CS148 Homework 4 - Lighting & Shading

Grading on Monday, Oct 20 – see the pinned posts on Ed for details!

0.1 Assignment Outline

This assignment has separate TODOs of varying lengths. The TODOs are covered within the first few pages of the document. The rest of the handout consists of instructional materials.

As a break from last week's coding heavy assignment, this assignment will be all done in the Blender GUI. Some parts are quite open ended on what you can submit for credit. Both the process of creating materials for objects and the process of camera placement and lighting for a scene lead to a lot of artistic freedom! We encourage you to get creative with what you make!

0.2 Collaboration Policy, Office Hours, and Grading Session

These policies are the same as they were in HW2. See the HW2 document for details.

0.3 AI Policy

You may use AI tools to generate HDRI images or geometry to place in your scenes for this HW. However, you may not use them to generate anything else, especially your rendered images. All rendered images must be the final ray traced result from Blender Cycles.

Quiz Questions (1 pt)

You will be randomly asked one of these questions during the grading session:

- How does the distance between a light and an object as well as the tilt angle between the two affect the irradiance on the surface of the object? How does the concept of a tilt angle also come up for the radiance of (area) lights?
- What causes color bleeding in real life? Give a high level description of what we would need to do with our objects in a scene to model color bleeding
- Conceptually, what is a BRDF? How are BRDFs involved in the lighting equation? For instance, if the incoming radiance is white light, and the BRDF involved models a blue material object, what might you expect for the color of the outgoing radiance?
- What is the difference between the way we compute Gouraud shading vs the way we compute Phong shading? Which one of these shading techniques produces a more realistic looking effect, and at what cost?
- Explain the ambient, diffuse and specular components of the Phong Reflection Model. What does each component model visually for the shading of an object?

1 Assignment

1.1 Blender Cycles

This week, we will explore lighting and shading using Blender's built-in, ray tracing render engine: Cycles. Cycles implements all that you coded up in the last assignment and much more (including color bleeding). It provides most of the features and power of modern ray tracing under the hood, allowing the average user to utilize it all without needing any programming experience.

Turning on the Blender ray tracer is incredibly simple. Whenever this HW prompts you to turn on Cycles, 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).

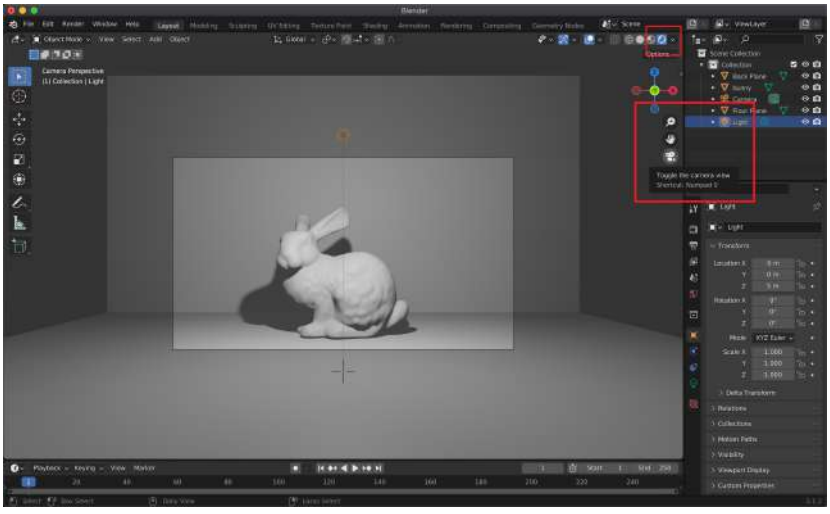


Figure 2: The bigger, red box shows how to toggle to camera view in the Blender Viewport. Above it is the Viewport toolbar, which has the button for toggling the render preview on the far right.

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/hw4/imgs/**. The following sections will 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 by clicking it, 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.

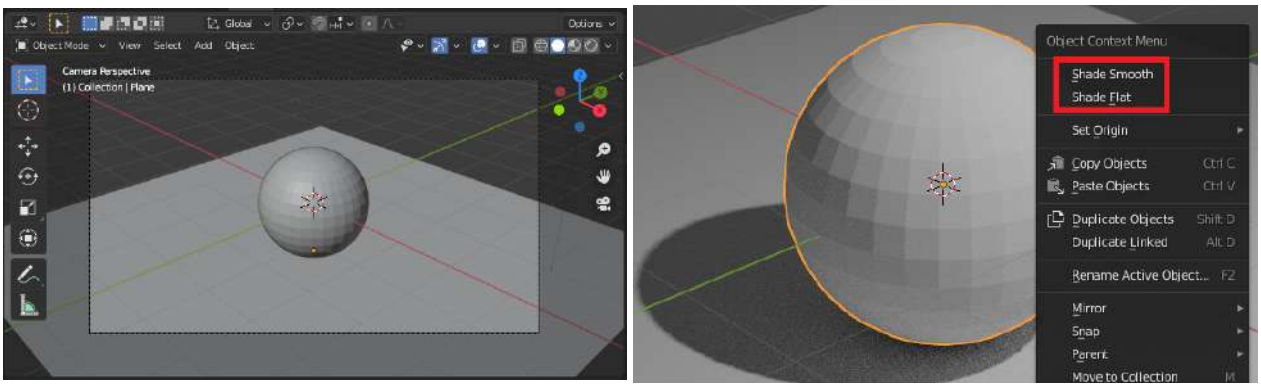


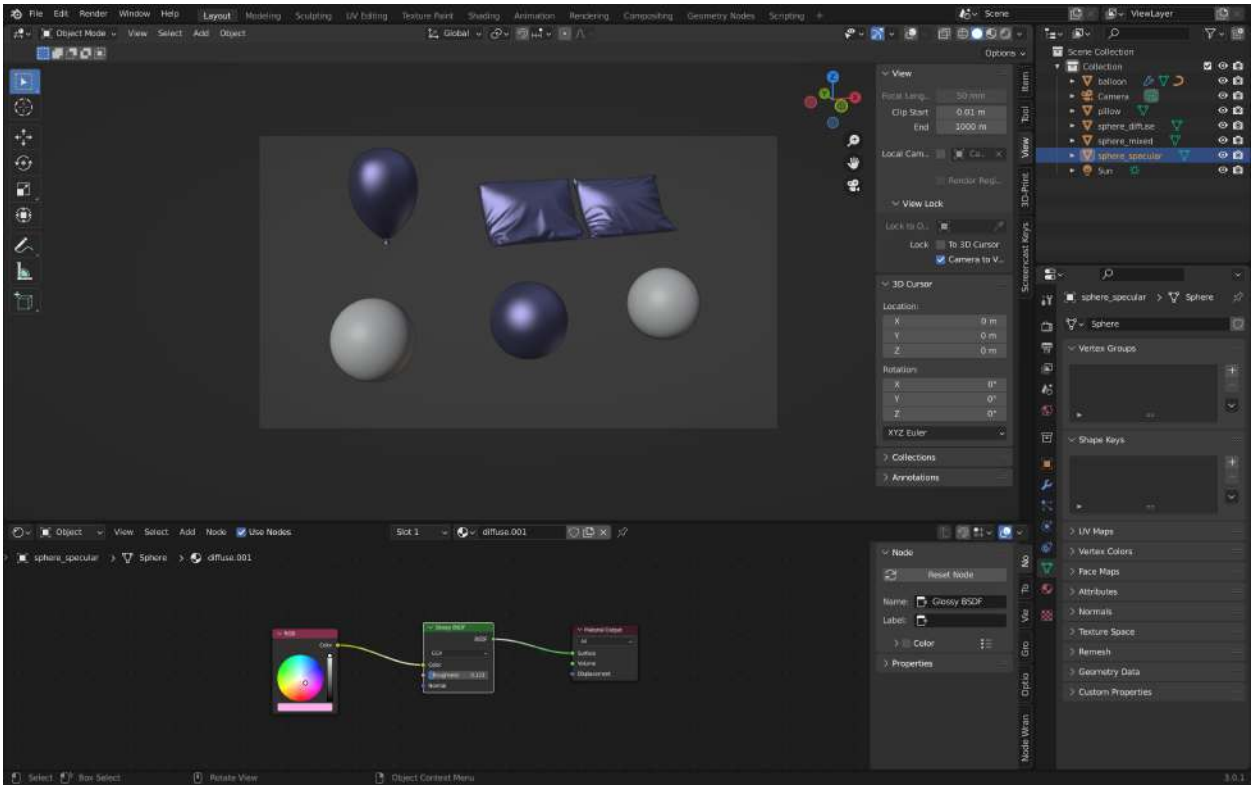
Figure 3: Left: Scene setup for this section. Right: Shading options.

Show us: (0.5 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 materials 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. Click 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: (0.5 pt) A rendered image of the balloon and the pillows having realistic looking materials.

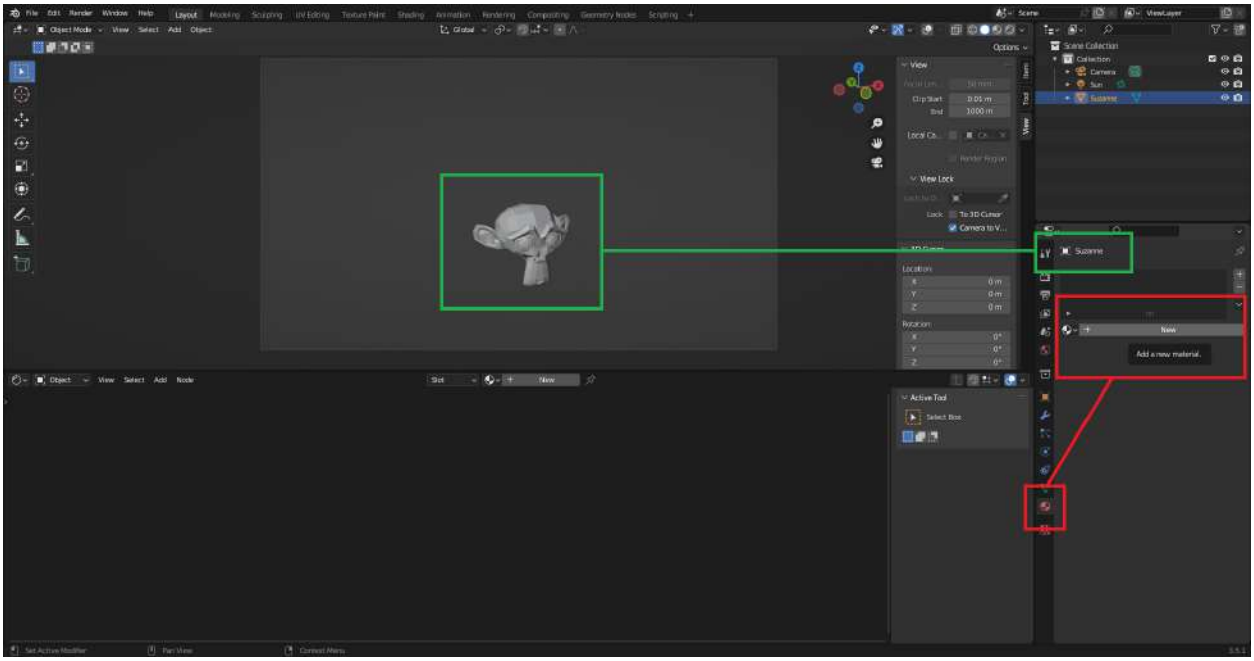
1.4 TODO 3: Create a material of your own using shader nodes.

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.

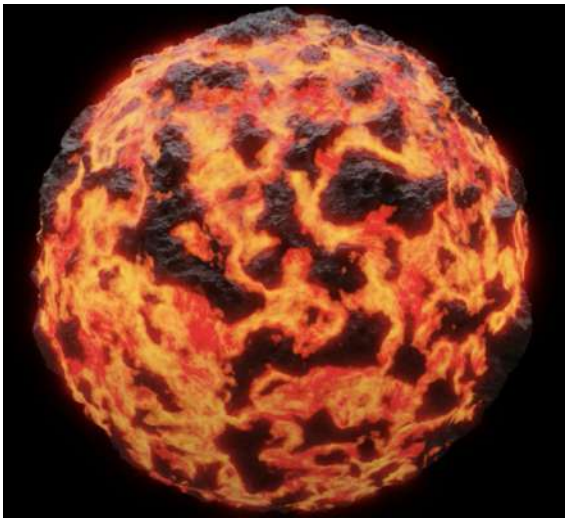
   Some objects might already have their own default materials if you downloaded them from online or are using some of Blender's default objects as a base. You can use the existing materials as places to start, but ultimately, **we want you end with a material that you've sufficiently edited enough to call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

   Note that for the Glass BSDF in particular, since it is supposed to be see through, you likely won't get any interesting results unless you have something behind the glass object to transmit. It's best to combine glass with a HDRI or Nishita Sky later in this HW.

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 2 scenes with (1) soft shadows and (2) dramatic lighting.

Blender by default provides 4 different types of lights for you to place into your scene: point, area, spot, and directional lights. We covered two of these in lecture (point and area), and the other two are pretty intuitive to pick up.

There are two ways to add any of these lights to your scene. One option is to add a new **Light** from the **Add** menu. The other is to modify an existing light in your scene in the Properties Editor under **Object Data Properties**. There, you can change your light between all four types.

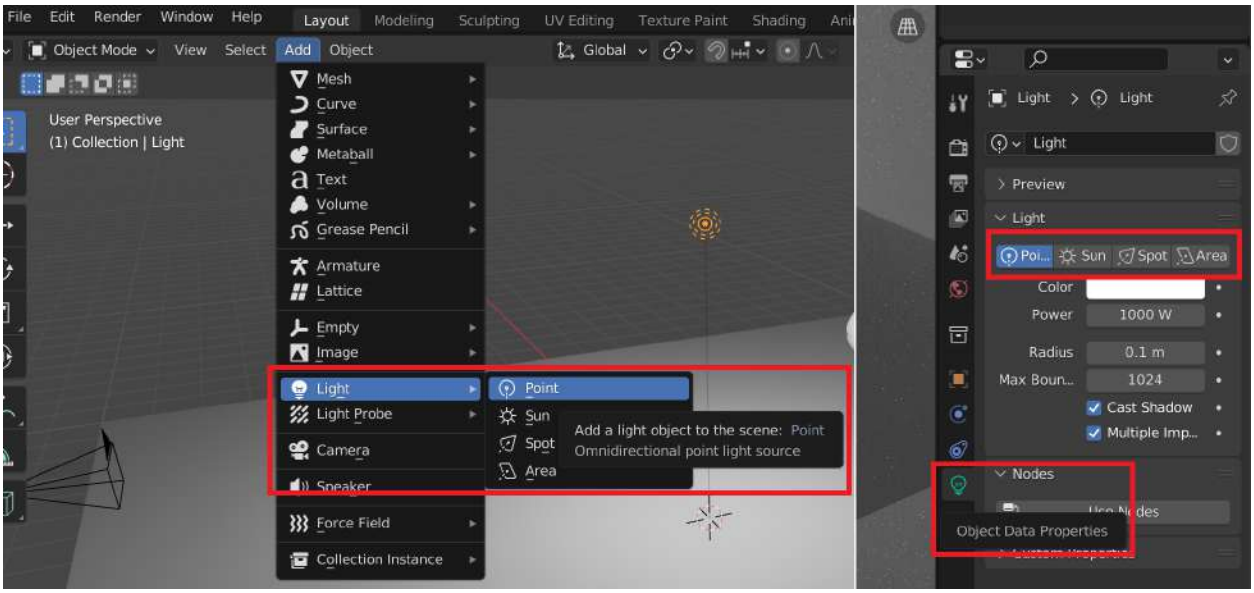


Figure 4: Ways to add a particular light type to your scene.

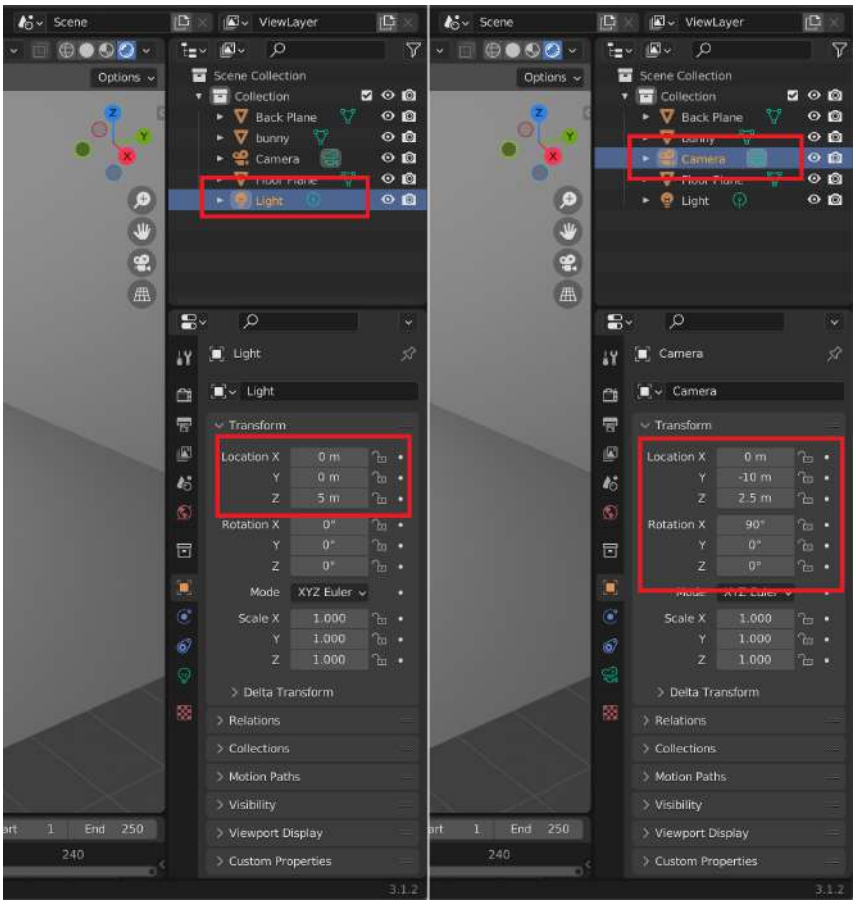


Figure 5: Transforming a light or camera using the Properties Editor. Note that in the case of a point light, only translations matter. Scaling has no effect on either.

Try using the different types of lights to light up your own scene. Use this as an opportunity to get a feel of how each light might come up for what you want to do in your final project. From Figure 6, we see that using a mix of lights can lead to a more natural look with softer shadows and more even illumination.

We want you to build two scenes, each with its own different lighting arrangements. For the first scene, we want you to find an arrangement of lights that give your objects soft shadows (try to get even softer shadow gradients than what's shown in Figure 6). For the second scene, we want you to try to intentionally create darker shadows for a dramatic lighting effect.

When you're satisfied with your two scenes, render and save them as two separate images using Blender Cycles (in an appropriate enough resolution to see the details) for the grading session. You can change the resolution of your output in the **Output Properties** tab of the Properties Editor, though the standard 1920x1080 at 50% should be good enough.

Show us: (1 pt) Two ray traced images using Blender Cycles of (1) your first scene with soft shadows and (2) your second scene with dramatic lighting. This part is extremely open-ended – as long as you make a good effort and can back up your lighting choices, you'll get credit!

Figure 6: An example of a scene illuminated with all 4 types of light: point, area, spot, and directional lights. We end up with a more natural look with softer shadows and more even illumination than from just using one light type.

1.6 TODO 5: Render a scene with environmental lighting.

In lecture, we talked about measuring incoming light and then using it to render synthetic objects in the scene. We can do this in Blender using (1) environment textures, also known as High Dynamic Range Images (HDRIs), or (2) the Nishita sky model. HDRIs are captures of real-world scenes, while Nishita Sky is a sky model that approximates the lighting of the sun and the sky.

Show us: (0.5 pt) A ray traced image using Blender Cycles of your own scene that has a HDRI or Nishita Sky applied and clearly visible.

1.7 TODO 6: Render a scene in your own unique camera view.

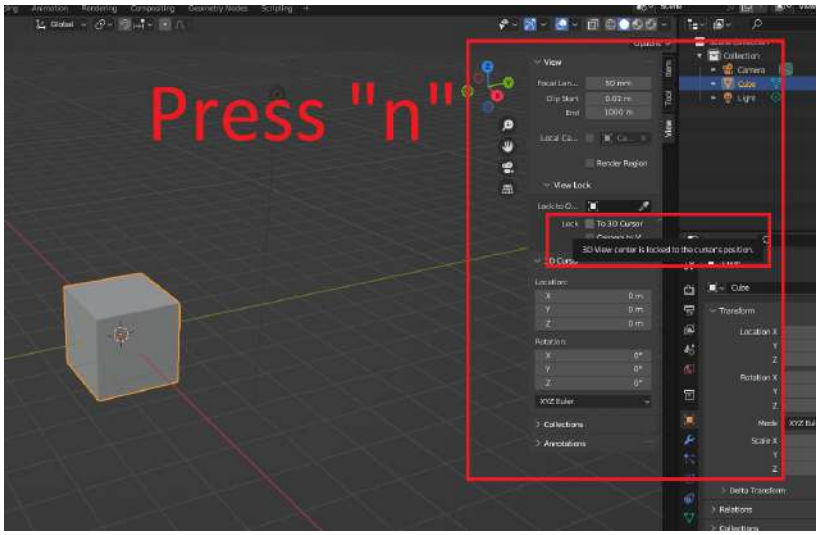
For this section, we want you to once again take a photo or find a reference image, but this time, it'll be of a scene layout that you might want to consider for your final project. You can also sketch your desired scene layout on paper if you want. In any case, we want you to have ready an image of 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. Choose a 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 lots of example scenes that you can choose from if you're not ready to start your own scene! Warning: because these are fancier scenes, it might take time for your computer to render properly in Cycles. If it takes too long to render these scenes as images, or if the images turn out blurry (due to lack of samples as we'll discuss later in this class), then a simple screenshot of your viewport **in camera view** and render preview is fine to show for grading. This doesn't have to look perfect!

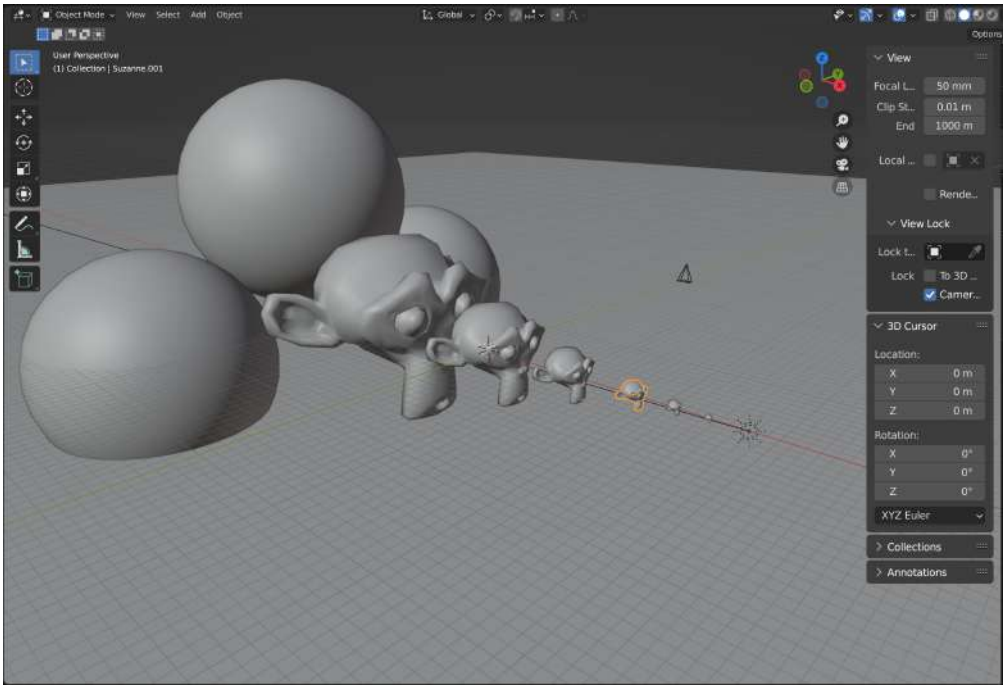
2. Lock the camera to our view so that when we're panning around and zooming in and out of the scene **in camera view**, our camera will move with us. To do this, open the transformation sidebar (View → Sidebar , or press N). Note that you can also change the camera's translation and rotation from here!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

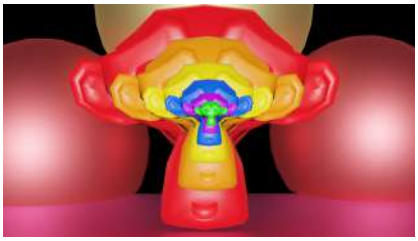


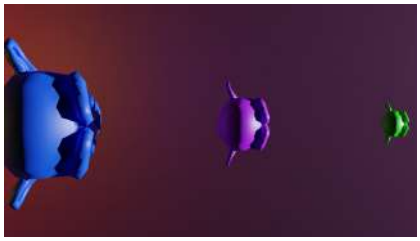
Now if you toggle to camera view and move your scene around in the viewport, then toggle back to the normal view, then you should notice that your camera has moved to accommodate for the movements you did!

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: (0.5 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 is a mixture of review from lecture as well as instructional (Blender) tutorials for you to reference.

2 Importing 3D Models

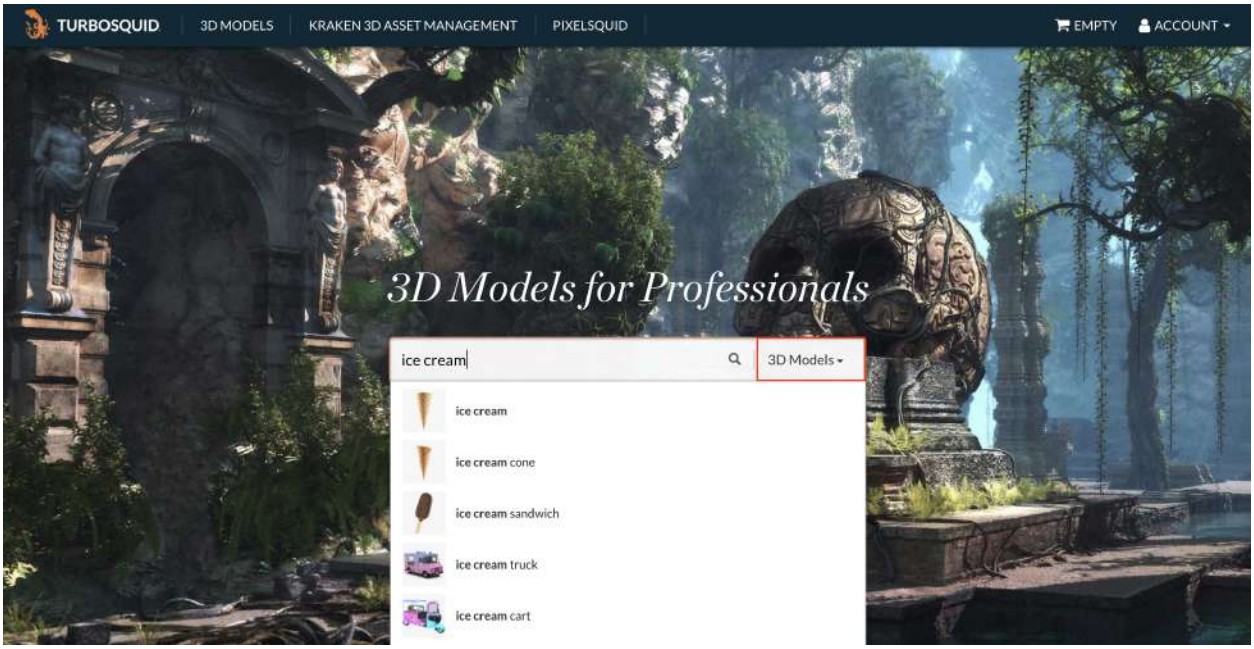
In this section, you will be given some pointers on how to start exploring the breadth of resources online and find models for your own scene. First, here is a list of popular websites for free resources:

- 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

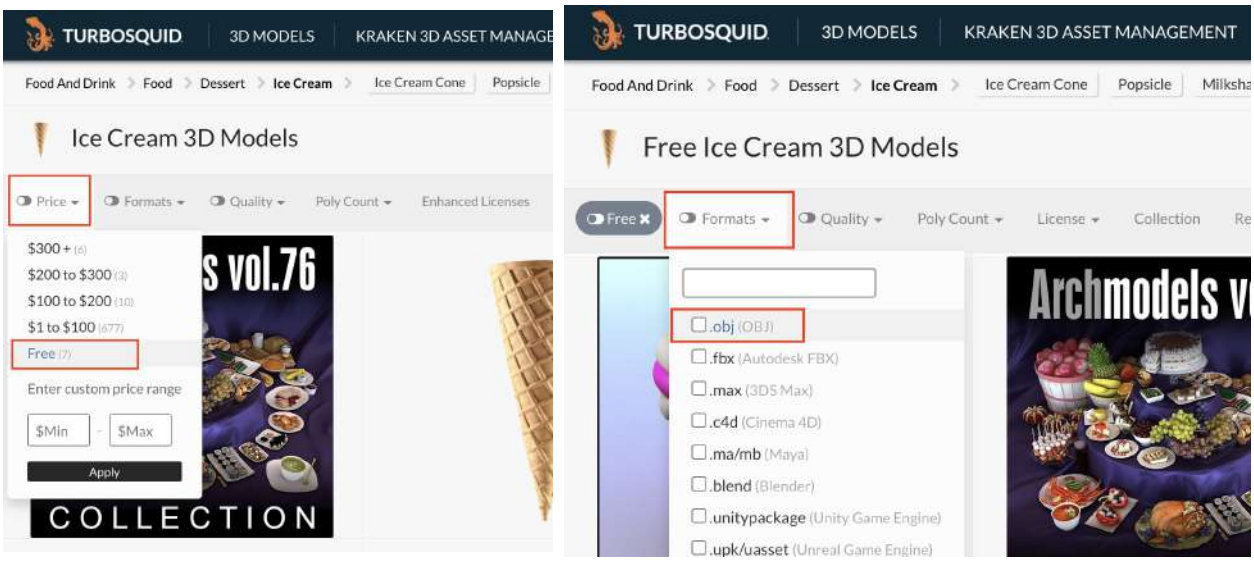
2.1 Finding a 3D Model

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.

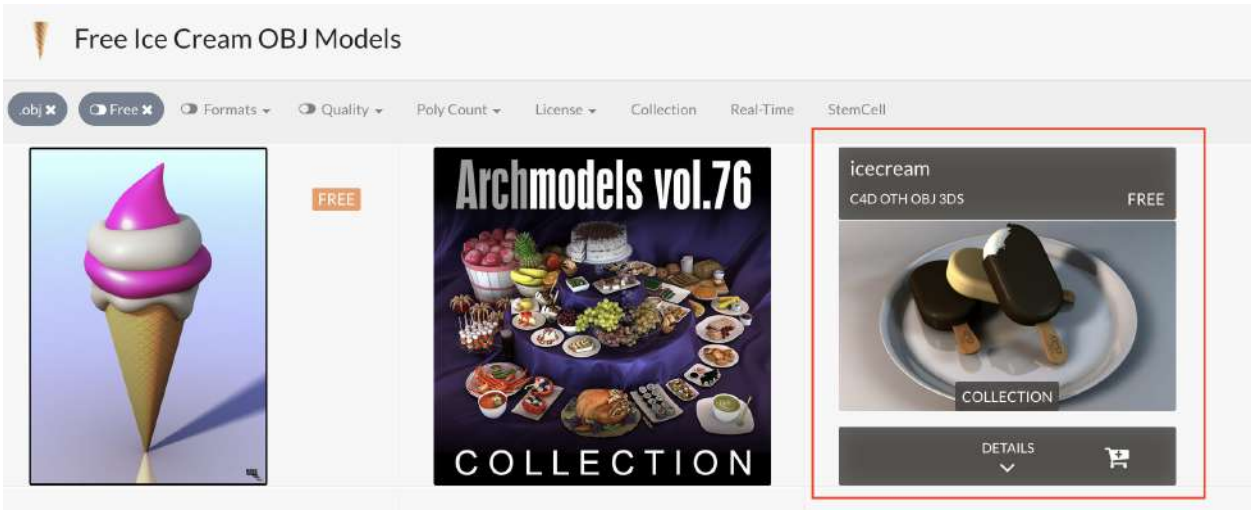
1. Search for a model:



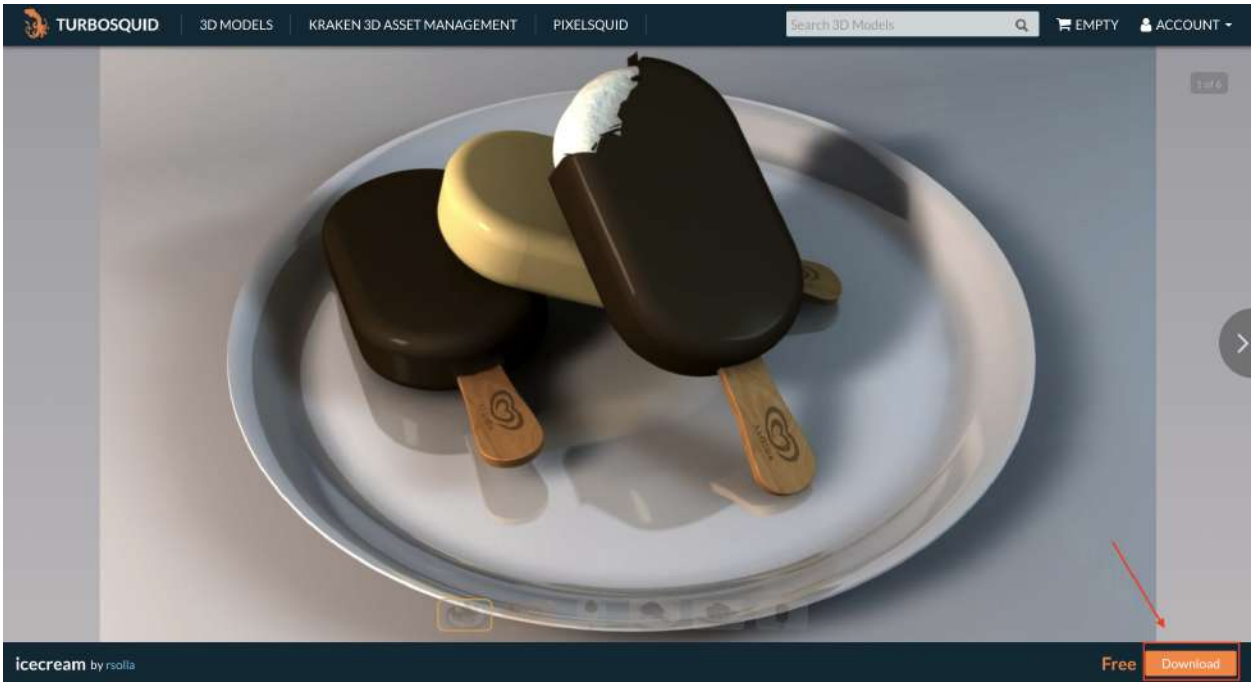
2. Filter for free models in the `.obj` or `.blend` file formats:



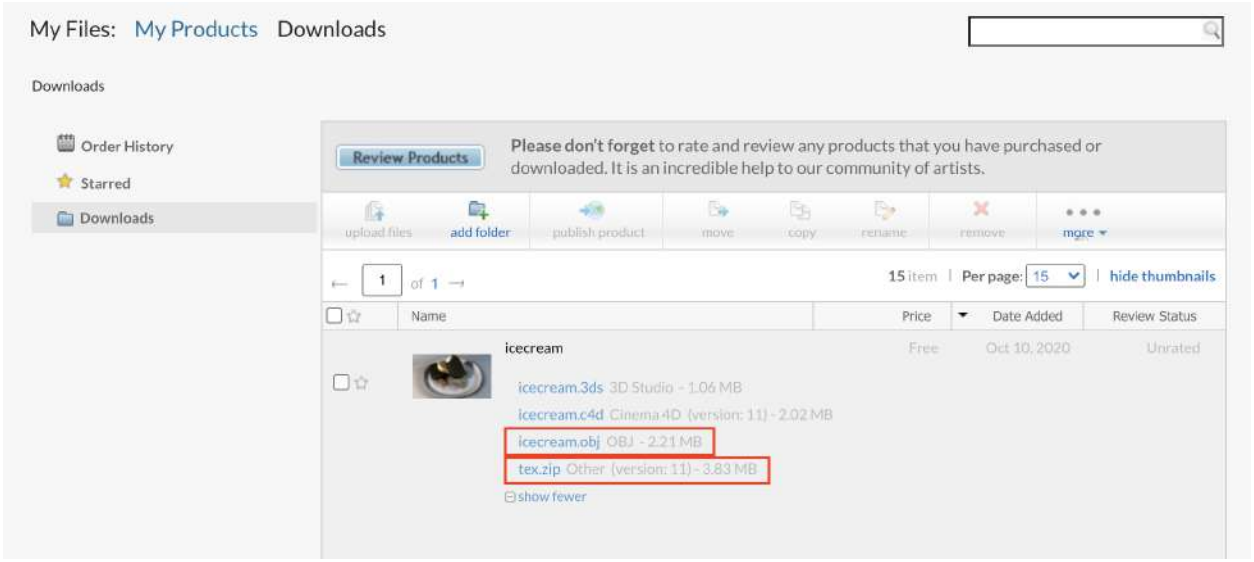
3. Choose which model you want to use and open its product page:



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.



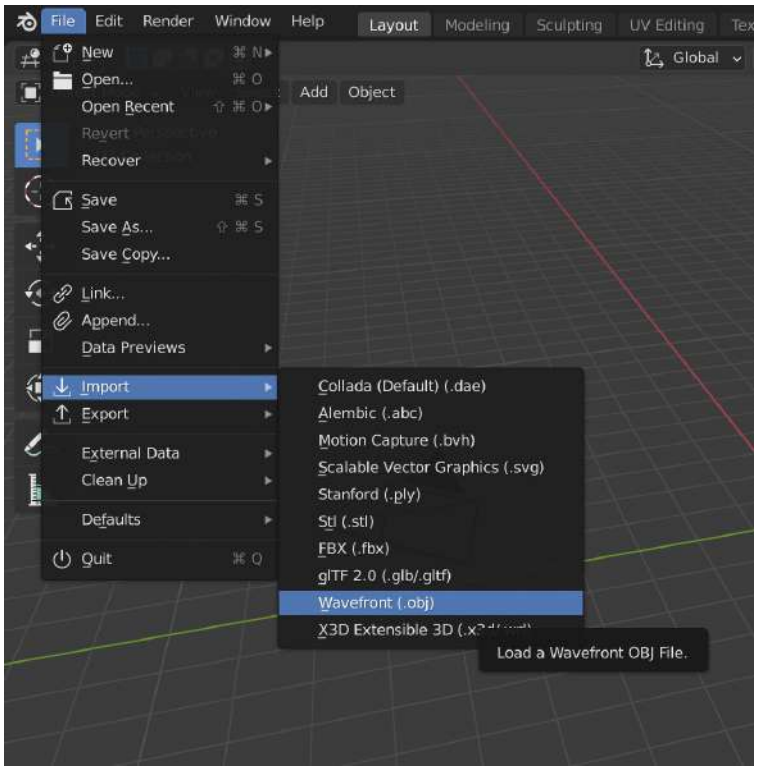
5. From your TurboSquid Downloads, save the downloaded `.obj` file by clicking on the file name (e.g. `icecream.obj`). Some models also come with texture and shading materials that you can download (e.g. `tex.zip`). We will take a closer look at these files in a later homework, but you may want to download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



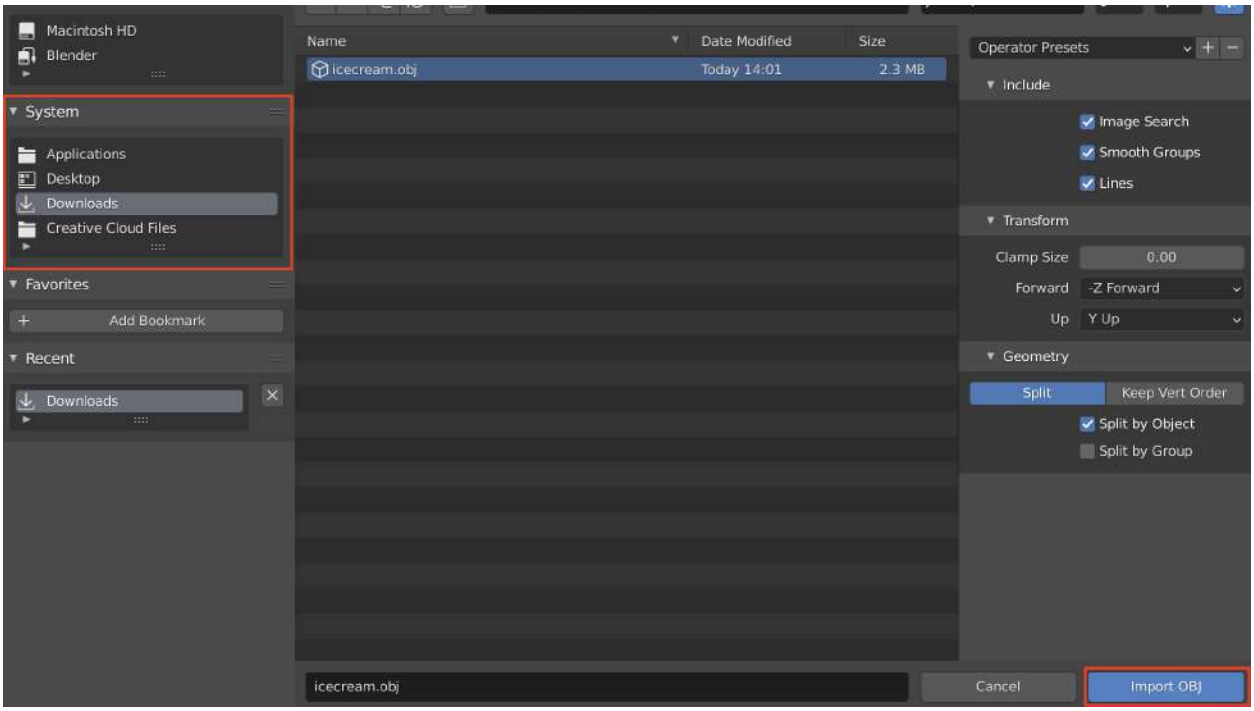
2.2 Importing a 3D Model into Blender

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:

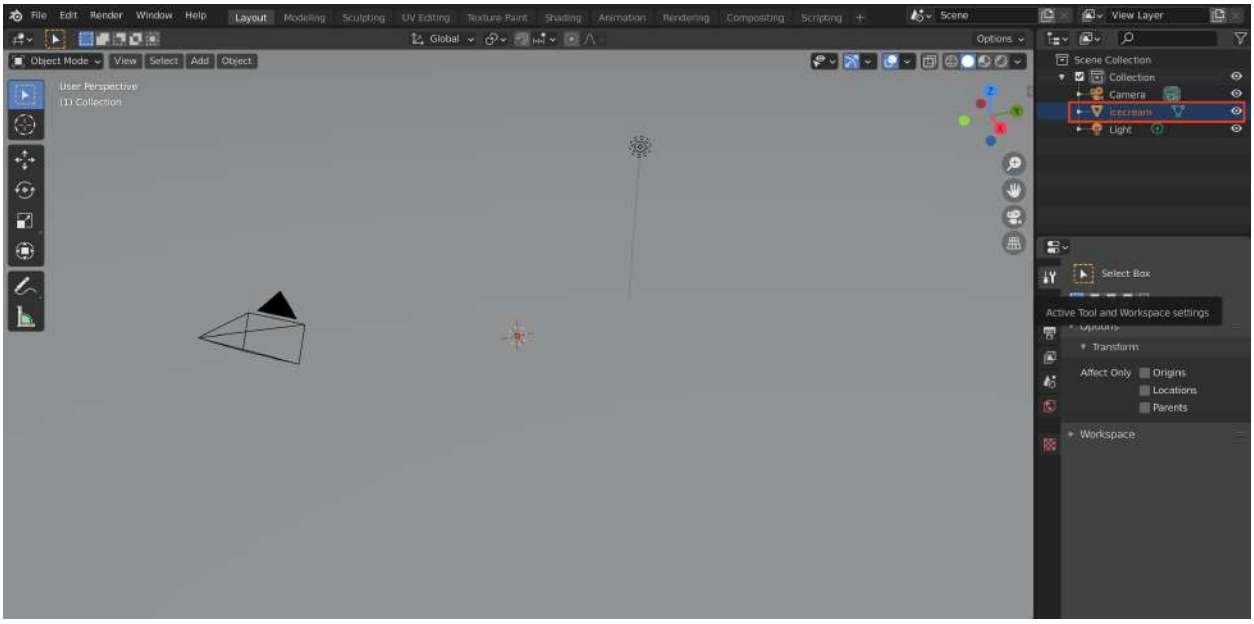


2. Navigate to the folder where you saved the model and select the **.obj** file. You can leave the import settings as default.

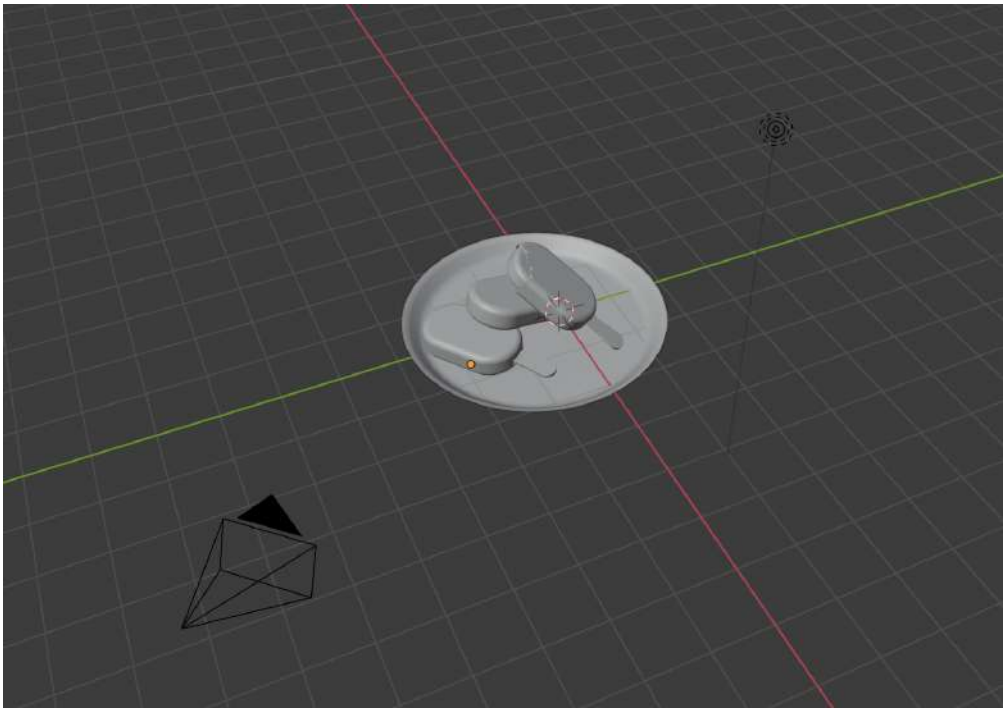


3. You should now see your imported object in the 3D Viewport, and the name of the object should appear in the **Scene Collection**. If the object is making the entire 3D Viewport gray,

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 7.

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 7. 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.

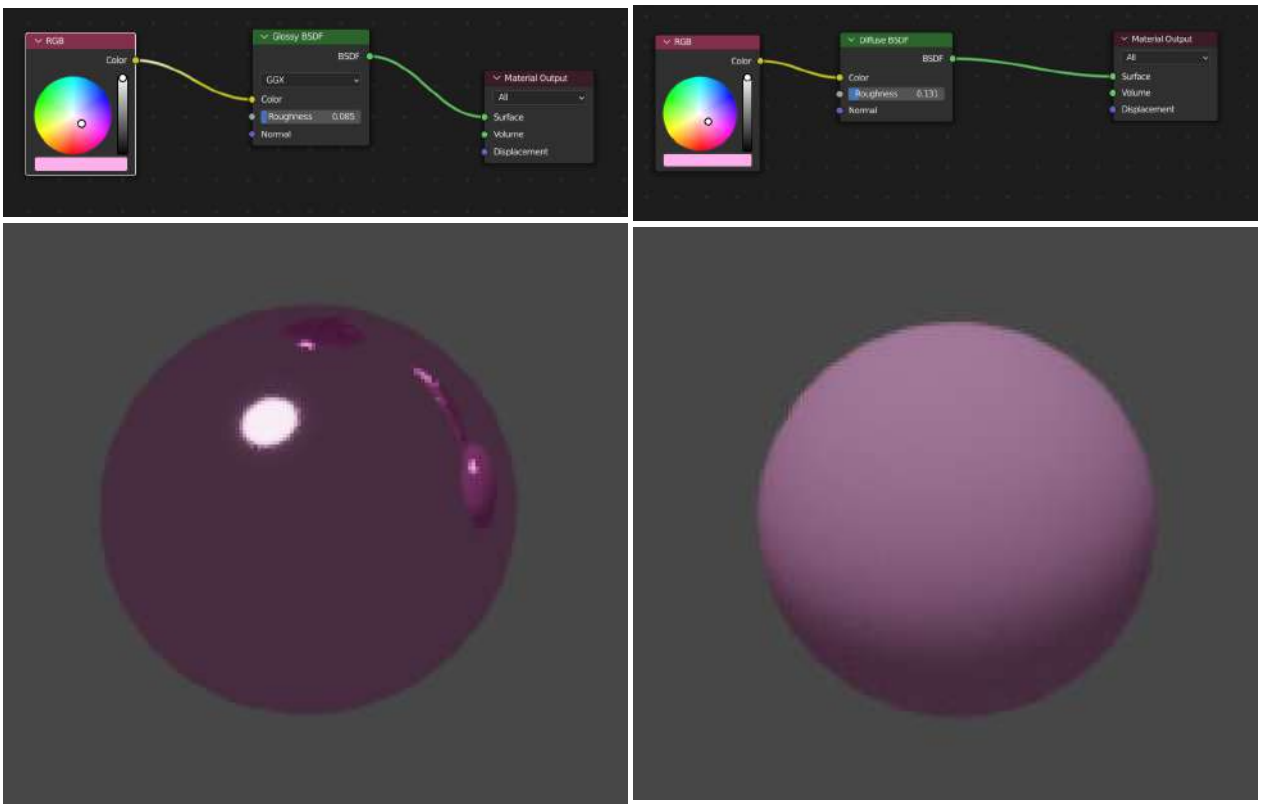


Figure 7: (Left) A sphere with a shader node graph to give it a pure (shiny) specular material. (Right) A sphere with a shader node graph to give it a pure (matte) diffuse material. The graphs are above, and the material results when rendered in Cycles are below respectively.

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.

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 from class affected the size of specular highlights.

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.

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.

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!

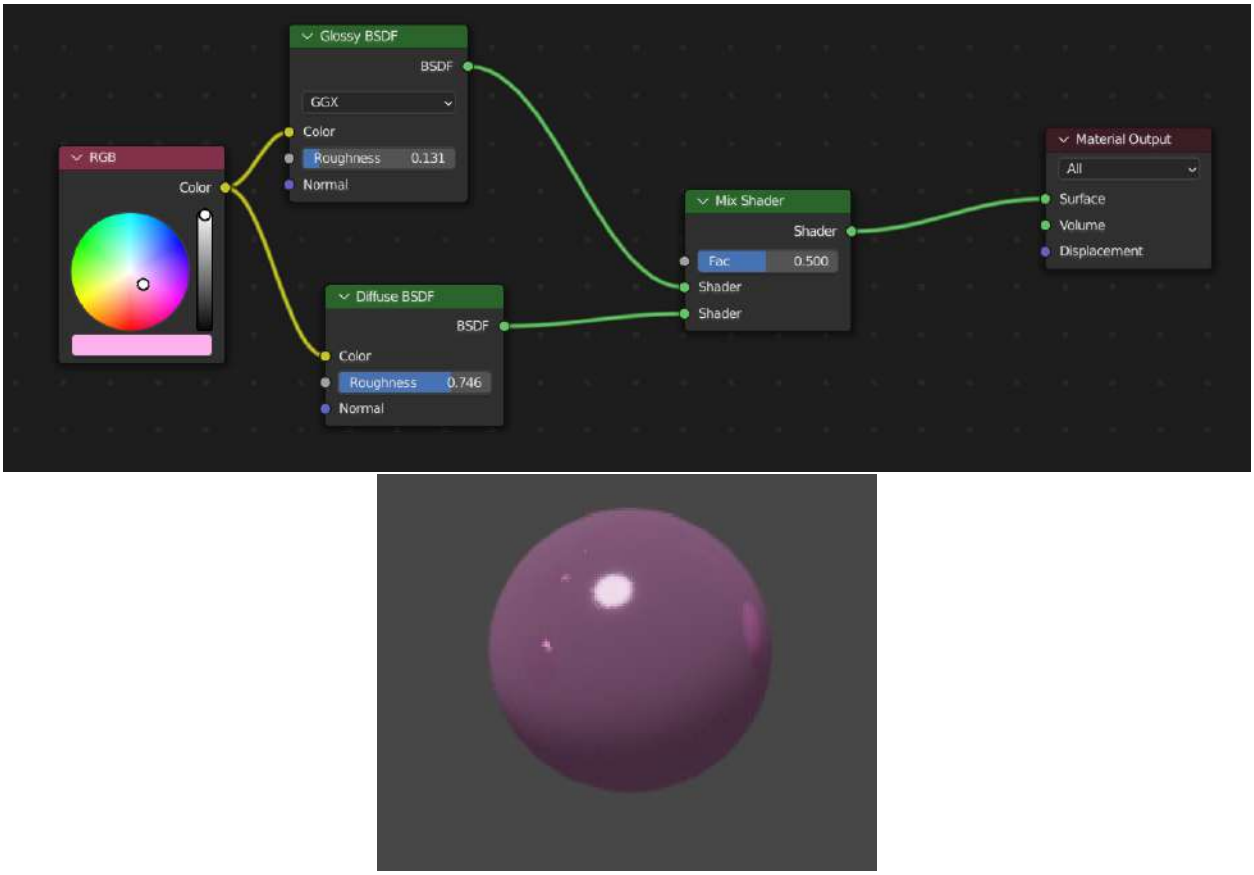
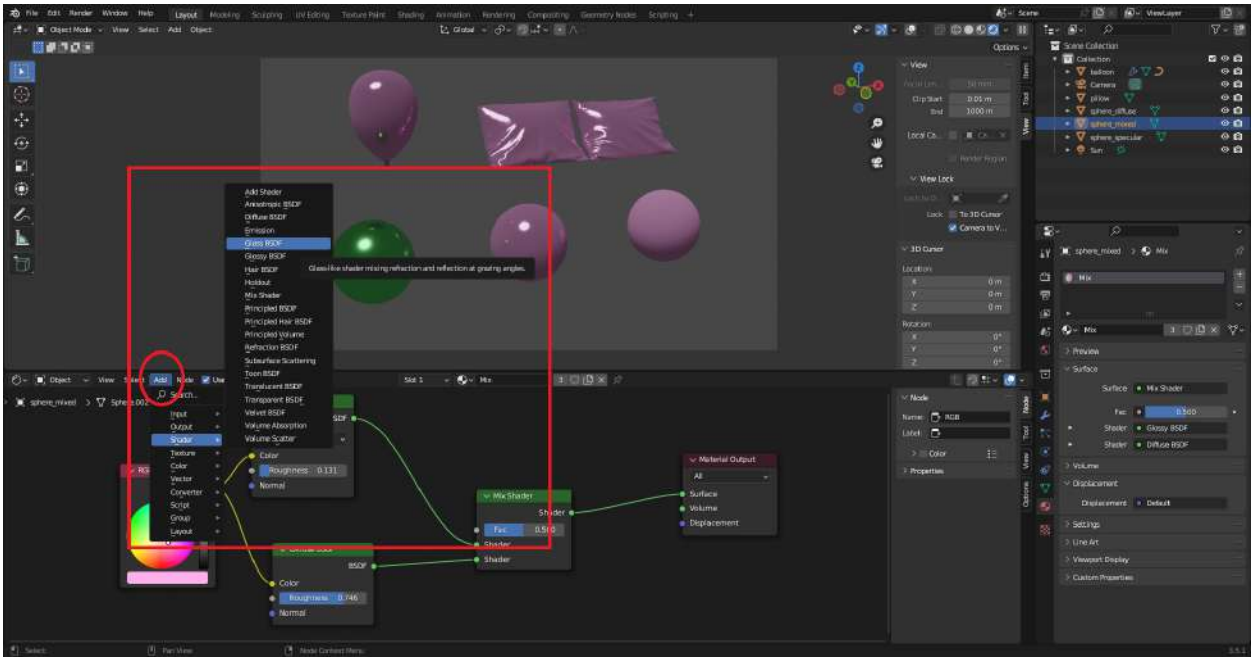


Figure 8: A shader node graph that mixes both a Glossy and a Diffuse BSDF node to give our sphere both specular and diffuse material properties respectively.

4.1 Other Shader Nodes

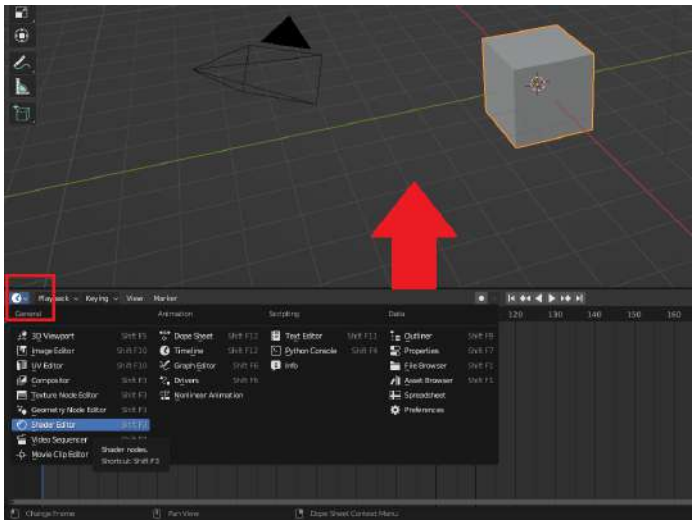
You can add other shader nodes such as the "Glass BSDF", which mimics glass material properties, via the **Add** menu in the **Shader Editor**. BSDFs are essentially Blender's equivalent of the BRDFs we discussed in class. You might also wonder what the other types of nodes like "Texture" are – we'll 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.

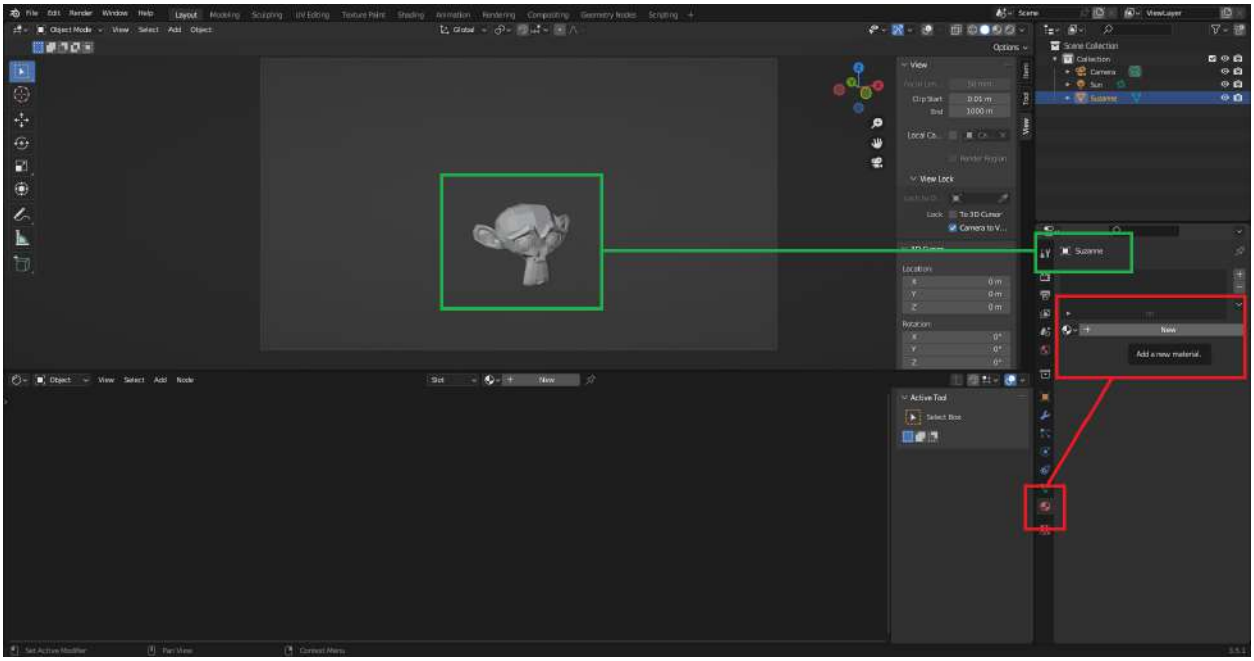
1. First, you want to view the **Shader Editor**. Click on the edge of the **Timeline** as we call it at the bottom of your Blender viewport and drag it up with your mouse. From there, click on the icon that looks like a mini clock near the bottom left and select **Shader Editor** to change the bottom-most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

5 Lighting in Blender

5.1 Point Lights

Let's try playing with a point light in Blender. First, we'll need to set up an actual scene. Here are the steps we took to set up the following scene:

1. Import this Stanford bunny model (with the default cube deleted), and translate it along the y-axis by 5. For speed and simplicity, you may just use the Properties Editor for transformations here, e.g. as shown in Figure 5 (for objects too in addition to the light(s) and camera).

2. Add a plane object via **Add** → **Mesh** → **Plane**, and scale it by 10 in both the x and y directions. This will be the floor plane.

3. Add another plane object, also scale it by 10 in both the x and y directions, then rotate it about the x-axis by 90 degrees, and translate it along the y-axis by 10. This will be the back wall plane.

4. Change the camera transformations from the default numbers to simply $(0, -10, 2.5)$ for the location and $(90, 0, 0)$ for the rotation.

5. Change the default light location to $(0, 0, 5)$. The rotation doesn't matter, since this is a point light (self check: can you answer why?).

6. Toggle to the camera view in the Viewport. Change your Render engine to Cycles, and toggle to the render preview. You should get a similar result to what's shown above.

Alternatively, if you've already begun to make objects for your final project, or just have some spare objects from previous work and experience, then you are very welcome to set up your own scene instead of the above! You may want to toggle the camera view back and forth when arranging your object(s), light(s), and camera. Additionally, you may want to toggle off the render preview and go back to the default solid mode view to lighten the computational load on your computer as you make modifications.

Once you're done setting up your scene, try experimenting with the placement of the point light(s). Remember from lecture that the irradiance (i.e. the light power or amount of light that is

hitting an object surface) varies based on the tilt angle of the surface with the light as well as the distance to the light. See Figure 9.

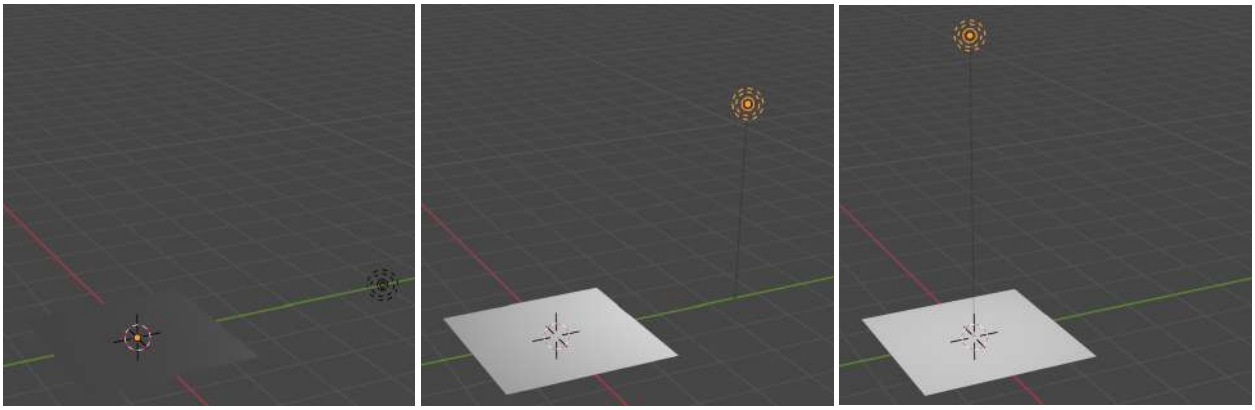
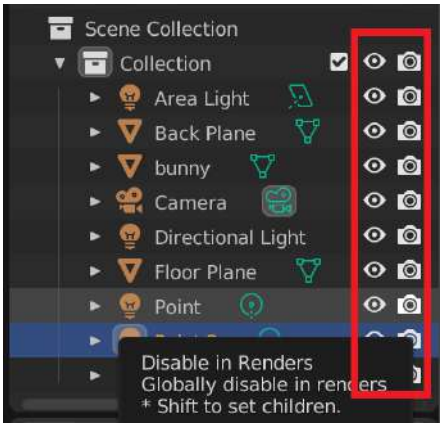


Figure 9: Irradiance varies based on the tilt angle and distance of the surface to the (point) light.

5.2 Area Lights

Now let's try playing with area lights. First, disable your point light(s) from Section 5.1 (using the eyeball and camera icons in the **Scene Collection** tree above the Properties Editor), and add an area light instead. Transform the area light however you see fit for your scene. Recall that the radiance of an area light varies based on the tilt angle between it and the object surface. See Figure 10.



Notice how the area light results in a much less uniform spread of light, as all the light rays are restricted to coming from a particular area. This is in contrast to the point light, where light rays are sent out indiscriminately in all directions around the point. **The larger area of an area light results in softer looking shadows (e.g. less dark contrast) than what might result from a point light** (see Figure 11 vs. Figure 12).

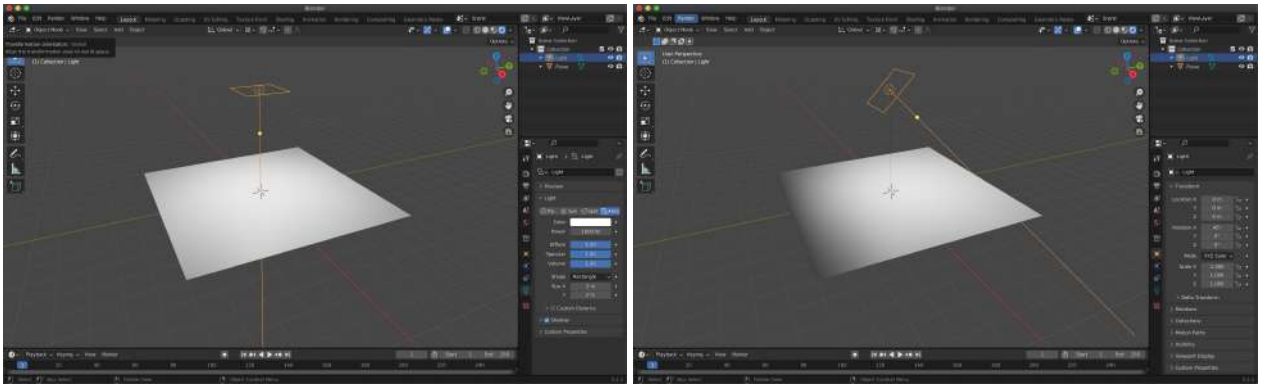


Figure 10: Radiance varies based on the tilt angle and distance of the surface to the (area) light.

Figure 11: An example of a scene illuminated with only an area light. The light rays are restricted to coming only from a particular area, though having multiple rays going in the same direction can lead to softer shadows than what would be produced by a point light.

Figure 12: An example of a scene illuminated with only a point light. Point lights tend to produce much harsher shadows with darker contrasts to that of the surroundings, as each direction only gets one ray of light.

5.3 Spotlights

Spotlights model lights that are engineered to emit light rays in a cone-like shape. Imagine lamps or streetlights in real life where the light is often surrounded by some sort of lampshade or outer object designed to concentrate the light. This leads to a light type that aims its light rays at a specific area, like area lights, but still spreads its rays in multiple directions, similar to point lights.

Spotlights are popular in modeling due to how common they tend to be in our everyday lives. They are also able to produce more dramatic lighting compared to a point light or area light. Consider Figure 13 compared to Figure 12. It's much easier to concentrate the light from a spotlight on a particular part of a scene than from a point light, whose light rays go in every direction.

Figure 13: An example of a scene illuminated with only a spotlight for more dramatic lighting. The spotlight excels at concentrating light all into one spot in the scene.

5.4 Directional Lights

The last type of light is the directional light or "sunlight" as Blender calls it. The idea of directional lights is that they don't have their own locations (i.e. no actual positions in world space), but rather, they shoot light rays uniformly across all of space in the same direction. Think of the sun in real life. The sun is so far away from us that all its light rays at any given instant might as well be shining in the same direction. Hence why Blender calls them sunlights.

Mathematically, when we consider directional lights for lighting computations (i.e. the lighting equation), we just ignore any concept of distance and use the same direction for the light vector regardless of where we are in the scene. So unlike what happens with point lights or area lights, an object that's twice as far away as another from a directional light still gets the same amount of light. Only the direction matters for a directional light.

Figure 14: An example of a scene illuminated with only a directional aka "sun" light. Directional lights have no concept of position. Instead, they shine uniformly across all of space in one direction.

6 Environment Lighting

6.1 Environment Texture - HDRI

High Dynamic Range Image, or HDRI, is a type of image frequently used for realistic lighting in 3D environments, usually in EXR format. It is created by combining pictures taken in the same scene with different exposure values, allowing it to contain much more color and illumination information than normal images. It used to be that we needed professional equipment to capture HDRIs, but nowadays you can do it yourself using 360 cameras. There are free CC0 HDRI libraries online, such as the popular HDRI Haven, which has a wide variety of environments from sunny outdoors to indoor studios.

To use HDRI textures in Blender, download an HDRI online (1k resolution is enough for our use) or pick one that comes with Blender (in the installation folder: **Blender x.x/x.x/datafiles/studiolights/world**, where x.x stands for your version number). Then, in the Properties Editor, go to **World Properties** and click on the yellow dot in the **Color** field to change the background color to **Environment Texture**. From there, click **Open** to browse to your HDRI. Swap to the render preview (with Cycles enabled) to see your HDRI in effect.

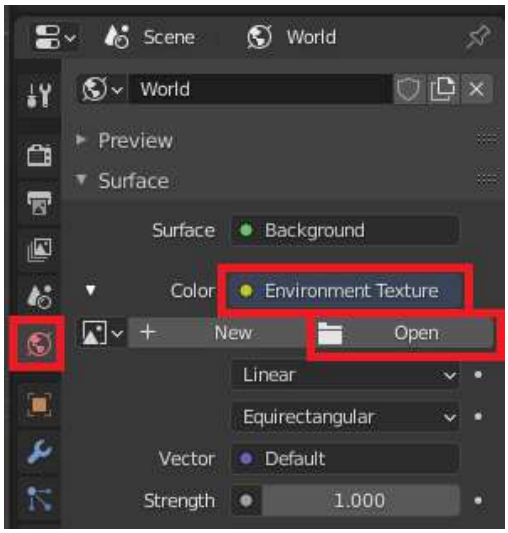


Figure 15: A scene rendered with Blender Cycles using a HDRI Haven texture.

6.2 Nishita Sky

Nishita Sky is an improved version of the sky model proposed by Nishita et al. in 1993. This model lets you control a sky texture with only a few intuitive parameters. You can read more about the Nishita sky model here and see a video demo in Blender 2.9's release notes.

In Blender, we can use Nishita Sky by setting the **Color** field to **Sky Texture** in **World Properties**. Play with the settings to get the sky you like. Note though that if you want a more detailed sky background (e.g. with clouds or buildings), then you might prefer a HDRI texture.

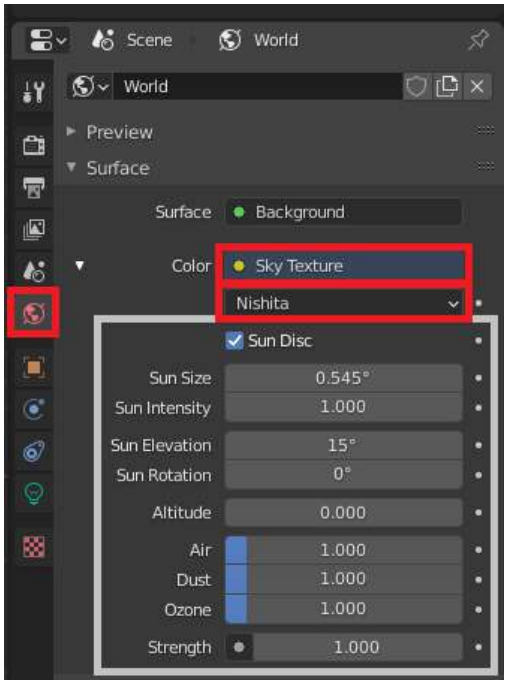


Figure 16: Nishita Sky Texture for Pavilion Barcelona. Source: Blender Wiki.