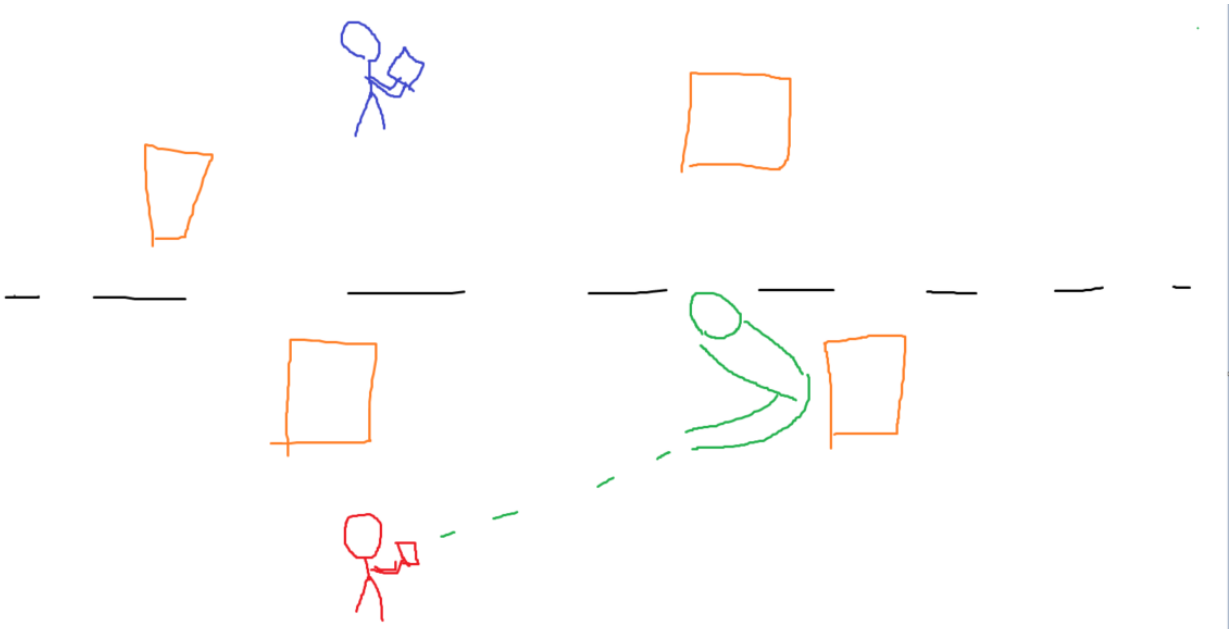
Before we jump into the boring text(reading is hard 🙂) here is a link to a short youtube video that shows you the basics of the game demo I made with Lightship!
https://www.youtube.com/shorts/j7DxJ801yaQ

Alright without further adieu let the saga begin!
Months before I was even aware of this Niantic Lightship Project Library Initiative I was creating demos of games that users could play outside on their smartphones. I love video games, the ability to provide fun challenging experiences is great, but I also knew I needed to get some more outside time in. Working on a computer all day and then playing video games isn't always the healthiest lifestyle for me, so when this Library Initiative started I was excited to apply.

I pitched the idea of Cube Crusher, in this game you would play outside and hit an AR ball into AR cubes to crush them. crushing all the cubes would win you the level. Here is the first concept art mockup for the game, I'd like to tell you that it was created by my very young and clearly artistically challenge child, but this masterpiece was created by yours truly 🙂



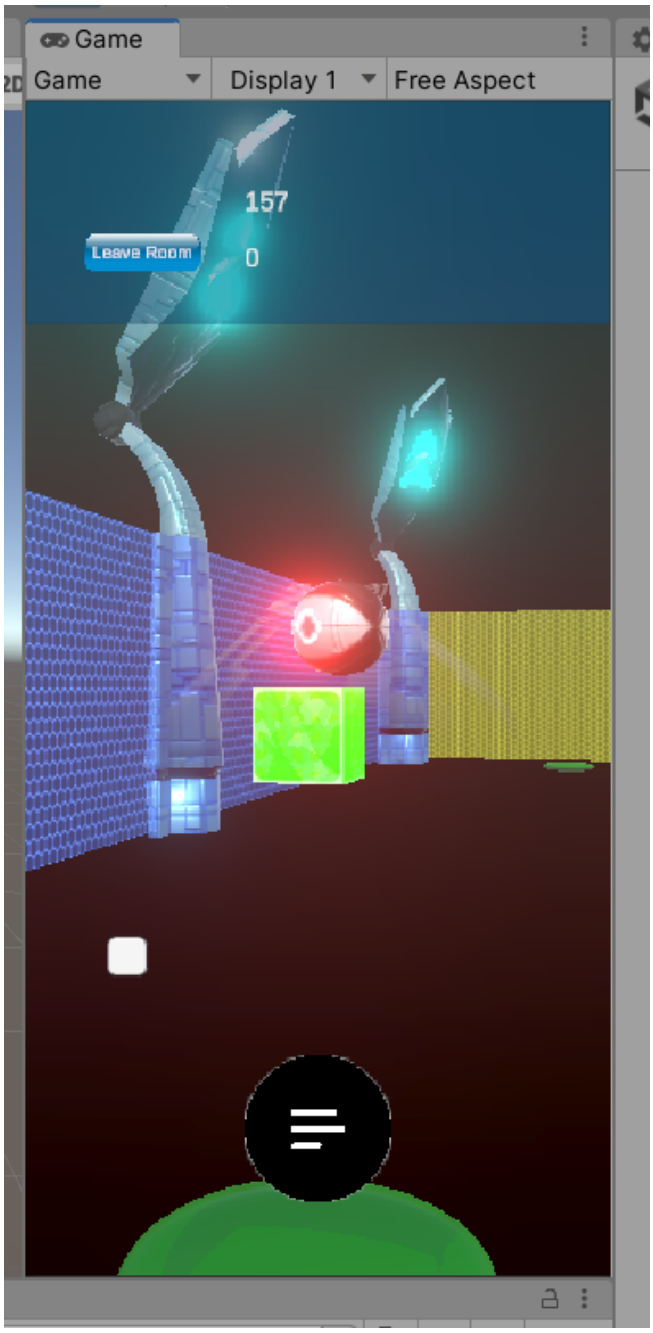
Easily my favorite thing about AR is being able to move around and in some ways use your physical body to interact with a fantastical game world, so Cube Crushers, the game I decided to make, focused heavily on that. If you really studied the above concept art in detail you might have noticed two stick figures(the red one and the blue one) this is because I wanted to make the game support Shared AR since I love playing video games with family and friends. Plus community is part of Niantic's core values and as an added bonus I've been wanting to get my hands dirty with Unity's new networking solution, NGO(Netcode for GameObjects), for awhile now. I love learning about new technology so I thought why not learn about multiplayer at the same time I learned about Lightship, what could possibly go wrong 🙂

Getting started with Lightship was pretty straightforward. I did run into some bugs when adding in SharedAR, but the Lightship support said those should be fixed in an upcoming release. For those of you who don't know Lightship supports two types of multiplayer, Image Colocalization and VPS colocalization. Basically you can either have everyone playing your game look at the same image to synchronize game worlds or they can look at a VPS location to synchronize game worlds. I chose Image Colocalization since I A) don't have many VPS locations around me(I live in a small town) and B) I wanted users to be able to play my game in wide open fields, like a park, baseball field or tennis court, none of which would make for a good VPS location. The fact that I use Image Colocalization means that to player Cube Crusher multiplayer you need to print off a copy of the picture you have to scan and then lay it on the ground between the players. This works pretty good but is certainly a bit of a pain, especially when it is windy outside and you have to weigh the four corners of the piece of paper down so that it doesn't blow away!
Alright so now that I've introduced you to my game let me dive into some of my thoughts on Lightship and NGO!

Lightship thoughts

Overall lightship is really easy to add to a project and has some super cool functionality. Testing/debugging can be challenging with realtime

multiplayer experiences but Playback is a great way to see how your AR features work in editor. I made my game playable via the Unity Editor with mouse and keyboard for testing which is super important to increase iteration speed. Here is a pic of what it looked like when playing, nothing fancy but it got the job done.



The online rooms were super easy to set up and use but I wasn't exactly sure how I should go about cleaning them up after matches were complete.

The multiplayer synchronization is very good and definitely one of the highlights when you can launch something towards your friend in AR! I did run into some issues with the shared AR experience desynchronizing during matches, I find that this specifically happens a lot when playing more then one match so I don't think the AR session is getting reset properly, even though I did some googling and tried to reset the AR session a couple different ways. The devices desynchronizing is definitely a pain, but it reminds me of how wii motes would sometimes get desynchronized. As long as the game dev accounts for this and builds in a pause functionality to allow the players to resync(maybe desynchronization could even be detected and the player could be prompted to resync) the impact can be minimized. That said I didn't get around to trying to implement this myself ;). Another thing that would be super cool is if the phones could communicate faster since they are right next to each other(maybe via bluetooth or something, I'm sure Niantic has put more thought into it then I have 🙂). Right now since all my messages are going through a relay server it seems that a hundred plus milliseconds of latency is pretty normal(which isn't bad by any means, but one can always wish). I also tried to implement Occlusion but it was always causing flickering, for example the in-game cyber walls would flicker, or the ball would flicker near the ground. I tried to add semantic segmentation suppression but still ran into the same thing so I decided to take it out, I might have just not configured it right though.

NGO implementation

A big part of any multiplayer experience is the way you implement the netcode. I found my demo to pose a special challenge since it was pretty real time/fast paced. I tried using network transforms but had multiple issues going down that route, so instead I had the balls motions simulated separately on each client and only synchronized during certain events like

when a player hits the ball or when the balls position gets reset because it hit the back wall(Which only happens in multiplayer). Also in order to help mitigate the effects of latency I had the ball move slower for the client that just hit it. That way the ball would hopefully get across the field to the other player at about the same time on each client. This is purely an implementation detail, but the way I used NGO a lot of times was by splitting what normally would be one function into 4.
Starter function : This is the function that would be called on the client that wanted something to happen, it would immediately call the Action function so that the local client could see the effect right away and also call the ServerRpc to synchronize the activity.
ServerRpc : This would just call the ClientRpcs to synchronize the action
ClientRpc : This would call the Action function on all the client except the initiating client(since the action function was already called on that client)
Action function : this would actually do the thing that I wanted to have happen, this function would pretty much just be a normal Unity function.

Overall the networking in the game is not production ready because it doesn't use enough interpolation and smoothing, instead the ball will teleport around sometimes, but it works and allows you to have a fun time with friends!