Omniverse

Omniverse uses two scripts to run its producer, and an additional two to run its consumer. We will go over here how to set up the producer and consumer after acquiring these files.

General setup:

To launch omniverse on server 7, use the following commands in the linux terminal:

```
cd Desktop/isaac-sim-new/

./isaac-sim.sh
```

Make sure that the computer is connected to the internet when opening Omniverse, or else the files will not properly load.

To load the interoperability test, use file > open. Next, navigate to the file:

```
home > dt_interoperability > hsml_api > src > firstTest >
omniverse_stage_interoperability_cadre.usd
```

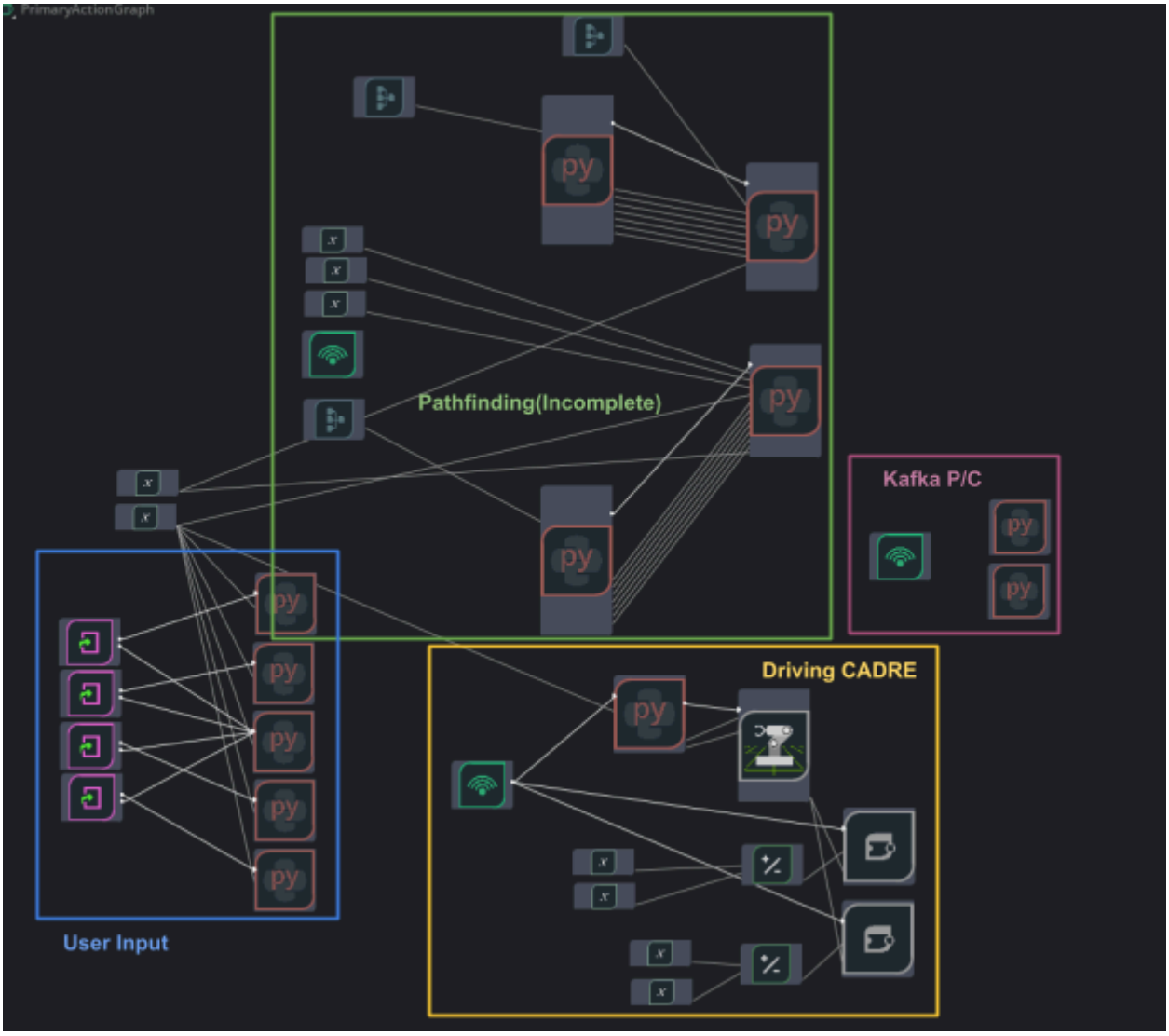
When the file opens, make sure to agree to running scriptnodes.

Simulation specific setup:

Omniverse takes heavy advantage of an action graph for controlling simulation properties in real time. This action graph can be viewed by selecting Window > Graph Editors > Action Graph in the top bar. There are two Action Graphs in the simulation, though only one (PrimaryActionGraph) is relevant.

This graph is complex, and consists of many scripts; however, few of these are relevant for the kafka consumer and producer. These other scripts control the articulation of the rover, the user's ability to control it via an input, and an unfinished pathfinding algorithm that can be enabled to enable autonomous driving. See the next page for a diagram of these sections.

Only the Kafka Producer and Consumer section (pink box) must be used to enable interoperability.



In the setup phase, one should also open the script editor via the top bar Window > Script Editor. This appears as a separate tab, though I prefer to drag the script editor over to the right side of the screen so I can see it as I work. The script editor is necessary as it displays the log of debug statements that can help one identify any issues in the Kafka setup.

There are two rovers in the simulation, CADRE_Demo and CADRE_2. CADRE_Demo runs in Omniverse and shares data to the Producer. CADRE_2 gets data from Unity and its position is updated via the Consumer.

In this diagram, neither the script for the producer or the consumer is connected to the controller. Ensure that both scripts are connected to the event timer (the wifi-symbol icon).

The Consumer runs entirely within the action graph, but the producer is called upon by the `producerOperator` script within the action graph. This points to the consumer file currently, but this can be changed by altering line 11:

```
sys.path.append('/home/luke-cortez/dt_interoperability/hsml_api/src
/firstTest')
```

Running:

Preconditions to run are as follows:

- ☐ Kafka is running (See Kafka readme)
- ☐ Unity is running (See Unity Readme)
- ☐ Both the Unity computer and the Server 7 are connected to the Futurama local network

To run the simulation, press the play button. The test cases for the scripts will appear in the script editor.