CS148 Homework 2

Homework Due: Jul 10th **(Thursday)** at 2PM - 5PM (In-Person) or 7PM - 8PM (Remote)

0.1 Assignment Format

This assignment has **4** `TODO` s of varying lengths. Like with HW1, the TODOs are covered within the first few pages of the document. The rest of the handout consists of instructional tutorials.

Unlike HW1, which had a mix of coding and Blender GUI exercises, this HW will be purely done through the Blender GUI. We'll return to coding next week with HW3, in which we'll write our own raytracer. As for this HW, it is quite open ended on what you can submit for credit. Both the process of creating materials for objects and the process of camera placement for a scene give you a lot of artistic freedom! We encourage you to get creative with what you make for this HW!

0.2 Collaboration Policy, Office Hours, and Grading Session

All policies from here on are the same as they were for HW1. See the HW1 document for details.

Quiz 2

You will be randomly asked one of these questions:

- Explain the high-level idea of how we use the cross product to determine whether a point is inside a triangle defined by 3 vertices. How is this technique useful for rasterization?
- Explain what the surface normal direction represents at a vertex on a mesh surface. How does the surface normal come into play for computing the color at that specific vertex (using the lightning model)?
- What is the difference between the way Gourad Shading and Phong Shading each do smooth shading? Which one gives better results and at what cost?
- How do the cones in our eyes process light as color, and how do we mimic this effect with the way we represent colors for images? How would we define a color to fool the human eye into seeing yellow? What about magenta?
- What makes the way our eyes process light different from how a simple, pinhole camera processes light? What kind of camera mimics the way our eyes process light, and describe two advantages that it provides over the simple, pinhole camera when it comes to capturing images.

1 Assignment

1.1 Blender Cycles

For this HW, we'll need to get ahead of ourselves and utilize Blender's ray tracer to visualize our results, even though we haven't quite gotten to ray tracing yet. Thankfully, turning on the Blender ray tracer is incredibly simple. Whenever this HW prompts you to turn on "Cycles", which is the

name of Blender's ray tracer, simply go to **Render Properties** in the Properties Editor and change the **Render Engine** to **Cycles**:

Figure 1: **Cycles** is Blender's ray tracer. By default, Blender has the render samples for Cycles set to some large number. You may want to change it to a much lower number, e.g. 4, to make the render faster (for now as you play around with it).

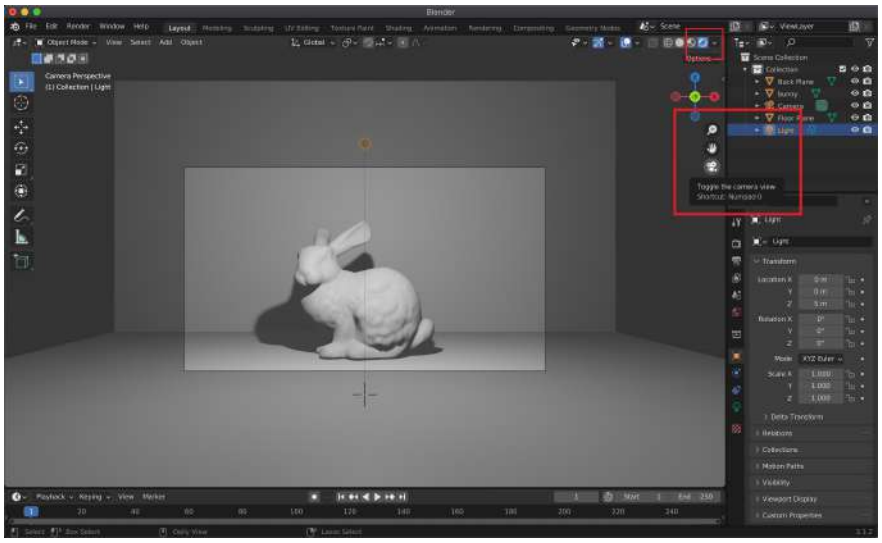


Figure 2: The bigger, red box shows how to toggle to camera view in the Blender Viewport. Above it is the Viewport toolbar, which has the button for toggling the render preview on the far right.

You can preview the render with the right-most button in the Viewport toolbar above the camera view toggle. Note that Blender may start running very slowly if you try to make scene edits while previewing the Cycles (ray traced) render. Ray tracing isn't well suited for real time rendering! Toggle back to wireframe or solid view (the left-most buttons) in the Viewport toolbar for smoother real time editing.

For now, as you experiment within Cycles, you may want to change the default **Max Samples** under the Render settings to a much lower number, e.g. 4 in Figure 1. We'll talk about sampling later in the class, but for now, know that the max samples is basically how many times Blender will repeat the render for better results. Too high of a number will cause your render to take an incredibly long time.

To render your scene as an image, you can go to **Render** near **File**, then click **Render Image**, or press the **F12** hotkey. When the Blender Render window finishes, save the result under **Image** to a directory, e.g. **$CS148_DIR/hw2/imgs/**. The following sections may have you set up various scenes to try rendering with Cycles.

1.2 TODO 1: Render a sphere with both flat and smooth shading.

In lecture, we talked about shading techniques to make geometry appear smoother without actually adding more triangles. More specifically, we talked about Phong shading. In Blender, we can toggle Phong shading for objects in the Blender Viewport and also for rendering with Cycles. Let's take a look at Phong (aka smooth) shading in Blender with an example scene.

1. Make a new Blender scene and replace the default cube with a UV sphere.
2. Add a plane and scale it by 5 in both the x and y directions. Then move it along the z-axis by -1.
3. Change your Render engine to Cycles, and toggle to the render preview.
4. The scene setup should look like the one in Figure 3. Select the UV sphere, then right-click to choose between **Shade Flat** and **Shade Smooth**. You can also find these options under the **Object** menu in Object Mode after selecting the sphere.

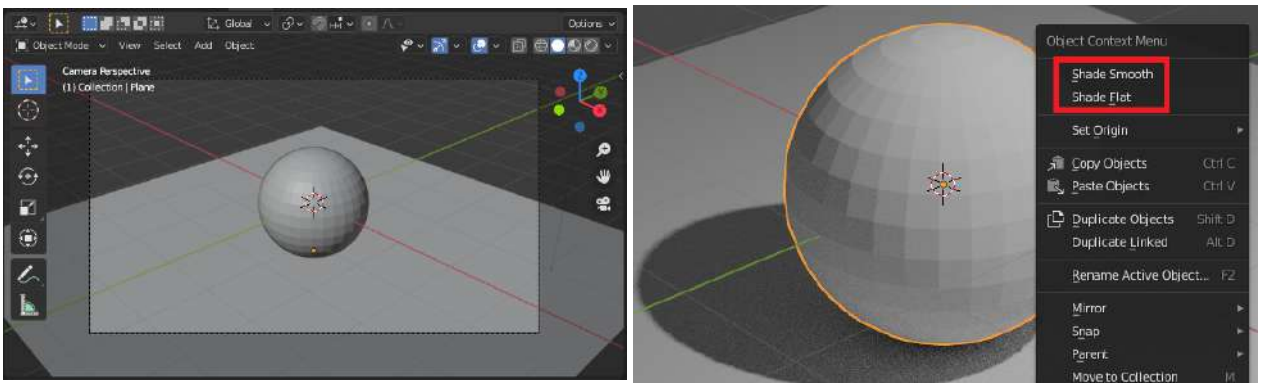


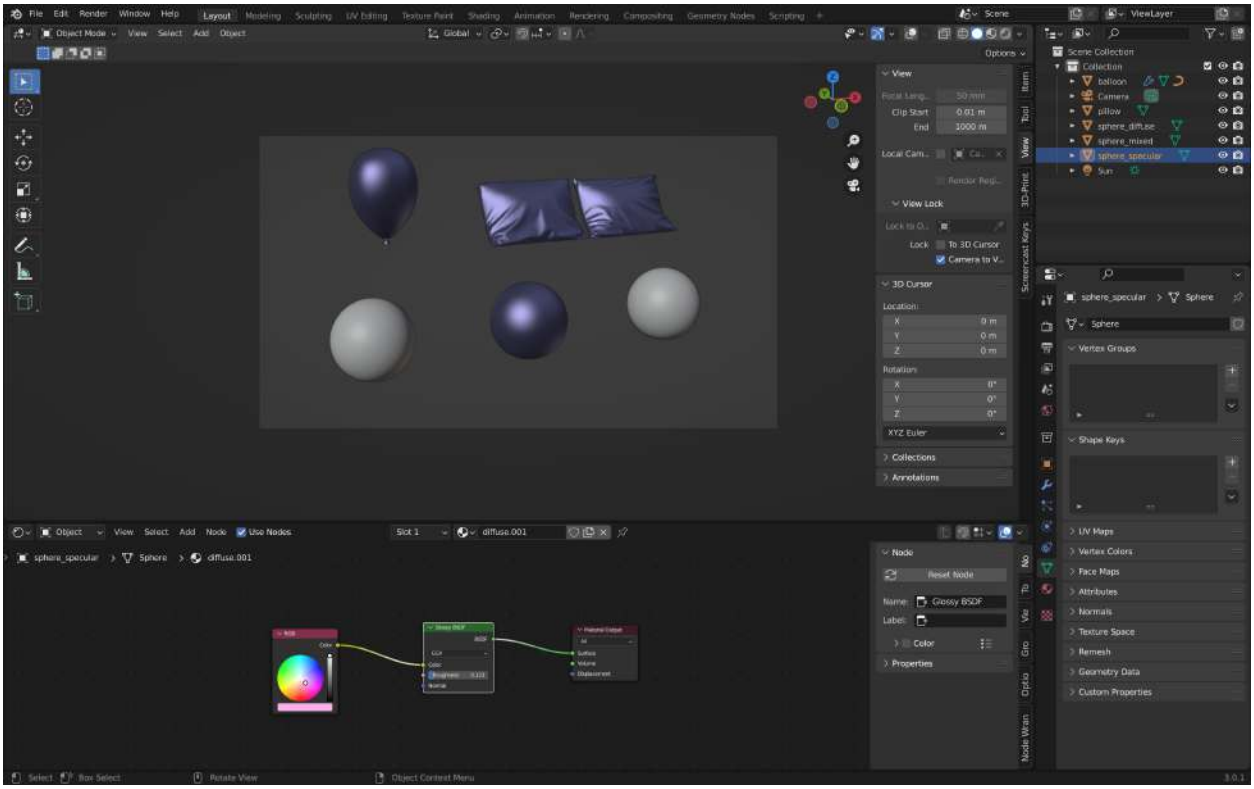
Figure 3: Left: Scene setup for this section. Right: Shading options.

Show us: (1 pt) Two rendered images: one of the sphere rendered with flat shading, and another of the sphere rendered with smooth shading.

1.3 TODO 2: Modify shader nodes for better looking materials.

We'll work with shader nodes here to get a more intuitive idea of the difference between **diffuse** color and **specular** color, as was discussed when we covered the lightning model (aka the Phong reflection model) in class. See the instructional tutorials for more details about shader nodes.

1. Open this example file to see a layout similar to below. The render engine for this file is already set for you to Cycles. Toggle on the render preview to see how the objects look when rendered with their respective shader nodes.



2. Think about how a balloon look likes in real life. Click on the balloon object to see its shader node graph below. Play around and edit any of the diffuse, specular (aka "glossy"), and/or mix shader nodes until the balloon looks like a real life balloon when rendered in Cycles.

3. Similarly, think about how pillows with fabric covers look like in real life. Clock on the pillow objects to see their shade node graphs below. Edit their shader nodes until the pillow covers look like fabric.

4. When you're happy with how your balloon and pillows look, render the scene as an image and save it for grading. Also save this .blend file under a different name, since you may want to edit it in the next TODO.

Show us: (1 pt) A rendered image of the balloon and the pillows having realistic looking materials. Be ready to explain what you did.

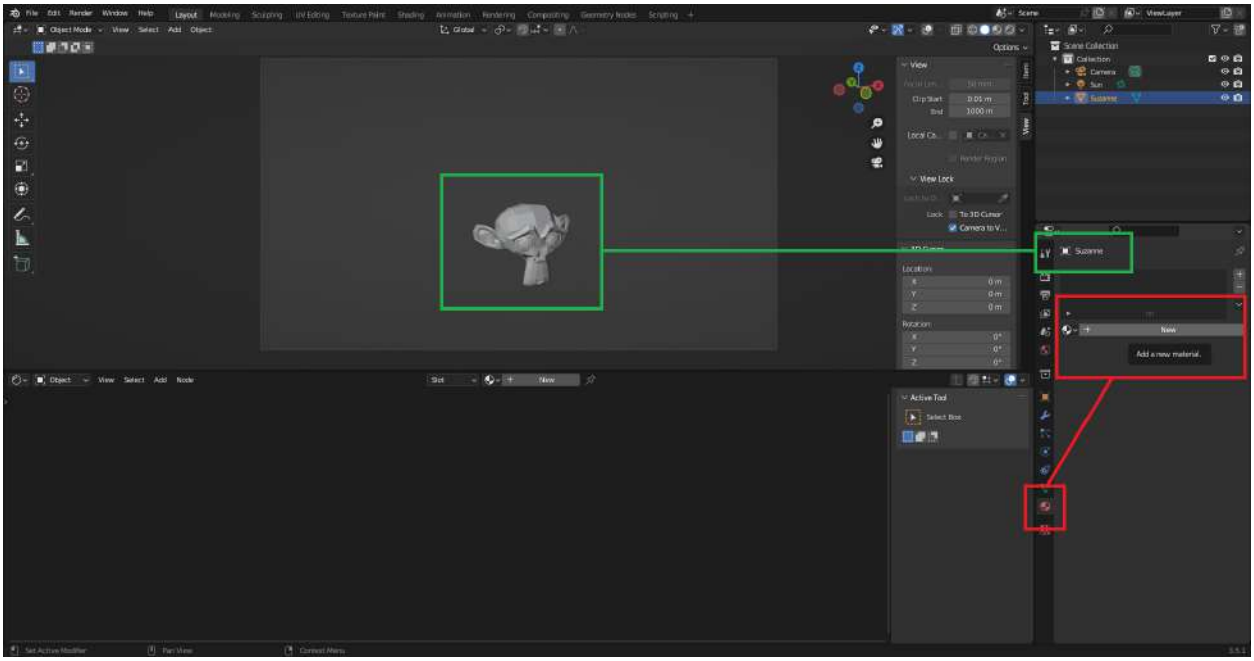
1.4 TODO 3: Create a material of your own using shader nodes.

For this section, we want you to take a photo or find a reference image of an object that you might want to include in your final project. Then, either model a rough start to that object yourself or find a similar model of that object online, and create a material for it that looks close to the real material in the reference photo or image.

1. You'll probably find it convenient to work with the .blend file from TODO 2, since it already has a nice window layout to see both the viewport and the node editor. Delete all the objects in the file by selecting them all and pressing Delete or x on your keyboard. Alternatively, you can make a new Blender file from scratch if you prefer and set up the window layout yourself – just remember to turn the render engine to Cycles as well.

2. Create or import the object of your choosing into your Blender file. Then, go to its Material Properties panel indicated by the small red box below and add a new material to start editing. After adding a material, you should see a shader node graph in the shader editor.

   Some objects might already have their own default materials if you downloaded them from online or are using some of Blender's default objects as a base. You can use the existing materials as places to start, but ultimately, **we want you end with a material that you've sufficiently edited enough to call your own.**



3. By default, Blender assigns objects the Principled BDSF shader. Try swapping this shader to other shaders such as the Diffuse BSDF and Glossy BSDF, and try adding more nodes such as the Mix Shader. You can even try the more exotic shader nodes like those for glass and hair! Experiment as much as you want until you're satisfied with how your objects look when rendered with your materials in Cycles!

   If you're in need of inspiration, then try these tutorials for making a cork material and another for making a lava material! When in doubt, look towards the internet for tutorials if you have something specific in mind!

Show us: (1 pt) Your reference photo or image, and a rendered image showing your object rendered in Cycles with your custom material.

1.5 TODO 4: Render a scene in your own unique camera view.

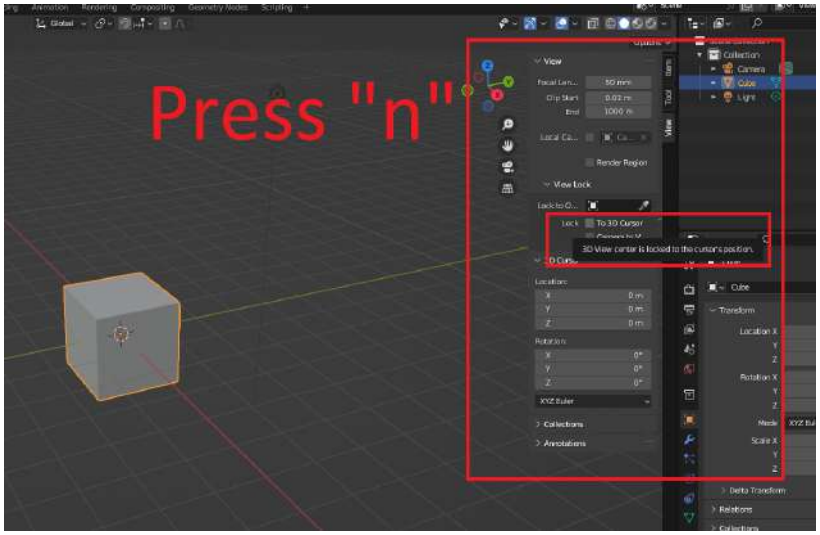
For this section, we want you to once again take a photo or find a reference image, but this time, it'll be of a scene layout that you might want to consider for your final project. You can also sketch your desired scene layout on paper if you want. In any case, we want you to have ready an image of a bunch of objects positioned and oriented in an interesting way in front of a camera. Cameras are not only technical tools, but also artistic ones. Where the camera is placed relative to the objects and environment can say a lot about a scene.

1. Create your scene to explore! Make sure this scene has a camera that you can move around, a light, and materials on every object. Blender scenes by default come with a camera and light if you decide to make a new .blend file from scratch. Some suggested options:

   (a) The ideal use of this TODO would be to set up a few objects for your final project scene. These objects don't have to be final or even look that good, but simply the practice of placing them and figuring out an appropriate camera angle for the scene can save a lot of time later!

   (b) This demo website from Blender has a lot of example scenes that you can browse from if you're looking for inspiration! Warning: because these are completed, more complex scenes, it might take time for your computer to render them properly in Cycles.

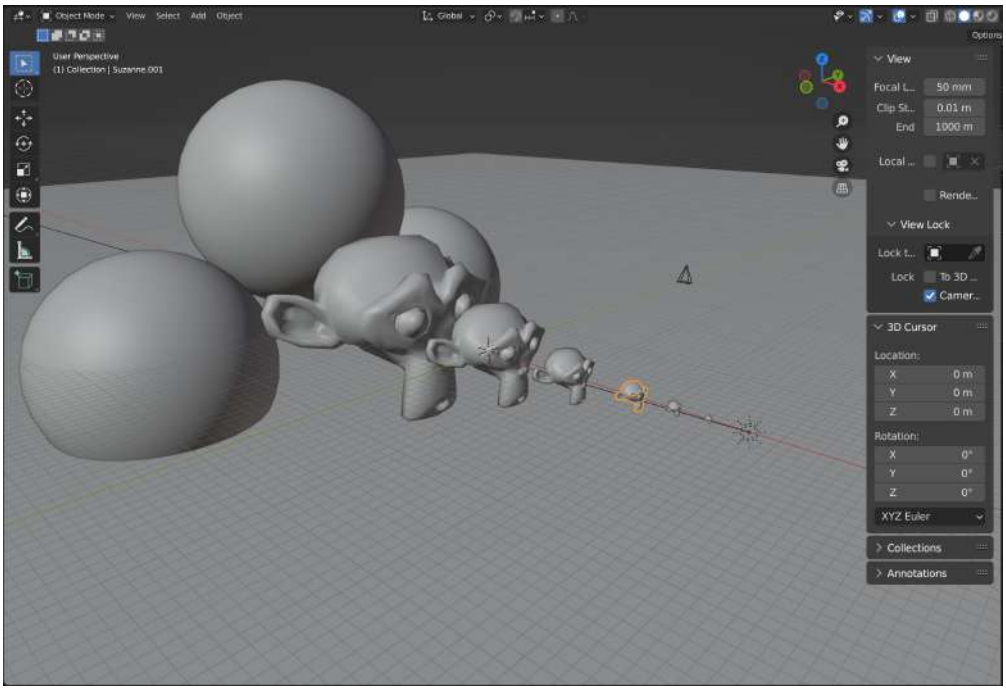
2. Lock the camera to our view so that when we're panning around and zooming in and out of the scene **in camera view**, our camera will move with us. To do this, open the transformation sidebar (View → Sidebar , or press N). Note that you can also change the camera's translation and rotation from here! Check the box that says Camera to View under the View tab to lock the camera.

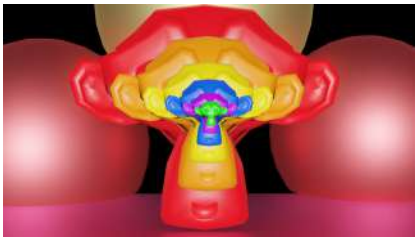


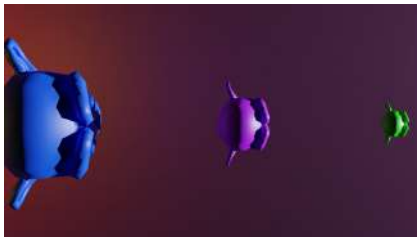
Now if you toggle to camera view and move your scene around in the viewport, then toggle back to the normal view, then you should notice that your camera has moved to accommodate for the movements you did!

3. Move your scene around in camera view until you find an interesting view (in order words, until you find a suitable place for your camera). Render the image and save it!

As an example, here are three viewpoints that we found interesting using the scene below. This part is very open-ended – get creative!







Show us: (1 pt) Your reference photo, image, or sketch of your scene, and a rendered image showing your attempt at recreating the camera angle and layout for the scene.

That covers all the TODOs! Everything else in this document consists of instructional (Blender) tutorials for you to reference.

2 Importing 3D Models

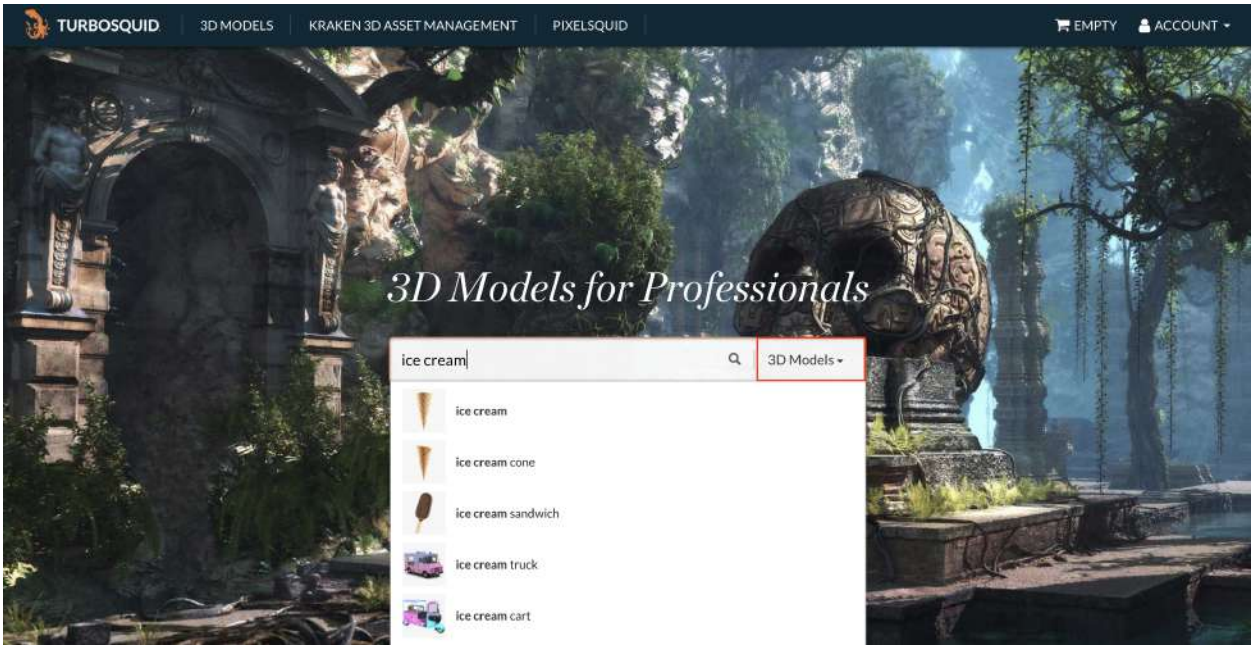
In this section, you will be given some pointers on how to start exploring the breadth of resources online and find models for your own scene. First, here is a list of popular websites for free resources:

- General Objects: Poliigon, TurboSquid, cgtrader, 3dsky, Dimensiva, Poly Haven
- Blender Assets: Blend Swap
- Unique Scanned Objects: Sketchfab CC0
- Landscapes: Quixel Megascans
- 3D Map Models: Map Models Importer

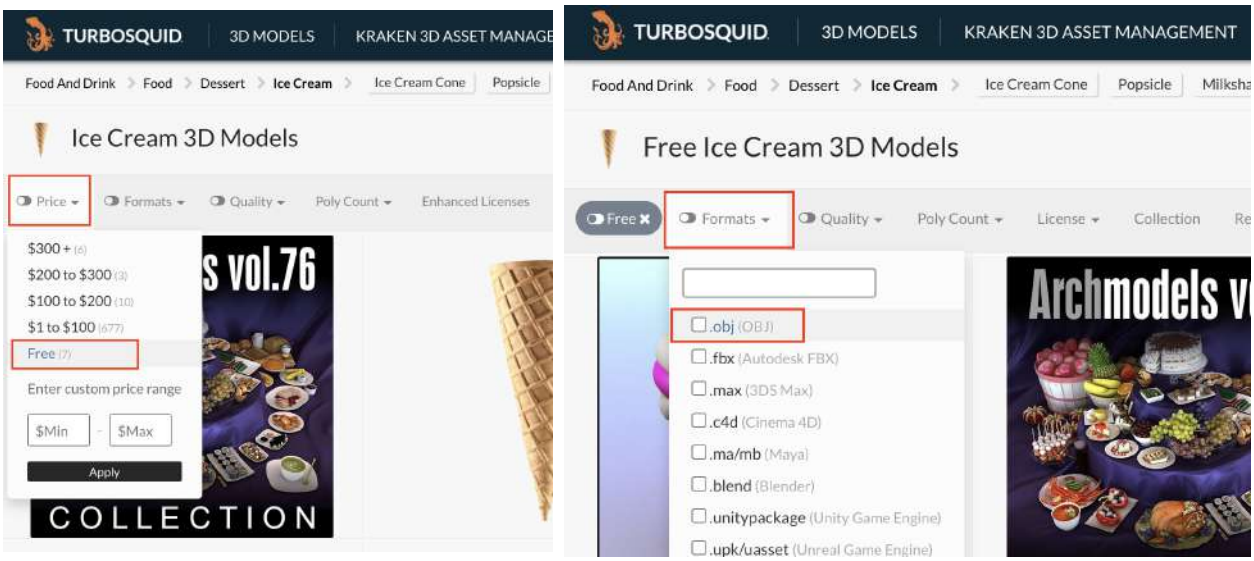
2.1 Finding a 3D Model

The steps for downloading a model depends on the website that you are using. Generally, you want to look for free mesh objects in the `.blend` or `.obj` file formats that we've worked with in previous assignments. Here is a step-by-step example walkthrough of how to obtain an object from TurboSquid. You will need to register an account beforehand, but afterward, you will be able to find a wide variety of free models to download.

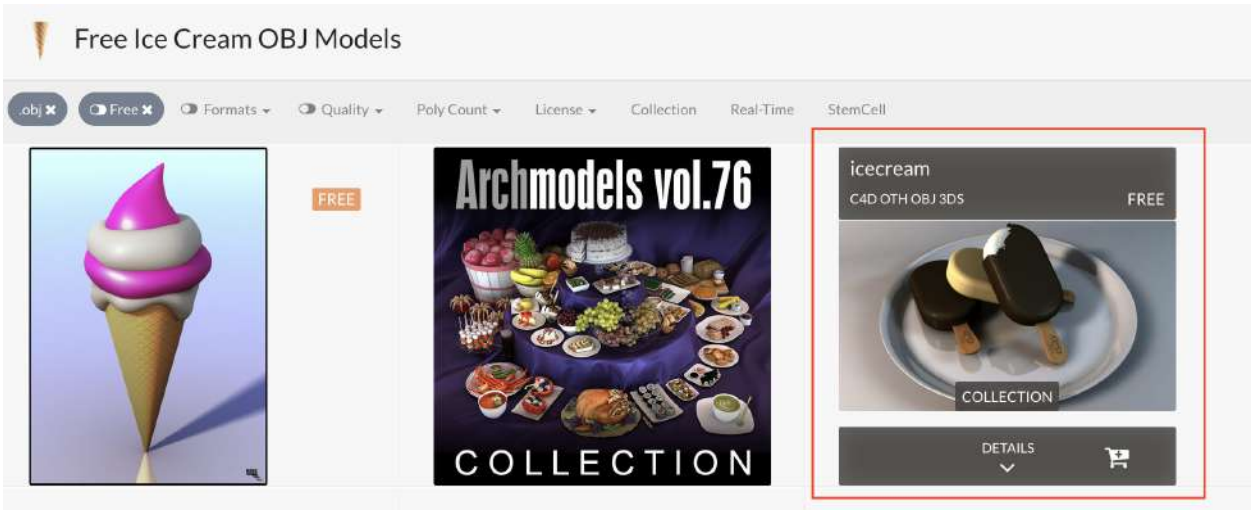
1. Search for a model:



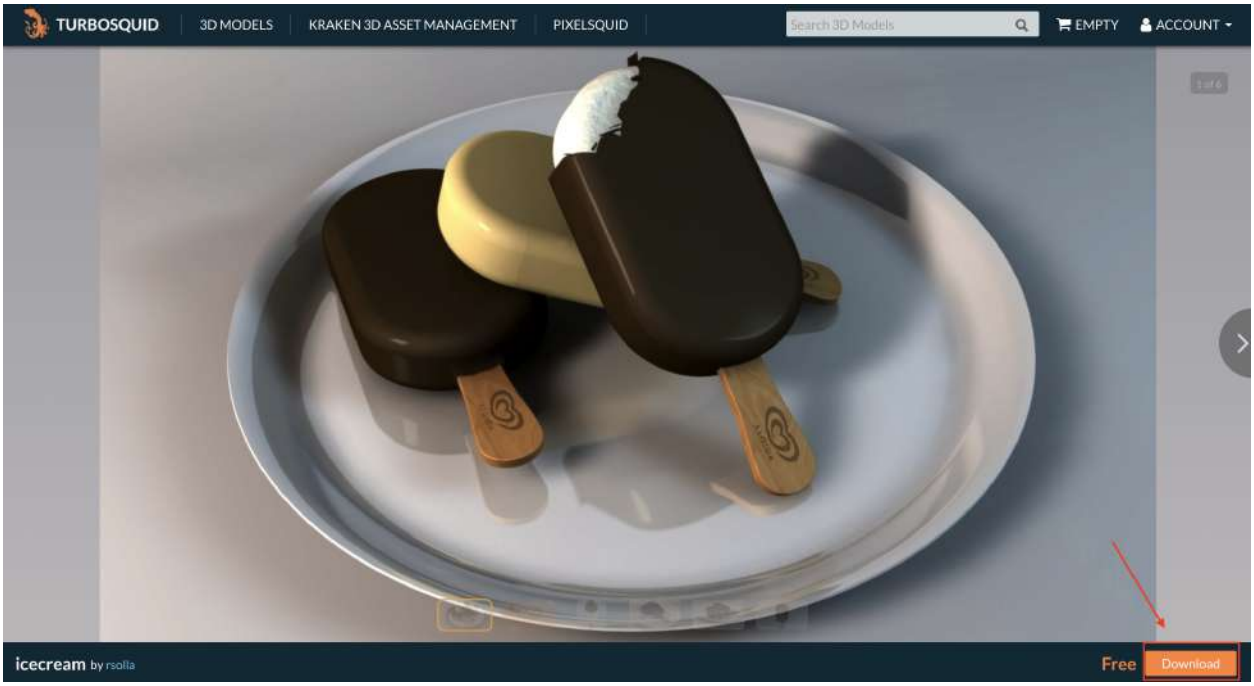
2. Filter for free models in the `.obj` or `.blend` file formats:



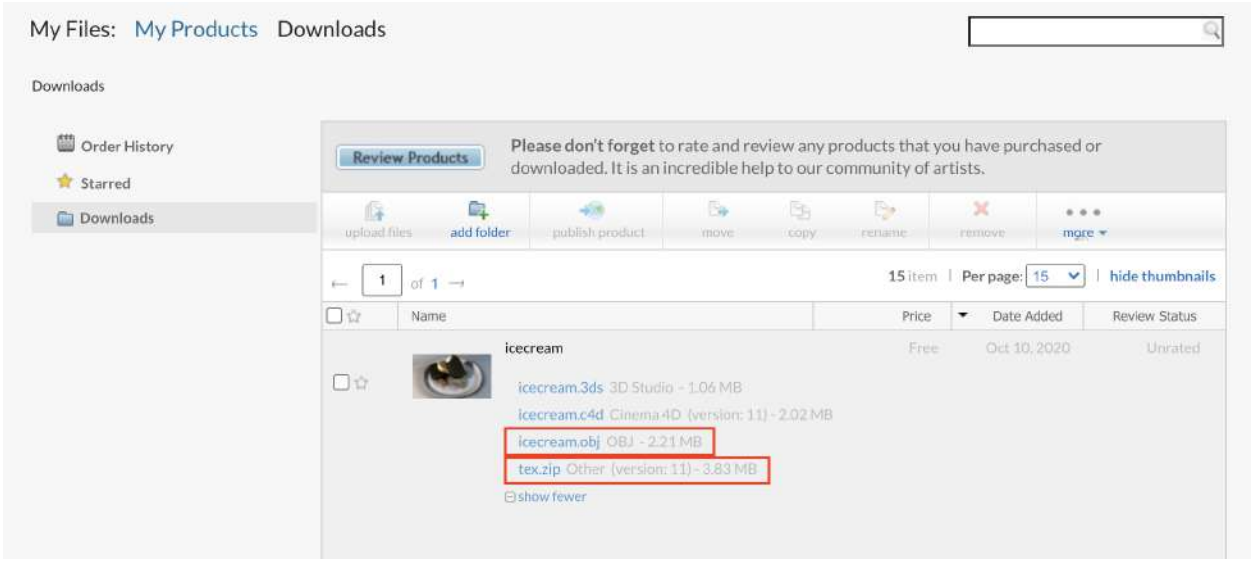
3. Choose which model you want to use and open its product page:



4. Download the model to your TurboSquid account. Note that this does not download it onto your computer yet. When you click "Download," it will redirect you to your TurboSquid Downloads asset manager.



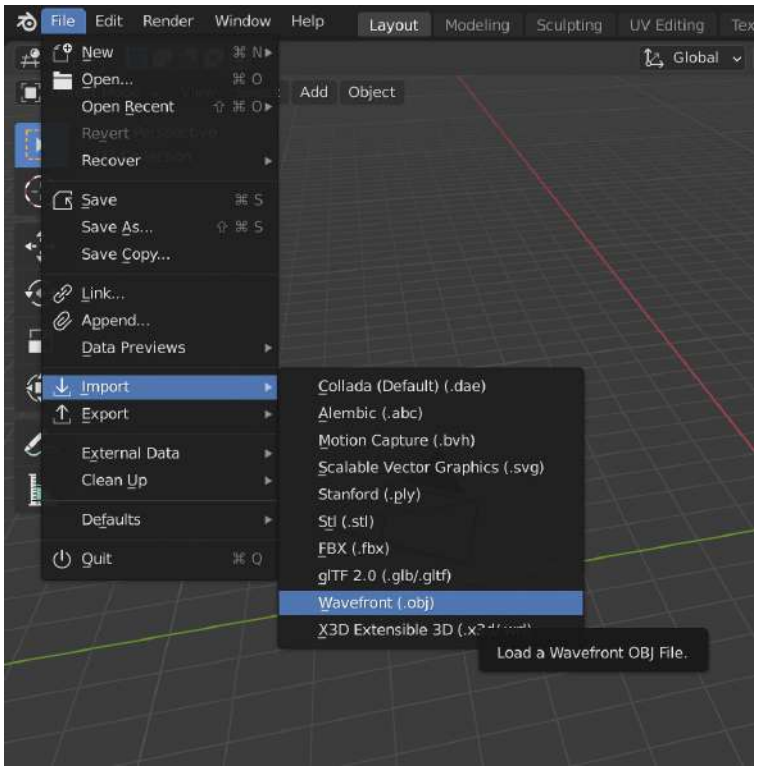
5. From your TurboSquid Downloads, save the downloaded `.obj` file by clicking on the file name (e.g. `icecream.obj`). Some models also come with texture and shading materials that you can download (e.g. `tex.zip`). We will take a closer look at these files in a later homework, but you may want to download and take a look at the materials now anyway.



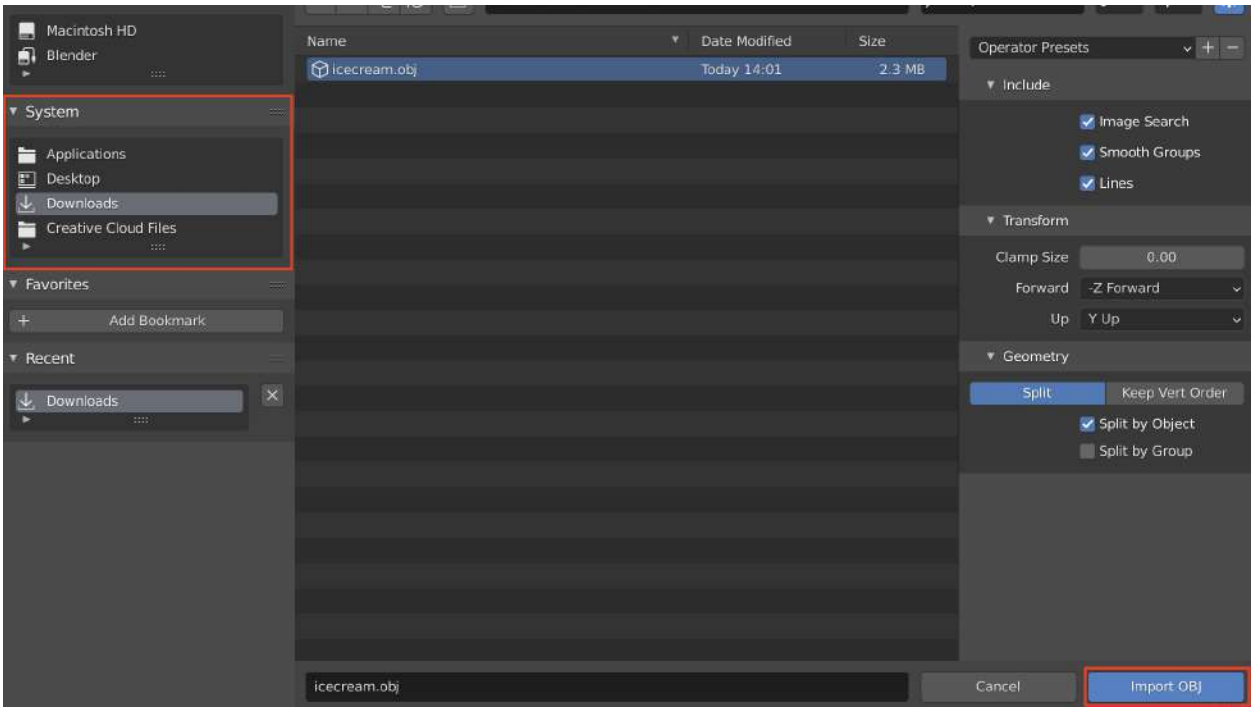
2.2 Importing a 3D Model into Blender

If you've forgotten how to import `.obj` files, then take a look back at HW1 on "The OBJ file format". Below is a recap of the steps for importing the icecream object that we found in the previous section into a Blender scene.

1. Start the process from the `File` dropdown menu:

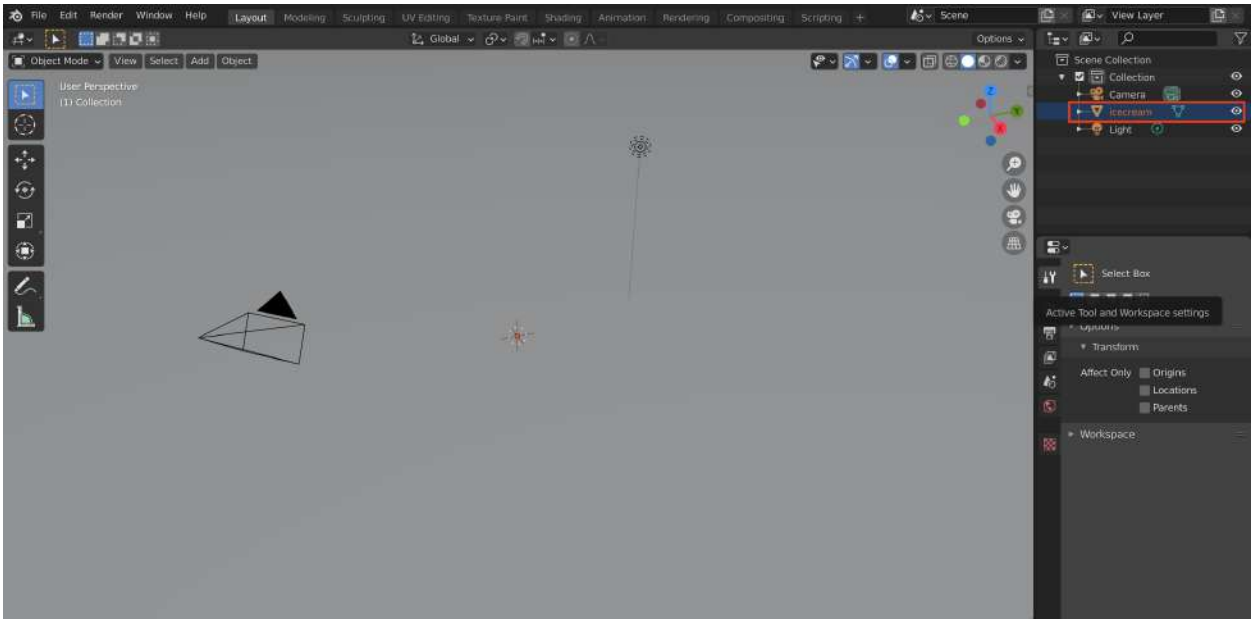


2. Navigate to the folder where you saved the model and select the **.obj** file. You can leave the import settings as default.

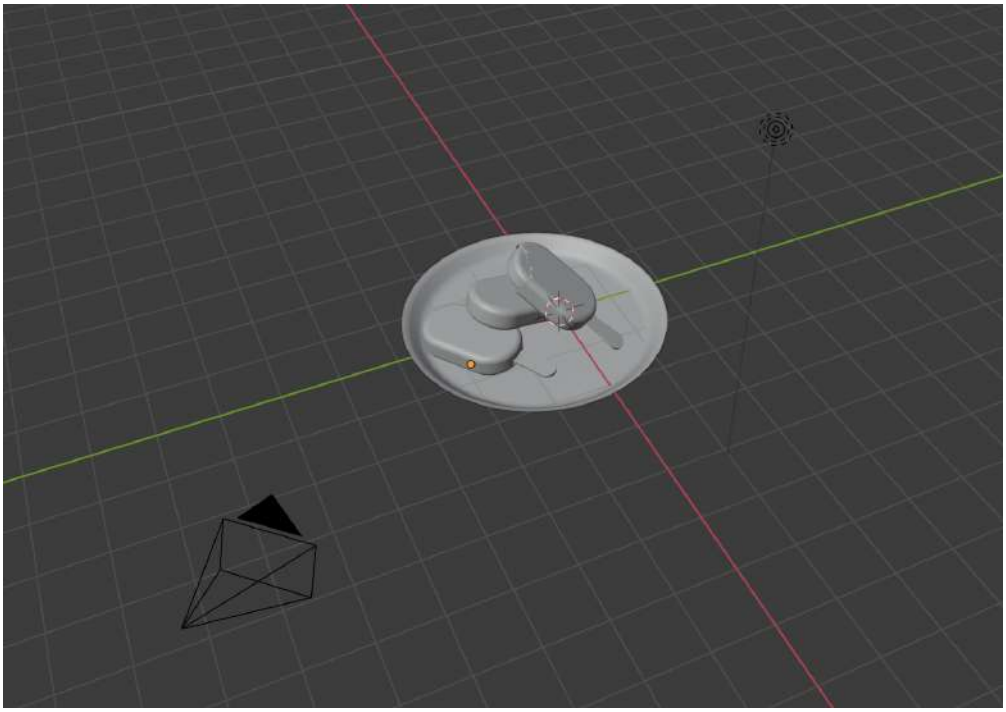


3. You should now see your imported object in the 3D Viewport, and the name of the object should appear in the **Scene Collection**. If the object is making the entire 3D Viewport gray,

then it is likely too big for the scene. You will need to scale it down.



4. From here, you can manipulate the object you imported as you did for objects in previous assignments. You can translate, rotate, or scale the object. You can also sculpt or model on top of the object. Note that a lot of objects will be automatically rotated 90 degrees upon import due to a compatibility fix between Blender and other modeling software.



3 Exporting 3D Models

You might have made your own custom object last HW or this HW and want to transfer it from one Blender file to another. There are two recommended ways to do so:

- The first and most ideal option believe it or not is to simply copy and paste your object from one .blend file to another. Simply make sure all parts of the object you want to transfer are selected in Object Mode, then press your OS's appropriate keyboard shortcuts for copy (e.g. **CTRL + C** on Windows). From there, go to the viewport of your other .blend file, and press the keyboard shortcut for paste (e.g. **CTRL + P** on Windows). Your entire object along with any materials, painting, etc you've done to it will have transfered over to the new .blend file!
- The second option is to export your object as its own .obj file. This is as simple as going to **File** in the top-left corner of the GUI, then **Export → Wavefront (.obj)** and saving it where you want it on your computer. Remember though that .obj files only store geometry. They do not store information regarding e.g. shader nodes. And so if you did any custom shader options to your object, then you're better off using the first option of copy and paste.

4 Shader Nodes

In lecture, we talked about the lighting model, aka the Phong reflection model, which had the terms c_a, c_d, and c_s to represent the ambient, diffuse, and specular material properties of an object respectively. These terms represented:

- How our object will (faintly) appear in the absence of light in the case of the ambient term.
- How our object will appear based on the angle at which the light is hitting it in the case of the diffuse term.
- How our object will appear based on the angle it reflects light back to our eyes in the case of the specular term.

We can construct what we call **shader nodes** to set parameters such as the diffuse and specular material properties of our objects. To see how we do this, open the file from TODO 2 and swap to the render preview. From there, click back and forth between the left-most shiny (specular) sphere and the right-most matte (diffuse) sphere. Notice how the two spheres have different flowcharts in the bottom most panel (called the **Shader Editor**) as shown in Figure 4.

These flowcharts consist of what are called nodes. Each node can have input(s) and output(s) that can be connected to create a graph. For example, consider the graph for the specular sphere in Figure 4. Let's break down what each node is doing, starting from the left and going right.

- The left-most node is the **RGB** node that simply generates a color. You can move the pointer in the color wheel to change the color. This node has no input because it serves as an input to other nodes. Its output is the color as chosen from the color wheel.
- The middle node is a **Glossy BSDF** node that gives us a way to control and fine-tune the specular c_s parameter of our object. Notice how it takes as input the color from our **RGB** node. You can think of this color as kind of like the base or ambient color of our object. When white light shines onto our object, this is the base color that we want the object to return. In this case, we decided to make the sphere purple.

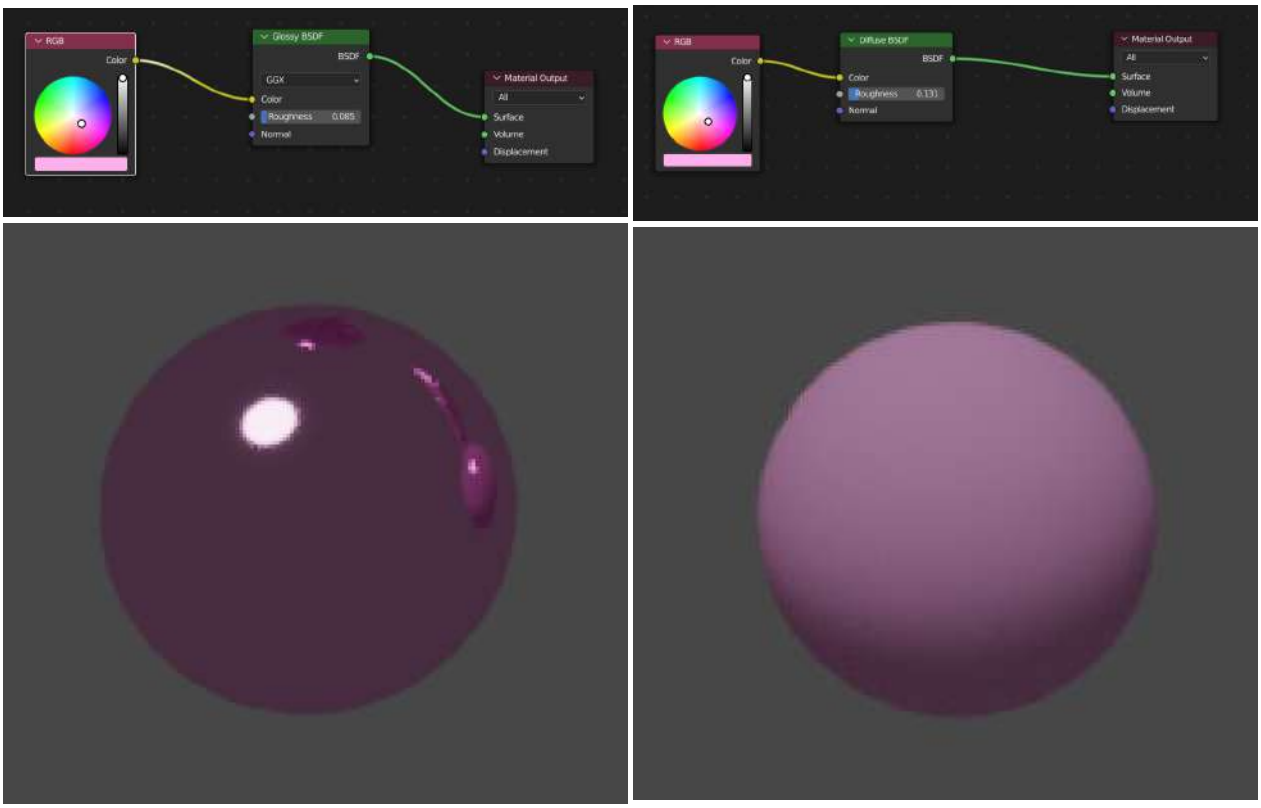


Figure 4: (Left) A sphere with a shader node graph to give it a pure (shiny) specular material. (Right) A sphere with a shader node graph to give it a pure (matte) diffuse material. The graphs are above, and the material results when rendered in Cycles are below respectively.

You might then notice a slider bar labeled **Roughness**. This lets us control how specular we want our object to be – i.e. how well does it reflect light back to our eyes. As you move the slider left to right, you might notice the specular highlight, the white bright spot on the sphere, get bigger and smaller. This might remind you of how the Phong exponent α affected the size of specular highlights from lecture.

All of this information regarding the specular properties of our object is encoded into the output, which is fed into another node on the right.

- The right-most node is what anchors all our previous nodes to our object. Notice how the **Glossy BSDF** node connects to the **Surface** input of our last node. That means we want all the information from the **Glossy BSDF** node to affect the surface of our object, thus all the specular information that we fine-tuned is applied to the sphere's surface to make it shiny.

And a similar idea applies to the shader node graph for the diffuse sphere. Play around with the slider for the **Diffuse BSDF** too and see what happens.

You might be wondering at this point what "BSDF" stands for – we'll cover that next week don't worry. For now, you can think of these BSDF nodes as ways for us to fine-tune those diffuse and specular and other such material parameters for our objects when we're making them.

Now click on the sphere in the middle of the example file. You might notice that this sphere's shader node graph is a bit more complex. This one has both a node for the specular and diffuse material properties, plus a new node called the **Mix Shader**. As you might guess, this Mix Shader node "mixes" the results of both the specular and diffuse nodes to output one color for the surface of our object. We've set it to 0.5 for 50% in the figure below, essentially giving both diffuse and specular equal weight.

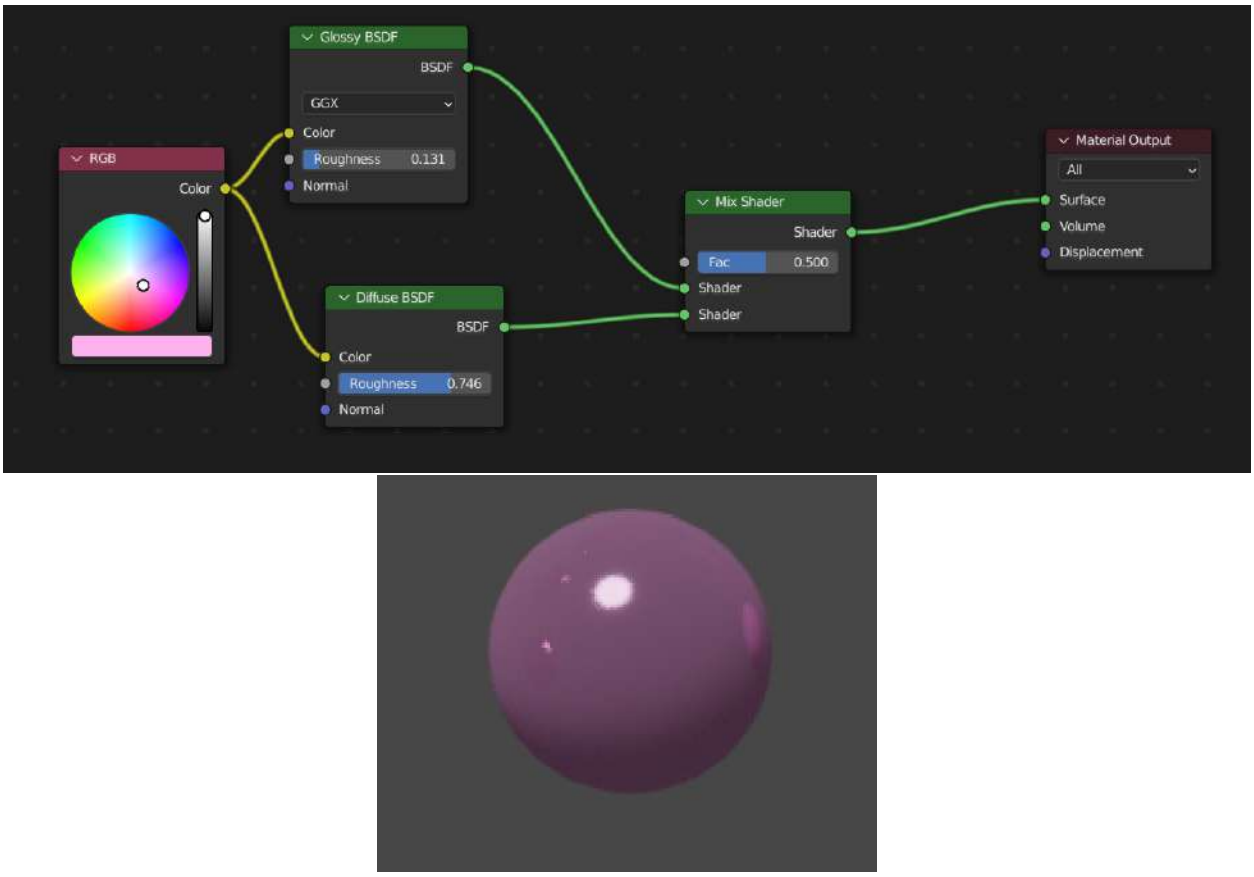


Figure 5: A shader node graph that mixes both a Glossy and a Diffuse BSDF node to give our sphere both specular and diffuse material properties respectively.

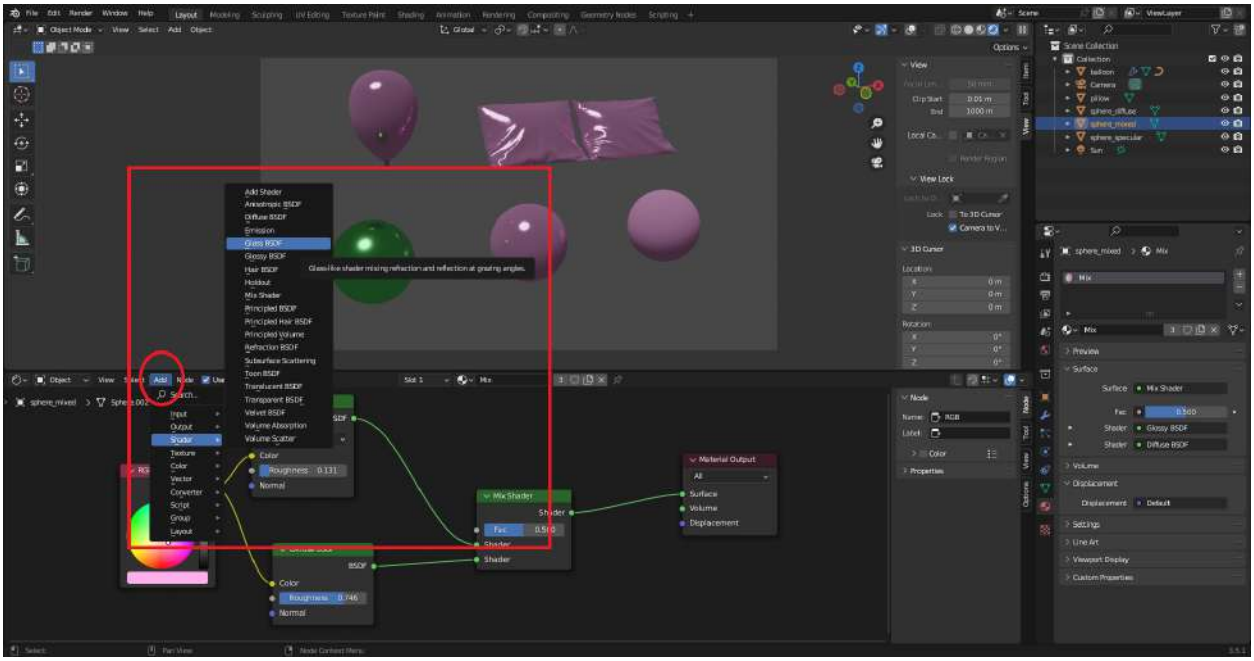
You can think of this final shader node graph in the context of our lighting model like this:

- The **RGB** node essentially represents our ambient c_a parameter, giving the sphere a base color.
- The **Diffuse BSDF** node fine-tunes our our diffuse c_d parameter to determine how our sphere will appear when interacting with light at an angle.
- The **Glossy BSDF** node fine-tunes our specular c_s and α parameters to determine how our sphere appears when reflecting light at an angle to our eyes.

Essentially, the entire material properties part of the lighting model is abstracted away into a simple flowchart GUI that you can casually edit in Blender!

4.1 Other Shader Nodes

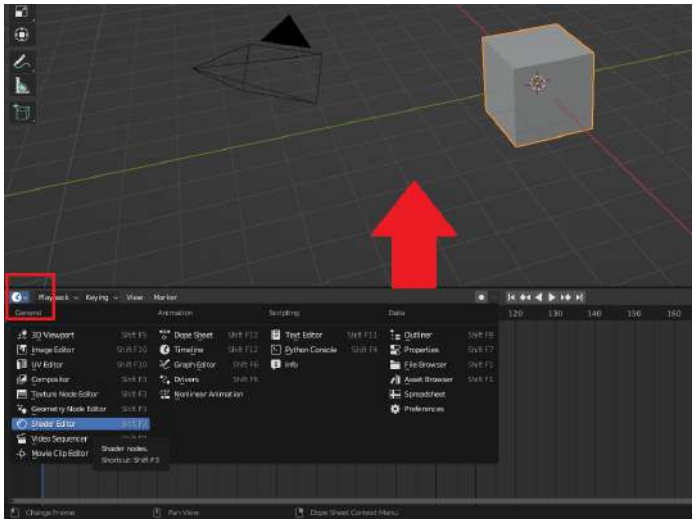
You can add other shader nodes such as the "Glass BSDF", which mimics glass material properties, via the **Add** menu in the **Shader Editor**. Again, don't worry about what "BSDF" means – we'll cover that next week. You might also wonder what the other types of nodes like "Texture" are – we'll also cover those later in the course.



4.2 Adding New Materials

Suppose you want to add materials to your objects from scratch instead of using our example file.

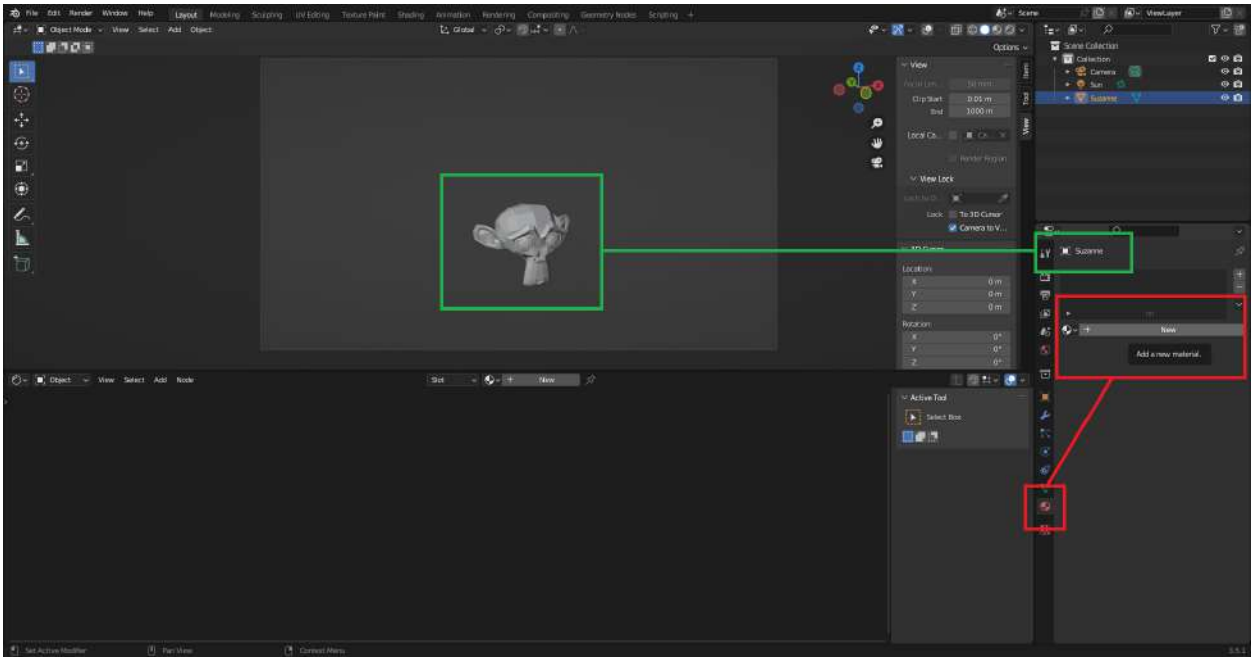
1. First, you want to view the **Shader Editor**. Click on the edge of the **Timeline** as we call it at the bottom of your Blender viewport and drag it up with your mouse. From there, click on the icon that looks like a mini clock near the bottom left and select **Shader Editor** to change the bottom-most window panel to the Shader Editor.



Depending on your object, you might already see a shader node graph in the Shader Editor by default. Some of the Blender built-in objects like the default cube already come with the Principled BDSF shader. You can swap this out for other shader nodes, or try playing with its many parameters. Play around as much as you want!

2. Some other Blender built-in objects don't have default materials though. For instance, the default monkey object does not come with its own shader node graph.

   To give it one, go to its **Material Properties** panel indicated by the small red box below and add a new material. After adding a material, you should see a shader node graph in the shader editor. By default, the graph uses the Principled BDSF shader.



3. You can expand the Preview panel to see how the material looks on template models.

For those wondering – the default material is the Principled BSDF, a shader based on the Disney principled model. It can emulate a wide variety of materials in our daily life. You can learn about the parameters and view some examples for the Principled BSDF in the Blender documentation as well as other BSDFs here.

We will discuss how to use the **Shader Editor** to make even more complex materials (like textures!) later in the course. If you'd like to try it out now, then you can watch the official tutorials here and here, or Blender Guru's explanation here.