**Modular Networked RTS Brief**

**Third Party Libraries:**

Mirror (Networking Framework)

The project will be using Mirror to handle the networking aspect of this, in the sense that it will deal with syncing between client and server. It will also provide the framework for allowing for a more complex and specific implementation of it to be used for a modular RTS system for example.

The Modular RTS System will expand upon the Network System provided by Mirror and create a more catered experience for making networked RTS games. In this way the system that is created should work seamlessly with Mirror and allow for a streamlined design path for future such RTS games using the modular system.

**Modular Network System (Core)**

There will be some aspects of the project that are detached from the RTS side of it, these will be components that can be used entirely separately to the Modular RTS System.

Network Lobby:

A very handy thing to have will be a fully functional Lobby where players can meet up and find games to play in. This isn’t essential but it will be a definite nice to have for any RTS game, or really any networked game in general.

**Modular RTS System (Core Mechanics):**

A modular RTS System entails a system that can handle staple design elements of a RTS System with ease of use. To build such a system there are a few necessary components that will be developed. As a side note, there are essentially two Modular systems that will be built here, the RTS System as well as an AI System that will work in conjunction with it.

Resource Management:

To manage any resources that are collected in the game to allow for player progression, e.g one resource allows you to upgrade tech and another is used to upgrade buildings.

It would be a good idea to make this as modular as possible, meaning that the end designer would be able to input their own resource types and cost of items and such.

Unit Movement:

Another important aspect of most RTS games is the ability to move pieces (or units) around the field of play, this would mean that a player would need to be able to select their units and in one way or another be able to manipulate them around the field of play (generally done by mouse click but other options exist).

In conjunction with unit movement there must also be a way to select the units, a popular implementation of this is to click on one of your units to select individuals, or drag over a group to select many.

Combat:

With the use of units there is generally some sort of combat involved (this isn’t necessary however) which will dictate the necessity to give the end designer the option to add combat in through this modular system if they so desire.

The idea will be to add in a basic (or seemingly basic combat system) into the modular system to allow for this as an option. It would be nice to be able to build something similar to StarCraft’s and Age of Empire’s combat in this regard, which is as far as I can tell units with different stats that have slightly different ways they target and attack (it is almost certainly more nuanced than this but this is a good starting point).

AI:

AI will be a large driving force for all of the previously mentioned systems so far, in its simplest form it will be used to make a unit go from point A to point B. For say Resource Management a unit would need to go from point A to point B, collect resources, go to a resource drop zone and then go back to point B to collect more.

To make this as re-usable and effective as possible a mini modular AI system will be developed alongside the Network and RTS specific content as it is a core part of a RTS. With enough time the goal will be to make an AI system that is capable of playing the game against the player (allowing for the option of a PvE mode).

A PvE mode is not the main goal but it would be very good to have given that almost all RTS games will have their own form of a PvE mode integrated alongside PvP.

(PvE – Player vs Enemy, PvP – Player vs Player)

Building:

Many, if not all, RTS games have some form of building mechanisms which serve to further your economy, army or anything else. A lot of these also tend to have building/tech trees that have prerequisites to for buildings.

It would also be necessary to create a simple building/tech tree system to allow for building upgrade paths, and tech upgrade paths. These would ideally be done in a way that can handle both buildings, tech and anything else the end designer could dream up. It will need to be as generic a system as it can so that it will work with any number of different use cases.

Action Queue:

An important part of RTS games is speed and efficiency of the player, one incredible way to free a player to do more things at once and lighten the load of micromanagement is the ability to make AI or buildings queue actions.

This could be as simple as telling a unit to go from point A to point B, and then to point C when it reaches point B. This is an extremely helpful asset to allow players to use and can make for some much more interesting and exciting gameplay. It will also allow for a designer to test many different/complex scenarios at once with much more ease.

**Runtime Debugging Tools**

Having debugging tools integrated into the system will be invaluable to anyone (including myself) using this modular system. Allowing the end designer to know what’s going on at all times will be helpful to spot issues and fix them a lot more efficiently than normal.

Ideally each of the larger subsystems will have their own debugging tools specific to their implementation, this will mean that it will be easy to debug the AI and the Building individually. While the debugging will be more useful for some systems rather than others it will be a helpful tool in either regard.

A solid debugging environment will be developed for each of the core mechanics (AI, Resource Management, Building, Upgrade Trees). I didn’t mention Combat and Movement separately because they will both be driven by the AI which will have its own debugging system already.

Many of these debugging systems can (and likely should) be implemented into actual gameplay so that things like Action Queues are exposed to the end user which are important for decision making. It's always nice to allow players the freedom to know and understand what their units and buildings are doing at any particular time (especially in a RTS like game).

Debug Interface – Either a HUD or world canvas

AI:

For the AI it would be handy to know what they’re thinking at any given time, the intention will be to have a Debug Interface that displays the AI’s current thought process and weighting behind it to see what’s going on in its brain and the current queue of actions of this AI.

For further detail it would also tell you the main function of the AI (is it combat, utility etc). For groups of selected AIs, the implementation will either be to show one or all of the thought processes depending on the desire of the debugger.

Building:

It would be nice to know the details of a building at runtime, such as the upgrade paths required and its current list of queued actions if it can queue any.

This would be another Debug Interface item that would display the currently selected building, other details may be present depending on other important information that may be present to that building.

**Test Program**

The test program of this modular system will essentially be a RTS game that incorporates all of the systems and subsystems created (AI, Resource Management etc) in a clean and simplistic way that highlights them effectively. This will be a useful exercise in seeing how all the different components work together and if any improvements need to be made depending on the interaction of the systems.

Separate test programs will be made for each individual subsystem to make sure that they each work within their own right and can be stress tested without worrying about something unrelated causing problems. They will also serve to highlight each subsystem on their own to allow for easier testing.

The end product of this will be the test program with hopefully some nice assets to use (not a necessity but makes it look nicer) rather than simple shapes which will be used for testing purposes. Using more practical assets will also show that the system can actually be used with anything which is the intention of a modular system.