

Final Individual Project

New Attempt

- Due Nov 26 by 11:59pm
- Points 50
- Submitting a media recording or a file upload

Objective

The goal of this assignment is to help you practice and develop a small project for CS 116A.

- 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.
- **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:

1. Imported SimpleITK and geometry3Sharp — **Skip this step, if you have done already**
2. Medical volume dataset:
 - In this assignment, you will use an open-source dataset from [Liver segmentation 3D-IRCADb-01](https://www.ircad.fr/research/datasets/liver-segmentation-3d-ircadb-01/) (<https://www.ircad.fr/research/datasets/liver-segmentation-3d-ircadb-01/>) similar to the work in Assignment 7
 1. Download the first patient case (DOB: 1944) and unzip the **3Dircadb1.1.zip** — **Skip this step, if you have already downloaded the dataset for Assignment 7**
 2. Inside this folder, unzip **MASKS_DICOM.zip**
 3. Let's work with the following datasets. However, if you have time, I encourage you to explore all datasets.
 - artery
 - liver
 - livertumor01

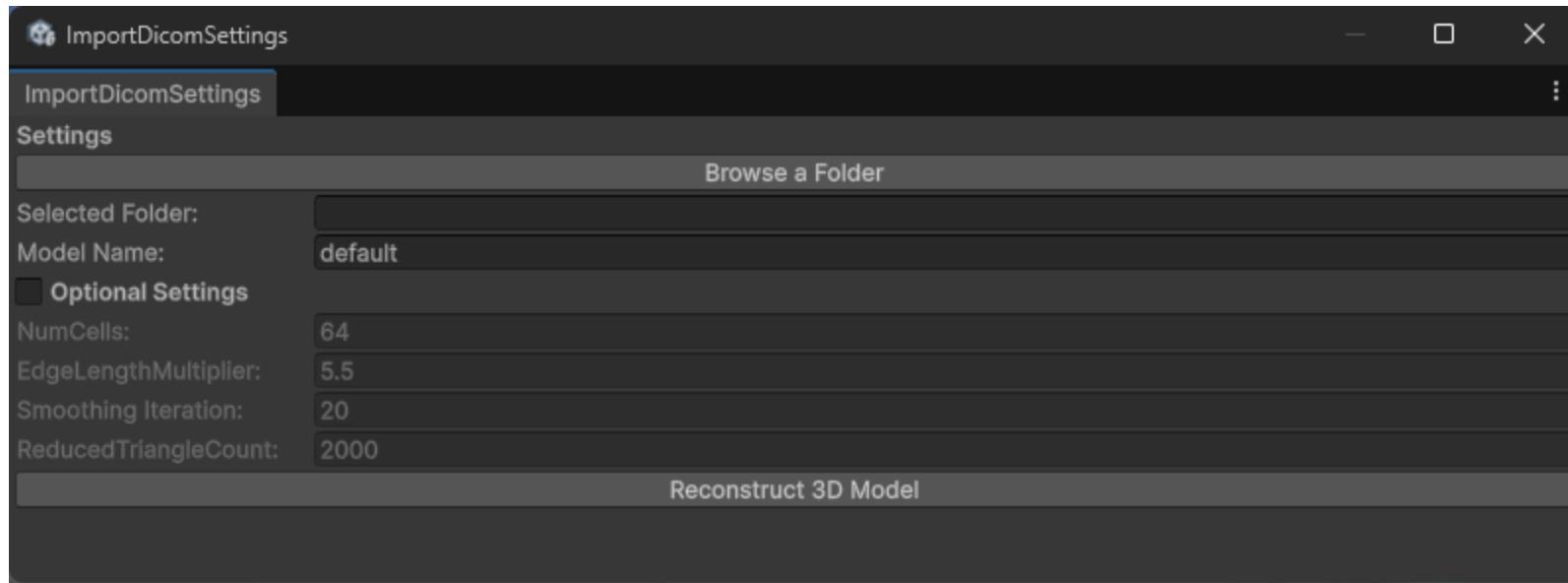
- livertumor02
- portalvein
- venoussystem

3. LLM Unity — Choose one the following methods:

- *Method 1: Install using the asset store*
 - Open the [LLM for Unity](https://assetstore.unity.com/packages/slug/273604)  asset page and click **Add to My Assets**
 - Open the Package Manager in Unity: **Window > Package Manager**
 - Select the **Packages: My Assets** option from the drop-down
 - Select the **LLM for Unity** package, click **Download** and then **Import**
- *Method 2: Install using the GitHub repo:*
 - Open the Package Manager in Unity: **Window > Package Manager**
 - Click the **+** button and select **Add package from git URL**
 - Use the repository URL <https://github.com/undreamai/LLMUnity.git> and click **Add**

Step 1: 3D Surface Reconstruction

- Similar to Assignment 6, you can create a menu, e.g., CS116A/Create3DModel, to reconstruct 3D surface model and save it to your Unity project
- In this assignment, let's create an Editor window to work with user inputs



- Instead of deriving from MonoBehaviour, you can work with **EditorWindow**
- Create a static function and add a menu item similar to Assignment 6

```
[MenuItem("CS116A/Create3DModel")]
```

- Show existing window instance. If one doesn't exist, make one.

```
EditorWindow.GetWindow(typeof(Your Class Name));
```

- Now, you can work with GUI to create input fields and buttons

```
private void OnGUI()
{
    GUILayout.Label("Settings", EditorStyles.boldLabel);

    if (GUILayout.Button("Browse a Folder"))
    {
        // Open a folder panel to select a folder
        datasetFolder = EditorUtility.OpenFolderPanel("Select DICOM Directory", "", "");
        modelName = Path.GetFileName(datasetFolder);
    }

    datasetFolder = EditorGUILayout.TextField("Selected Folder:", datasetFolder);
    modelName = EditorGUILayout.TextField("Model Name:", modelName);

    groupEnabled = EditorGUILayout.BeginToggleGroup("Optional Settings", groupEnabled);
    numCells = EditorGUILayout.DoubleField("NumCells:", numCells);
```

```

edgeLengthMultiplier = EditorGUILayout.FloatField("EdgeLengthMultiplier:", edgeLengthMultiplier);
remeshPasses = EditorGUILayout.IntField("Smoothing Iteration:", remeshPasses);
reducedTriangleCount = EditorGUILayout.IntField("ReducedTriangleCount:", reducedTriangleCount);
EditorGUILayout.EndToggleGroup();

if (GUILayout.Button("Reconstruct 3D Model"))
{
    if (datasetFolder.Length > 0)
    {
        // Your DICOM image reading and mesh reconstruction codes
        //this.Close();
    }
}
}

```

- For datasets in this assignment are series of DICOM volume images, so you need to read image series similar to the work in Assignment 7
- In assignment 5, the spacing of dataset is 1, 1, 1, which works very well with Marching Cube. However, in these datasets, the spacing are different, which are challenging to work with Marching Cube for surface reconstruction. You can consider using Resampling technique to resample an image to have the spacing of 1, 1, 1 as an equal size of a cube (cell) to work with Marching Cube.

```

private Image ResampleVolumeImage(Image _volumeImage)
{
    // Create a ResampleImageFilter
    ResampleImageFilter resample = new ResampleImageFilter();

    // Set the interpolator (e.g., Linear)
    resample.SetInterpolator(InterpolatorEnum.sitkLinear);

    // Set the output direction and origin to match the input image
    resample.SetOutputDirection(_volumeImage.GetDirection());
    resample.SetOutputOrigin(_volumeImage.GetOrigin());

    // Define the new spacing
    VectorDouble newSpacing = new VectorDouble { 1.0, 1.0, 1.0 }; //To get it works with Marching cube's cell
    resample.SetOutputSpacing(newSpacing);

    // Calculate the new size based on the new spacing
    var originalSize = _volumeImage.GetSize();
    var originalSpacing = _volumeImage.GetSpacing();
    VectorUInt32 newSize = new VectorUInt32();

    for (int i = 0; i < _volumeImage.GetDimension(); i++)
    {
        newSize.Add((uint)Math.Ceiling(originalSize[i] * originalSpacing[i] / newSpacing[i]));
    }
}

```

```

    resample.SetSize(newSize);

    // Set a default pixel value for points outside the original image bounds
    resample.SetDefaultPixelValue(0.0);

    // Execute the resampling
    var volumeImage = resample.Execute(_volumeImage);

    //SimpleITK.WriteImage(volumeImage, "E:/portalvein.mhd");

    return volumeImage;
}

```

- For 3D surface reconstruction, you can use similar techniques in Assignment 5. However, the parameters, including numCells, edgeLengthMultiplier, remeshPasses, and reducedTriangleCount should be given by the user in the Editor window.
- You may notice in Assignment 5 that the position of reconstructed model is not at its centroid. It's because it's relative to an origin point of the volume image. To make it's easier for interaction, you may need to change their default positions to the center of volume. It's noteworthy to mention that Unity utilizes a left-handed coordinate system.

```

    MeshNormals.QuickCompute(_result);
    var centroid = new Vector3d(width, height, depth)/2;
    MeshTransforms.Translate(_result, centroid * -1);
    MeshTransforms.FlipLeftRightCoordSystems(_result);

```

- You can create a preview version by creating Unity gameobject in the scene. However, the ultimate goal is to save the 3D mesh to the project folder, e.g.,

```

var savePath = $"{Application.dataPath}/{_modelName}.obj";
Debug.Log(savePath);
g3UnityUtils.WriteOutputMesh(_result, savePath);

```

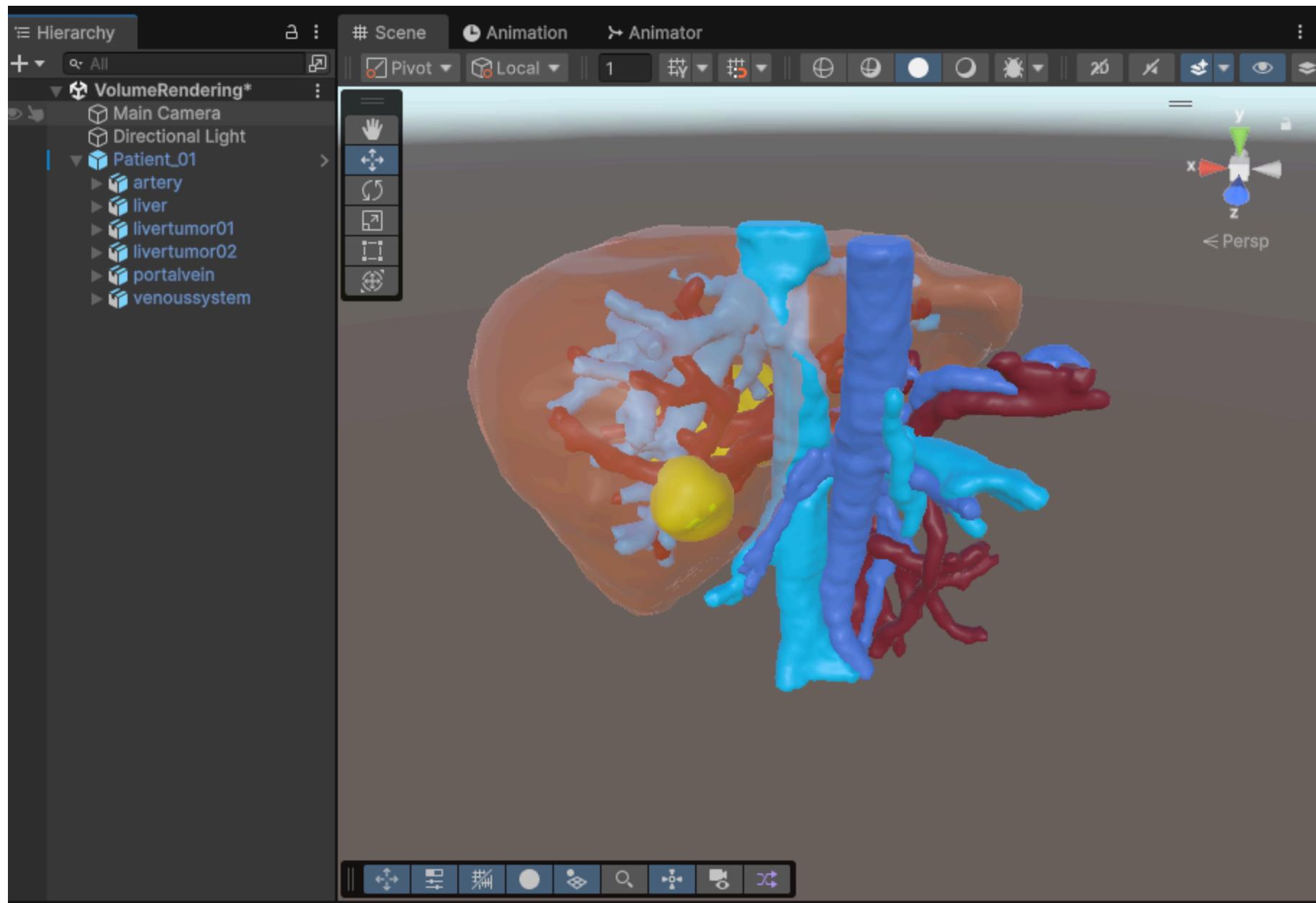
- Finally, you can create surface meshes for the above datasets. Here are the suggested parameters for the datasets:

- **Portalvein, artery, venoussystem:**

- numcells = 256
- reducedTriangleCount = 10000
- edgeLengthMultiplier = 2
- remeshPasses = 20

- **Liver, livertumor01, livertumor02:**

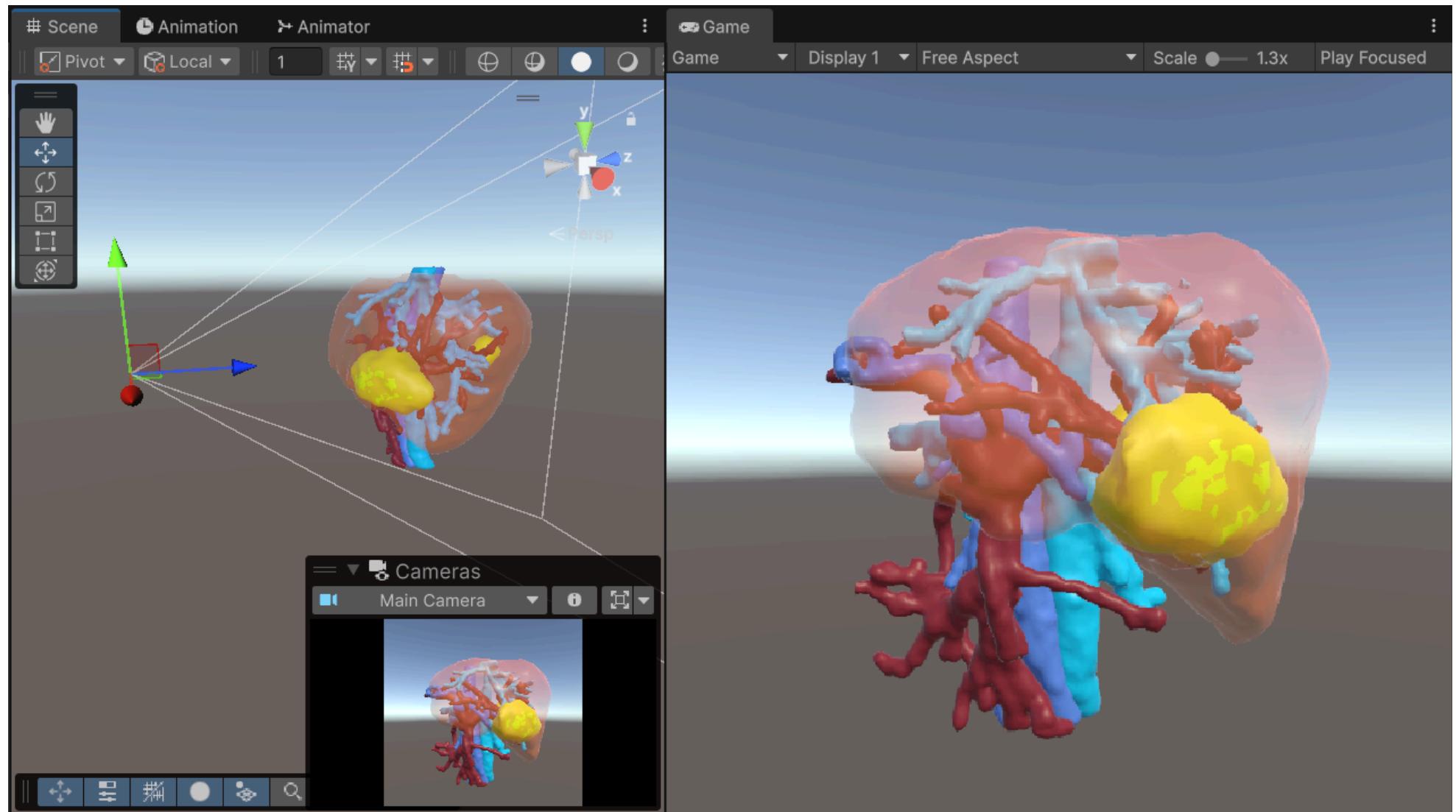
- numcells = 64
 - reducedTriangleCount = 2000
 - edgeLengthMultiplier = 5.5
 - remeshPasses = 20
- To get the credit for this step, you should create **materials** for those meshes and create a parent node called "**Patient_01**" and drag those meshes to the scene similar to the screenshot below.



Step 2: Interaction

- Make a **new recording video** to earn credit for this step. Also walk me through the code for the new recording.

- In this step, the goal is to allow users to interact with your 3D models in the Play mode (Game view). It's crucial when you deploy your application to the end users.
- You may reset all transforms inside the Patient_01 to default.
- You may reset the transform of the Patient_01, and update X axis of Rotation to 90 if you want to have a similar visualization like my screenshot.
- Update your camera transform to capture the models to make sure they're inside the view frustum and visible in the Game mode (see screenshot as an example).



- Create a new script and attach it to the Patient_01 node.

- I would declare a few parameters, e.g.,

```
public float Speed = 5;  
public Camera mainCamera;  
private float cameraZDistance;
```

- In the start function, you can assign the camera and get the Z distance from the main camera

```
mainCamera = Camera.main;  
cameraZDistance = mainCamera.WorldToScreenPoint(transform.position).z;
```

- For real-time interaction, you can work in the update function by detecting mouse left and right click. Let's assume:

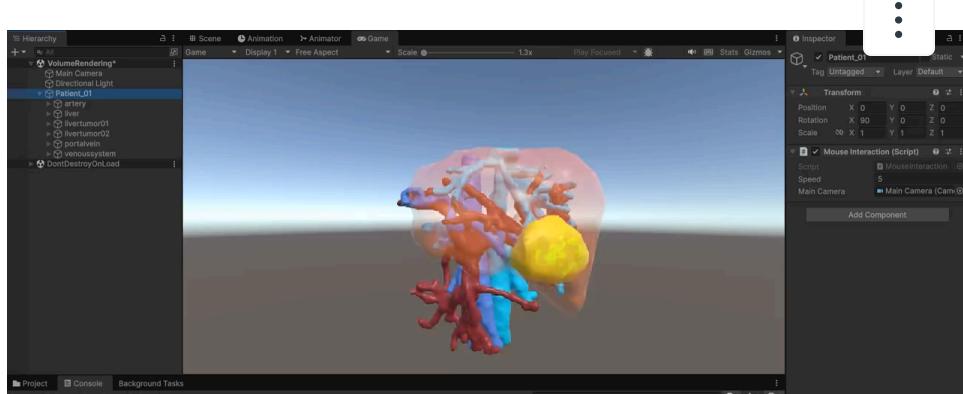
- Mouse left click for changing the object's rotation

```
if (Input.GetMouseButton(0)) //Mouse left click  
{  
    transform.eulerAngles += Speed * new Vector3(-Input.GetAxis("Mouse Y"), Input.GetAxis("Mouse X"), 0);  
}
```

- Mouse right click for changing the object's position

```
if (Input.GetMouseButton(1)) //Mouse right click  
{  
    Vector3 screenPosition = new Vector3(Input.mousePosition.x, Input.mousePosition.y, cameraZDistance); //z axis added to screen point  
    Vector3 newWorldPosition = mainCamera.ScreenToWorldPoint(screenPosition); //Screen point converted to world point  
  
    transform.position = newWorldPosition;  
}
```

- The results of Step 2 should show the interaction similar to the demo video below:



▶ 🔊 0:00/0:00



Step 3: UI and AI-integrated assistant

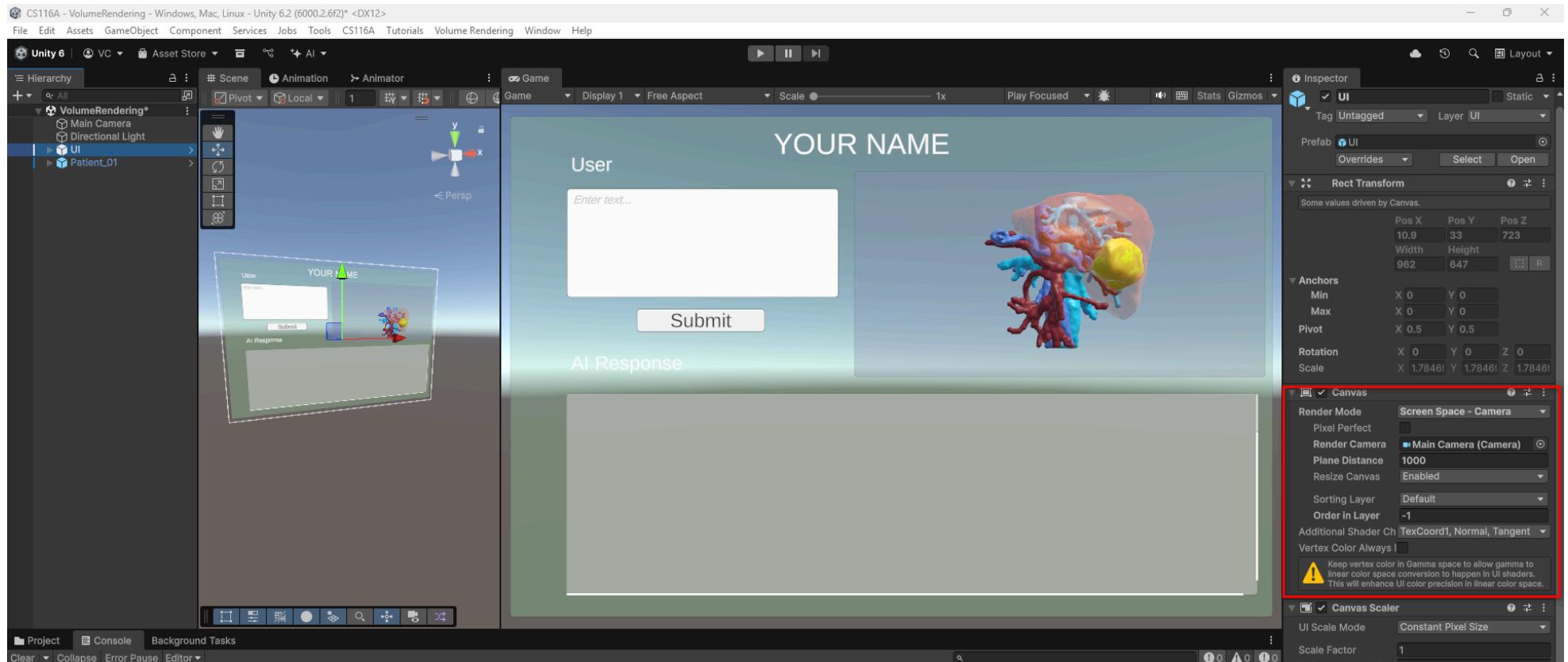
This is the final piece I wanted you to learn, and I hope it proves useful in the future if you decide to incorporate AI-related features into your games or applications.

- Make a **new recording video** to earn credit for this step. Also walk me through the code for the new recording.
- In this step, you will learn how to create and update user interface for interactions as well as integrate AI-related features into your application. Let's assume we create a medical anatomy education system with AI-based assistant that allows users to ask questions related to human anatomy.

Step 3.1: UI Setup

- Create UI components, including panels, text inputs, and buttons.
 - I encourage you to create your own UI if you have time. However, I created a prefab [UI.prefab](#) (<https://sjsu.instructure.com/courses/1613868/files/84600131?wrap=1>)
 - (https://sjsu.instructure.com/courses/1613868/files/84600131/download?download_frd=1) that hopefully you can easily make use of it.
- Assign Main Camera into the Render Camera in the Canvas as shown in the screenshot and change the value of plane distance to 500 or 1000 depending on your scene.
- Modify your Patient_01 node's transform to fit the model panel similar to the screenshot.

- Change the label '**YOUR NAME**' with your real name :)



Step 3.2: LLM configuration

- You can create a new empty node (game object) in the scene called '*controller*'.
- Click **Add Component**, search and **LLM** and then search and add **LLM Character**
- In the LLM script, click Download model -> Medium models -> Llama 3.2 3B as we use Meta AI model as an example. You can also try tiny, small, or large models depending on your needs.
 - The main difference between small, medium, and large AI models lies in their size (number of parameters), capabilities, resource requirements, and deployment needs.
 - Learn more about Meta Llama model 3.2 <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/> (<https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>).
 - Once you downloaded, make sure that you selected the model in the editor

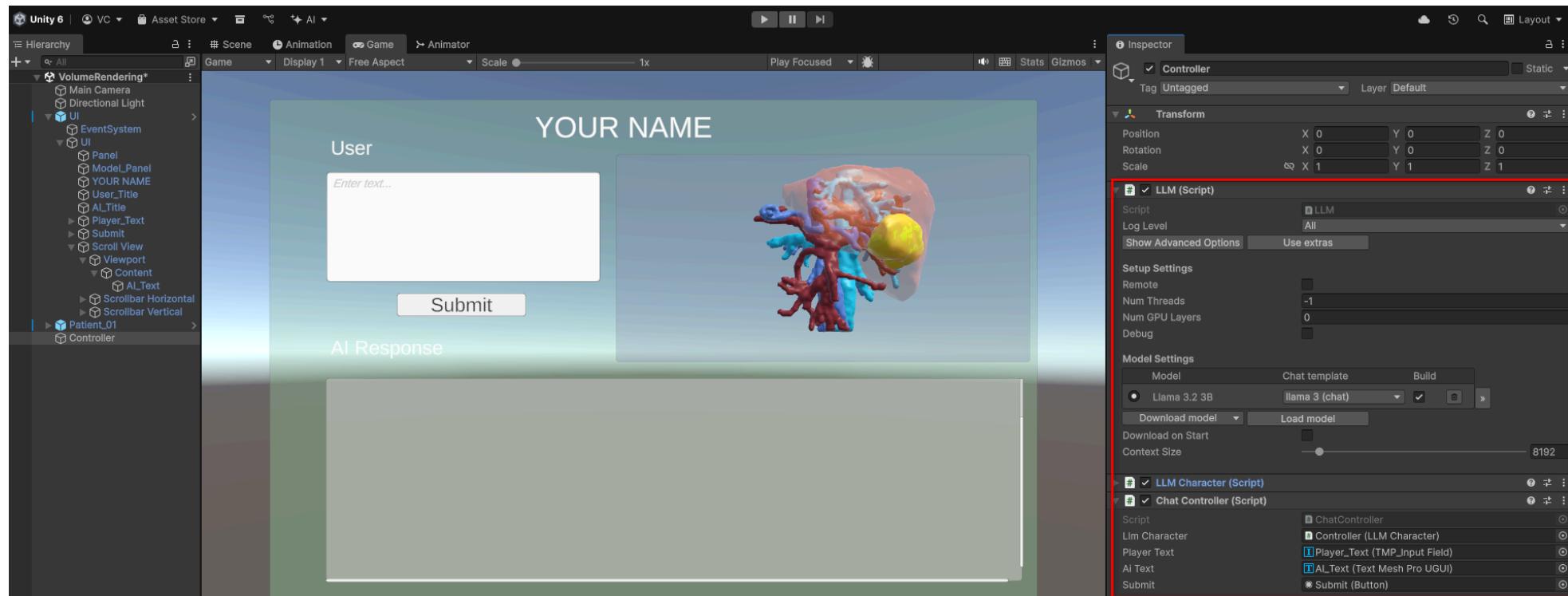
- In this assignment, I would use the default configuration for LLM Character. The idea is just to show the capabilities and get your first impression that you can work with AI-related features in Unity that you can take advantages of in the future.

Step 3.3: UI Controller

- In this step, you can now configure your UI with LLM. To do that, you need to create a script and add it to your *controller* node in the scene
- You can make references to the UI components, e.g.,

```
public LLMUnity.LLMCharacter llmCharacter;
public TMPro.TMP_InputField playerText;
public TMPro.TMP_Text aiText;
public UnityEngine.UI.Button submit;
```

- Make sure you correctly reference those objects in the editor



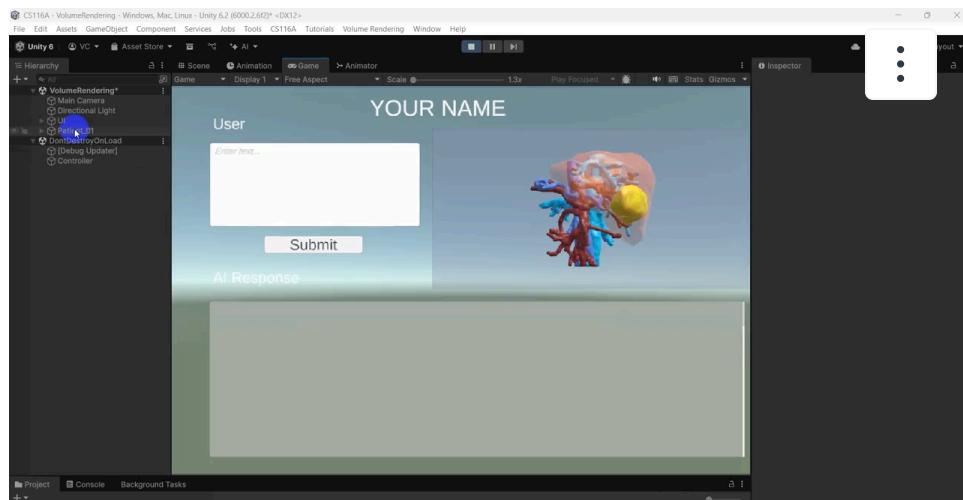
- In the start function, you can add listener when the button click, e.g., when the submit button is clicked, it calls `OnSubmitButtonClick` function

```
submit.onClick.AddListener(OnSubmitButtonClick);
```

- In the `OnSubmitButtonClick` function, you can now map the player's input text to the LLM Character to process and get the AI's response
`llmCharacter.Chat(playerText.text, HandleReply);`
- To handle the reply, you can create a new function called 'HandleReply' with the parameter *reply* to display the AI response text to the UI (AI_Text)

```
private void HandleReply(string reply)
{
    Debug.Log(reply);
    aiText.text = reply;
}
```

- If everything configures correctly, you can now test your application. See the demo video below as an example:



- If you have time, I recommend trying out function calling and Retrieval-Augmented Generation (RAG) if you're curious and want to learn more about AI integration.

Individual_Project

Criteria	Ratings		Pts
Step 1	20 pts Full Marks	0 pts No Marks	20 pts
Step 2	10 pts Full Marks	0 pts No Marks	10 pts
Step 3	20 pts Full Marks	0 pts No Marks	20 pts
			Total Points: 50