CS148 Homework 1 - Blender Setup

Grading on Monday, Sept 28 – see the pinned posts on Ed for details!

1 Assignment Outline

This first assignment is primarily setup and installation to ensure that you have the proper Blender environment for the class. The policies for this assignment differ from the usual course policies, so please read through everything carefully. **Instructions for what to submit and how to do so are at the end of this document.**

A common theme for the assignments in this class will be a heavy use of blue hyperlinks to embed outside resources and references. For instance, the following blue link should send you to the class Ed page. You will see many more hyperlinks throughout the document.

2 Collaboration Policy

For this setup assignment, you may consult with anyone and any resources you may find to make sure you get your Blender environment working. As a heads up, for future assignments, you may work with a partner to produce and submit the same deliverables together. The collaboration policy for that will be specified in the next HW document. Discussion of the quiz questions is allowed as long as the answers are not explicitly posted anywhere public (e.g. Ed).

3 Office Hours

Office Hours will start Week 2 with the release of the second assignment, and the logistics will be posted on Canvas. For this first installation assignment, please post any questions that you may have on the class Ed. We will also primarily post announcements (e.g. homework releases, office hour rescheduling, etc.) on Ed.

4 Blender Basics

What is Blender? Blender is a free and open-source 3D creation suite. It supports the entirety of the 3D pipeline – modeling, rigging, animation, simulation, rendering, and compositing, and even supports motion tracking, video editing, and game creation. See the official Blender website if you're interested in looking more into any of these features.

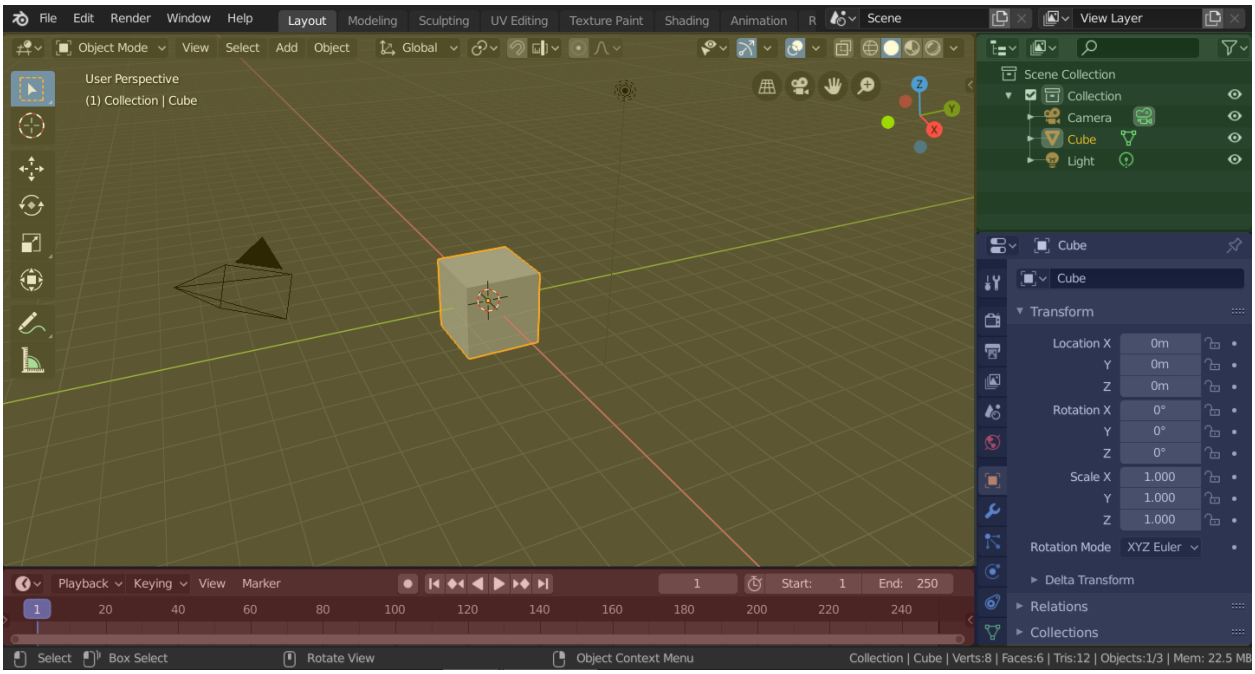
For this course, we will use Blender as a way to learn the basics of modeling, lighting, shading, texturing, and rendering. We will also use Blender's Python scripting API for a hands-on experience in coding various algorithms as well as our own, custom raytracer.

4.1 Installing Blender

Download and install the latest version of Blender from the official website. If you already have Blender installed, then make sure it's at least version 3.0 or higher. The screenshots in these

assignments are from an earlier version of Blender and may have slight text or graphical user interface (GUI) differences from the latest version. These differences should be minor enough that they won't affect any of the assignments, but if a major difference does come up, then don't be afraid to post on Ed!

When you run Blender, you should see a default scene with a cube, a light, and a camera. Alternatively, you can also get the default scene by creating a new project via **File → New → General** on the upper left of the screen.



This is the default workspace for Blender (more in the Blender documentation), with the:

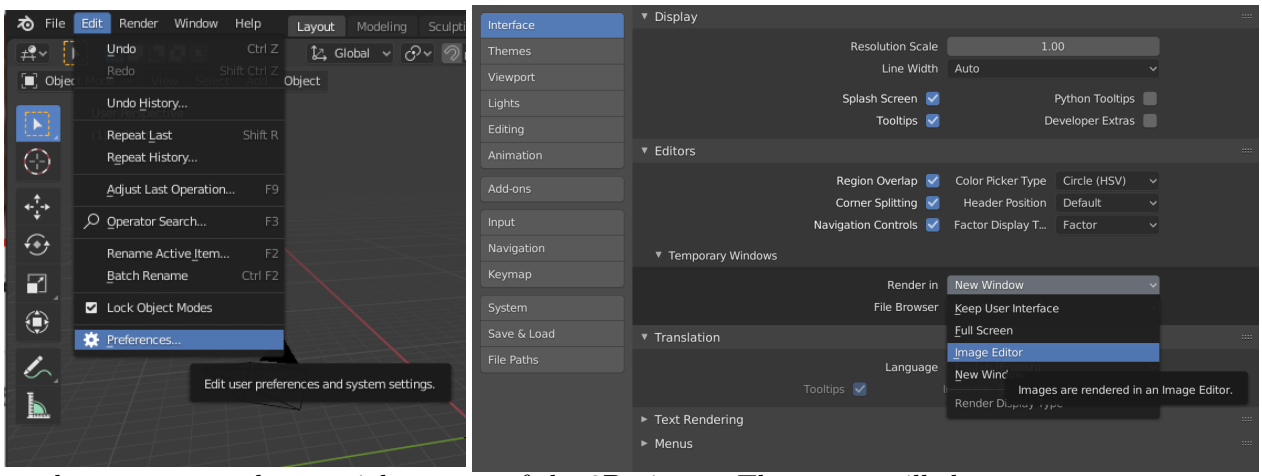
- 3D Viewport on the top left (yellow)
- Outliner on the top right (green)
- Properties editor on the bottom right (blue)
- Timeline on the bottom left (red)

Blender has a nice official introduction playlist for versions 2.8 and higher on their Youtube channel. These should still be applicable for the newer versions as well. If you haven't used Blender before, then we suggest watching the first two videos to learn the basics.

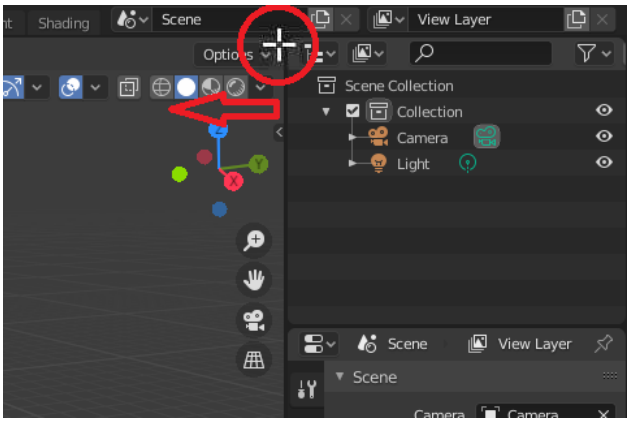
4.2 Editor Setup

We will first add an Image Editor to our workspace. This will be where we see our rendered results inside Blender. It will be more convenient than (as you may try later) saving your image each time you make changes to the scene, going to your file explorer, and then opening the rendered image. Setting up the editor will keep the render in the same window without popping out a new window every time.

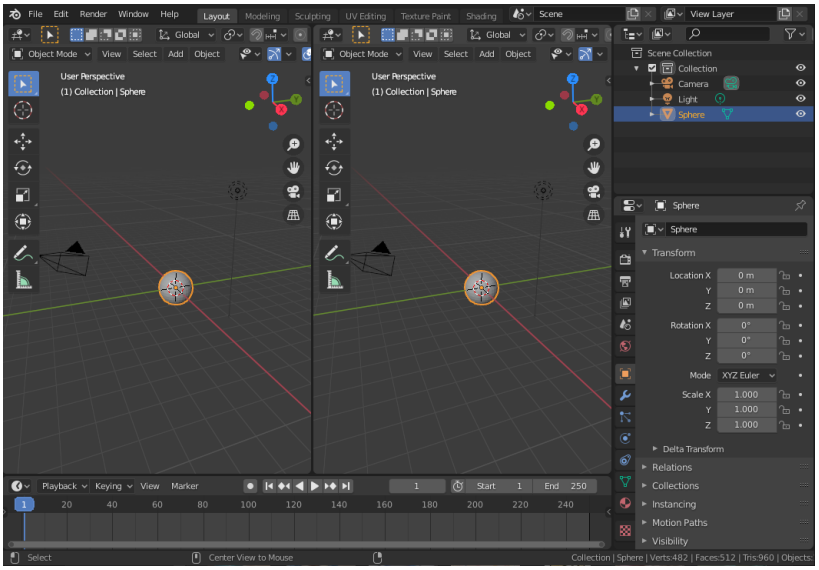
Go to the top bar, and click on **Edit → Preferences**. A new window should pop up. On the sidebar of this window, select **Interface**, then change the option for **Editors → Temporary Editors → Render in** to **Image Editor**.



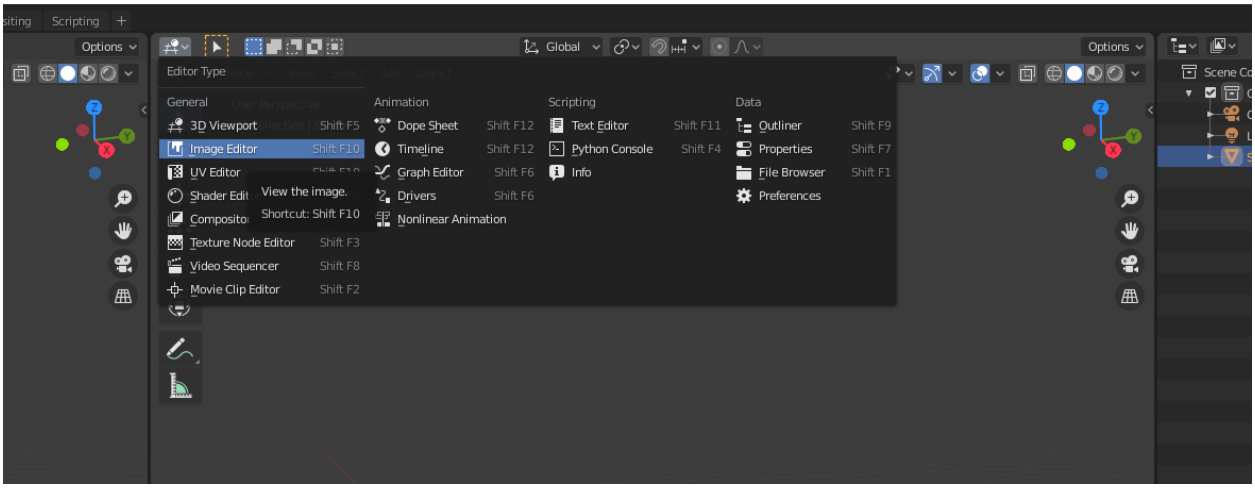
Hover the mouse over the top right corner of the 3D viewer. The cursor will change to a cross.



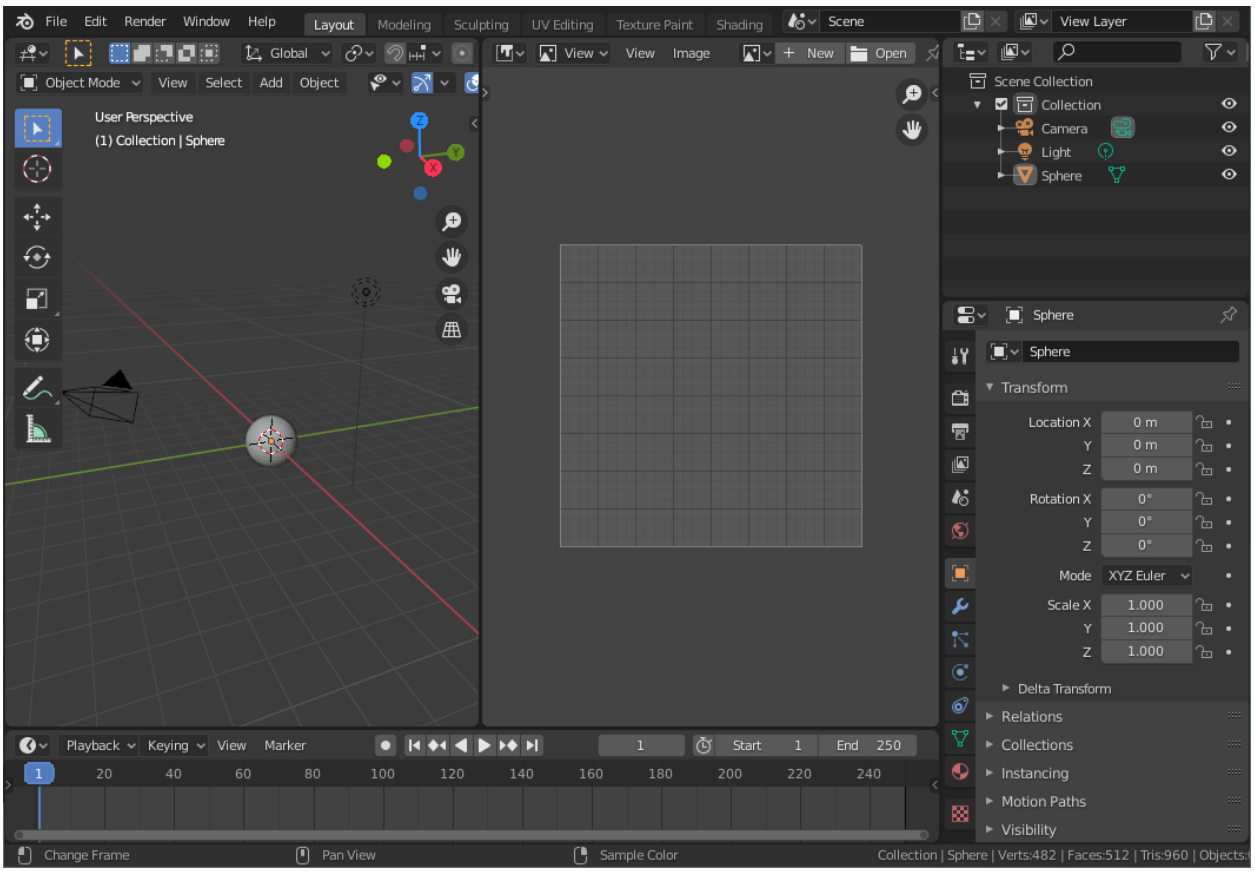
Drag left to create a new area.



In the top left of the new area, change the editor type to Image Editor.



Your screen should now look like this:



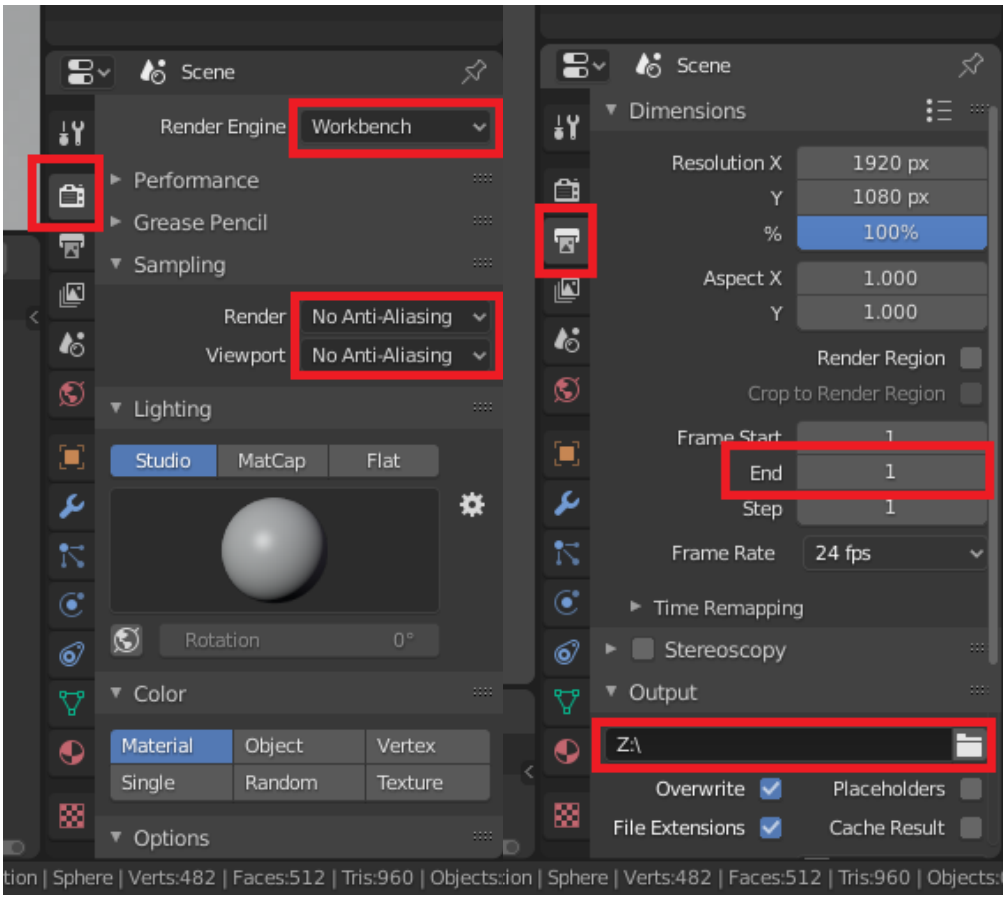
For more detail about areas and how to manipulate them, we optionally recommend the tutorial video.

4.3 Render and Save Image

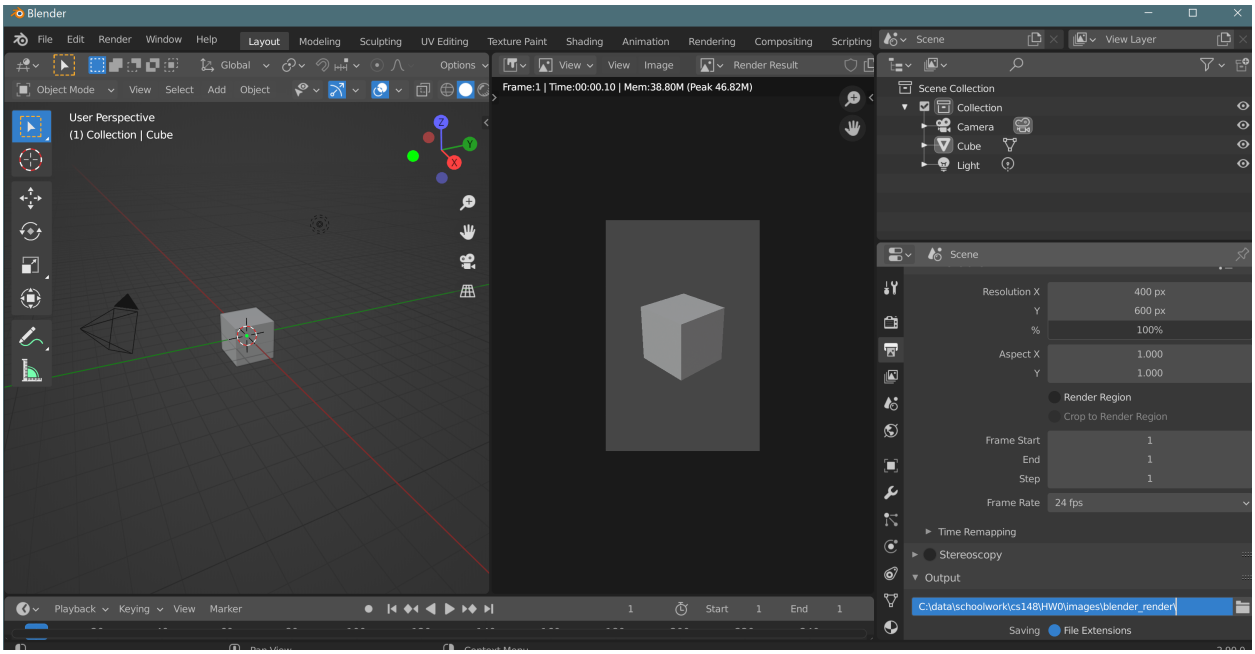
We can now change the render settings and render the image. By default Blender only saves animations, so here we change the length of the animation to 1. Also, we are playing with the

viewport scanline renderer for this homework, so we will use the Workbench render engine. For future assignments, we will use different render engines.

Go to the Properties Editor (bottom right) and select the camera icon (Render Properties) on the sidebar. Change the **Render Engine** to **Workbench**, and the **Sampling** for Render and Viewport to **No Anti-Aliasing**. (We will talk about about what this all means later!) Then click on the printer icon (Output Properties) on the sidebar. Change **Frame End** to **1**, the Output Path to where you want to save the image (e.g. **$MY_CS148_DIR/hw1/**), and the resolution to **X=480px,Y=640px**.



Now render the image by going to **Render → Render Animation** on the top bar (or shortcut Ctrl/Cmd F12), and it should save to the specified path automatically. (You can verify this yourself by going to the directory where you told Blender to save the image.) The rendered image will also show up as "Render Result" in the Image Editor we created in the previous step.



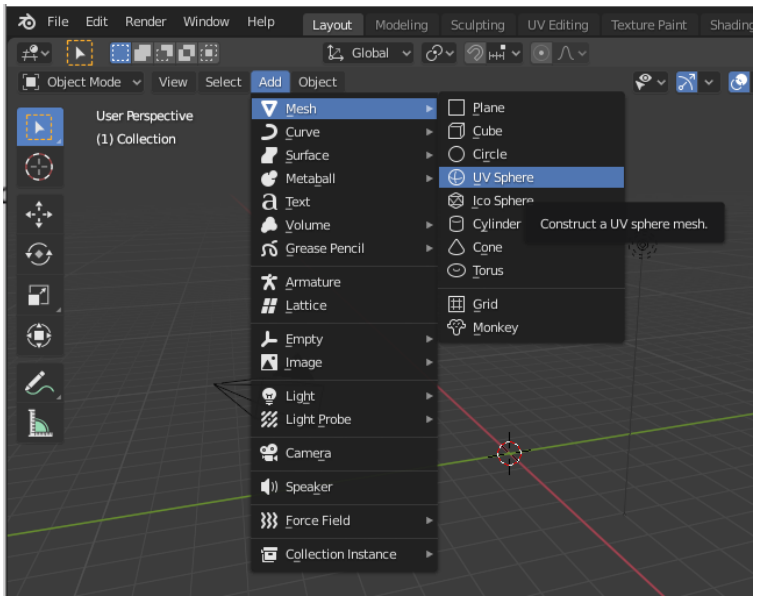
Now you have rendered your first image in Blender! Note that this image is rendered from the default camera's perspective.

4.4 Scene manipulation

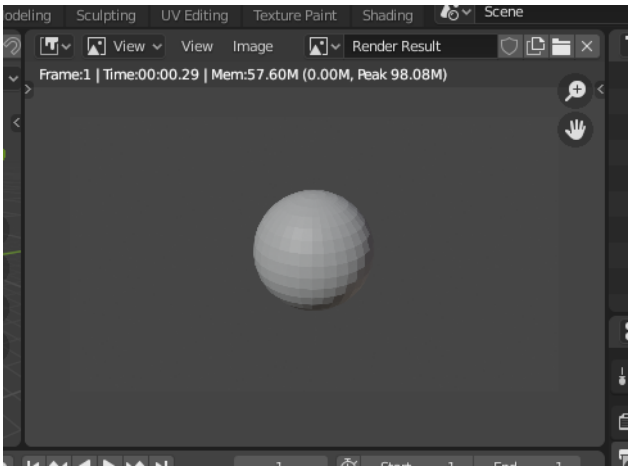
The default scene comes with a cube. We can add/delete objects in the scene. Blender stores some basic objects that we can directly add to the scene without constructing them from scratch. Here we will delete the cube and add a UV sphere.

Move your mouse over the cube, left click to select. An orange outline will appear around the cube, indicating it's being selected. Hit the Delete key to delete the cube.

Then go to the menu bar on top, and navigate to **Add → Mesh → UV sphere**. A sphere should appear in the center of the scene.



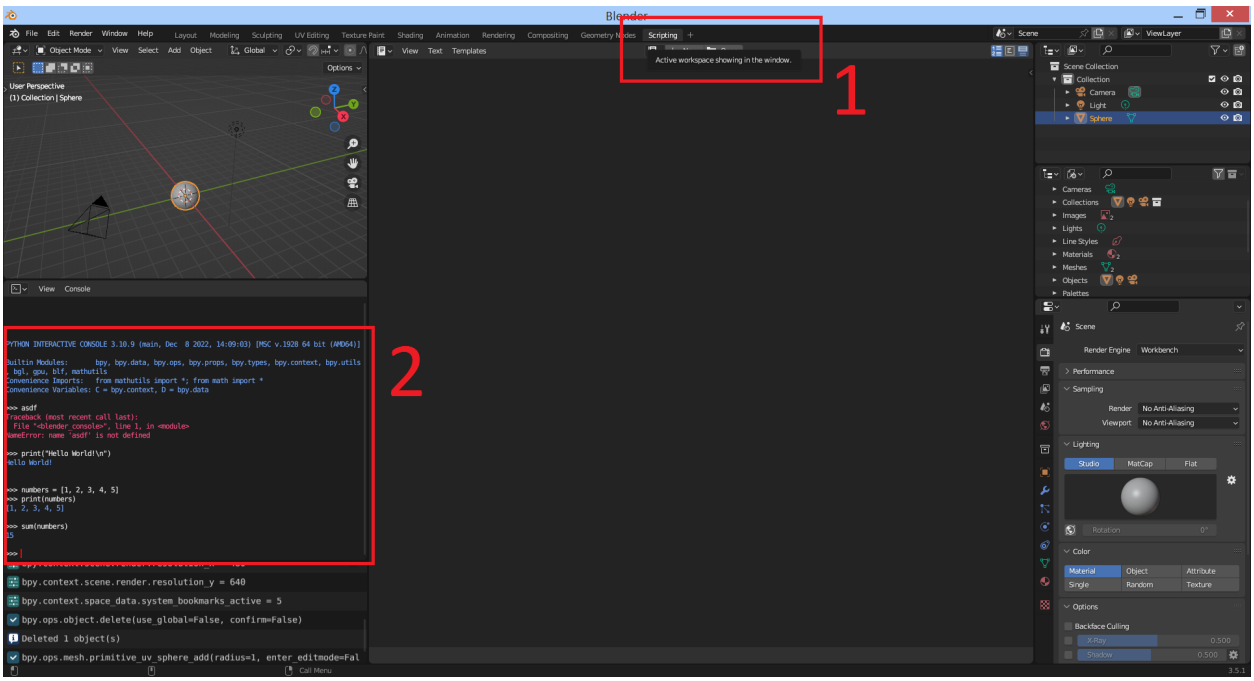
Press Ctrl/Cmd F12 (or **Render** → **Render Animation**) again to update the render. You should see a sphere instead of a cube now. Check that the rendered image of the sphere is saved correctly to your directory, having overwritten the render of the cube.



4.5 Scripting

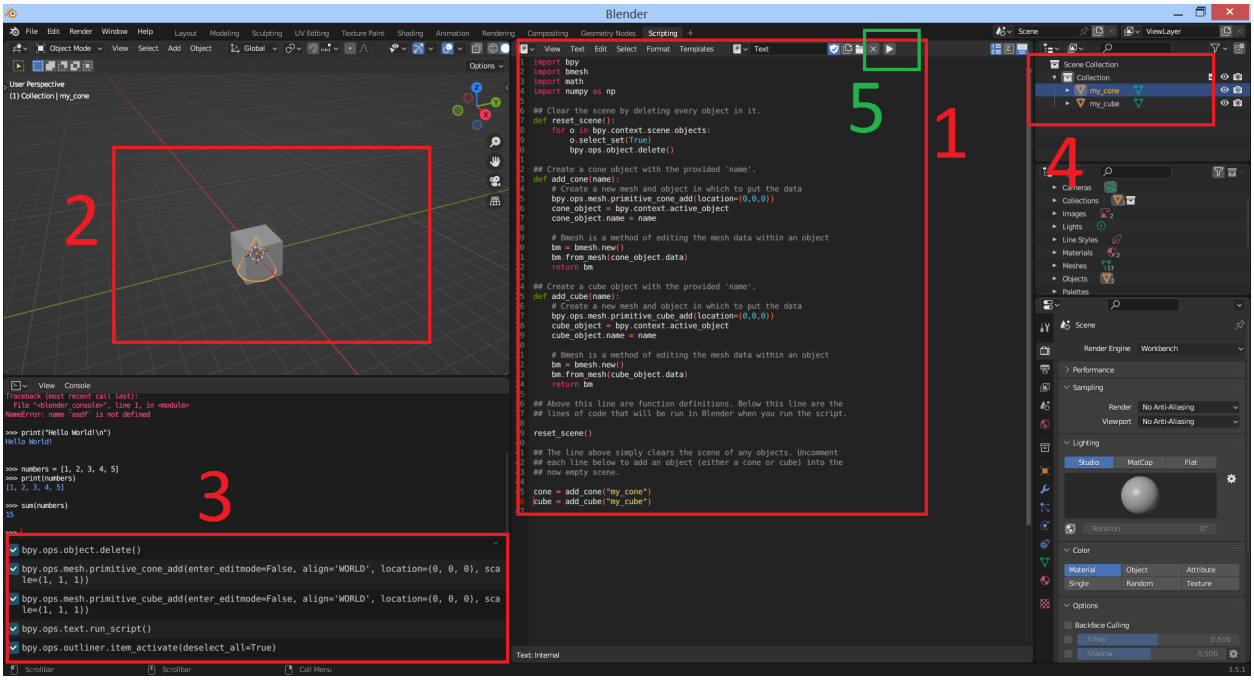
Blender provides a scripting environment that allows us to develop using Python in addition to using the GUI. By default, the Blender installation should have included a Python installation. You can check by locating your Blender installation folder and finding a Python folder within (e.g. on Windows, it may be located in **Program Files** → **Blender Foundation** → **Blender 5.2** → **5.2** → **Python**).

To access the scripting environment, click **Scripting** on the top bar (labeled **1** below). Depending on the size of your window or monitor, the **Scripting** tab might be cut off from view. If this is the case, then you can hover your mouse over the top bar and use the scroll wheel on your mouse to horizontally scroll through the top bar. Once you find the **Scripting** tab and click on it, the GUI should then resemble the following.



In the boxed area labeled `2` in the above image, you have the Python interactive console. Similar to other Python shells that you may have used in the past, this console provides you a space to experiment with snippets of Python code. You can see the documentation here. If you find the default layout not to your liking, then you can always move your mouse to the borders of each window and drag to resize them.

To write an actual Python script, press the `New` button that should be under the top bar, likely under `Scripting` if you haven't changed the default layout. You should now have an area in which you can type (labeled `1` in the picture below).



To try it out, copy and paste the code in this test script into the text editor. If you try to mess with the code in the text editor, then you might notice that it feels a lot more restrictive than your typical Python editor. Blender has a very friendly development environment compared to other 3D software, but it is still not on par with professional Python IDEs or professional text editors. **If you already have a preferred Python environment in which you write your code, then we recommend doing any code writing and edits in that first, THEN copy and paste the final code over to Blender to run.**

If you do decide to use the Blender text editor, then here are some handy shortcuts that can help with formatting:

- `Tab` for indent
- `Shift` + `Tab` for unindent
- `Ctrl` + `/` or `Cmd` + `/` to toggle comments

For more shortcuts in the text editor, please see the Blender documentation.

Press the run button (labeled `5` in green above) to run the script. Alternatively, if you do not see the button, then you can still run the script from the menu bar above the text editor with `Text` → `Run Script`. Without any changes, the script will simply delete every object in the scene

displayed in the upper-left "3D Viewport" window (2 in the image above). The log and any warnings and bug messages that may occur will be displayed in the Info Editor in the lower-left (3).

Try uncommenting the lines at the bottom of the script to add a cone and/or cube object to the scene after clearing it. You should then see the object(s) being added to the scene in the upper-left 3D Viewport as well as being listed in the upper-right "Scene Collection" (4). We'll discuss these parts of the GUI in more detail in later assignments.

5 Testing a Sample Final Project

To make sure your computer can handle the workload of the final project, we're having you play around with an example of what could be the start of a final project for this class. In this Google drive (log into your Stanford email), you will find four files. Open "DemoLecture.mp4" and you'll see a recording from a previous offering of this class that shows how one can start with the "DemoStart.blend" file and end with the "DemoFinished.blend" file in 30 minutes. The "market_background.jpg" image is the one used in the recording for the scene background. **NOTE: The files here CANNOT be used for your own final project later in this course, and the rendered image at the end of the recording is what we might consider a passing final project, but not quite enough to get one an A in the class.**

Skim through this recording and try your hands at doing similar operations in your own Blender environment as shown in the recording. It is OK if you don't fully follow everything shown – we'll eventually go over it all by the time you need to turn in your final project proposals. Use this as an opportunity to navigate the Blender UI with the recording as your guide, and take note of any lag and issues that you may notice from your computer. **If your computer shows extreme lag or freezes when opening the .blend files or when you're interacting with the Blender UI, then that's an indicator it unfortunately might not be capable of handling this class. Please reach out to the staff mailing list for potential alternative computing resources!**

6 How to Submit / How to get Graded

Instructions on the live grading component of this class have been posted on Ed. Please see the pinned post titled, "HW1 Posted + Grading Session Sign Up".

7 Quiz Questions

During the grading session, the CA will choose one of the following questions at random to ask. Please be prepared to give a brief (1-3 min) answer based on the lecture material. **You may reference your notes during the quiz session, but your answer cannot be read from a script, web search results, or live AI. The CA will ask you follow-up questions to make sure you understand the material beyond a prepared script!**

Additionally, **the use of AI tools to directly generate answers to these questions is prohibited and considered an honor code violation!**

- Explain what it means for light to be absorbed or reflected. If an object absorbs all incoming colors of light except red, then how does that affect the way we perceive the object with our human eyes? What if an object absorbs all incoming light? How would we perceive that? What about an object that reflects all incoming light?

- Explain the RGB color space in terms of trichromatic theory. Why is it sufficient to represent colors with just 3 numbers? What do the numbers mean, e.g. what color would be represented by the vector $(1, 0, 1)$ (aka $(255, 0, 255)$) and why? What's the difference between using numbers between 0-1 vs 0-255? What would halving the numbers do to the color, e.g. $(1, 0, 1) \rightarrow (0.5, 0, 0.5)$?

- What is the difference between a black and white (gray scale) image vs an RGB (colored) image? Think about the Y-component (aka the luminance channel) of the YUV color space, or alternatively think about the concept of luminance in general. Why do you think a single "Y" is enough to represent a gray scale image when a fully-colored image needs an "R", "G", AND "B"?

8 What to Demo / Submit

During the grading session, please screenshare with your CA and:

- (1 pt) Create a new Blender project within Blender and replace the default cube in the Blender UI with a UV sphere.

- (1 pt) Show a saved image of a UV sphere rendered using Blender's **Workbench** render engine. (You can pre-render this ahead of time before the grading session and simply open the image from your file directory.)

- (1 pt) Show the result of running a script through the **Scripting** tab of the Blender UI. (You can use the provided test script from the tutorial above.)

- (1 pt) Show that the finished sample final project file ("DemoFinished.blend") can be navigated without any lag or issues on your computer.

- (1 pt) Answer the quiz question given by the CA correctly.

The CA will ask follow-up questions for any of the above to make sure you have everything running smoothly and are firm in your understanding of homework and course concepts. CAs are encouraged to further question and look for tangents in order to create a more interesting conversation. As stated under "Engagement" in the class website, students who do exceptionally well will earn an extra Mastery Point. These points are used to upgrade some students from an A to an A+ final grade (there is no other way to earn an A+ in the course). There is no partial credit for any of the points for this assignment – you either get the point or don't get it.

If you suspect that your internet connection may not be stable enough to live demonstrate everything over screenshare, then you may take screenshots and video captures of your Blender UI ahead of time and present those files instead during the call.