How-to-use
AAXLP Software Module

Introduction

This document aims to teach the reader how to use AAXLP's software module in their VIROO-based Unity VR application.

Installation

Packages

The software module is packaged in a standard Unity package format, which can be opened in Unity version 2022.3.40f1 or higher. The Unity project must have all the necessary packages (including VIROO itself) installed for AAXLP's package to work.

The following is a list of packages utilized by AAXLP, which you can install using Unity's Package Manager or by following VIROO's documentation (excluding the packages you would have by default when making a new project using Unity's Universal 3D template):

- Viroo and all of its dependencies (Follow VIROO documentation to install, AAXLP is made with Viroo Studio version 2.6.941)
- OpenAI Unity (Add by git URL: https://github.com/srcnalt/OpenAI-Unity.git)
- Unity Sentis (Add by name: com.unity.sentis)

After installing the necessary packages, you can install the AAXLP package file. The package can be imported into Unity by using the "Assets/Import Package/Custom Package" menu. Additional integration requirements are listed below:

Layers

Both VIROO and AAXLP's physics systems use a custom layer configuration. You can automatically set VIROO's needed layers from the Viroo Project Validation window, but you will need to manually assign AAXLP's layers 16-21 exactly as follows:

16- Floor

17- Placing

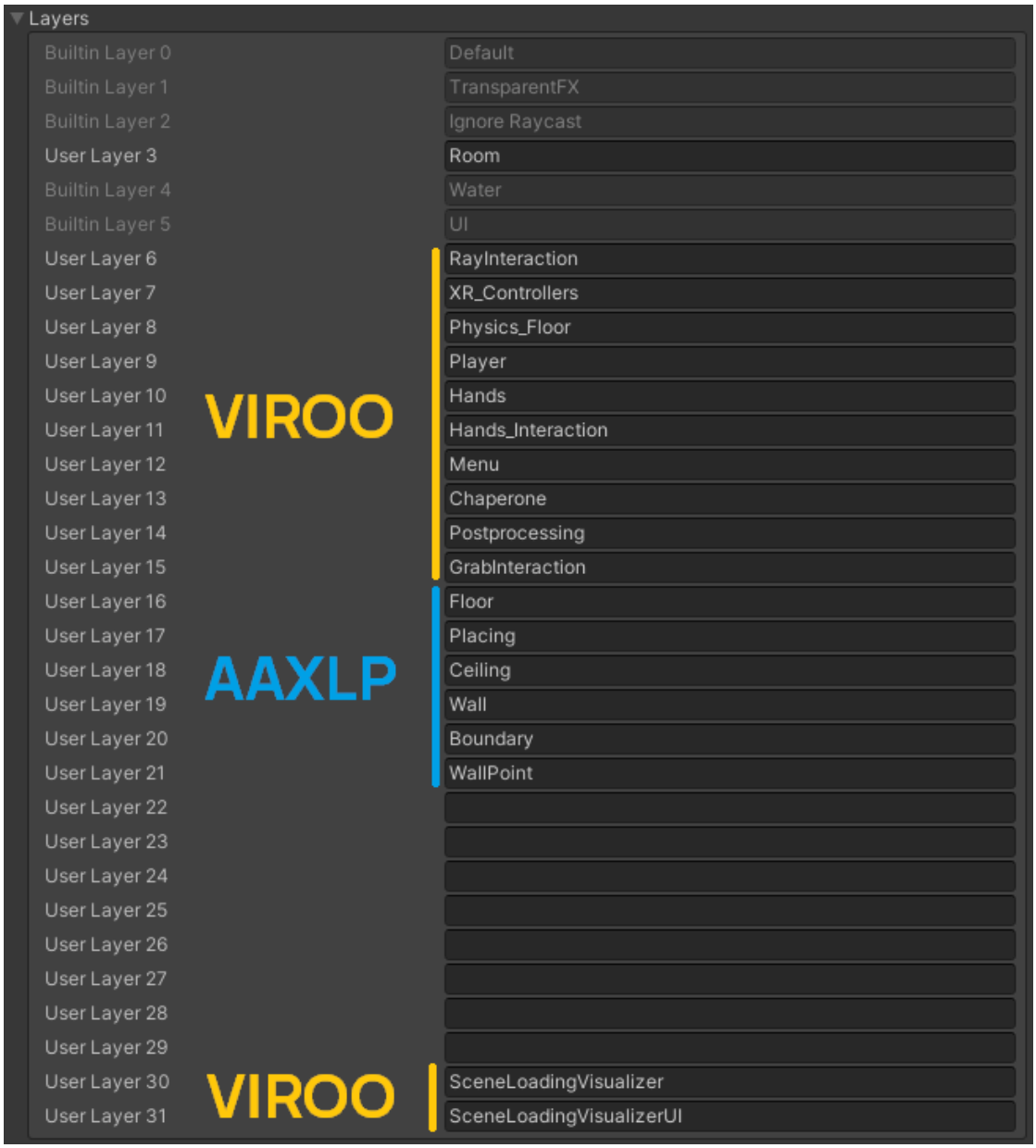
18- Ceiling

19- Wall

20- Boundary

21- WallPoint

Your Layers tab should look like this (you're free to assign your app's layers in 22-29):



Additional Configurations

- Go to the “VR” scene found in the “Assets/Scenes” folder and enter your OpenAI API key and Organization ID in the “ChatGPTController” script on the “Manager” game object.
- Go to Project Settings > Player > Other Settings > Configuration menu and set the “Active Input Handling” to “Both”.
- Add a new tag named “VIROORoot” and assign it to the “Root” game object in the “VR” scene.
- Remove the space key from the “Alt Positive Button” field of the “Submit” input in Unity’s Input Manager window.
- It is possible to test AAXLP in the Unity editor without a VR headset. If you wish to use keyboard and mouse to test in the editor, configure VIROO’s bootstrap mock as follows:

Updating and Maintenance

It is possible to update the following components of the project if needed:

- VIROO Studio
- Unity Engine
- Unity Sentis (now rebranded to “Inference Engine”)

Follow VIROO’s documentation on how to upgrade the VIROO version and use Unity’s Package Manager to update Unity Sentis to Inference Engine in a newer Unity editor.

Testing

You can test if the base project is working properly by following the steps below:

1- Open the “Assets/Scenes/VR” scene file

2- Press the play button in Unity Editor or press the Ctrl + P keys together to enter the play mode

Desired outcome: You should see the “Welcome to AAXLP” starting menu after a while with no error messages.

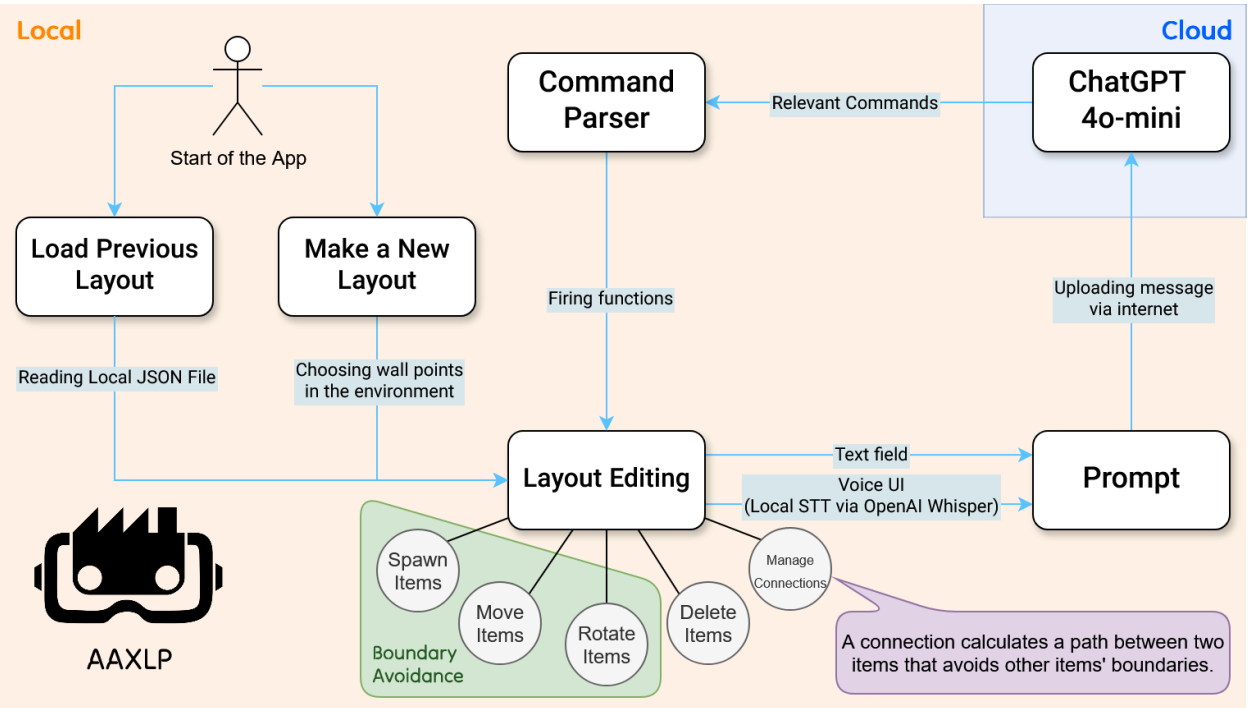
3- Test the functionality of the app:
- Free mode:

- **New layout**: You should be able to select four or more wall points in the environment and successfully make a new layout with your chosen name without errors.
- **Spawn** one or more items by opening the commands menu (Q)
- **Delete** an item by opening the options menu (Left Mouse Button) while looking at an item
- **Move** the items around by holding the (E) key while looking at an item
- **Connect** two items by opening the options menu and selecting “Manage Connections” and “Add a New Connection” and then selecting another item (Left Mouse Button)
- Enter the miniature mode by changing the “World Scale” slider on the top right side of the screen (it is expected that the connections are disabled while the miniature mode is active)
- The “Go back to starting menu” button does not work in the Editor because of how VIROO is configured. Instead, follow the next steps to go back to the starting menu:
- **Stop the play mode** by clicking the stop icon or pressing Ctrl + P together again
- **Enter the play mode** again
- **Choose “Load Layout”**: You should be able to see the layout you made in the previous steps and load it successfully

- Scenario Mode: Load each of the scenarios and follow the voice instructions. You should be able to finish the scenarios with no errors.

Integration

AAXLP stands for AI-assisted XR layout planning; it is essentially a layout planner application enhanced with different AI technologies. The diagram below explains the systems of this project:



The responsible script for each step of the above diagram is shown below:

- Load Previous Layout:
  - **Ainak JSON Reader**
  - **Room Generator**
- Make a New Layout:
  - **Wall Placer**
  - **Room Generator**
- Layout Editing:
  - Spawn, Move, Rotate, Delete Items: **Item Handler** and **Command Wheel**
  - Boundary Avoidance: **Boundary Checker**
  - Manage Connections: **Path Visualizer**
- Voice UI: **Run Whisper** and **Audio Recorder** (both in “Assets/STT Whisper Tiny/Scripts” folder)
- Uploading the message via the internet: **ChatGPT Controller**
- Command Parser: **ChatGPT Controller**

You can find these scripts in the “Assets/Scripts” and “Assets/STT Whisper Tiny/Scripts” folders.

To integrate the functionalities of this app with your VIROO-based app, you need to copy the contents of the “VR” scene found in the “Scenes” folder and paste them into your scene. All AAXLP scripts (found in the “Scripts” folder) have description headers in the Unity inspector, and if you hover your mouse over the variables, you can see a tooltip description for each one.

You may educate yourself with the information found in each script and the system diagram to integrate your solution with any part you see fit. However, there are two integration opportunities with little to no coding prepared for you. The next sections will describe how you can use AAXLP’s software module to integrate with your project without additional programming.

Using Your Own Models

One example use case is to customize the catalog by assigning your own 3D models in the “Item Handler” script on the “Manager” game object, and enter custom information regarding the items. You can also change or customize the “Boundary Canvas” prefab on the same script to display additional info on the items based on your needs.

Here are the requirements for the “Items” variable in the “Item Handler” script:

- The “Name” field of the items must be unique and have no whitespaces in it.
- The “Type” field of the items can be shared to categorize the items together, but they also must not have any whitespace in them.
- The object assigned to the “Prefab” field needs to have one or more child mesh objects with either a MeshRenderer and a MeshFilter component or a SkinnedMeshRenderer component (there could also be a single object with no children that has the aforementioned components). Additionally, make sure that the layer of the prefab is set to “Default”. There must not be any collider components attached to the prefab object.

If you want to show additional information which are shared across all of your items, you can edit the “Boundary Canvas” prefab and include your custom indicators or interface alongside AAXLP’s boundary canvas. Do not remove anything from the boundary canvas prefab, and do not rename or change the hierarchy order of the four ruler objects.

Unity Events

There are over 20 Unity event endpoints where you can fire a custom function when something happens in any of the AAXLP systems. You can add the (+) icon in the inspector tab to manipulate the game objects or call your scripts via the Unity Event System without additional coding.

Here's a list of all the implemented endpoints and which script they're assigned to:

Ainak JSON Loader	File Control	Item Handler	Command Wheel	ChatGPT Controller
onLayoutLoaded	onLayoutSaved	onItemsProcessed	onStartedMoving	onResponseFinishedProcessing
		onItemSpawn	onFinishedMoving	
		onItemDelete	onConnected	
		onItemMove		
		onItemRotate		

Menu	Settings	Notification	VRCanvasUtilities
onPause	onVolumeChanged	onNotification	onBringNearHand
onUnpause	onQualityChanged		
onReset	onSnappingChanged		
onExit			
onError			
onLoadingStarted			
onLoadingEnded			

A potential use of the above Unity Events is to disable an element in your menu when onLoadingStarted is called and enable it back upon onLoadingEnded. Another example is to call a custom function that processes a layout to perform the processing upon onLayoutLoaded and update it with onItemSpawn, onItemDelete, onItemMove, and onItemRotate.