ABA AIOT PVT LTD
Pratik Rohane

ROS Reality

Introduction

Creating a piece of software to interact with the various sensors and actuators, communicate with the different nodes of the robot as well as perform the visualization and simulation. This attempt can be made by integrating ROS with a popular game engine Unity3D with the help of some set of tools and libraries that are already available. Here we are testing the Unity with the ROS# extension. Using Unity gives several benefits like the high degree of graphical fidelity, also build-in support for Virtual Reality devices. This allows the creation of a custom user interface for controlling and visualizing robots, in the real or virtual environment.

The ROS-Unity framework chooses the ROS#, which is developed by Siemens. That provides a general-purpose interface between Unity and ROS. It is based on the ROSBridge, which enables cross-platform communication with ROS from the Web-socket client interface. ROS# provides the Unity-side implementation for the interface. Communication between ROS and Unity can happen via TCP/IP protocol. Currently, there are a certain number of message types are supported into the ROS#, new message types must be created by the user.

One more reason to use Unity over the other game engine is the ability to assign a closely matching collision model to each object, called mesh collider in Unity used for collision detection.

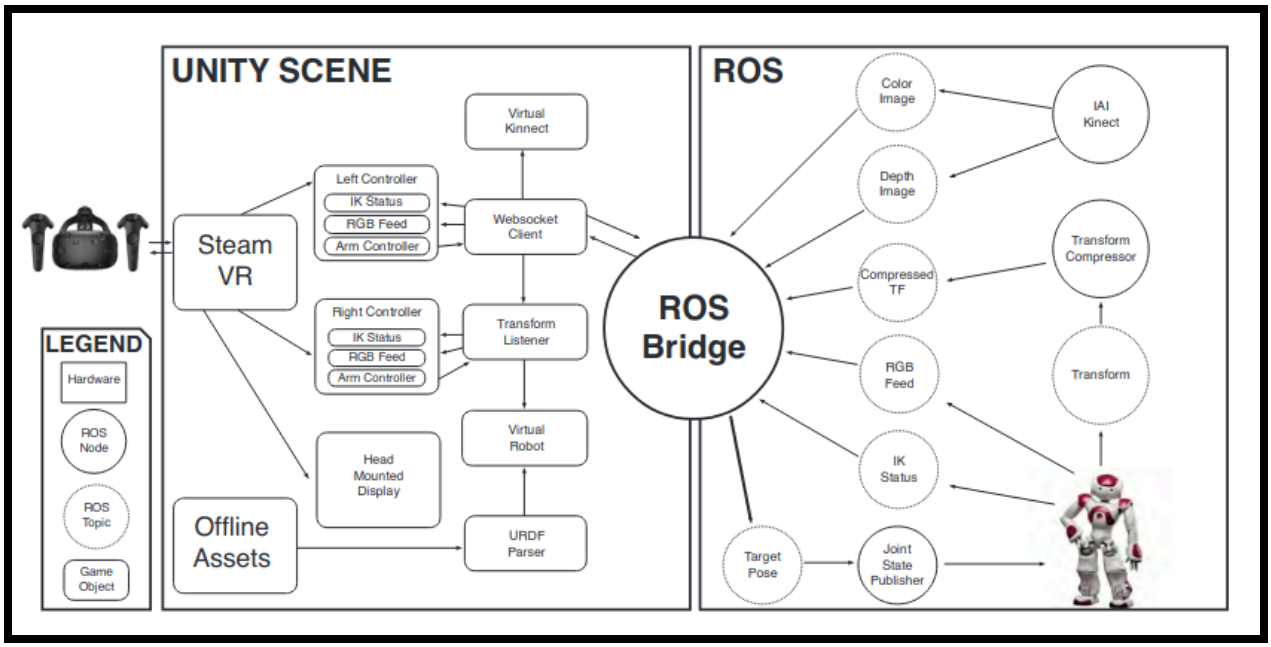
ROS Reality

ROS Reality is an open-source framework, over the internet teleoperation interface between any ROS-enable robot and any Unity-compatible VR headset. ROS Reality is a VR and AR teleoperation interface using consumer-grade VR and AR hardware with ROS-enable robots. It allows users to view and control robots over the internet using VR and AR hardware.

Overview

VR Hardware will be connected to the computer running Unity. Unity built a copy of our robot by importing the URDF from ROS. Unity connects to a ROS network over the internet via a ROSBridge Web-socket connection. ROS messages will be sent via Web-socket connection, color and depth image will be built into a point cloud in Unity with GPU shader, robot control commands will be sent from the user control to the robot.

Architecture



ROS

Robot Operating System, open-source framework for development, testing, and running robotics software. ROS is a set of tools and libraries that helps to create a robot application. Nodes communicate by streaming data over the topic on a local TCP network, called the ROS network.

ROS Reality launches a Camera ROS node, one for RCB image data and RGB-D data as well as a ROS message listener for the Transform, Twist, or Joint State.

Unity3D

Unity3D is a powerful cross-platform 3D engine and a user-friendly development environment. Easy enough for the beginner and powerful enough for the expert; Unity should interest anybody who wants to easily create 3D games and applications for mobile, desktop, the web, and consoles.

Unity has high-quality rendering engine and a physics engine also provides extensibility via a C# programming interface.

VR Hardware

It consists of 3 objects, one is a head-mounted display(HMD) and two hand controllers. The controllers are fully wireless and HMD is connected to the computer via USB. Each controller has a touch-pad joystick, trigger, and buttons for user input. VR hardware will connect with unity through a software package StreamVR.

ROSBridge

This application helps to build communication with ROS over a network through a WebSocket and proves to be the interface between ROS-Unity. It includes 3 packages contained in the rosbridge_suite package.
rosbridge_library: Handle the conversion between JSON string and ROS messages
rosbridge_server: Provides web-socket for communication
rosapi: Defines services that expose a subset of ROS commands to the network

ROS#

ROS# is open-source extension for the Unity editor which implements the ROSBridge protocol, It provides tools to improve integration with ROS such as import robot 3D models (URDF). ROS# can be used for robot and sensor visualization, as a simulation environment, and for teleoperation.

COORDINATE FRAME

Position

$$X_{Unity} = -X_{ROS}$$
$$Y_{Unity} = Z_{ROS}$$
$$Z_{Unity} = -Y_{ROS}$$

Rotation

$$qX_{Unity} = qX_{ROS}$$
$$qY_{Unity} = -qZ_{ROS}$$
$$qZ_{Unity} = -qY_{ROS}$$
$$qW_{Unity} = qW_{ROS}$$

SCENE RENDERING

Scene rendering is a core task of VR visualization to create a realistic and natural visual frame of the environment, by fusion sensor data, joint position, and robot model and other available objects. To responde a operation in real time, without disorienting lag, required a high-performance rendering system. Strategy to develop that system is to stream sensor data to a powerful local computer, which builds and displays a local model of the world using Unity.

Virtual Scene in Unity Consist of:

1. 3D model of robot: importing URDF description of robot model
2. 3D point-cloud of the scene: Obtained by calibrating Kinect sensor mounted on the robot
3. Display the robot camera: high resolution live image of the environment forward to the robot

VISUALIZATION

RGB Camera Visualizer

To visualiza camera feeds from the robot, the image subscriber script subscribes to a specific topic. When image data is received that will convert it in form base64 and texture or rendered that on the selected plane GameObject. ROS reality always uses JPG for bandwidth reasons.

PointCloud Visualizer

PointCloud visualizer script uses a GPU shader to build a point cout from the RGB camera image and raw depth data from the Depth Camera. The script subscribes to a topic of PointCloud data. Each pixel carries a depth information (distance in millimeter of that pixel from the camera) and RGB color data.

INSTALLATION

ROS

- ROS_Bridge_Suite
- File_Server Package

ROSBridge_Suite Installation

sudo apt-get install ros-<rosdistro>-rosbridge-suite

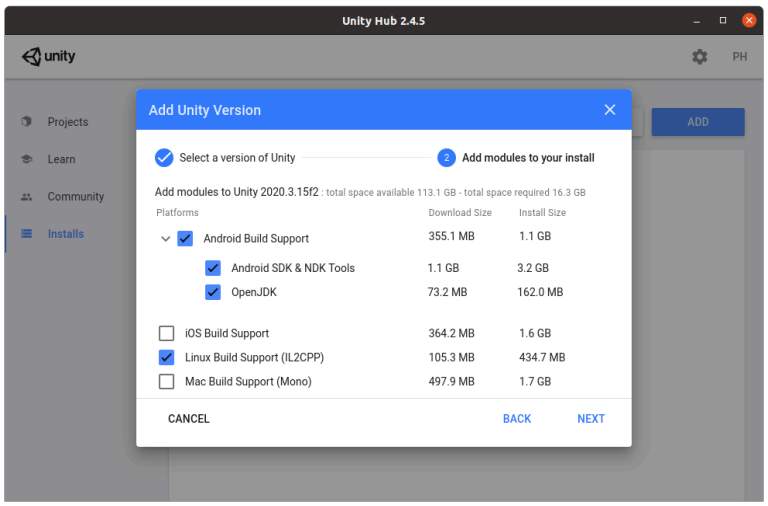
File_Server ROS Package

Step1: Download Zip and Extract ros-shape github
Step2: Copy the file_server package from the ROS folder into catkin_wc/src
Step3: Do catkin_make and source your workspace

Unity3D

- Linux Build Support
- Android SDK
- ROSSharp Package
- Oculus Integration
- XR Plugin Management

Install Android SDK and Linux Build Support in Unity



ROSShape Package

This package is also included in the ros-shape github, in the Unity/Asset folder, copy that from there and put it into the project Asset folder.

Installation of Oculus Integration and XR Plugin Management are mentioned below in the Oculus setup section.

SETUP

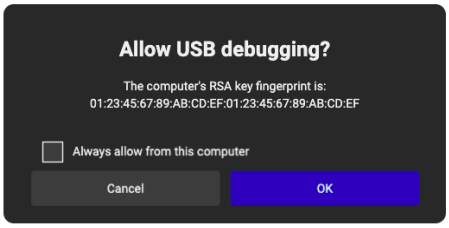
Oculus with Unity

Step1: Turn on Developer Mode of Oculus

Oculus Companion App: Settings > More Settings > Developer Mode

Step2: Connect Oculus Rift with USB

Allow USB Debugging



Step3: Enable VR support in Unity

Go to Edit > Project Settings > XR Plugin Management

- Install XR Plugin
- Select Oculus

Step4: Import Oculus Integration Plugin in Unity

- Got Window > Asset Store

- Search “Oculus Integration”
- Import “Oculus Integration”

Unity with ROS

-Import plugins into Unity Project

-Generate C# code for the ROS msg files

Publishing a message from the Unity to ROS Topic:

-Create instance of ROS message in scripts

-Publish message to ROS topic

Subscribing to a ROS Topic from Unity:

-Extend ROSSubscriber class in scripts

-Override Async function to read and instantiate ROS message

MINIMUM SYSTEM CONFIGURATION REQUIREMENT

Unity3D

- Intel i5-4590 equivalent or greater.
- NVIDIA GTX 970 or AMD R9 290 or greater.
- 8GB RAM or more.
- HDMI 1.3 and 3x USB 3.0 plus 1x USB 2.0

Gazebo

- CPU that is Intel i5 or equivalent
- Dedicated GPU (Nvidia cards tend to work well in Ubuntu)
- At least 500MB of free disk space
- Ubuntu Trusty or later installed