VR Dueling - Final Report

A Beginning to Multiplayer VR

ROBIN SCHMIT, University of Washington

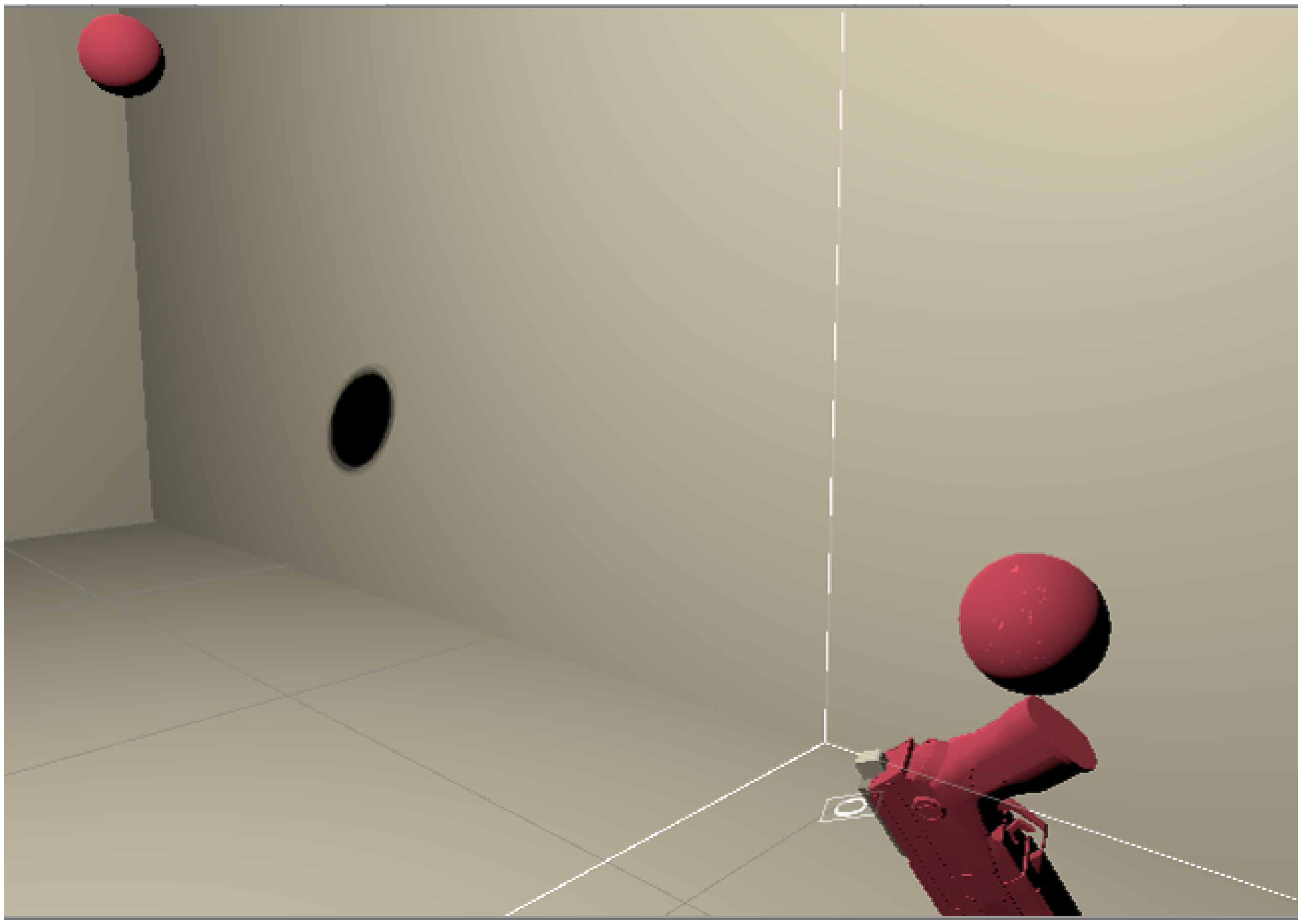

Fig. 1. For my project, I investigated multiplayer solutions for VR game development in Unity. Using Photon 2, I built a room in Unity that multiple clients could log into simultaneously. Each client appears as a ball to themselves and others, and carries a blaster that was intended to be the core of gameplay.

I wanted to try developing a multiplayer game for VR, in order to create something fun with only the essentials of a game. In order to do this, I started with existing tutorials for multiplayer development in Unity using Photon and for singleplayer VR game development. I was ultimately unsuccessful in creating an actual game, though I did get multiplayer VR working over a network. I reached a point where I have the knowledge to create a more up-to-date, helpful tutorial on the subject than the ones I followed.

Author's address: Robin Schmit, schmirob@cs.washington.edu, University of Washington.

1 INTRODUCTION

Multiplayer games are extremely popular on all gaming platforms, and that includes VR. Once I decided that I wanted to try VR game development, I realized that investigating multiplayer would both be a unique challenge and be potentially more satisfying to play given my own abilities. One of the cores of interesting 3D single-player games is satisfying visuals, which require substantial artistic work on the development side. Multiplayer games, on the other hand, can more easily get away with simplified visuals in favor of clean gameplay.

Multiplayer VR also offers unique opportunities to players, who can take ridiculous actions in VR that can then be riffed on by other players in a way not possible in standard multiplayer games. Many traditional multiplayer games offer taunts, emotes, or other actions that have no in-game purpose and serve only to joke around with other players. Multiplayer VR doesn't have to write the functionality for such behaviors, and other players can interact with whatever a player decides to do.

The necessary parts of game development in VR are a game engine, a way to handle HMD and controller input, and a network solution. I used Unity 2019, SteamVR SDK, and Photon 2, respectively. In order to familiarize myself with these systems, I looked for introduction-level tutorials. I used two of these, one for singleplayer SteamVR and one for using Photon in VR.

Unfortunately, these two tutorials were somewhat out of date, especially the Photon one. This means that it is currently quite challenging to go from no development experience to a functioning multiplayer VR game. This paper will document the challenges involved, ideally making the initial learning process easier for others in the future.

1.1 Contributions

- I created a scene in Unity of a closed room into which players load.
- I added local blaster functionality. The blaster spawns attached to the player's right controller and fires a projectile on trigger input.
- I successfully loaded two players into the same room from different executables, and they were able to see each other's avatar without being able to control it.

2 EXISTING TUTORIALS

Our First Social VR App [1] is one of the only existing video tutorials on using Photon in Unity for VR, and it is a tutorial for the original Photon, now referred to as Photon Classic. It covers creating a network handler to load players into the same Unity scene, and also discusses audio solutions. Combining this with the up-to-date Photon 2 tutorial[2] that exists on their website but does not cover the details of Unity integration gets almost all the way to a functioning multiplayer scene.

Blaster Weapon for SteamVR 2.0 [3] gives an introduction to building a singleplayer VR shooter using SteamVR SDK 2.0. It's relatively up to date, since SteamVR SDK is currently at version 2.5. It doesn't cover creating enemies for a game, but it covers creating a weapon, using it to fire projectiles, and detecting when those projectiles hit other objects. These are very useful things to cover for introductory shooter gameplay, but generalizing them to multiplayer using Photon is surprisingly challenging.

3 METHOD

In order to create any introductory multiplayer VR application, there are three distinct necessary components: a game engine (and development environment), a device handler, and a network handler. Each of these already exists in several different forms, the challenge comes in integrating them successfully. In order to achieve this, having a general understanding of each component is vital. Without such an understanding, it is nigh-impossible to step away from examples in any way. Now, the place to integrate the above components is the game engine/development environment. This is the system designed for the developer to create and customize, unlike prefabricated device handlers.

Additionally, having a fully-functioning development environment that allows you to fully test your application is crucial. For multiplayer, this means having the ability to host the game and connect to it from multiple devices simultaneously. Many challenging bugs will not show up when testing with a single player, and testing every step of development is key.

4 IMPLEMENTATION DETAILS

In order to create a multiplayer game in VR, I used Unity, SteamVR SDK 2.5, and Photon 2 as my engine, device handlers, and network handler respectively. As of time of writing, these would be my recommendations for introductory multiplayer VR development. Unity is the classic introductory game engine, and Photon is the only option for networking. Unity used to have its own networking solution, but it is currently deprecated. SteamVR provides a hardware-agnostic Camera Rig prefab which handles HMD and controller input perfectly. Many sets of prefab objects and visuals also exist on Unity for common game objects. I used one set of blaster prefabs from VR with Andrew [3] for my specific application, which was intended to be a firearm dueling setup. As for hardware, I used two sets of Windows Mixed Reality (WMR) headsets and controllers. The relative cheapness of these devices is a win, since having two HMDs for testing purposes is a necessity, and the application will not function without an attached VR device.

SteamVR SDK and Photon can both be installed into Unity as plugins from the Asset Store. SteamVR SDK requires Steam and SteamVR installed, and using SteamVR with WMR headsets requires Windows Mixed Reality for SteamVR, available on Steam. Photon, meanwhile, requires an account and an app on their site (both free). The app id of that app is used in the import of Photon to connect the Photon plugin to a server, which the Unity build will use as its network point. Photon will also generate a file called PhotonServerSettings, which can be configured in Unity. One thing to make sure to do is to set your fixed region to be whatever region Photon believes to be the region with the lowest ping for you. This will prevent you from later discovering that your app does not connect players on disconnected devices because they are somehow in different regions.

Once the Unity project has everything installed, the creation of one multiplayer scene is relatively simple. Following the Photon 2 tutorial [2] guides you through the creation of a network manager script which needs to be added to the scene, as well as basic room components for the scene. There are apparently a number of challenges with Photon objects in multi-scene games, but I did not reach those challenges in my project and so they are outside the scope of this paper. The network manager handles which room a player loads into, allowing lobby-style matchmaking (with some extra work) or simple random matchmaking.

The final challenge for a multiplayer VR scene is projecting the controller movements over the network, so that each player can see the other and their movements. In Photon, the way to create objects visible over the network is to use PhotonNetwork.Instantiate() on GameObjects which are public variables in the scripts, and actually chosen in Unity. This tells Photon to create a copy of the object in each other player's local scene. Additionally, you add a PhotonView object and a PhotonTransformView object so that the rotations and translations of the copied objects are synchronized to the original object's behavior. Note that unlike Unity's regular Instantiate(), the first parameter to Photon's Instantiate should be the name of the object rather than the object itself.

Unfortunately, PhotonNetworkInstantiating a SteamVR CameraRig object is a mistake. It successfully creates another CameraRig object on the other player's local scene, but this has a significant problem. The two CameraRigs in the local scene both respond to the local player's devices, so moving your device moves where the other player appears to be. The traditional Photon solution to this problem is to check in the controller input handling whether the local object's PhotonView.IsMine(), which is a variable that Photon objects have that says whether they were created locally or as the copy of an object somewhere else. This would probably work here, but it is difficult since the code handling controls is part of the SteamVR SDK. Instead, the solution suggested by deraggi on a Unity forum in 2016 [4] is to Photon Instantiate 3D objects (spheres in my case) as children of the CameraRig object, which is only instantiated locally. This (plus a short script) means that the children are moved when the devices that control them are moved, but are not moved by controllers that should not control them.

5 EVALUATION OF RESULTS

Overall, while I did not successfully create a multiplayer VR game, the part that I failed to deliver was the game part, and that was simply a function of running out of time debugging the network part. I did successfully create a scene into which multiple users could load and see each other's movements approximately correctly. Unfortunately, while the networked player avatars seemed to move relatively correctly according to the movement of their headsets, I never got networked projectiles working in order to really test the latency and accuracy of Photon.

6 LIMITATIONS AND FUTURE WORK

Obviously the future of this multiplayer game development project would be to finish developing the multiplayer game. I believe that it would not take substantially more work in order to get gameplay functionality: projectiles firing across the network and projectile-player collisions having some kind of effect (score, hit point loss, etc.).

However, this would leave one significant unsolved problem, and solving it would require different logic for networked projectiles than for networked players. That would be detecting projectile collisions with player avatars. The best way to handle this would be to detect weapon fire and to spawn projectiles locally in the opponent's scene, so that it could handle collision detection there. This would likely be done with a remote procedure call. This would make sure that if a projectile hit a player, that player would believe that they were hit, which is important for fun.

Additionally, the current scene does nothing to prevent a player from walking through the scene walls. The way to do this in single-player is to move the scene with the player when they try to do this. A similar solution might work in multiplayer, but it would require both moving the copied avatars in the local scene and not moving the copies of your avatar in the non-local scenes.

7 CONCLUSION

This section is somewhat redundant with the abstract, but is often included in publications. This can be a single paragraph that is focused on emphasizing the “take home” messages of your work. This is also a good place to frame why your work matters within the AR/VR community and what might happen as a result of it, or if more researchers started working on this topic.

While I was not able to create a multiplayer VR game, I was able to create a multiplayer VR scene. This would be an excellent starting point for any inexperienced developer interested in multiplayer VR, as I cover the current relevant plugins while also collecting useful past tutorials and mentioning a couple of the major potential pitfalls. I believe that with access to this, another student could make progress on the same project far more rapidly than I was able to.

ACKNOWLEDGMENTS

Special thanks to Kirit Narain, whose advice on existing solutions led me to Photon, and who helped me troubleshoot multiple persistent issues.

[1] FusedVR (2016). From Single to Multiplayer, Creating our first Social VR app. https://www.youtube.com/watch?v=GEi_j7JUG-4

Note that underscores have to be escaped for LaTeX, and that the URLs did not originally contain backslashes.

[2] Photon 2 Tutorial (2020). https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

[3] VR with Andrew (2019). [Unity] Blaster Weapon for SteamVR 2.0. https://www.youtube.com/watch?v=QUCPh9ZuryU

[4] deraggi (2016). Vive multiplayer problem - Unity forum.
https://forum.unity.com/threads/vive-multiplayer-problem.427730/