Assignment 6 — 3D Texture Volume Rendering

New Attempt

- Due Nov 13 by 11:59pm
- Points 15
- Submitting a media recording or a file upload

Objective

The goal of this assignment is to help you practice and reflect on the lecture covering Volume Rendering based on **3D Texture**. You will learn how to create 3D texture and volume shader and to visualize it directly using optical properties, without reconstructing an isosurface.

- You'll **upload a screen-recorded video** that includes a walkthrough code review and a demonstration of your scene.
- Make sure to show the folder with **YOUR NAME** clearly visible.
- **Upload your scripts** along with your submission, and **walk through the code** during the screen recording.
- **I don't expect AI-generated code**—please do your best. I'm happy to support you and answer any questions along the way.

Follow the steps below for detailed instructions.

Preparation:

- This assignment is similar to Assignment 5 — Surface Rendering. But instead of reconstructing it as isosurface, we create a 3D texture and visualize it directly.

1. Project setup — **Skip this step, if you have already imported UnityVolumeRendering**:

- Import **UnityVolumeRendering (https://github.com/mlavik1/UnityVolumeRendering/releases/tag/2024.4)** to your Unity project
  - If you use Unity 6 editor, download **UnityVolumeRendering-2020.3+.unitypackage (https://github.com/mlavik1/UnityVolumeRendering/releases/download/2024.4/UnityVolumeRendering-2020.3+.unitypackage)** and import it to your existing project
  - Once you imported, enable SimpleITK by clicking on Volume Rendering in the menu -> Settings. A window will pop-up, click Enable SimpleITK.
  - If you use Mac with Apple Silicon and face issues with SimpleITK, try the following:

- First, you could try to Disable SimpleITK in the UnityVolumeRendering settings if you previously enabled it.
- Delete all the files inside SimpleITK folder under Assets\UnityVolumeRendering\Assets\3rdparty\SimpleITK
- Download the new version of SimpleITK here **https://github.com/SimpleITK/SimpleITK/releases/tag/v2.5.2 (https://github.com/SimpleITK/SimpleITK/releases/tag/v2.5.2)**
  - **(https://github.com/SimpleITK/SimpleITK/releases/tag/v2.5.2)** Download SimpleITK-2.5.2-CSharp-macosx-11.0-anycpu.zip
- Unzip and copy those files to the SimpleITK folder in your Unity project Assets\UnityVolumeRendering\Assets\3rdparty\SimpleITK
- Test it and let me know if you need support.

2. Medical volume dataset — **Skip this step, if you have already downloaded and worked on Assignment 5**

- You will use an open-source dataset from **MedSeg Liver segments dataset (https://figshare.com/articles/dataset/MedSeg_Liver_segments_dataset/13643252)**
  - I cleaned and made it ready for your practice.
  - Download **Liver_Mask_01.tif (https://sjsu.instructure.com/courses/1613868/files/84330643?wrap=1) (https://sjsu.instructure.com/courses/1613868/files/84330643/download)** to your folder.

Step 1: Create 3D Texture from volume data

- In this step, you can learn to create a menu item in Unity to quickly read volume data without pressing the play button.
- Create a new folder called Editor in your Script folder. It's to avoid any errors during building if you want to build an executable file later on.
- Create a new script and import namespaces including

```
using UnityEngine;
using UnityEditor;
using System;
using System.Runtime.InteropServices;
```

- Create a new static function with MenuItem as keyword, e.g., [MenuItem("CS116A/3DTexture")]

```
[MenuItem("CS116A/3DTexture")]
static void CreateTexture3DVolume()
{
```

```
}
```

- Check in the Unity Editor if you have a new menu appeared on the top bar.
- If everything is fine, you can use OpenFilePanel to ask user to browse the file.

```
string path = EditorUtility.OpenFilePanel("Volume Image", "", "tif");
if (path.Length != 0)
{

}
```

- Similar to Assignment 5, you can read the volume image using SimpleItk and debug the number of pixels if it works correctly.

```
var imageFileReader = new itk.simple.ImageFileReader();
imageFileReader.SetFileName(path);
var volImage = imageFileReader.Execute();
Debug.Log(volImage.GetNumberOfPixels());
```

- Then, cast the image to sitkFloat32 and read the buffer image to float array.

```
        //Caset Image to Float32
        var volumeImage = itk.simple.SimpleITK.Cast(volImage, itk.simple.PixelIDValueEnum.sitkFloat32);

        // Assign volume to grid
        int depth = (int)volumeImage.GetDepth();
        int height = (int)volumeImage.GetHeight();
        int width = (int)volumeImage.GetWidth();

        int length = width * height * depth;
        IntPtr bufferImg = volumeImage.GetBufferAsFloat();
        float[] bufferAsArrayImg = new float[length]; // Allocates new memory the size of input
        Marshal.Copy(bufferImg, bufferAsArrayImg, 0, length);
```

- Instead of creating a grid to work with Marching cube, you can create an empty 3D color array to store color data and to work with Texture3D in the next step.
- You can then filter the pixel value similar to previous assignment.

```
         // Create a 3-dimensional array to store color data
          Color[] colors = new Color[length];

          for (int k = 0; k < depth; k++)
          {
              for (int j = 0; j < height; j++)
```

```
        {
            for (int i = 0; i < width; i++)
            {
                int idx = i + width * (j + height * k);
                float pixel = bufferAsArrayImg[idx];

                if (pixel > 0)
                {
                    colors[idx] = new Color32((byte)pixel, (byte)pixel, (byte)pixel, 255); //Assign pixel to color array
                }
            }
        }
    }
```

- Now, it's time to create and assign color values to Texture3D.

```
TextureFormat format = TextureFormat.RGBA32;
TextureWrapMode wrapMode = TextureWrapMode.Clamp;

// Create the texture and apply the parameters
Texture3D texture = new Texture3D(width, height, depth, format, false);
texture.wrapMode = wrapMode;

// Copy the color values to the texture
texture.SetPixels(colors);

// Apply the changes to the texture and upload the updated texture to the GPU
texture.Apply();
```

- Finally, you can save the texture to Assets folder in the Unity Project.

```
// Save the texture to your Unity Project
AssetDatabase.CreateAsset(texture, "Assets/MyCS116A_3DTexture.asset");
```

- During your screen recording, make sure to click on the `MyCS116A_3DTexture` file in your Unity project to inspect and preview the volume data to get full credit for this step.

Step 2: Create a volume shader to render 3D Texture

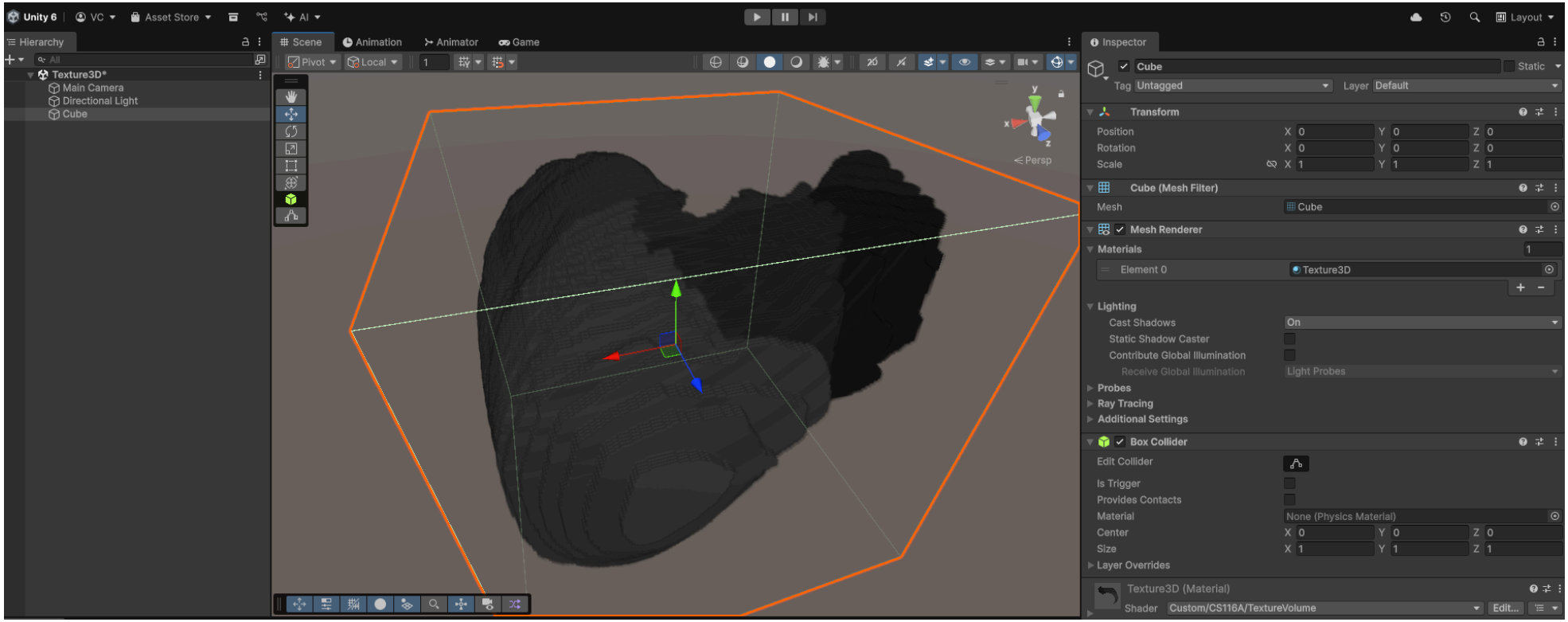
- In the **Project** window, use **Add** > **Shader** > **Unlit Shader** to create a new basic shader.
- Replace the shader code with the following example code.

```
Shader "Custom/CS116A/TextureVolume"
{
    Properties
    {
        _MainTex("Texture", 3D) = "white" {}
        _Alpha("Alpha", float) = 0.02
        _StepSize("Step Size", float) = 0.01
    }
    SubShader
    {
        Tags { "Queue" = "Transparent" "RenderType" = "Transparent" }
        Blend One OneMinusSrcAlpha
        LOD 100

        Pass
        {
            HLSLPROGRAM
            #pragma vertex vert
            #pragma fragment frag

            #include "UnityCG.cginc"

            // Maximum number of raymarching samples
            #define MAX_STEP_COUNT 128

            // Allowed floating point inaccuracy
            #define EPSILON 0.00001f

            struct appdata
            {
                float4 vertex : POSITION;
            };

            struct v2f
            {
                float4 vertex : SV_POSITION;
                float3 objectVertex : TEXCOORD0;
                float3 vectorToSurface : TEXCOORD1;
            };

            sampler3D _MainTex;
            float4 _MainTex_ST;
            float _Alpha;
            float _StepSize;

            v2f vert(appdata v)
            {
                v2f o;

                // Vertex in object space. This is the starting point for the raymarching.
```

```
                o.objectVertex = v.vertex;

                // Calculate vector from camera to vertex in world space
                float3 worldVertex = mul(unity_ObjectToWorld, v.vertex).xyz;
                o.vectorToSurface = worldVertex - _WorldSpaceCameraPos;

                o.vertex = UnityObjectToClipPos(v.vertex);
                return o;
            }

            float4 BlendUnder(float4 color, float4 newColor)
            {
                color.rgb += (1.0 - color.a) * newColor.a * newColor.rgb;
                color.a += (1.0 - color.a) * newColor.a;
                return color;
            }

            fixed4 frag(v2f i) : SV_Target
            {
                // Start raymarching at the front surface of the object
                float3 rayOrigin = i.objectVertex;

                // Use vector from camera to object surface to get ray direction
                float3 rayDirection = mul(unity_WorldToObject, float4(normalize(i.vectorToSurface), 1));

                float4 color = float4(0, 0, 0, 0);
                float3 samplePosition = rayOrigin;

                // Raymarch through object space
                for (int i = 0; i < MAX_STEP_COUNT; i++)
                {
                    // Accumulate color only within unit cube bounds
                    if (max(abs(samplePosition.x), max(abs(samplePosition.y), abs(samplePosition.z))) < 0.5f + EPSILON)
                    {
                        float4 sampledColor = tex3D(_MainTex, samplePosition + float3(0.5f, 0.5f, 0.5f));
                        sampledColor.a *= _Alpha;
                        color = BlendUnder(color, sampledColor);
                        samplePosition += rayDirection * _StepSize;
                    }
                }

                return color;
            }
            ENDHLSL
        }
    }
}
```

- Right-click the shader in the **Project** window and select **Create** > **Material** to create a new material from the shader.

- You can choose the shader you just created "*Custom/CS116A/TextureVolume*"
- Then, drag and drop the 3D Texture `MyCS116A_3DTexture` to the texture field.
- Adjust the transparency — alpha value to 1.
- In your Unity Scene, create a Cube gameobject and replace the material with the one you just created.
- Reset the transform to default position and rotation.
- Hooray! You should see the results without pressing the play button. And the volume data should be rendering directly without geometric surface.



Instructions to earn credits for the following steps:

- You can make a **new recording** and **submit a new video addition to the previous one**.
- Also walk me through the code for the new recording.

Step 3: Extension to Liver Segments

- Similar to previous assignment 5, in this step you can visualize individual liver segment with different color.
- Tip: instead of creating a list to store and create the surfaces like the way you did in assignment 5, you can simply create a condition and assign color directly to the 3D color array, e.g., if pixel == 1 => colors[idx] = Color.yellow;
- Change the each segment to the following color:
  - Liver Segment 0 = Yellow
  - 1 = Blue
  - 2 = Black
  - 3 = Red
  - 4 = Brown
  - 5= Pink
  - 6 = Orange
  - 7 = Purple
  - 8 = LightSkyBlue

3D Texture Volume Rendering

Criteria	Ratings		Pts
Step 1	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Step 2	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Step 3	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
			Total Points: 15