David Wright

CPSC 2460 Distributed Computing

1 Dec 2024

Project Proposal

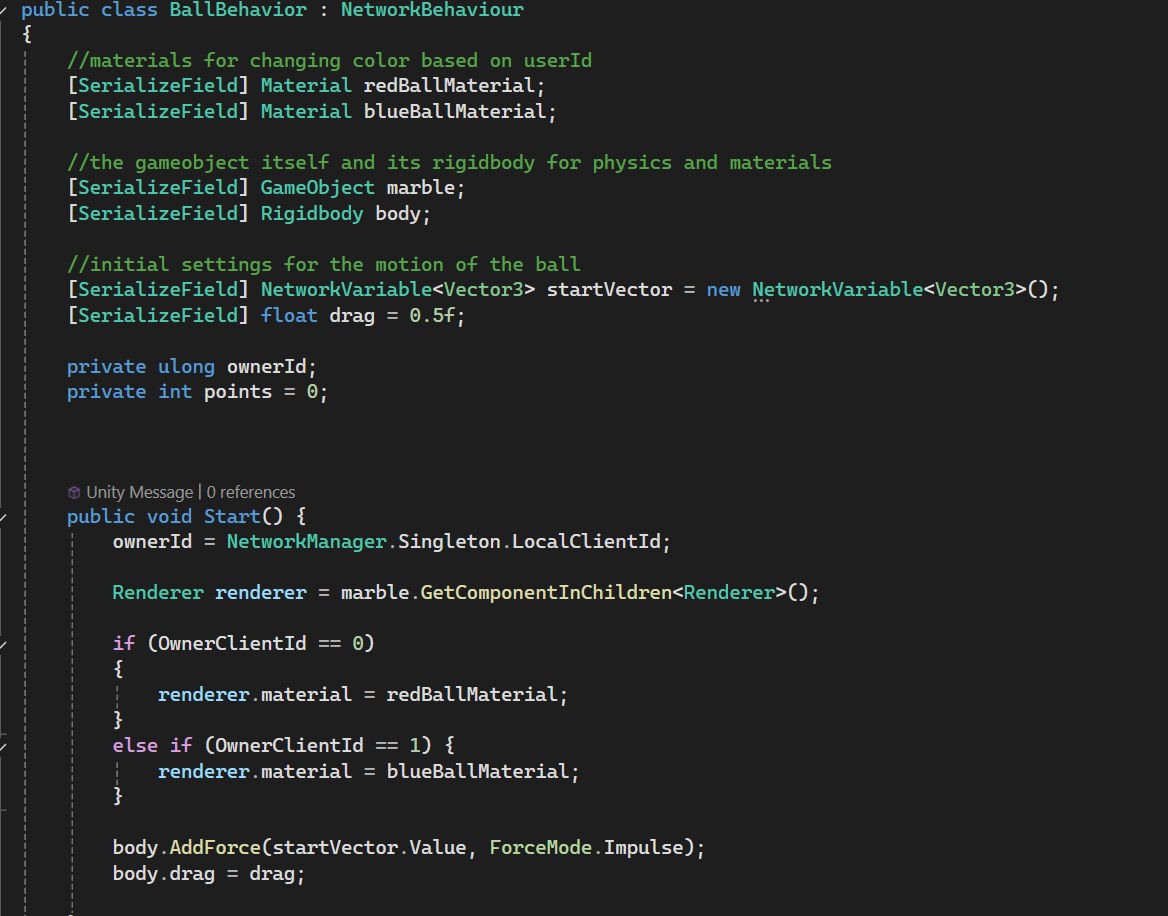
1. **Title**: Micro Mini Marble Madness
2. **Introduction**: “Micro Mini Marble Madness” (4M, for the rest of this paper) is a game designed around distributing the workload of each player in the game. It is a game in which players take turns dropping marbles into an arena in which they strike the other marbles already played and earn or lose points based off the other objects they hit before coming to a stop. Marbles that have been played will continue to gain or lose points based on the players who take turns after the marble is placed. The “distributed” piece of 4M is twofold. The first distributed piece is a mutual exclusion, which allows only one player to take a turn at a time. The second distributed piece will be in the keeping of the scores, which will have each client tracking the scores of their own marbles and then using an RPC to send it to the server to keep the overall game score updated.
3. **Background**: The basic materials used for this project will be:
   1. **Unity Game Engine**
   2. **Netcode for Game Objects**
   3. **Quantum Console (for debugging and scorekeeping help)**
4. **Methodology (with Visuals and Code Snippets):** The first step to making this project work was to set up the network environment. This involved importing Netcode and getting the network manager environment up and running, and ensuring that both client and host could see each other’s actions in the game. I also set up the basic network prefabs so that I could start adding objects into the program. I tested the project along each step, and added the player prefab in order to make sure that the clients were connecting to each other and the host over the network. I set up a basic movement structure for the player prefabs, though the players prefabs would later be removed because having them display on screen is going to be unnecessary for the project.

A screenshot of a computer

Description automatically generated

My next step was getting started on the marble objects themselves, getting the players to be able to spawn marbles of different colors, and getting the physics to work. To accomplish this, I created a prefab with an empty object and attached a sphere collider to it, then created a few different materials in order to be able to attach them to the individual players. When the marble spawns, it checks who the owner of the marble is, and changes the color of the material based on which player is spawning it.

Additionally, the marble has a networked Vector3D. I had some trouble getting this aspect of the spawn to work, and for a while the marbles would spawn with different initial vectors on each client. It turned out that the problem was being called by the server spawning the Vector, but not passing the information along to the clients, so while the client could see the marble’s changes after it spawned, it was receiving the marble with the default movement vector so the marbles were always spawning with the same (wrong) initial movement vector. I fixed this by attaching the initial vector as a network variable and adding the spawn behavior as an RPC in the player network code.

 A screenshot of a video game

Description automatically generated

The next step was creating the scoring for the individual marbles. I initially tried doing this with a simple collision detector, but I ran into an issue where marbles were counting their scores twice on all their collisions when they were hitting a wall or other marbles. After playing around with some debugging, I discovered that what was happening was that the marbles were actually bouncing, which was causing an unintended collision with the floor of the arena. I fixed this by adding tags for each of the game objects, so that they would only score when they hit targets that I intended for them to gain or lose points from. Additionally, I added a check for the collision to see if the marble was hitting an opponent’s marble or their own. As of this writing, I am still working out how I want the scoring to work, but the marbles gain points for hitting other players marbles and lose points for striking their own or hitting a wall. If I have time after getting the distributed portions working, I plan on adjusting the scoring to scale based off of the velocity of the marbles, but that may not make it into my final project.

A computer screen shot of text

Description automatically generated

After this, I needed to work on the ability for the player to choose which direction and speed their marble should spawn with. I thought about a couple different paths for this, including allowing them to type in a trajectory or a click and drag, but in the interested of convenience for the player I decided on using a button system. The player would be allowed to choose which direction and speed their marble should spawn at and then click on the screen to drop the marble, which would take its initial movement vector and proceed accordingly. I did use ChatGPT at this point, because I was having trouble getting the buttons to communicate with the network. I essentially ended up creating a server Rpc for each button, which sends the movement vector and/or speed out and sets it for each player in the game. One pitfall with this, is that if a player clicks immediately after their opponent takes a turn, they could accidentally spawn a marble with the same velocity and movement vector that their opponent just spawned one with.

A white line on a blue surface

Description automatically generated A screen shot of a computer program

Description automatically generated

The speed controls work in much the same way as the directional controls, but act as a multiplier for the initial placement vector.

A screen shot of a computer program

Description automatically generated

For these to work from the player, I need to create a method to initialize the balls before spawning in the BallBehavior script, which takes both the initial vector and speed, and initializes the ball before spawning it on the network.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

The turn based portion of the game is accomplished by a counter on the game manager that checks to see which turn it is in the game, and uses the modulo operator to determine which player’s turn it is. The player script only allows the player to spawn a marble on the server is the turn counter matches their client id. For the purposes of this project, I am only implementing a two player game, but it should be possible to expand to more players in the future by using a dynamic network variable that tracks how many players are in the game, and an array denoting which player id belongs to each player.

A computer screen shot of a program

Description automatically generated

The main distributed part of the game is in the placement of the marbles. Initially I had the players calculating their scores by going through all of the marbles on the table and adding their scores up if they belonged to the playing in question. What I realized was that this was an ineffective way of distributing work, because every client had to go through every marble in the game, and it wasn’t really “distributing” work, so much as duplicating it. I went through a few iterations of how I wanted to make it work, and I ended up settling on adding game objects, or “buckets” to the game manager for each player to place their marbles in their own bin, so that instead of searching the entire program for marbles, the client can look for its own bin, and simply add the points from their own marbles from there.

A computer screen with white text

Description automatically generated

A computer screen shot of a program

Description automatically generated

My final steps were getting the scoreboard to work, which included adding network variables to the game manager to track each players’ score and update them on the screen. When each player calculates their score, they send it to the server which updates this variable and then changes the scoreboard display on the screen.

A computer screen shot of a program

Description automatically generated

Once all of this was complete, I wanted to make some adjustments to the individual marble scoring system. It had been scoring 5 points when a marble hit an opponent’s, but I wanted to have this score change based on the speed of the marbles involved, to make hitting a marble going at a higher velocity worth more points. Initially, my plan was to look at the velocities of both marbles and subtract the velocity of the opponent’s marble from the attacker’s marble velocity to calculate the score to add, but I ran into an issue where they were not calculating properly. After some research, I discovered a thread on the Reddit forums, and it looked like the problem was happening because I was calculating the velocity inside of the collision method, which was actually capturing the speeds of the two marbles after the collision happened, when the attacker was slowing down and the defender was already speeding up. In order to fix this, I moved the velocity variable to the class instead of making it local, and set the updates to it in the FixedUpdate method.

A black background with white text

Description automatically generated

A screenshot of a video game

Description automatically generated

1. **Deviations**: For this project I managed to implement all of the features that I was planning on implementing, so there were no true deviations. I did end up changing the way that the game tracks marbles and calculates their score, by the previously discussed binning process, and I had not initially planned on having speed changes in the game but was able to add that as an additional feature.
2. **Conclusion**: In conclusion, 4M provides a fun environment that allows players to compete against each other by putting their physics skills to the test. It is a simple, game, but the amount of computations that will take place during collisions and the ability of players to see in advance how their marbles, and their opponents’ marbles, will react will add a good amount of strategy to the game, that, on its face, is a relatively basic concept. On the distributed system side, the game will leverage the power of both the host and the clients’ systems in order to spread the work around and perform computation for potentially hundreds of collisions at a time in a much more efficient manner.
3. **Reference**:
   1. Quantum Console, qsfw.co, <https://qfsw.co.uk/docs/QC/articles/quickstart/quickstart.html>
   2. Reddit forums, “rigidbody.velocity showing strange numbers”, <https://www.reddit.com/r/Unity3D/comments/nohm44/rigidbodyvelocity_showing_strange_numbers/>, accessed Nov 29, 2024