Lab 04

Lab Overview

- **This lab has two parts**
  - Part I: create a project to do the demo shown in the lecture and then enhance it as follows:
    - ✓ Create an entire box, not just 3 sides. Put the camera in the center
    - ✓ Change the sphere bounciness to 1 so that the ball doesn't lose energy when it collides
    - ✓ Add a starting direction and speed to the ball (i.e. a velocity) so that it heads off at an angle and bounces off of multiple walls
    - ✓ Write a script to:
      - – Change the color of the ball every time it bounces off a wall
      - – Change the speed of the ball each time it bounces (toggle between a fast and slow speed after each bounce)
      - – Change the diameter of the ball each time it bounces
      - – Make the materials, sizes, and speeds public variables in your script so you can easily edit them before a build
      - – Alternate between two different "bounce" sounds of your choice
    - ✓ Add a menu to restart the simulation
    - ✓ Verify you can watch the ball in your headset as it moves around!

Lab Overview

- **This lab has two parts**
  - Part II: Create a dynamic mesh

Part II: Dynamic Mesh

- **Create a new project through the Unity hub. This is how I do it**
  - Select new 3D project
  - Swith platforms to Android; verify can build project
  - Adjust build settings (“Marshmallow” min API level, product name, etc.)
  - Install XR plugin management and pick Oculus
  - Install Oculus Integration package
    - ✓ Must use v18!
      - Later versions give a misleading message on Oculus Go about “system updating”...but nothing ever happens after that, it gets stuck

Create a Mesh Object

- **Create a Mesh object in Unity as follows:**
  - Create an empty GameObject
    - ✓ Rename to something meaningful like DynamicMesh
  - Add MeshFilter and MeshRenderer components to it
  - Create a Material for debugging (make it red) and assign it to the MeshRenderer
    - ✓ We’ll change it to a texture later, this is just to get things going
  - Attach a script to the GameObject that defines the mesh and animates it

Animate the Mesh with a Script

- **The script will be a class that inherits from MonoBehaviour**
  - It will have at least these class member variables
    - ✓ A mesh object, e.g. call it m_your_mesh (google Unity Mesh for more info)
    - ✓ A Vector3[ ] array to hold vertices (each array entry is (x, y, z))
    - ✓ An int[ ] array that defines triangles
  - The Start( ) function will
    - ✓ Allocate the vertices array, initialize its values, and assign it to the class member variable
    - ✓ Allocate the triangles array, initialize its values, and assign it to the class member variable
    - ✓ Allocate a mesh and assign it to m_your_mesh
    - ✓ Call m_your_mesh.Clear()
    - ✓ Assign your class member vertices and triangle arrays to m_your_mesh.vertices and m_your_mesh.triangles
    - ✓ Call m_your_mesh.RecalculateNormals() and m_your_mesh.RecalculateBounds()
    - ✓ Assign m_your_mesh to the MeshFilter of the game object to which the script is attached
      - – GetComponent<MeshFilter>().mesh = m_your_mesh

Animating the Mesh

- **The Update() function of the class will**
  - Update the vertices array class member variable
    - ✓ Only the z component changes. This periodic updating produces the animation
  - Assign the updated vertices array to m_your_mesh.vertices
  - Recalculate normals and bounds as before (see previous slide)
- **Specifics for this Assignment**
  - Create your square mesh to have size at least 40 by 40 rectangles (I used 100 x 100); subdivide each rectangle into two triangles
  - Let x vary from -2.5 to 2.5 and y vary from -2.5 to 2.5 when you define your vertices
    - ✓ initialize z to 0 for all vertices

Animating the Mesh

More details

- You will animate the mesh as if it is the surface of a square vibrating drum. The vertex height at each instant t will depend on both spatial position (x,y) and on the the time t as follows
  - ✓ vertex_height = cos(pi*x)cos(pi*y)*sin(a*t)
  - ✓ You can obtain the elapsed time in Unity with Time.time
- You should experiment with the value of "*a*" in the equation above to see what gives a nice effect. You might start with *a*=2
- You may need to play around a bit with the position of your camera to make it look nice in the oculus headset
  - ✓ I used a point light and adjusted its intensity to get a nice brightness
  - ✓ Make sure your camera is looking at the side of the mesh that the shader reveals! (Can try flipping mesh by 180 degrees if you see nothing)

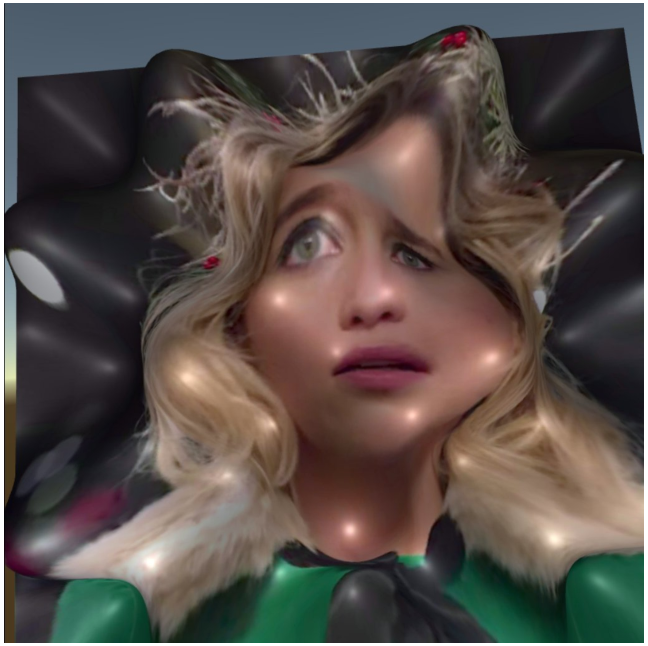
Step 2: Add a Texture Map to Your Mesh

- **To add the texture to your mesh**
  - Pick an image of size NxN that amuses you
  - Add the image to your project, create a material, name it, and then assign the image to your material by clicking on “albedo” and selecting the image you added (now you have a texture material)
  - Assign the material to your mesh renderer (replace the earlier “red” material with your texture)
  - Finally, you need to prepare the “vertices to texture” mapping in your script
    - ✓ Declare a Vector2[ ] array, e.g. m_myUV, as a class member variable
    - ✓ in Start() allocate space in m_myUV for every vertice in your mesh (i.e. the arrays are the same length)
      - NOTE: m_myUV is an array of Vector2’s, i.e., (u, v)
    - ✓ Where you initialize your vertices in your script, initialize the UV coordinates for each vertex too. This does the mapping
      - U and V both vary from 0 to 1; upper-left hand corner is (0,0)
        - » U <-> x and V <-> y
    - ✓ Before exiting Start( ), set m_your_mesh.uv = m_myUV

Step 2: Add a Texture Map to Your Mesh

- **Continued**
  - I got the best visual results by going to "window->rendering->lighting settings" and then selecting "color" source in "Environment Lighting" and choosing its color to be white
    - ✓ This creates ambient lighting everywhere so your picture is still visible as it makes its animation movments (things aren't hidden by shadows)

Raw picture



Captured during animation

Dynamic Mesh Deliverables

- Include captured images in your report that show you succeeded
- Check your code into github