# Necessary Context

My mechanic is bullet ricochets where you can only do damage to enemies with the ricochets and not direct shots. My concern with this is the accuracy of the shots and how this might be too inaccurate for player to put this mechanic into a full-fledged game.

# Research

I didn’t do much research on hit-scan FPSs that have ricochets because there aren’t many, and the ones that do don’t feature it as a main mechanic, just like a special gun or something. This further made me think about how inaccuracy might be the main concern with constructing a game around this mechanic.

# Design Question

My design question for this is: Does the requirement of ricocheting a bullet off a wall to hit and do damage to enemies significantly affect the player’s accuracy.

# Design Measurable

My design measurable is the difference between calculated accuracy scores of sustained direct shots and ricochet shots. These accuracy scores are calculated using a target that gives points based on what ring you hit, with 1 point for the outer ring and 10 points for the bullseye. Divide this by the amount of shots that hit the target to result in the accuracy score.

# Changes to the Design

My base design was a very basic hit-scan fps with player controller that could perform bullet ricochets up to a given number of ricochets.

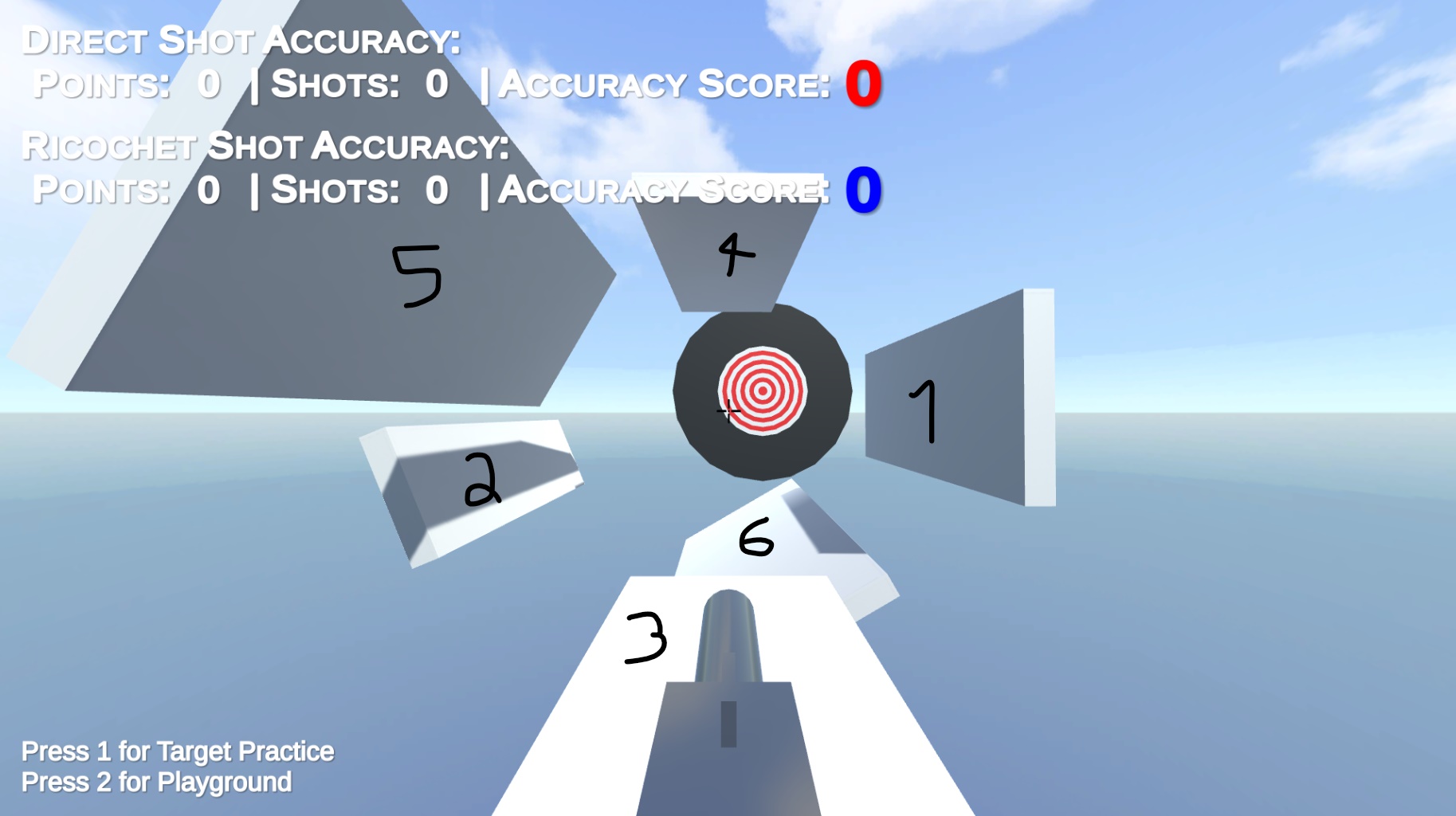
The first change I made to this design is that I added a scalable random spray pattern to the initial shots to increase the base inaccuracy of each shot. Before when there was no bullet spread it was way too easy to rack up perfect bullseyes on both the direct and ricocheted shots after you positioned your aim correctly.

My second change was to make the target move in a pattern to loosely replicate how you would be shooting at moving targets in an actual game.

# Results

After testing multiple shots with different ricochet paths off of walls I placed around, I got the following data (see *figure 1* for panel numbers):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Shot Path | Points | Shots | Accuracy Score | Difference to Direct Accuracy | Aiming Notes |
| Direct Fire (baseline) | 1001 | 106 | 9.443397 | N/A | Normal mouse aiming |
| Ricochet Fire (off panel 1 to target) | 961 | 113 | 8.504425 | 0.938972 | Aiming with flipped X axis |
| Ricochet Fire (off panel 2 to target) | 1002 | 124 | 8.080646 | 1.362751 | Aiming flipped in x axis and skewed in both axis |
| Ricochet Fire (off panel 3 to panel 4 to target) | 778 | 109 | 7.137615 | 2.305782 | Aiming with y axis majorly scaled |
| Ricochet Fire (off panel 5 to panel 6 to target) | 642 | 121 | 5.305785 | 4.137612 | Aiming with both axis skewed and majorly scaled |



With the results I got, it’s obvious that the more ricochets means the lower the accuracy is, but it’s also apparent that extreme angled panels, such as panels 2, 5, and 6, are also a big factor in the accuracy. So for this ricochet mechanic to serve to be useable in a full game, I should stick to basic angles and only care about single ricochets.

*Figure 1*

# Technical Requirements

I implemented inheritance in multiple effect scripts in order to gain more control over effects in my games that can execute all on their own. My base effect script used a basic life timer to handle automatic destruction when its effect is complete. I also calculated a 0 to 1 ranged number to dictate its lifetime, this was very useful for multiple animations over lifetime. An example is my DecalEffect.cs which scales the decal over its lifetime according to a given curve.