Cobot simulation in Isaac Sim

Mini report

By Kashif Ansari

Phase. 1

Imported assets into Isaac Sim and defining rigidity and colloidal presets (including a table, manipulator, fruits, and other items to set up the environment).

The simulation environment in Isaac Sim was set up by first importing necessary assets, including a table, manipulator, fruits, and other items. Rigidity settings were then defined to determine the solidity or flexibility of these objects within the simulation, with specific objects like the table being set as rigid. Following this, colloidal presets were established to govern how objects interact with each other, encompassing collision behavior, friction, and material properties. Once these settings were configured, the environment was assembled by placing the objects in the simulation scene, ensuring accurate positioning and defining initial states as required. This systematic process facilitated the creation of a realistic and interactive simulation environment in Isaac Sim, ready for further analysis and experimentation.

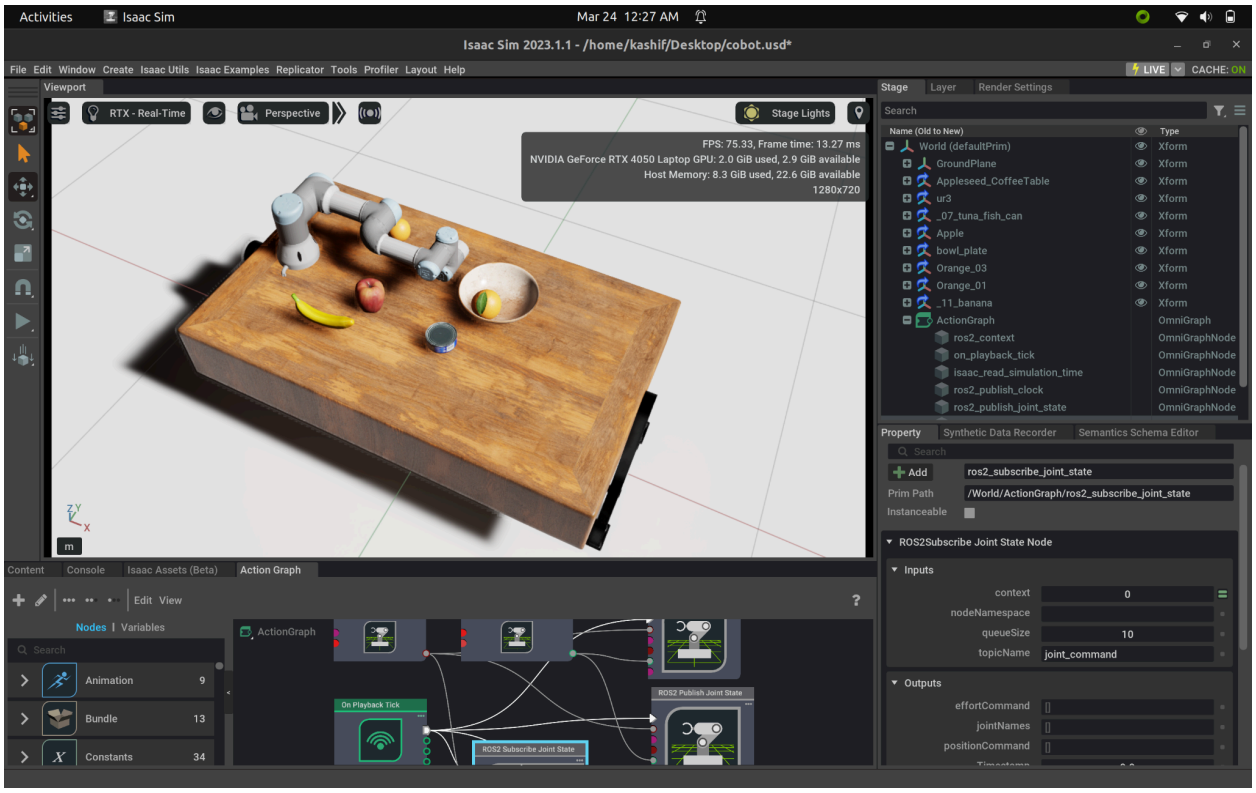

Figure. 1

Installed and tested the MoveIt package for ROS2 Humble, and tested it in RViz with the Franka Emika Panda.

The MoveIt package for ROS 2 Humble was successfully installed and tested, showcasing its functionality with the Franka Emika Panda robot. The testing phase involved integration with RViz, where the capabilities of MoveIt were explored and validated. This included planning and executing motion tasks with the Panda robot within the RViz environment, demonstrating the package's effectiveness in motion planning and control. The seamless integration and successful testing of MoveIt in conjunction with ROS 2 Humble and the Franka Emika Panda in RViz highlight its utility for robotic applications, particularly in the context of motion planning and manipulation tasks.

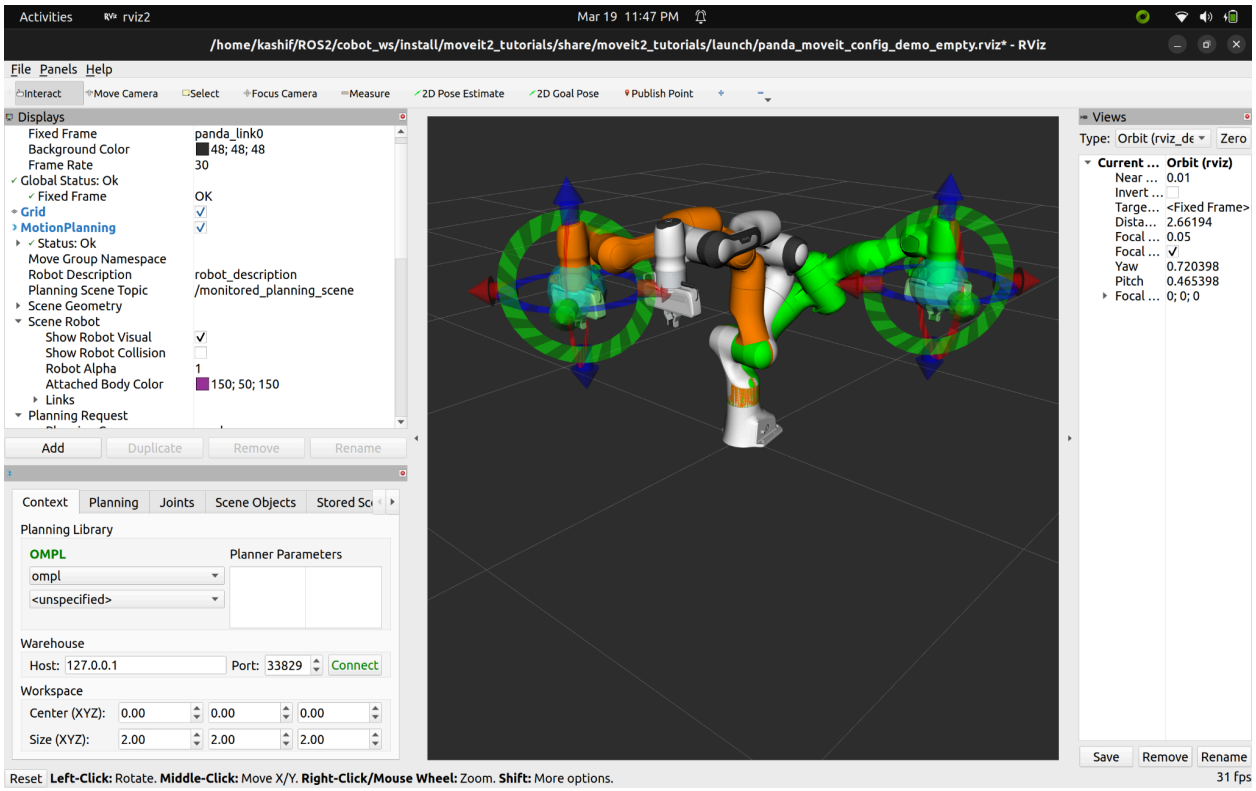

Figure. 2

Selected the UR3 to set up the MoveIt Package from scratch for further progress in Isaac Sim.

The UR3 robot was selected as the platform to set up the MoveIt Package from scratch in order to advance the progress within Isaac Sim. This involved initializing the MoveIt configuration for the UR3 robot, including defining its kinematic model, joint limits, and collision objects. The configuration process also encompassed setting up motion planning parameters such as planner algorithms and planning groups tailored to the UR3's capabilities. Additionally, integration with Isaac Sim was ensured to enable seamless simulation and testing of motion planning tasks within the simulated environment. This setup lays the foundation for MoveIt functionalities effectively with the UR3 robot in Isaac Sim, facilitating further progress in robotic motion planning and control.

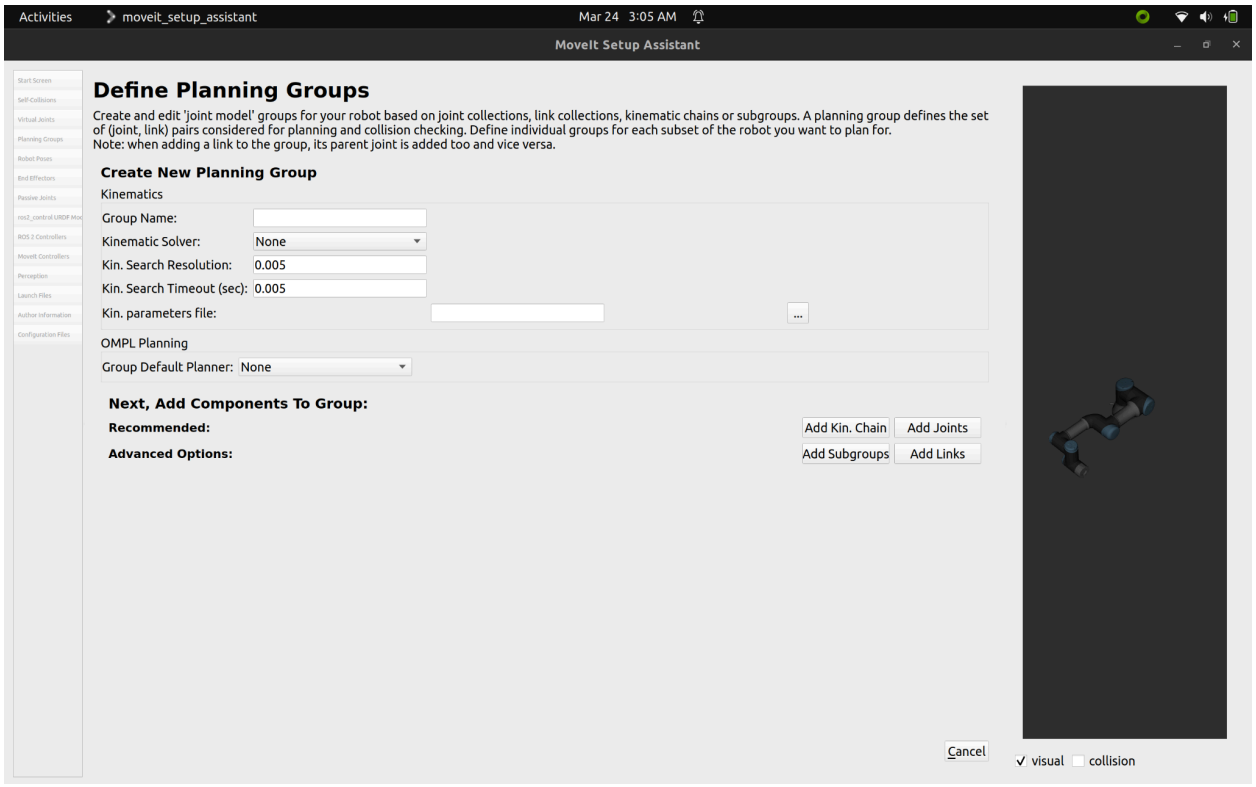

Figure. 3

Set up the action graphs in Isaac Sim to integrate ROS2 (MoveIt) and the Isaac simulation environment.

Action graphs were configured in Isaac Sim to facilitate the integration of ROS 2 (specifically MoveIt) with the Isaac simulation environment. This involved defining actions within the graph to enable communication and data exchange between ROS 2 nodes, particularly those related to MoveIt functionalities, and the simulation environment in Isaac Sim. Key actions included subscribing to MoveIt topics for motion planning commands and publishing robot states back to ROS 2 for feedback and control synchronization. Additionally, appropriate message formats and data structures were specified to ensure compatibility and seamless communication between ROS 2 and Isaac Sim. The successful setup of these action graphs establishes a robust framework for bidirectional interaction between ROS 2 (MoveIt) and the Isaac simulation environment, paving the way for comprehensive robotic simulations and testing scenarios.

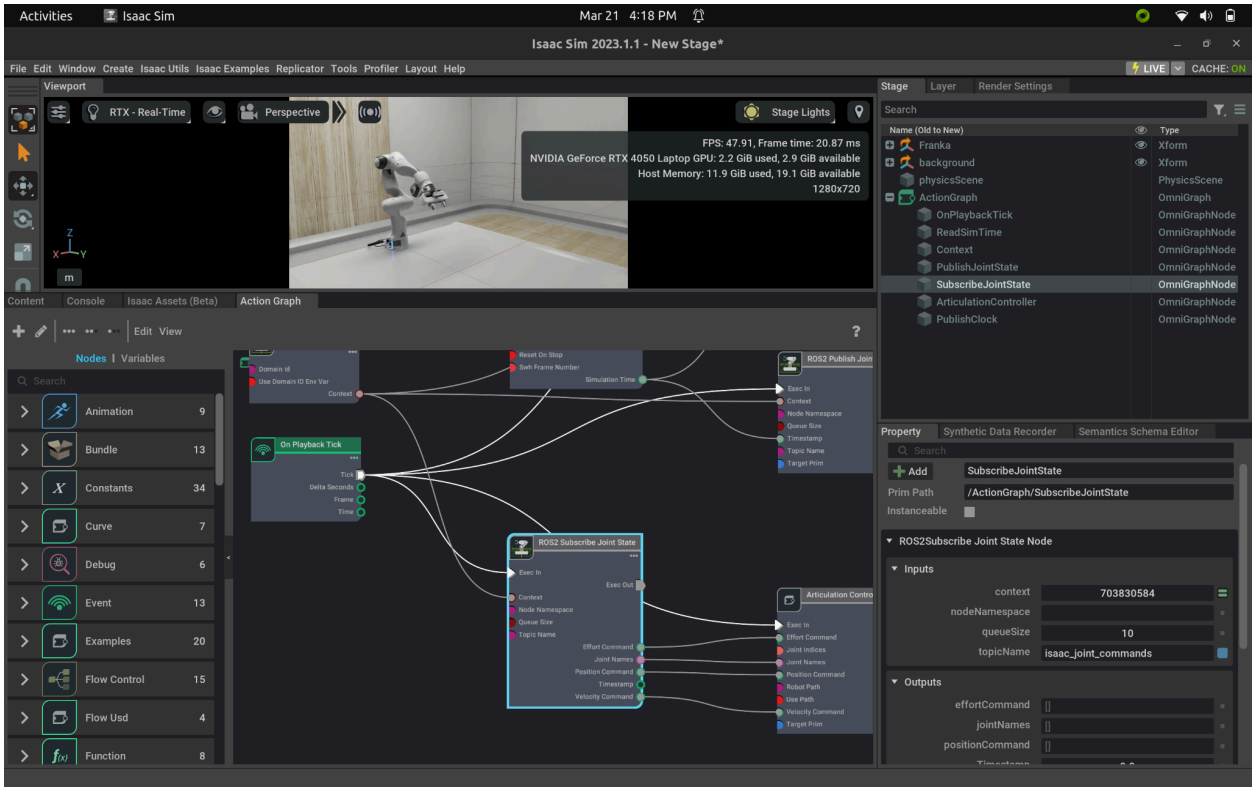

Figure. 4

Above shows the testing phase over Franca Panda that comes as an inbuilt tool in Isaac Sim.

Developed and tested action graphs in Isaac for further visual scripting and control of the differential drive (mobile robot)

Action graphs were developed and tested in Isaac to enhance visual scripting capabilities and control functionalities for a differential drive mobile robot. This involved creating custom action nodes within the graph to handle various tasks such as motion control, sensor data processing, and decision-making algorithms. I designed it to communicate with the differential drive robot's hardware interface, enabling real-time control and feedback integration.
This included testing for smooth motion execution, accurate sensor data processing, and robust decision-making in response to dynamic environments. Debugging and optimization were also carried out to fine-tune the action graphs. This technique can also be useful in designing custom dynamic environments in future projects.

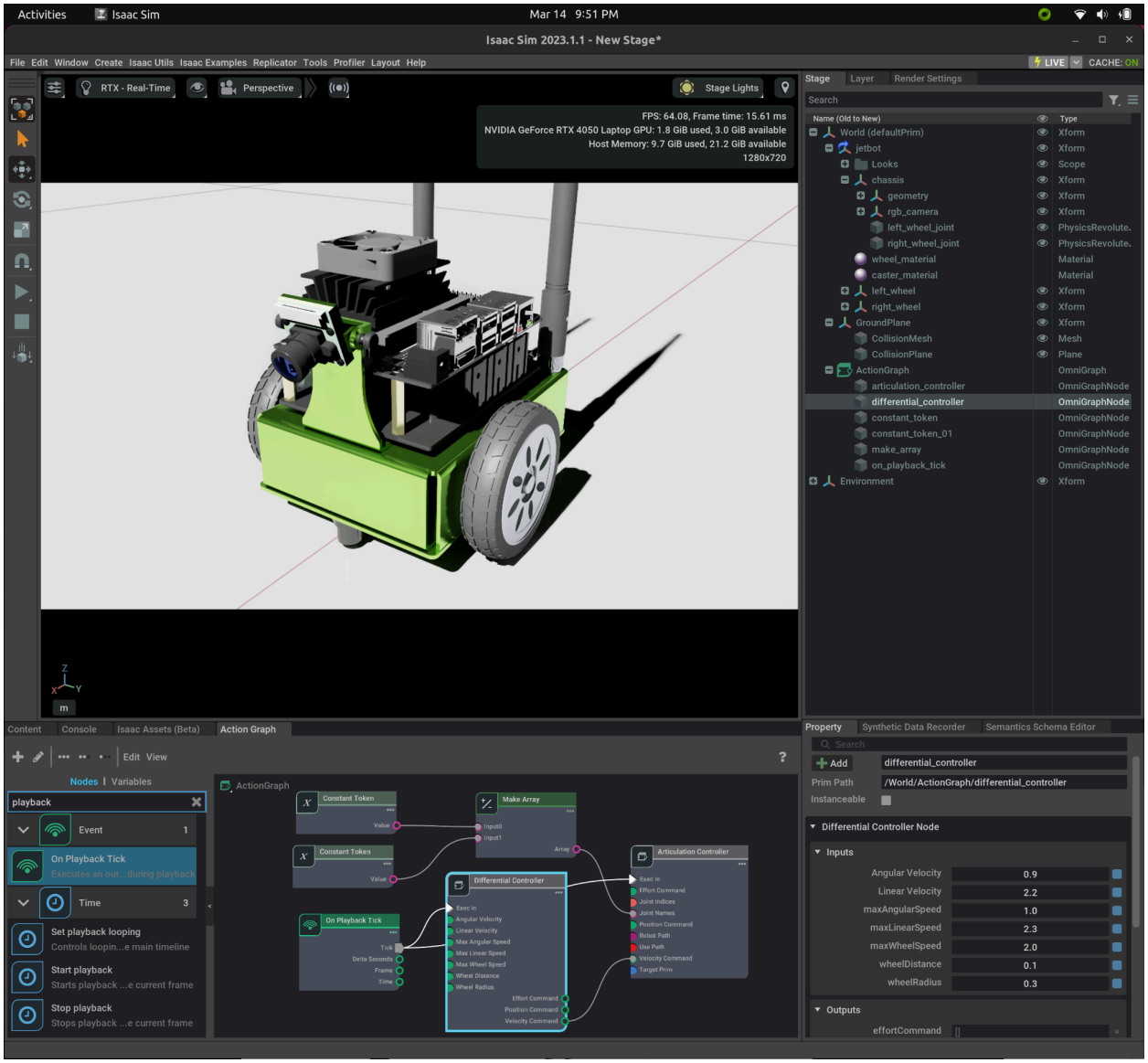

Figure. 5

Motion planning and testing MoveIt for Rviz & Isaac Sim cross communication

Video and image link: isaac sim & RViz

Motion planning using MoveIt was tested in RViz and Isaac Sim with cross-communication facilitated by a custom ROS node acting as a publisher and subscriber. In RViz, MoveIt was used for trajectory planning and execution visualization, while the ROS node handled communication between MoveIt and Isaac Sim. This setup allowed for coordinated motion planning and execution validation in both environments, ensuring seamless integration and compatibility across platforms. This means the changes on either platforms ie. RViz and Isaac were reflected in either platform in real time.

Next task and planning so far:

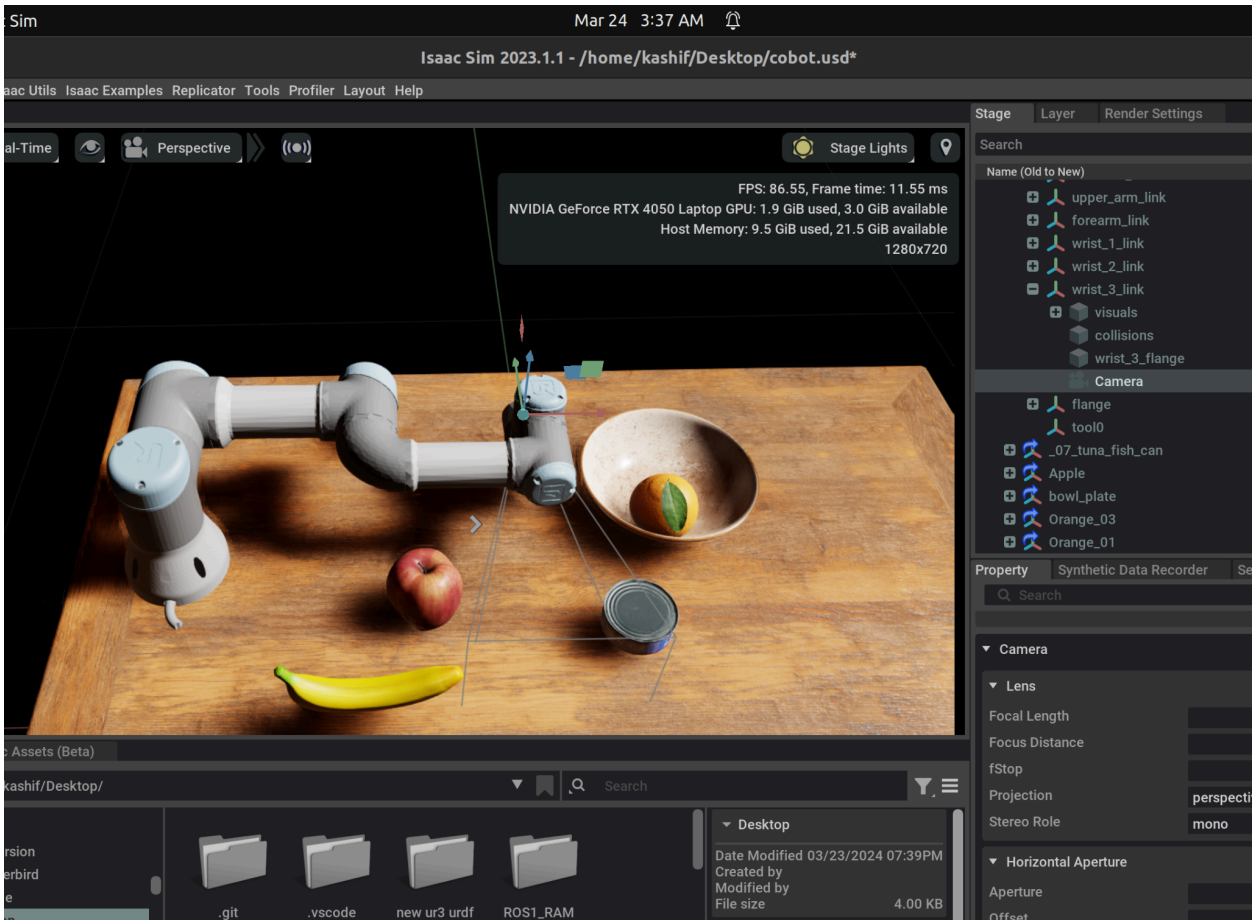

Figure. 6

Now in order to identify and reach up to the fruit following are my planned strategies:

> Capture a video in 360* direction using RGB camera (preferably intellisense depth camera).
> Extract 100 frames from the video
> Train CNN to identify apples when it's projected in the cam frame.
> Sense and orient the camera coupled with an end effector towards the apple.
> Reach to the apple measuring the total distance using depth sensing feature of intellisense. (This will reduce the complexity arising by eliminating any requirements of sensor fusion for this case)

Challenges faced so far: *A lot of ROS debugging and new learnings from limited resources over the internet on newly emerging simulation tool 'Isaac Sim'.*
