**PROJECT GREEN** (because we don’t have a cool name yet)

*Project overview*

Project Green is a game about building spaceships from the ground up, and taking them into fleet warfare with and against other players and their ships.

The core of the game is the interaction between the crafting and combat mechanics. Losing parts of a ship in combat should affect the abilities of the ship, and protecting your ship’s key components while destroying the enemy’s is the main challenge of combat.

There are a fair number of open questions about the design of the game.

One of the main ones is of course how do all the different parts work – how do they affect gameplay, how do they affect the cost of your ship, how do they all fit together, etc.

Another one of the key problems is how people are going to engage in combat – whether it’s an open world, some sort of matchmaking system, etc. For this, an open world would be incredibly interesting, where people compete with each other naturally over resources for trade and territory. However, it’s a lot more work, though probably possible.

*Building the project*

Download Unity (<http://unity3d.com>) and clone the git repo (<https://github.com/liam-mitchell/green>). Open Unity, select ‘Open project’ and choose the folder containing the git repo. Open the ‘Builds’ menu from the toolbar and build each of the three options.

*Playing around*

Now, open the git repo in your file explorer and go the the ‘Builds’ folder. Start the two servers (RegionServer.exe and PlayerDataServer.exe) in windowed mode and wait for them to load past the splash screen. Then, start the client (also in windowed mode – the UI doesn’t work in full screen), enter a username and click ‘Start Game’.

Right now you can’t really do much – so press ‘escape’ to enter the ship editor. Drag parts from the parts menu (the UI may be pretty messed up, depending on your resolution) into the center to add them to the ship, and press ‘escape’ again once you’re done. The game should load the previous scene with your newly constructed ship in the middle. If you have engines on your ship, you can press W and S to accelerate forwards and backwards.

*Exploring the code*

Client code (UI, cameras, editor stuff, etc.) is in the Assets/Scripts/Client directory, while server-specific stuff (connection management, player data storage, messaging, etc.) is in the Assets/Scripts/Server directory. Things that need to be shared (generally, player state updates like movement, adding/removing ship parts, etc) between both are in the Assets/Scripts/Shared directory.

Networking-specific code is in three places. Assets/Scripts/Client/PlayerClient.cs has code for managing the player’s connections to the servers. Assets/Scripts/Server/PlayerDataServer/\*.cs has code for managing the player data server (ie. Persistent storage of player data when they’re not signed on – this will probably be a database wrapper eventually, but for now most things are in memory). Assets/Scripts/Server/RegionServer/\*.cs has code for managing active player connections (currently in the poorly-named ShipServer.cs, but that should change sometime in the near future). Most of the other \*.cs files in Assets/Scripts/Server are from earlier experiments before the crafting system was in place, though some of it will be relevant when we start adding those pieces back onto ship parts.

Code for managing the state of a ship, serializing it, spawning it and synchronizing it is in Assets/Scripts/Shared/Ship\*.cs.

Finally, code for the ship editor is in Assets/Scripts/Client/ShipEditor.cs.