Kinect to Unity depth visualization

Nathan Larson
December 18, 2018

1. Introduction

This project captures the raw depth data from a Kinect v1 using the libfreenect library, processes it, and displays it in Unity where the user can view it as well as do some simple interactions in the scene.

2. How to use

2.1. Startup

To begin running the program, make sure that a Kinect v1 is plugged in to a usb port as well as a power source. It is recommended to have at least a 2 meters across and 4 meters of space in front of the Kinect to allow for ample movement.

2.2. Interaction

Once loaded into the scene, the plane will be visible directly in the center of the window. At this point the depth is being collected from the Kinect and should be updating the texture in real-time. Based off of the lower and upper bounds for tracking, moving into this range should result in pixels being drawn in the overlay texture like painting. The default paint color is green, although it can be changed to red by pressing the ‘R’ key or blue by pressing the ‘B’ key. The space bar will clear all current pixels painted onto the overlay.

3. Documentation

3.1. Plugin

3.1.1. Libfreenect library

In order to communicate with the Kinect v1 device this program uses the OpenKinect communities libfreenect library. The library allows full control of a connected Kinect device such as camera mode, tilt angle, and capturing raw camera data.

3.1.2. Data processing

In order to get the raw depth data from the Kinect, a callback function was assigned to be used by the libfreenect library whenever a new frame is ready. This function is called independently from the rest of the program, only being used by libfreenect when there is new depth data available. The depth data is returned as a pointer to a 16-bit unsigned short with a size of 307200 elements since the Kinect has a resolution of 640x480. To process this data I implemented two separate threads, one to handle grabbing the depth data as it becomes available and another to process the data and get it ready for Unity.

3.2. Unity

3.2.1. Data Manager

3.2.1.1. Kinect Script

This script acts as an object to communicate with the plugin from the Unity side. All of the external functions imported from the .dll file are located here, as well as handling the initialization and shutdown of the Kinect device. From the initialization function the script manually allocates memory, establishes a connection with the Kinect, and starts the threads for the plugin. Since nearly all other scripts in the environment need information from the Kinect in order to operate, it is a static class thus does not need an active instance to operate.

3.2.1.2. Manager Script

The manager acts as the “main” driver for the scene, being the only script that contains an update function in order to maintain straightforward synchronization. Although the kinect script does not require an instance to run it will not begin the initialization until it gets the signal from the manager to begin setting up the device.

3.2.2. Display

To display the depth data returned from the Kinect, there is a plane object that acts as the “window” to draw to. Attached to this plane is a custom rendering script that dynamically creates and modifies the texture at runtime. The depth data is drawn to the texture in a RGBA32 texture format with 8-bits per channel (red, green, blue, alpha) and a resolution of 640x480 to match to resolution of the Kinect depth camera. As the depth data is read into the texture, the values are analyzed to check if any pixels are within the drawing threshold, if so they are added to a list of indices to be used to draw images onto the overlay texture.