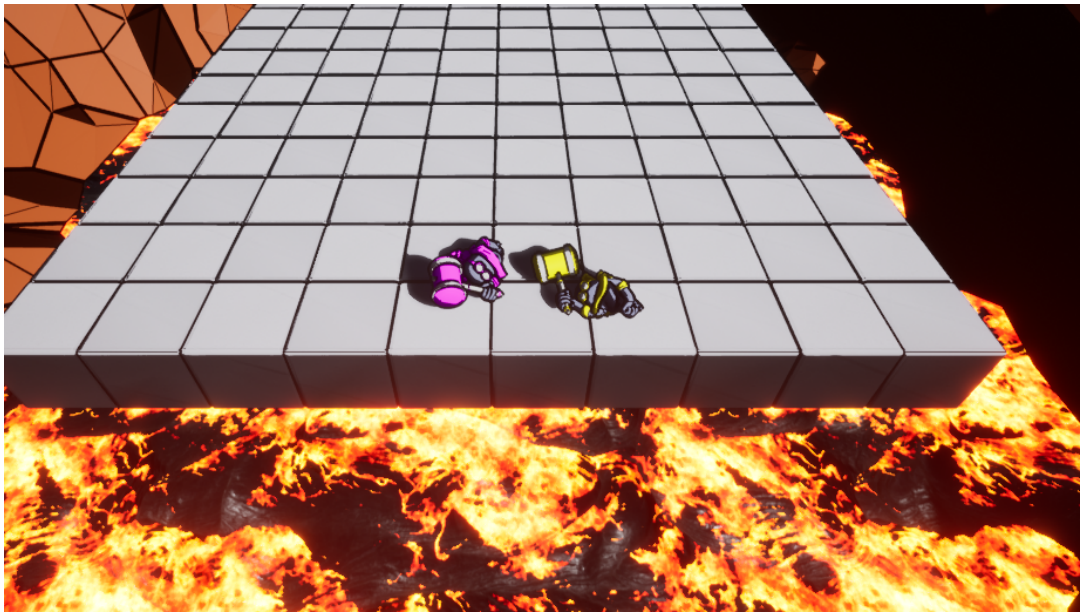

Figure 2: Fully Functional Ragdoll Physics

Figure 3: Tuned Jumping Physics

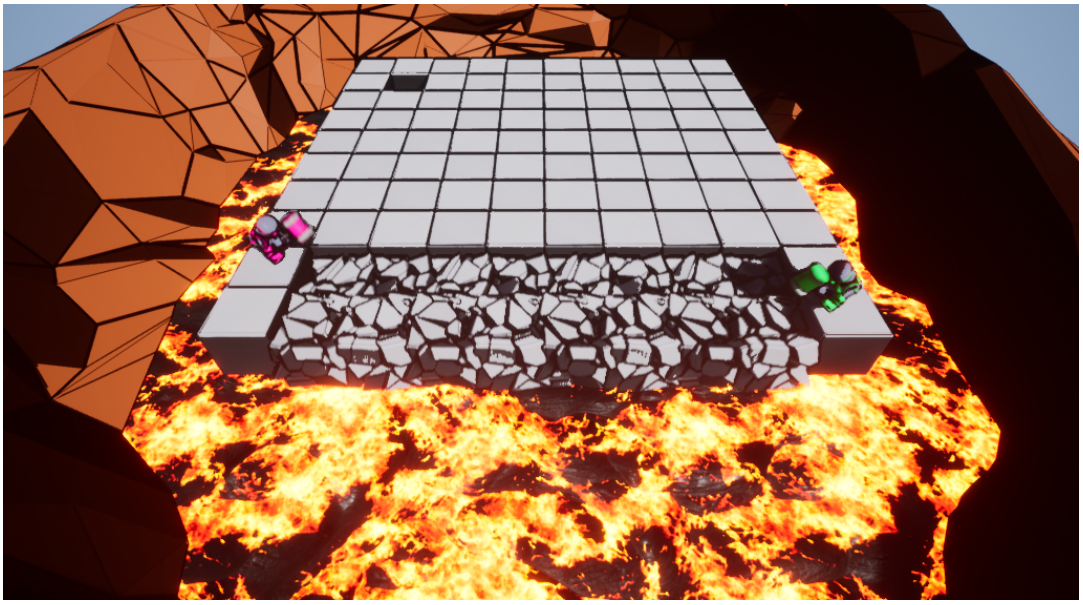

Figure 4: Physically Accurate Destructible Mesh for Blocks Breaking

they can do fake-out jumps, and they can even use the jump time as a strategy for messing with an opponent.

Variable run speeds also provide another level of control. Not only can you gain enough momentum to clear a gap, but you can walk slowly to get right up to the edge of a block for a quick escape.

While it's hard to pull off it is also very much possible to jump off another player's head. Every little aspect of this game comes down to a single point where it's simply just fun to play and fun to get distracted by.

The game even supports full local multiplayer with up to 4 players. All players use a different gamepad. The game was tested with the USB Xbox 360 controller. This functionality can be seen in Figure 5.

2.3. All Asset Design

Nothing in this game was borrowed, used, or stolen from someone else or even any other organization. Every asset, texture, animation, physics body, mesh, and more were all created by me.

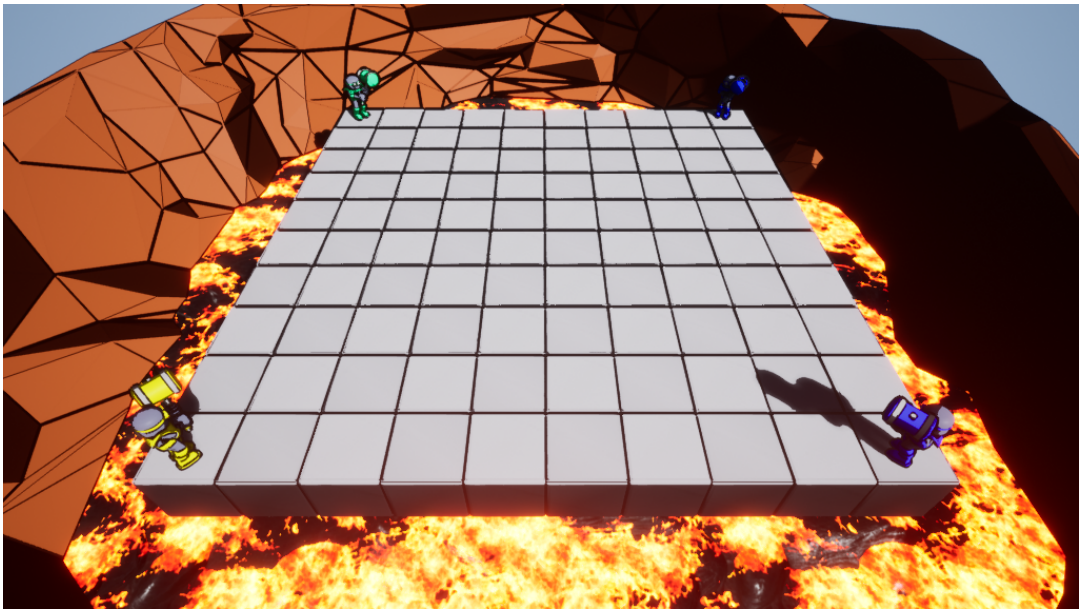

Figure 5: 4 Total Spawn Points for Multiplayer Play

While it doesn't show as a core strength at face value, once one realizes how much work goes into even just one asset of a game, this could be the biggest strength.

3. Changes to Original Concept

The overall concept of the game was kept the same throughout the duration of work. However, Many features were thought up during the development process, and many were sacrificed as a result of the time constraints.

3.1. Minimal Player Customization

While modeling and playing with textures, I realized how easy it would be to make a dynamic material in Unreal Engine so that players could change the color of their character. This ended up being implemented very well.

Harder customization such as swap-able heads and hammers had their core functionality implemented. The character asset is designed to have 2 sockets that other meshes can connect to. One is the 'head' socket, and the other is the 'hammer' socket. Changing the color of these meshes was also achieved.

3.2. Simple Menu Screen

Without some kind of menu screen color customization would not be achievable, also there would always be 4 players on the screen. If you have only 3 controllers, there would still be another player spawned which could mess up how the game plays. It was chosen to add in a menu to make the game more playable.

3.3. Audio

No game is complete without audio. Without audio a game can be very bland and the player won't feel as connected to it. The choice to add audio was just so that the game would have a feeling of completeness.

3.4. Block Breaking Options

Originally the character could break as many blocks in any 360 degrees of direction. This ended up feeling weird and unpredictable since humans are unable to quickly determine which blocks will break in a certain direction. All 360 degrees of rotation also would decreased the strategy involved and it would end up playing as a 'quick draw' type of game. So the options on block breaking is limited to 1 in the diagonal or any number in the horizontal or vertical directions. This can be seen in Figure 6 and Figure 7.

4. Play-ability Changes

As the game grew, it became apparent that original ideas for how things would work would need to be changed in order to optimize the flow of the game. Many of these just had to do with how the player controlled their character and the pacing of the game itself.

4.1. Physics Changes

With the base physics, even having the character look moderately real, The character still felt too springy and hard to control. As it was it was possible for a character to jump over 5 blocks at once, which would make the game last forever and reduce the amount of strategy required. Physics

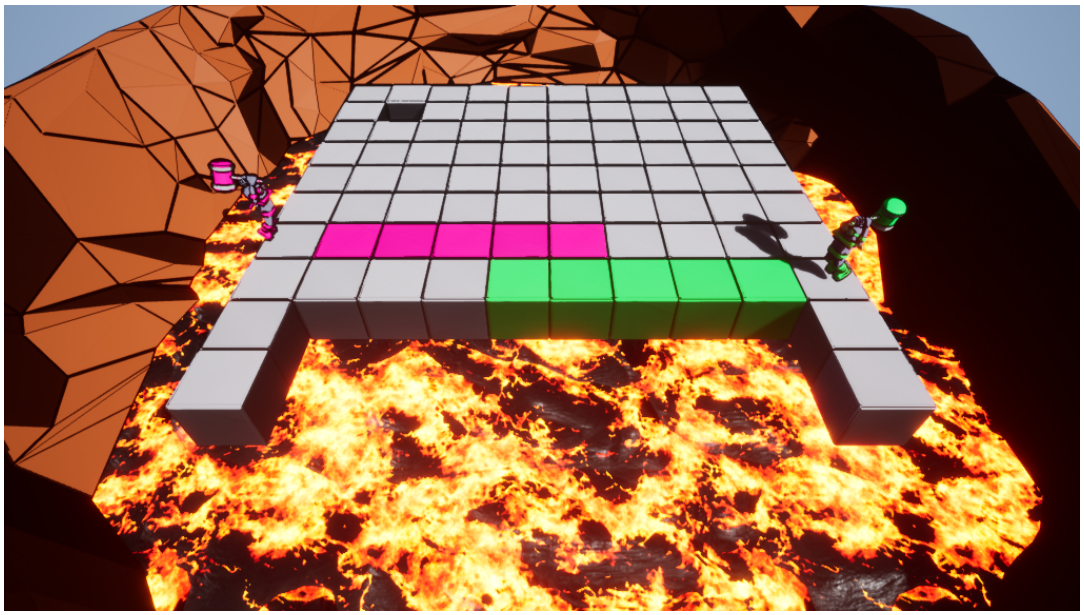

Figure 6: Hammer Mode Charges Up So Player Can Break Multiple Blocks

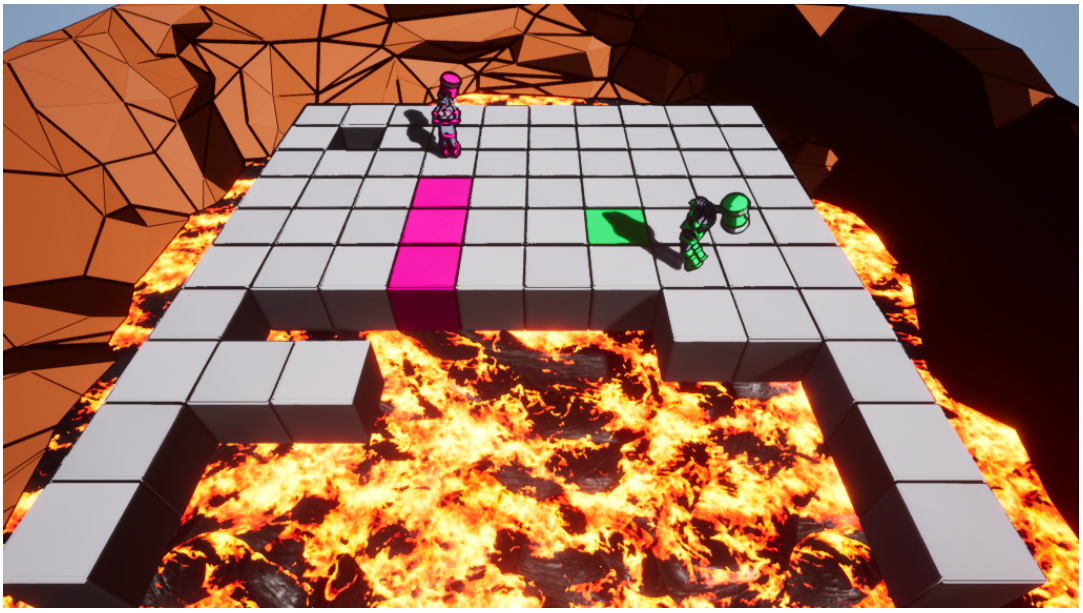

Figure 7: Hammer Can Only Break 1 in the Diagonal, and Up To 8 In Horizontal or Vertical

were tuned to still be realistic but reduce the jump height to just one character in height, and friction was added to the air so that characters could only jump over 1 block easily, and 2 blocks if they were really experienced.

```
1  GetCharacterMovement()->bOrientRotationToMovement
       = true;
2  GetCharacterMovement()->RotationRate = FRotator(0.
       f, 800.f, 0.f);
3  GetCharacterMovement()->bConstrainToPlane = true;
4  GetCharacterMovement()->bSnapToPlaneAtStart = true
       ;
5  GetCharacterMovement()->MaxWalkSpeed = 1100.0f;
6  GetCharacterMovement()->MaxAcceleration = 10000.0f
       ;
7  GetCharacterMovement()->JumpZVelocity = 800.0f;
8
9  GetCharacterMovement()->BrakingDecelerationFalling
       = 800.f;
10 GetCharacterMovement()->BrakingFriction = 0.5f;
11 GetCharacterMovement()->AirControl = 50.f;
12 GetCharacterMovement()->GravityScale = 2.2;
```

Listing 1: Biggest Physics Engine Tuning Parameters

4.2. Animation Speed

While I wanted to show off the time I spent on the animation, it ended up being a big detriment to game-play. Jump animations were too slow and took too long from when the player clicked 'a' till the player jumped. The same problem was true when landing and swinging the hammer. Animations were sped up to feel fluid to the player and help the player feel as if they were in full control.

5. Play-Testing

The game was play tested several times throughout development, which is how many issues were found. However, the final test proved to be very beneficial. What I saw as a developer as needed to be fixed first, was a completely different priority from what the players thought should be changed first.

5.1. Testers had Fun

One group of play testers got carried away and kept starting new games. It was fun and exciting for them.

"It was fun. The rounds were quick and engaging. It was easy to keep jumping into the next round. I found there was more strategy to it than I had expected. Looked good. Felt good. Intuitive controls." -Elliot

"The game was a lot of fun, and hard to put down. It's great for small groups of friends, and results in a lot of laughs. The volcanic texture and the controller vibration were really cool. I liked being able to change the color of my avatar to any shade of the available colors." -Rachel

All may play testers thought it was fun. So I consider that a huge success.

5.2. Bugs Galore

The players also had fun finding small bugs. As one player stated 'there is nothing game braking'. Many of the bugs are simply funny or just something that makes one go 'oops'.

"The main menu looked a little incomplete, but it's functional. There are probably a few bugs that still need to be worked out, but nothing game breaking. I think there needs

Figure 8: Player Falling Off the Map and Ragdolling

to be some thought put into preventing/resolving potential stalemate scenarios, like random blocks breaking after a certain amount of time" -Elliot

"The time it takes to charge the hammer is a bit long, but that doesn't impede game-play or enjoyment too much." -Rachel

6. Testing Changes

Since many of the bugs could not be fixed within the time-frame, my focus was directed to the game-play itself.

6.1. Player Movement

The acceleration had to be tuned. Some players found that quickly turning around resulted in a total loss of momentum, and while this is physically accurate, it leaded to them feeling like they were being cheated by the game instead of being in control of their character. The max acceleration of a character was tuned so that a character could quickly turn around and have enough momentum to jump over one block. The speed of the hammer charge rate was also tuned. I found through testing that players ended up using the single block hammer the most because it gave other players less time to react.

6.2. Collisions

When trying to tune the physics for the rag-doll and the effect for landing on lava, I ended up breaking the collisions for the destructible meshes used for the blocks. Players found that they could still jump off of blocks as they were falling. This made it hard to feel like you were eliminating another player and made it instead feel like you were just trying to survive the longest without mistakes. Fixing these collisions fixed that problem.

7. Time Constraints

The biggest problem in the entire project was time constraints. As the project progressed I learned that every single component takes way longer than I thought it would. It gave me a huge appreciation for many of the games that come out today. Hours were spent on just tuning one physics component, days were spent on the overall physics, hours again were spent on modeling and animation. The list goes on and on. I slowly began to realize that my goals were too lofty for the time allowed. While this could be a fully polished game with another work-weeks of work, I simply didn't have another set of 40 hours to spend on the game.

7.1. Achievements and Score Tracking

Achievements and score tracking would've taken too much time to implement and I prioritized having a functional game. Post screen achievements and stats would have made the game really enjoyable and definitely more competitive, but in the end, the game-play was much more important.

7.2. Audio

Some of the functionality for audio was there, but I did not have the time to make my own sounds, and it was important to me that every single part of this game was made and designed by myself. For footsteps I did not have time to set up a mic and record sounds, however, the controller does slightly vibrate every time the character steps as a drop in replacement (and maybe addition because everyone really

liked it). Game music was not implemented as composing songs takes hours alone.

7.3. Code Organization

Rushing to get a complete project done within the time frame lead to some bad coding practices. If I had more time I would go back and re-organize where much of the code is, add in comments, and add variables where variables are needed. I found myself having to change code in 3 places at the end when I wanted to change something.

8. Lessons Learned

In the future I would also get a broader range of play testers. Having just friends and family limits the feedback. I fear that many of them are too scared to say much that is bad about the game.

8.1. Physics is Difficult

Getting physics to look correct while also making sure the game-play still feels good is extremely difficult. There was also much difficulty in figuring out how to fight the built in Unreal Physics Engine. Many of the commands I had to

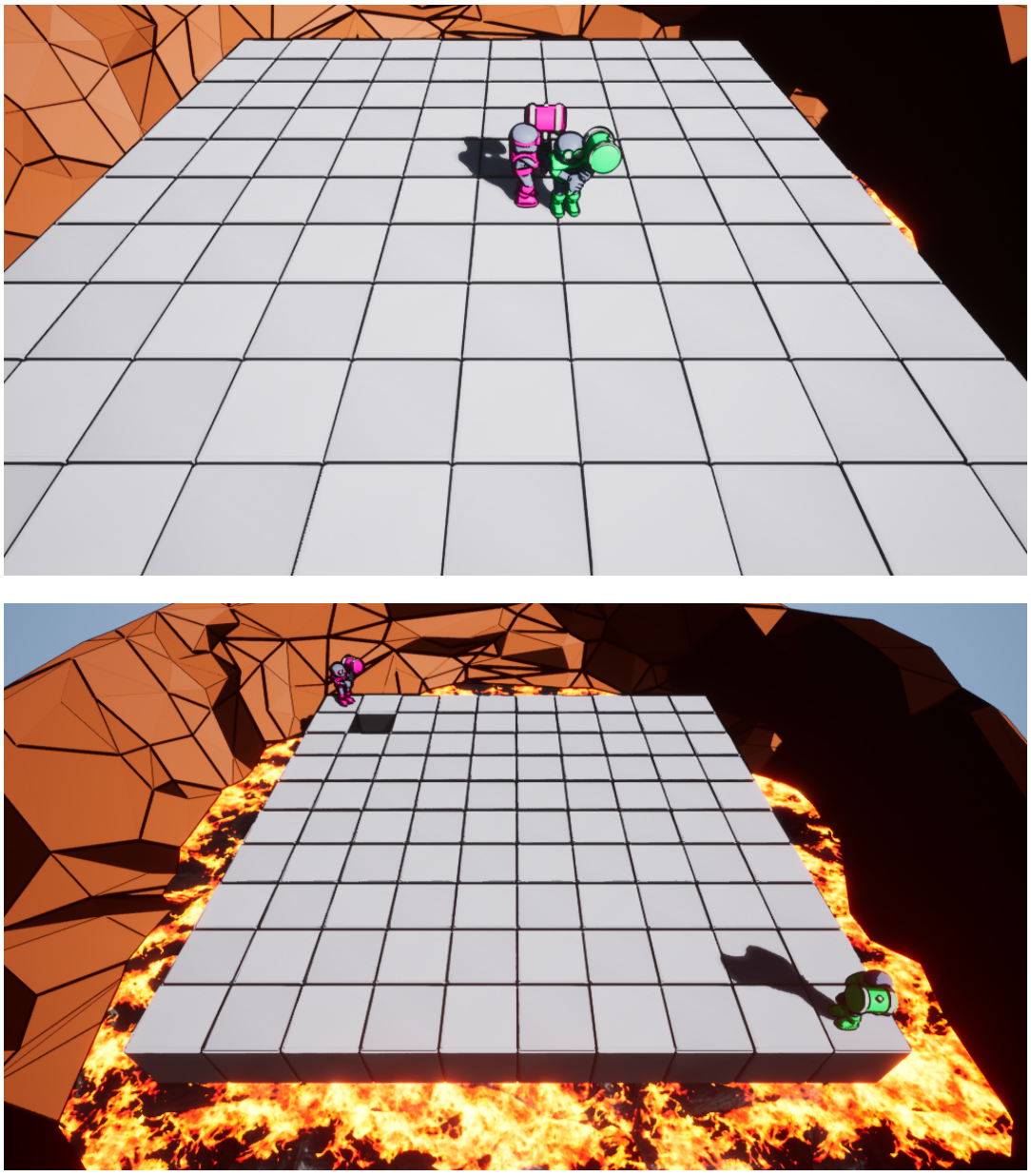

Figure 9: Camera Position and Zoom Levels Based on Player Position

use were hard to find in documentation such as AirControl and BrakingFriction.

8.2. Time Estimation

The biggest lesson I learned was how long everything takes. Stuff I've already done, I'm sure I could do faster the next time around, but in general, It takes much longer to do one thing, and you will always run into problems. Sometimes it's a missing vertex, a face that's too small, a joint that won't move right, a collision that won't trigger, or any other hundreds of things, but there are always small issues that will take much longer to fix than if everything had just gone perfectly the first time.

8.3. Code Organization

Since I plan to work on this game in the future and add in some final polish, it would have been very beneficial to maintain the code in a very meticulous manner. Even though it would have come with the sacrifice of not having a full game to submit, it would have made my life easier in the future. It's always better to put in the work to have good project organization ahead of time.

Figure 10: Fully implemented Main Menu and Player Customization

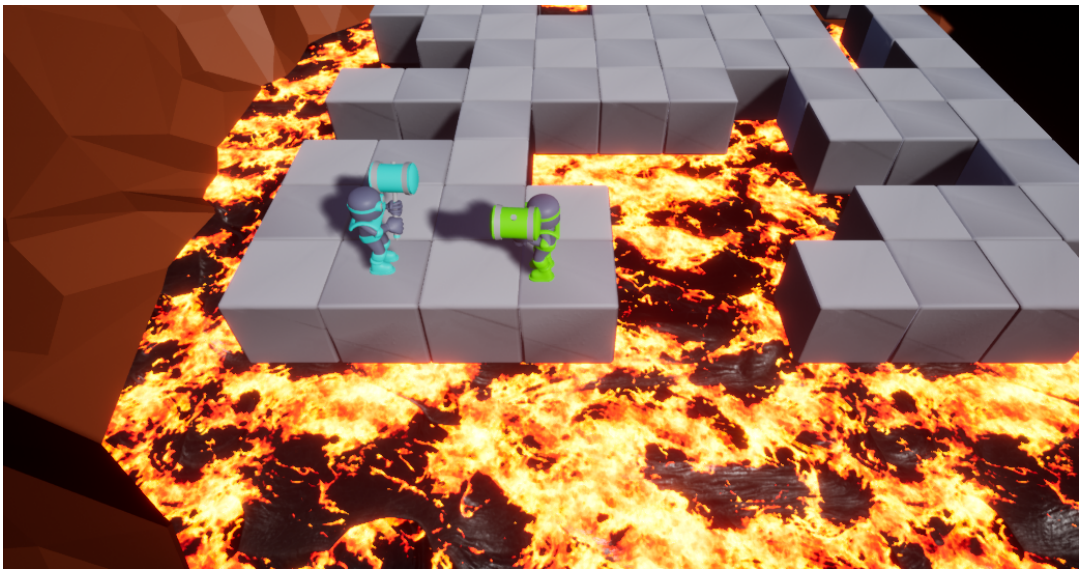

Figure 11: Cell Shading and Outline Disabled