**ROS integration with HoloLens**

This project aimed the integration between ROS and the Microsoft augmented reality glass, HoloLens. The goal was to be able to access and obtain full control of a ROS node from HL. The chosen node was the turtlesim, a default node that mostly ROS users have access.

The idea was to build a HL application that could simulate and control the turtlesim node using HL features. HL apps are built and deployed to the device using Unity, so the environment needed to be built using this game engine. The turtlesim environment built in Unity consists in:

* A background plane that simulates the 2D blue plane where the turtle can move in ROS.
* A cube that simulates the turtle.
* 3 buttons responsible for connecting/disconnecting, clearing and resetting.
* A DPAD, an additional option for moving the turtle

In this application some HL features are being used. The gesture manipulator applied on the cube is responsible for the movement. Gazing on the DPAD can also move the turtle. The select gesture on the connector button is responsible for starting and finishing a ROS connection, on the reset button can take the turtle to the initial position and on the clear button can clear the ROS turtle path. A select gesture can also be applied in the plane, while there is no connection with ROS, so that the user can place the turtlesim environment around the room.

The connection between HL and ROS is made through Rosbridge. Rosbridge provides a JSON API to ROS functionality for non-ROS programs. In this project we are accessing the ROS service “teleport\_absolute” to move the turtle, where the turtle position in ROS is constantly updated based on the cube position in the simulated ROS environment running on HL. To clear the environment, we are accessing the ROS service “clean”.

It is also possible to access ROS and move the turtle while on Unity Play mode. Press “C” to connect with ROS, move the turtle with WASD, press “E” to disconnect, “I” to return to the initial position and “R” to clear.

**Instructions**

To run this project, you will need:

* Unity Hololens 5.4.0b16-HTP
* Linux running ROS (I am using ROS Jade running on a Linux VM mounted on HyperV-Manager).
* Microsoft HoloLens device or emulator
* holoROS folder

1. **Configure and open the Unity Project:**

* Extract the holoROS folder.
* Open Unity.
* Select Open.
* Select the holoROS folder you previously extracted.
* Unity will open the project. There is a scene ready to go.
* Select the Scenes folder and double-click the holoConnection scene.
* The scene will load.
* In Unity select File > Build Settings.
* Select Windows Store in the Platform list and click Switch Platform.
* Set SDK to Universal 10 and Build Type to D3D.
* Check Unity C# Projects.
* If there isn’t any scene loaded, click Add Open Scenes to add the scene.
* Close the Build Settings window.
* Go to Edit > Project Settings > Player
* In the “Other Settings” tab, find “Rendering” to make sure “Use 16-bit Depth Buffers” and “Virtual Reality Supported” are checked and “Windows Holographic” is added.
* In the “Publishing Settings” tab, find “Capabilities” to make sure “InternetClient”, “InternetClientServer”, “PrivateNetworkClientServer” and “SpatialPerception” are checked.
* Go to Edit > Project Settings > Quality
* Make sure Windows Store App level is set to Fastest.

1. **ROS IP address**

* Before deploying your app to HoloLens or running in Unity play mode, remember to check your VM ROS IP address.
* With ROS installed and running, open a terminal and type “ifconfig”. You should see and use the first provided “inet addr” on your code.
* To do this, simply change the value of the “host” variable on Source.cs (Scripts folder) for the desired value.

1. **ROS Commands**

* With ROS installed, running and the right IP address configured you should run the following commands on ROS terminal before trying a connection: (Note: Rosbridge package need to be installed. For further information, visit: http://wiki.ros.org/rosbridge\_suite)
* source /opt/ros/jade/setup.bash (Using ROS Jade)
* roscore
* \*new terminal\*
* rosmake turtlesim
* rosrun turtlesim turtlesim\_node
* \*new terminal\*
* roslaunch rosbridge\_server rosbridge\_websocket.launch

1. **Connecting and running on HoloLens or Unity Play Mode**

After all the previous steps implemented you should be able to connect and access ROS from HoloLens or Unity Play Mode. Since this HL app is deployed through Unity, it is easier to test the connection in Unity Play mode before deploying it to HL, but you can directly deploy the Unity project to HL without step a).

1. **Connecting and running on Unity Play Mode**

* In Unity, press the play button to enter in Unity Play Mode. You should see the rosturtle environment with a white plane, the connector, reset and clear button and the DPAD.
* Now, press “C” to connect. If everything works fine, you should see that the plane turned to blue, a cube appeared in the center of the plane, the connector button is now highlighted and you can see in the Rosbridge node on ROS that now we have a client connected.
* As you use WASD you should be able to move the turtle.
* To clean the path left by the turtle, press “R”.
* To return to the initial position, press “I”.
* If you wish to disconnect from the ROS environment, press “E” or exit Unity play mode. After that, you should be able to see in the Rosbridge node on ROS that a client disconnected.

1. **Connecting and running on HoloLens**

* In Unity select File > Build Settings.
* Click Build
* In the file explorer window that appears, create a New Folder named "App".
* Single click the App Folder.
* Press Select Folder.
* When Unity is done, a File Explorer window will appear.
* Open the App folder.
* Open (double click) holoConnection.sln.
* Using the top toolbar in Visual Studio, change the target from Debug to Release and from ARM to X86.
* Click on the arrow next to the Device button, and select Remote Device.
* Set the Address to the name or IP address of your HoloLens. If you do not know your device IP address, look in Settings > Network & Internet > Advanced Options or ask Cortana "Hey Cortana, What's my IP address?"
* Leave the Authentication Mode set to Universal.
* Click Select
* Click Debug > Start Without debugging or press Ctrl + F5. If this is the first time deploying to your device, you will need to pair it with Visual Studio.
* The project will now build, deploy to your HoloLens, and then run.
* Put on your HoloLens and look around to see your new holograms. Try to look to a “cleaner” environment so that the hologram does not appear behind a wall or other obstacle from the beginning.
* You should see the rosturtle environment with a white plane, the connector, reset and clear button, the DPAD and virtual meshes draw around you.
* While we are not connected, if you perform a Select gesture in the white plane, we enter in the placing mode. You should now be able to place the rosturtle environment in a specific location by gazing at it, using the Select gesture and then moving to a new location, and using the Select gesture again. Try to place the environment in a position where none of its area gets compromised by the virtual meshes. The movement of the cube/turtle can be compromised by interferences with unwanted meshes particles.
* After placing the environment in a desired position, perform a select gesture in the connector button. If everything works fine, you should see that the plane turned to blue, a cube appeared in the center of the plane, the connector button is now highlighted and you can see in the Rosbridge node on ROS that now we have a client connected.
* Now you are able to move the cube. For this, you have two options:
* Gazing at the DPAD: Gaze at the arrows and see how the turtle moves in the hologram and ROS simultaneously, accordingly to the arrow you are gazing.
* Gesture action: If you gaze at the cube, four arrows should appear around the cursor to indicate that the program will now respond to Manipulation events. Lower your index finger down to your thumb, and keep them pinched together. As you move your hand around, the cube will move too, as the ROS turtle. Raise your index finger to stop manipulating the cube.
* To clean the path left by the turtle, perform a select gesture on the clear button on the top of the plane.
* To return to the initial position, perform a select gesture on the reset button on the top of the plane.
* If you wish to disconnect from the ROS environment, perform another select gesture in the connector button that is now highlighted. After that, the button and the plane should return to its standard state and you should be able to see in the Rosbridge node in ROS that a client disconnected. While disconnected you can enter in placing mode and place the environment anywhere you want (as long it is in a position where none of its area gets compromised by the virtual meshes) and connect to ROS again.