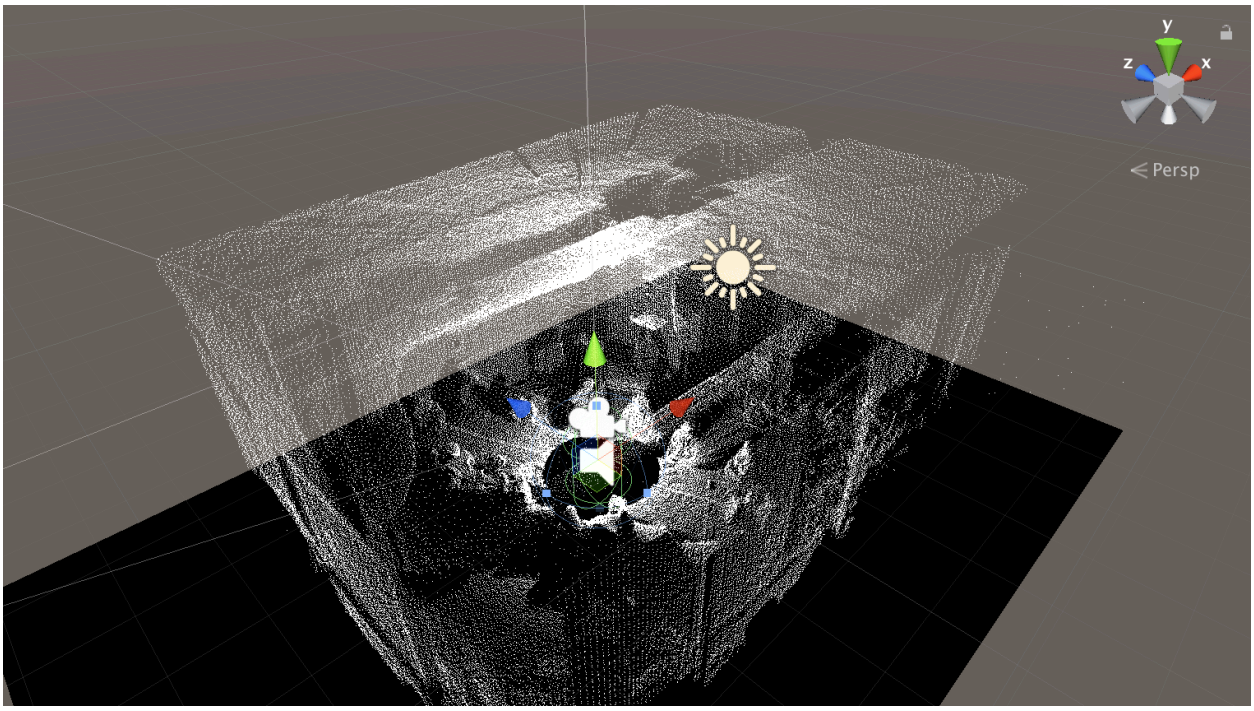
AD41700 Augmented and Virtual Reality Art
(Variable Topics in Electronic and Time-Based Art)
Prof. Fabian Winkler
Fall 2019



Visualizing Point Cloud Data in Unity3D (2018.4)

Step-by-step instructions:

1) Prepare and export your point cloud data from Cloud Compare as a .ply file (binary), Cloud Compare is a great free software tool for this: http://www.danielgm.net/cc/ you can watch tutorials for Cloud Compare here: http://www.danielgm.net/cc/tutorials.html

2) Download and install the Pcx - Point Cloud Importer/Renderer for Unity – there is information about this tool including the source code at: https://github.com/keijiro/Pcx

   a. Download the latest v0.1.5: Unity 2019.1 support from: https://github.com/keijiro/Pcx/releases (choose the Pcx.unitypackage file)
   b. In Unity import the Pcx.unitypackage file as a custom package: Assets > Import Package > Custom Package…

3) Now you can import the point cloud data into Unity
   Assets > Import New Asset…

Check the import settings in the Inspector for container type and Material/Shader. I have gotten good results with Container Type: Mesh and the Default Point material and the Point Cloud/Point shader.

4) Drag and drop the imported point cloud file into the Scene window. You probably have to transform your point cloud to make it roughly life size (it's probably too big, you can add a 1x1x1 or 10x10x10 cube to give you a good comparison – the units are in meters, this is especially helpful later when we bring in a first person character controller).

   With the FPRD204.ply file I used a uniform scale of 0.01 to make it match somewhat the scale in Unity. I rotated the point cloud -90 around the X axis and I also changed the point size of the shader to 0.005

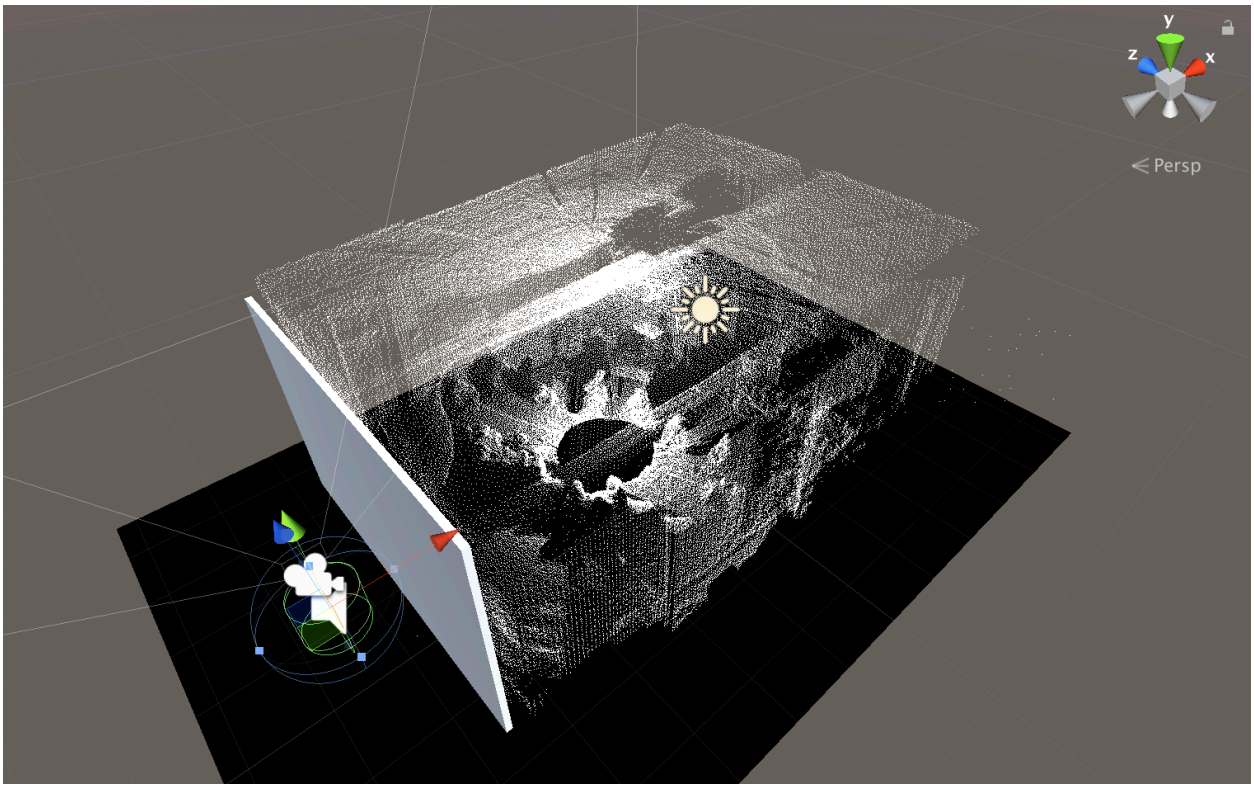
5) Next, add a plane underneath the point cloud model, this will become the surface the first person character controller can walk on. Gameobject > 3D Object > Plane… - Translate the plane so it lines up with the points on the floor of the point cloud model (Y position around -1.55). It will be helpful to change the color of the plane to black for example to create a contrast with the white points.
   Create a new material and make it black: Asset > Create Material… name this material (black, for example) and in the inspector change its properties to: Shader: Particles/Unlit and under Maps - Albedo change the color to black. Here you could also drag a .jpg or .png file that you have imported into unity onto the square in front of Albedo if you wanted to apply a texture to this material. Drag and drop the material onto the plane in the Scene window.

6) Now we can add a first person character controller to the scene. We have to get it from the Standard Assets Package from the Unity store. You get to the Unity store by clicking on the tab on top of the Scene window (it changes the Scene window to the Store window). Log into the store with your Unity ID and search for "standard assets." Download and import this package (you do not need to import the sample scenes).

7) In the Project window go to Assets then open the Standard Assets folder then the Characters folder and finally the FirstPersonCharacter folder. In there open the Prefabs folder and drag and drop the FPSController prefab into your scene. Translate its Y position so the capsule sits slightly above the ground plane (in my case around -0.6).

8) There is a camera included in (and attached to the capsule of) the first person character controller, so you can now delete the scene's main camera. Hit the play button and navigate the point cloud data using standard WASD navigation (or the arrow keys for forward, back, left and right). You can use the mouse to orbit the view and the space bar allows you to jump. Holding down the shift key while walking doubles your speed. You can fine tune all of these settings in FPSController's First Person Controller script in the Inspector. I changed my walk speed to 2 and the run speed to 4 as well as the jump speed to 5 to make the movements a bit more natural.

9) You also see that we still have the standard skybox (horizon line and blue sky) in the background which makes the point cloud less legible. Highlight the FirstPersonCharacter (a child of the FPSController) and go to its Inspector settings. In the Camera section change Clear Flags to “Solid Color” and the Background to black.



Add Interactive Trigger Zones

I would like to play a sound when the first person character controller moves through the front wall of FPRD 204 (the one with the projection screen).

1) Add a cube and scale it accordingly, so it sits just outside the wall (see picture above) – Used the following scale transforms (X 0.1, Y 4, Z 6).

2) Import a sound file into Unity (as either .mp3, .wav or .aiff) Asset > Import New Asset… I am using a file called “beep.mp3”

3) Import the triggerEnter.cs script Asset > Import New Asset… and drag and drop it onto the cube you created in step 1.

4) Select the cube in the scene window, now go to its Inspector settings and look at the Trigger Enter (script) section. Drag and drop the FPSController onto the box next to “My Target” and drag and drop the “beep” sound file from the Project window onto the box next to “My Sound.”

5) Finally, go to the cube's Box Collider settings in the Inspector and make sure the "is Trigger" property is checked. This will ensure that you can walk through it (otherwise the FPS controller would not be able to pass through it) and that it triggers the events in the OnTriggerEnter function of the triggerEnter.cs script that is attached to the cube.

6) Hit the play button and try it out, notice also the counter at the bottom of the Unity window that counts how often you have passed through the wall. Finally, if you'd like to make the trigger cube transparent, simply uncheck its mesh renderer property in the Inspector.

Where To Go From Here?

If you are new to scripting in Unity, go over this excellent beginner video tutorial that explains the most basic concepts creating and working with C# scripts in Unity:
https://learn.unity.com/tutorial/coding-in-unity-for-the-absolute-beginner

Using just the simple programming structure of the example above you can already think about:

1) Setting up multiple interactive trigger zones
2) Changing a game object when entering a trigger zone (transforming it or switching it out for another game object)
3) Changing elements of the scene (sounds, colors, lights, textures, game objects) based on how often a certain trigger zone has been entered
4) Instantiate new point cloud models in your scene based on the position of your first person character controller.

These are just a few of the possibilities you can experiment with, but with a good understanding of scripting you can create a lot of interactive scenarios in Unity that are only limited by your imagination.