CT-Unity: Virtual Reality Stream Sharing

Unity (http://unity3d.com) is a cross-platform 3D game engine that builds 3D immersive games and Virtual Reality (VR) worlds. Unity is widely used, and claims “more games are made with Unity than any other game technology”, and “5 Billion downloads of Made with Unity games in Q3 2016” .

Here, we discuss a prototype interface and demonstration of how CloudTurbine (CT) enables real-world data streaming to the Unity virtual-world environment. Significant applications for this include “hybrid simulation” and "live, virtual, constructive" (LVC) systems, in which a combination of measured, analytical, and simulated real-time events are coordinated in a seamless virtual environment.

Figure 1 shows the basic interface between CloudTurbine streaming data sources and CT/Unity (“CTunity”) in-world displays. The figure shows in-world CT displays floating over the Unity “roll a ball” tutorial scene. In this example, the “CTstream” and “CTmousetrack” sources record CT data, which in turn is served via the CTweb application. CTunity display modules make HTTP data requests from CTweb for associated data. These prototype CTunity displays include:

- CTxyChart - Cross-plot of two streaming channels, e.g. X/Y mouse track
- CTstripChart - Channel vs time scrolling display (multiple color-coded channels)
- CTvideo - Animated image (video) display, e.g. from CTstream/image.jpg
- CTchartOptions - Embedded UI window to adjust chart display options in-world
- SineChart - Self-contained simple sine-wave scrolling stripchart

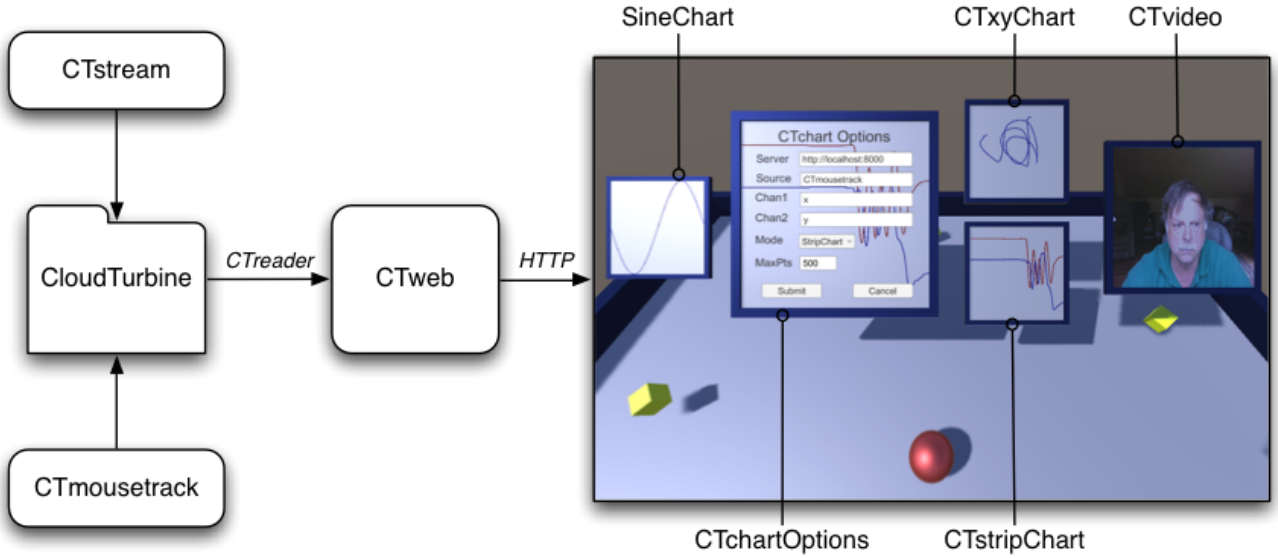

Figure 1: CloudTurbine to Unity VR Streaming Data Displays

Each of the CTunity display objects in Figure 1 is comprised of simple Unity 3D objects (e.g. cubes), with one or more custom CT scripts. Unity supports both JavaScript and C#, the latter being utilized here to interact with CloudTurbine data streams via HTTP calls to the CTweb utility.

The upper part of Figure 2 outlines the interaction between the various scripts developed for the prototype demonstration. In the bottom part of Figure 2, a potential improved future version implements a master "CTunity.cs" script that controls all or most interaction with CTweb, freeing other CTunity objects to utilize standard-Unity inter-object communication versus each needing to deal with CloudTurbine specific protocol.

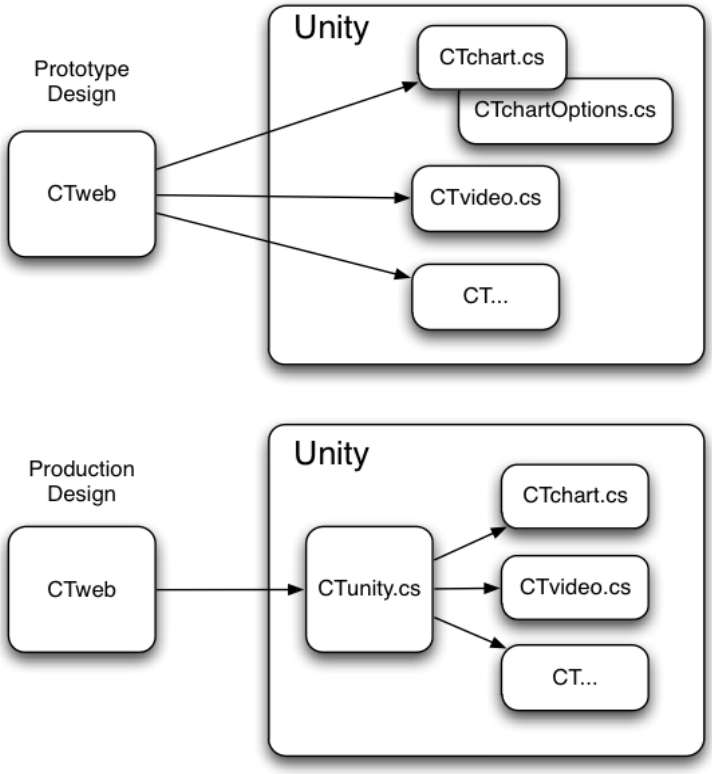

Figure 2: CTunity Interface Scripts, Prototype and Production Designs

In Figure 3, a sample CTunity script (CTvideo.cs) is shown to illustrate how powerful yet relatively simple scripting logic provides a live streaming data interface between CloudTurbine and the Unity environment. Of note, the "url" parameter specifies the link to CTweb for the video channel data. A runtime loop ("DownloadImage") makes sequential requests and paints them as a texture on the CTunity object (a simple 3D cube). Other scripts, not shown here (e.g. CTstripChart), are more complex in that they interact with the CTchartoptions menu and book-keep a back-to-back sequence of time-interval requests for gapless streaming data.

Open Source code for this CT-Unity demo are available on Github at:
https://github.com/cycronix/cloudturbine/tree/master/UnityCode.

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

// Simple video display (no options)
// Matt Miller, Cycronix, 7-6-2017

public class CTvideo : MonoBehaviour {
    public string url = "http://localhost:8000/CT/CTstream/webcam.jpg";
    public float pollInterval = 0.1f;       // polling interval for new data (sec)
    private Boolean showImage = false;
    private Texture startTexture;

    // Initialization
    void Start () {
        startTexture = GetComponent<Renderer> ().material.GetTexture ("_MainTex");
        StartCoroutine("DownloadImage");
    }

    // Download image updates
    IEnumerator DownloadImage()
    {
        while (true) {
            yield return new WaitForSeconds (pollInterval);

            if (showImage) {
                WWW www = new WWW (url);
                yield return www;

                Texture2D tex = new Texture2D
                   (www.texture.width, www.texture.height, TextureFormat.DXT1, false);


                www.LoadImageIntoTexture (tex);
                GetComponent<Renderer> ().material.mainTexture = tex;

                www.Dispose ();
                www = null;
            } else {
                GetComponent<Renderer> ().material.mainTexture = startTexture;
            }
        }
    }

    // toggle show image on mouse click
    public void OnMouseDown() {
        showImage = !showImage;        // toggle
    }
}
```

Figure 3: Example CTunity Script to Fetch and Display Streaming Video