Technical Documentation

System overview

The artefact is designed around a menu system for spawning interactive entities. The user controls the system using gestures appropriate in a realistic environment - for example, grabbing a virtual helicopter is done by clenching the fist upon the real world position where the helicopter is represented. Buttons are activated by *pushing* them with a finger, and a list of buttons is navigated by *scrolling* a fingertip across their surface.

These actions are utilised to quickly create mission simulations. The user builds a terrain of mountains, trees, obstacles, and targets, before planning out troop deployments and manoeuvres. 3D line drawings can annotate positions, flanking routes, and supply lines. This can then all be played out. Manually controlled troops will follow their chosen routes, meanwhile AI units will find the fastest journey to their goal. This is all editable in real-time, allowing units to adjust on the fly to dynamic terrain changes.

Requirements

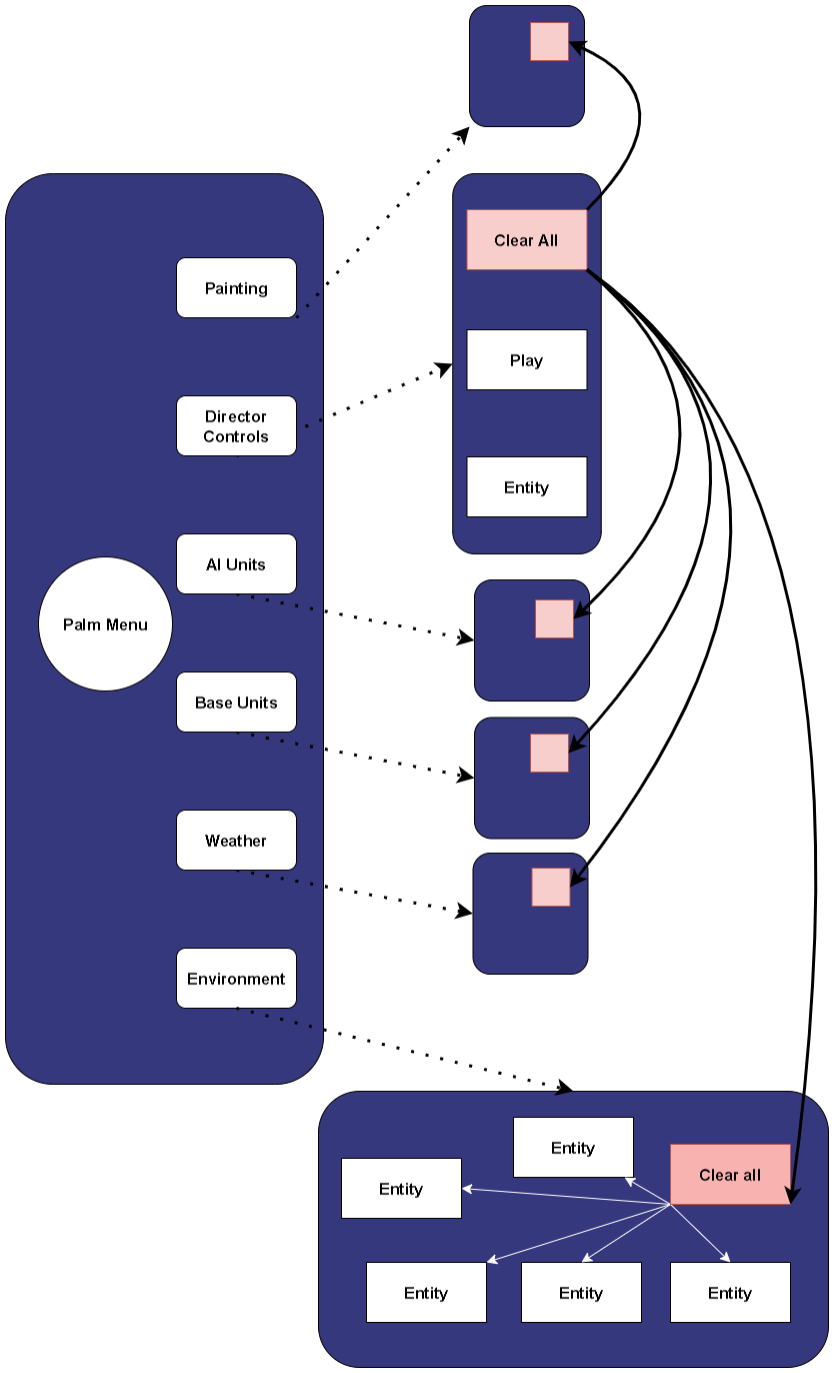
The project is installed using the apk build. Details on this are provided in the *Installation Guide* document. This is installed onto a Head Mounted Display (HMD) with hand tracking. The team's choice of HMD is the Oculus Quest 2 as our organisation, the Queensland University of Technology, has allowed us to borrow a few sets during the development period.

Opening the project files in Unity (2021.3.15 or later) and accessing the Package Manager will reveal key dependencies for the Virtual Reality libraries:

1. The Oculus XR Plugin - version 3.2.3
    a. This provides display and input support for Oculus devices. This is vital as the project was edited, tested, and installed on an Oculus Quest 2. The installation guide is focused on this device too.
2. The Open XR plugin allows developers to target a wide range of AR/VR devices. This allows the team to build the project for non-Oculus devices.

Detailed Description

Menu Layout Diagram



The Palm Menu can be hidden and shown by pressing the index and middle finger of the right hand onto the left wrist - like checking for a pulse. Choosing a Palm Menu toggle activates that submenu and deactivates the rest. This leaves only 2 menus - 1 of which can be hidden - shown at all times. Each submenu contains a list of entities available to create, and a button to delete all spawned entities of that type. For example, choosing the Clear All button in the environment submenu will delete all environment entities on the board. The Director Controls submenu has a Clear All button that combines all of the other submenus Clear All. Using this will wipe the desk clean for a fresh start.

Spawning an entity will place it within the Game Manager's list of spawned entities. From here, operations can be applied en-masse, such as removing all of one entity type, or running unique functions per troop pathing method.

Code samples

The code written in this project is all relatively simple algorithms that iterate through lists of gameobjects, whether they're spawned entities or waypoint markers. The Oculus Library code was successfully implemented to handle the VR interactions. Thus, there is no amazing or ground-breaking algorithm that solves a major problem – the code instead focuses on being simple to understand, with generic functions that can apply in different situations. For example, the script to stick troops to the table height is the same script to stick waypoint markers onto the table. Likewise, the waypoint markers and all troops are derived from a singular *grabbable* gameobject with variations on the visual designs and interactions.

Regardless, here are a few pseudocode examples to display how common actions were organised.

Remove All Items via Tags

function RemoveAllItems (tagName);

Input: One string

for each *item* in the *list of spawn items* **then**

if the *item's tag* = to the *tagName* **then**

if the *gameManager's units list* contains the *item* **then**

remove *item* from the *gameManager's units list*

end if

if the *gameManager's targets list* contains the *item* **then**

remove *item* from the *gameManager's target list*

end if

destroy the *item*

end if

end for

Spawn Items via Names

function SpawnItem (itemName)

Input: One string

for each *item* in the *item list* **then**

if the *item's name* = to the *itemName* **then**

instantiate the *item* as *newItem*

if the *item's tag* = to "Manual" **or** "Automatic" **then**

add *newItem* to the *gameManager's units list*

end if

end if

end for

Follow a List of Waypoints

Function Follow Waypoint()

Input: Ordered List of Vector 3 positions

if followingPath **then**

Set *moveTarget* to *markerHolderList[markerIndex]*

if distance between (*this* position xz, *moveTarget* position xz) > 0.03 **then**

Set *destination* to the *moveTarget*

Look at *moveTarget*

Set *moveTarget* colour to Red

else if *markerIndex* < *totalMarkers* **then**

Set *moveTarget* colour to Black

Increment *markerIndex* by 1

If *markerIndex* equals *totalMarkers* **then**

Set followingPath to false

End if

End If