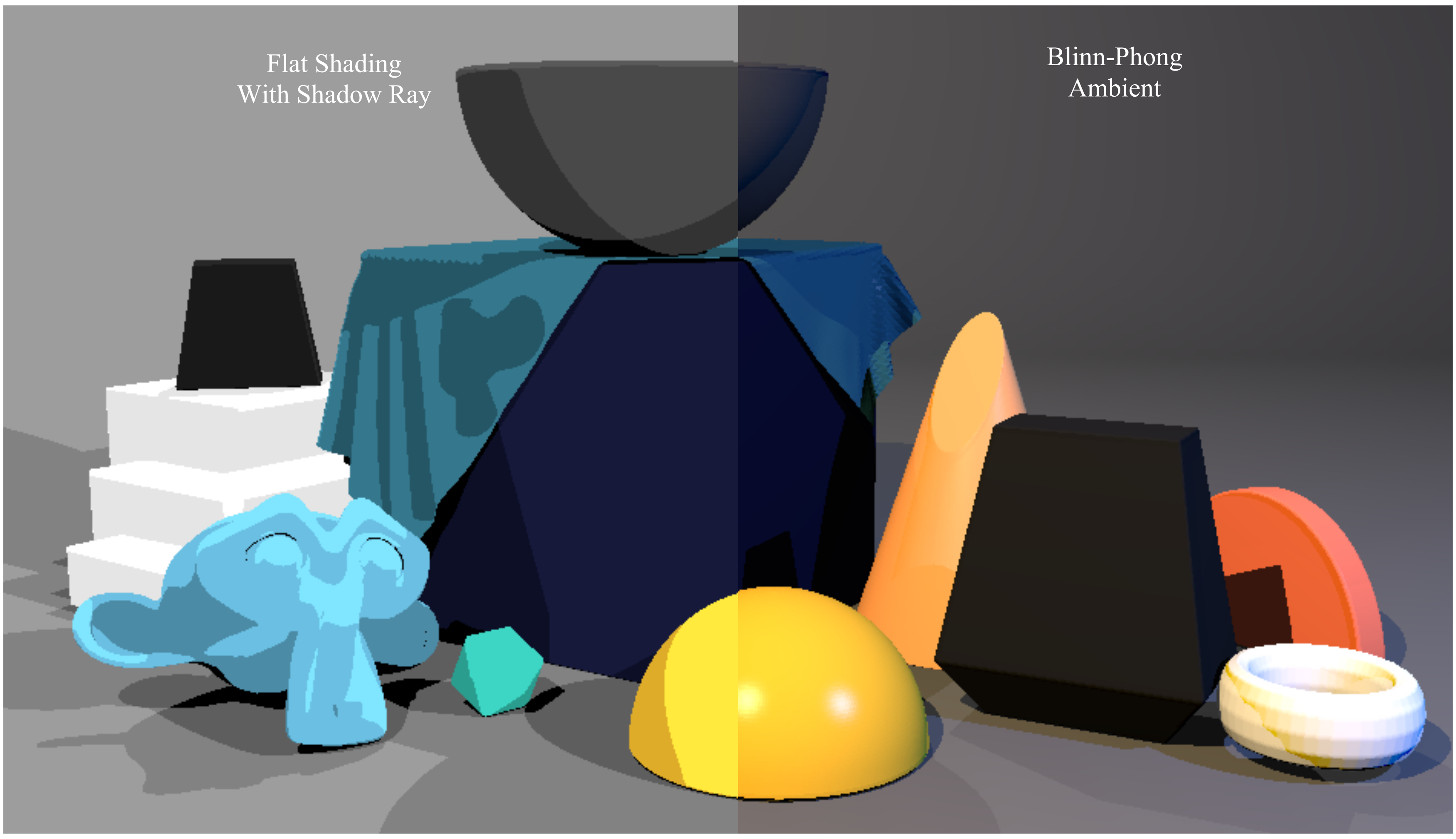
CMPE 360

Fall 2023

Project 4

Ray Tracing 1

Follow the instructions carefully. Be aware of the Checkpoints below. Make sure the complete each one since we will them.



This week, we will create a ray tracer using Python in Blender. The scene is created in Blender, and we will use scripts to get information from the scene through Blender's Python API, and create our own routine for bouncing lights in the scene to get rendered images. The image above shows the two main steps to our final image. For shading, we will adopt a popular shading model called Blinn-Phong. It is a simple but sufficient (at least for this simple ray tracer) approximation of how light interacts with objects, and is adopted by OpenGL and Direct3D as the default-shading model. Read more here

- Please read through this handout before you start coding. We provide many helpful tips here.
- There will be a lot of math and logic that you need to comprehend before writing your own ray tracer. If you get overwhelmed at any point, we suggest you review the lecture slides and make sure you understand the high-level idea

  First, download the .blend file for this homework from LMS.

  There's nothing more to it other than two text files, an already set-up scene, and a better workspace layout for doing scripting work. You can copy the scripts to a blank Blender file, add/delete objects and lights, and everything should work as well. However, please do not change anything in the scene for submission.

1. Scripting in Blender

In this homework, we will use Blender's scripting functionality to write a ray tracer.

1. 1. Start Blender from command line

For Python development in Blender, it is easier to use the console/terminal since the Blender output will be there, making it easier for us to print debug information, such as **print()**. To use the console, we need to launch Blender from the command line. See the Blender manual for more information.

First, start Blender from the command line. The command line will be the place we see debug information.

Windows

Press Windows+R to open the "Run" box. Type "cmd" and hit enter to open the command prompt.

First,

cd to the Blender installation directory. The default location is **C:\ProgramFiles\Blender Foundation\Blender 2.90**

Then type **Blender.exe** to launch Blender.

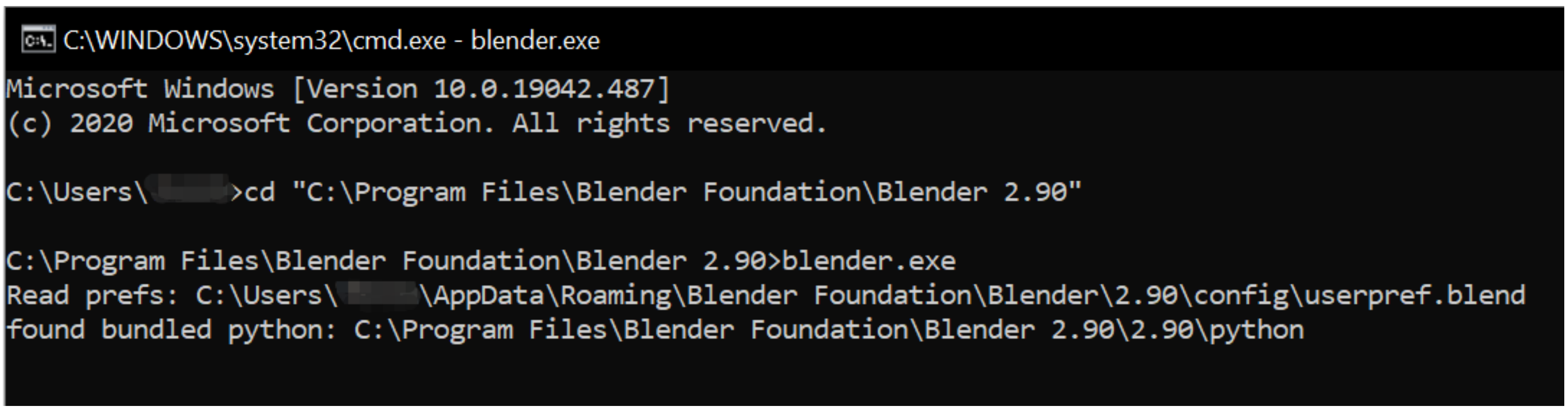


Figure 1

Mac

Please refer to the official document :

https://docs.blender.org/manual/en/2.90/advanced/command_line/launch/macos.html

Linux

Please refer to the official document:

https://docs.blender.org/manual/en/2.90/advanced/command_line/launch/linux.html

Blender will launch as usual. Open the downloaded .blend file from the menu bar: File→Open. You should see the workspace layout like this:

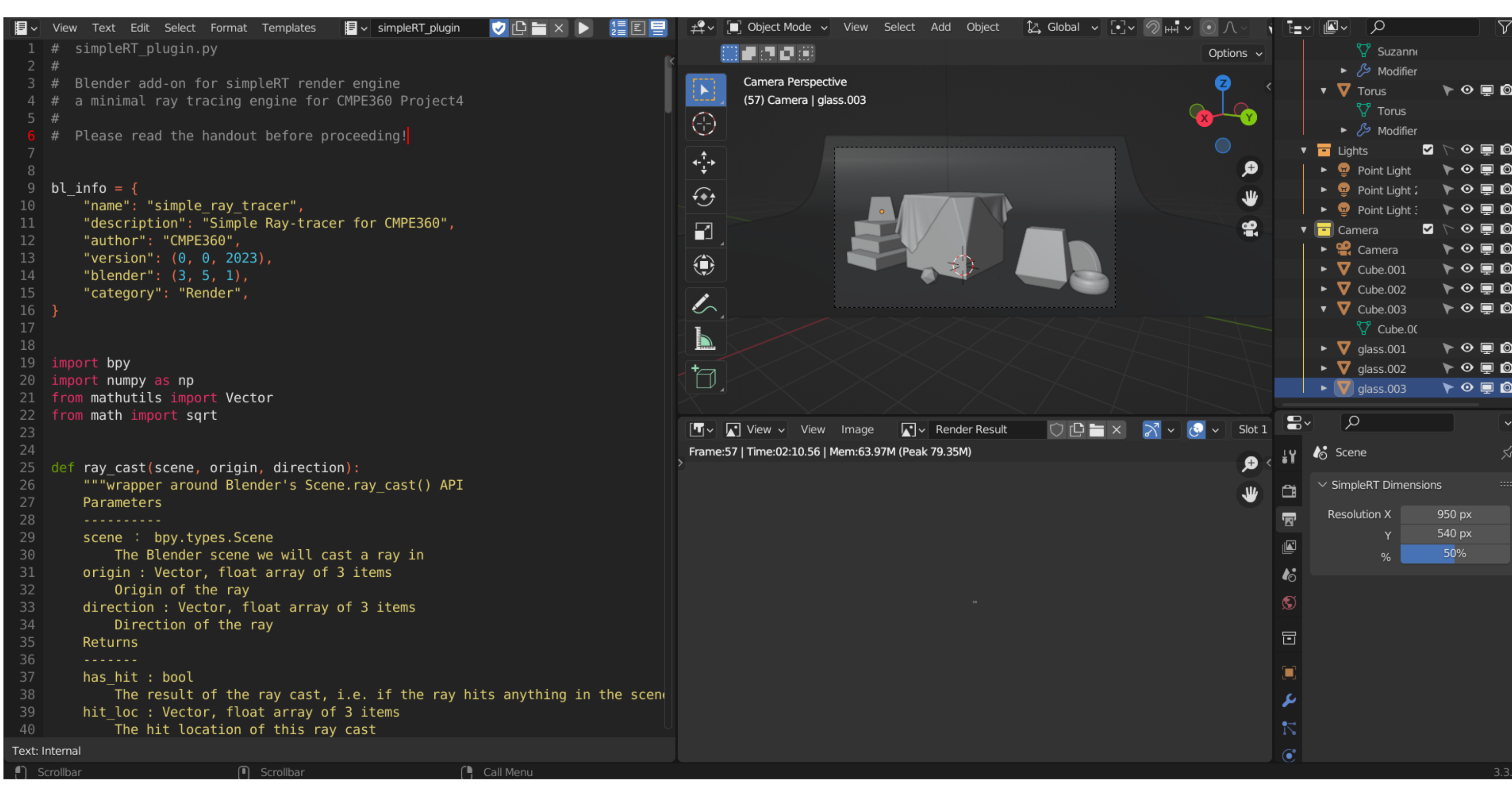


Figure 2

There are three main sections:

1. Left: Text Editor. This is where we edit and run the code.
2. Top right: 3D viewport. We can manipulate and preview the scene in this area.
3. Bottom right: Image Editor. We will view our rendering results here.

1.2. The Text Editor

First, let's browse to a script called simpleRT_UIpanels and run it. In the Text Editor, you can switch between files using the dropdown button (Figure 2). To run a script, press the "Run Script" button on the top bar of the Text Editor (Figure 2). 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". Nothing should appear in the command line if the script executed successfully.

The simpleRT_UIpanels script creates custom properties that will be used for our renderer such as material properties, light parameters, etc.. It then exposes these properties through the Blender UI so we can change them easily from user interface widgets.

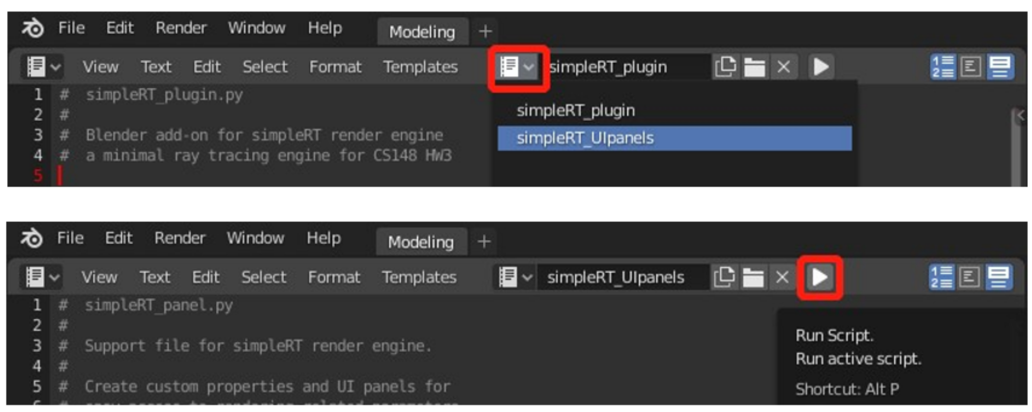


Figure 3

Then, switch back to the simpleRT_plugin and run it.

To check if everything is working:

From the Properties Editor, switch the Render Engine to SimpleRT (Figure 2). This is the new render engine that we just added to Blender by running the script. We will work on this render engine in this homework. The UI in Properties Editor should be refreshed (Figure 3). You should be able to see the "SimpleRT Render Settings" panel in the Render tab, and the "SimpleRT Dimensions" panel in the Output tab. If you still see the UI panels from other render engines, press F3 with your cursor in 3D Viewport, and type reload script and Enter to reload all the plugins (Figure 4).

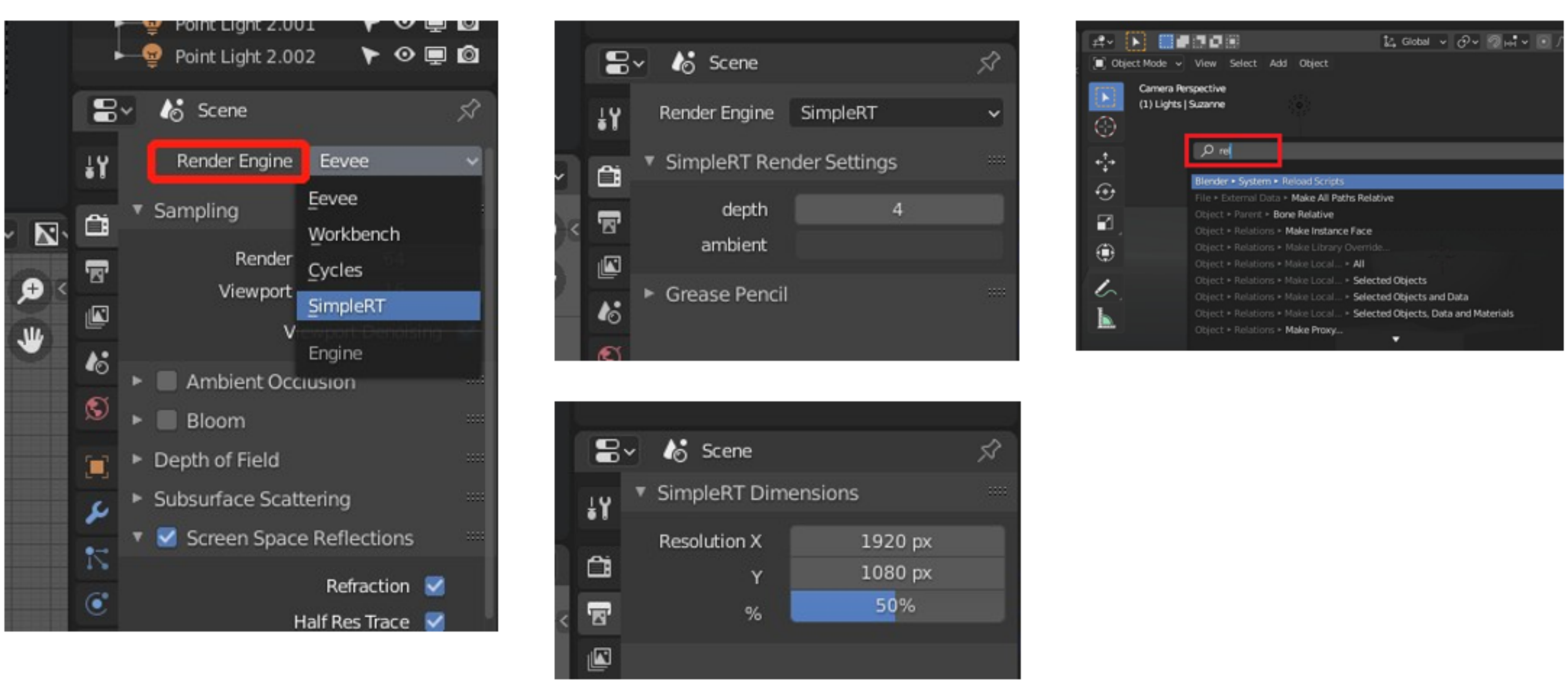


Figure 4

1.3. Start Coding

Now your task is to edit the simpleRT_plugin to finish all the TODOs.

After you make edits, run the script again, and hit F12 to start the render. You should see the rendered image appear in the Image Editor and a progress bar in the bottom status bar (Figure 5). When rendering is done, use Image → Save Image (Alt/Option S) to save the result. You can press Esc at any time to stop the render.

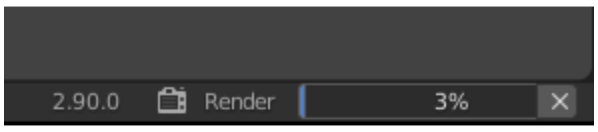


Figure 5

1.3.1. Editing and debugging

Blender has a very friendly development environment compared with other 3D software but it is still not on par with professional Python IDEs or professional text editors.

There are some handy shortcuts in the text editor. Formatting might be the most common operation: Tab for indent, Shift+Tab for unindent, and Ctrl / or Cmd / (mac) to toggle comments. For more shortcuts in Text Editor, please see Blender doc.

Most debugging information will be displayed in the command line window you start Blender from, but Blender still displays some information from the user interface. There is a Python Console (Blender doc) that you can use as an interactive Python shell and the Info Editor (Blender doc) which displays logs, warnings, and error messages.

1.3.2. Speed up the render with lower resolution

To speed things up, we should keep the image resolution low so the renders are fast, which means faster iterations. After the low-resolution image looks correct, crank up the resolution and render a higher resolution image for submission.

A quick and easy way to change the resolution is to change the resolution percentage from Properties Editor → Output Tab → SimpleRT Dimensions → %. This will effectively take the original resolution and shrink it down. So you will render a smaller image, with the same aspect ratio.

Usually 25-50% will be a good place to start, which will keep your render time under 10 seconds. This will of course give you a very aliased result, but you should be able to tell if your code is working from this image.

After the image looks right to you, change to 100% for a higher resolution image. This will take around a minute. With this resolution, you will be able to check the details.

- For submission, use 100% for resolution percentage, and keep resolution X and Y to 480 and 270 respectively (the default value for this .blend file.

1.3.3. Vector manipulation

In the code, we mostly use the data types that come with Blender. mathutils.Vector is the one we use to representlocation, direction, and color. See basic usage here. The most common functions are:

- vec.normalized() to return a normalized copy of vec
- vec1.dot(vec2) to take the dot product of two vectors
- vec.length_squared to get the squared vector length

To perform element-wise vector multiplication, the best way would be to convert the vectors to numpy arrays using arr = np.array(vec) , then use arr1 * arr2 to get the result.

1.3.4. Compare rendering results

In the Image Editor, you can store up to 8 rendered images in slots and toggle between them afterward. This could be helpful when comparing the results from different versions of code, or different material properties.

To render image to a certain slot, select that slot before rendering:

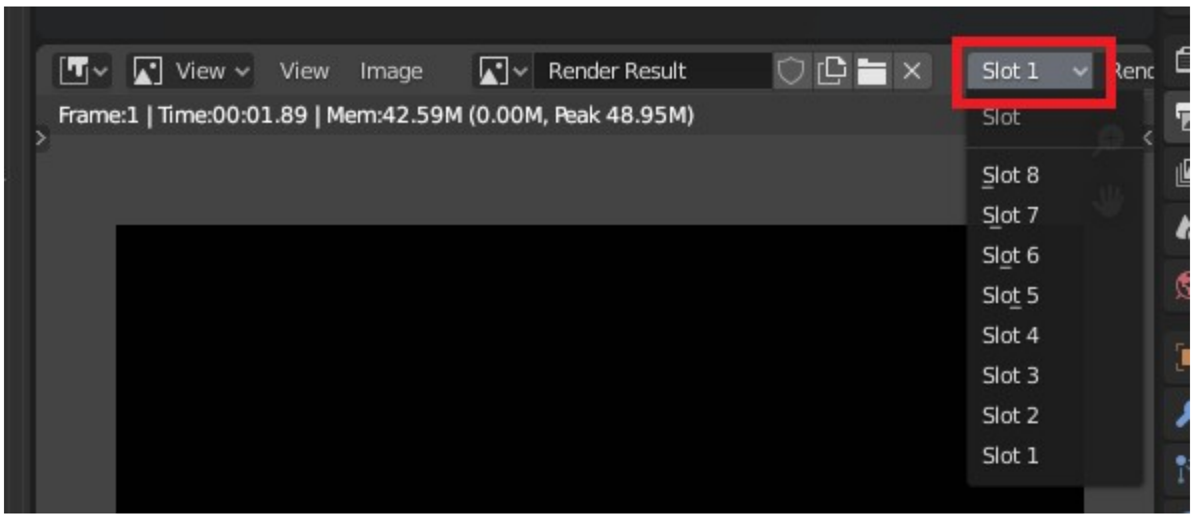


Figure 6

To toggle between the slots, use the number keys to go to the slot with the corresponding number (remember to turn off Emulate Numpad from the Preferences!), or press J and Alt J to cycle forwards and backward through saved renders, respectively. See Blender doc for more.

2. Checkpoints

All the TODOs are listed as comments in the Python scripts. Please read them to get started. Here we will provide general guidelines and checkpoint images you can compare with.

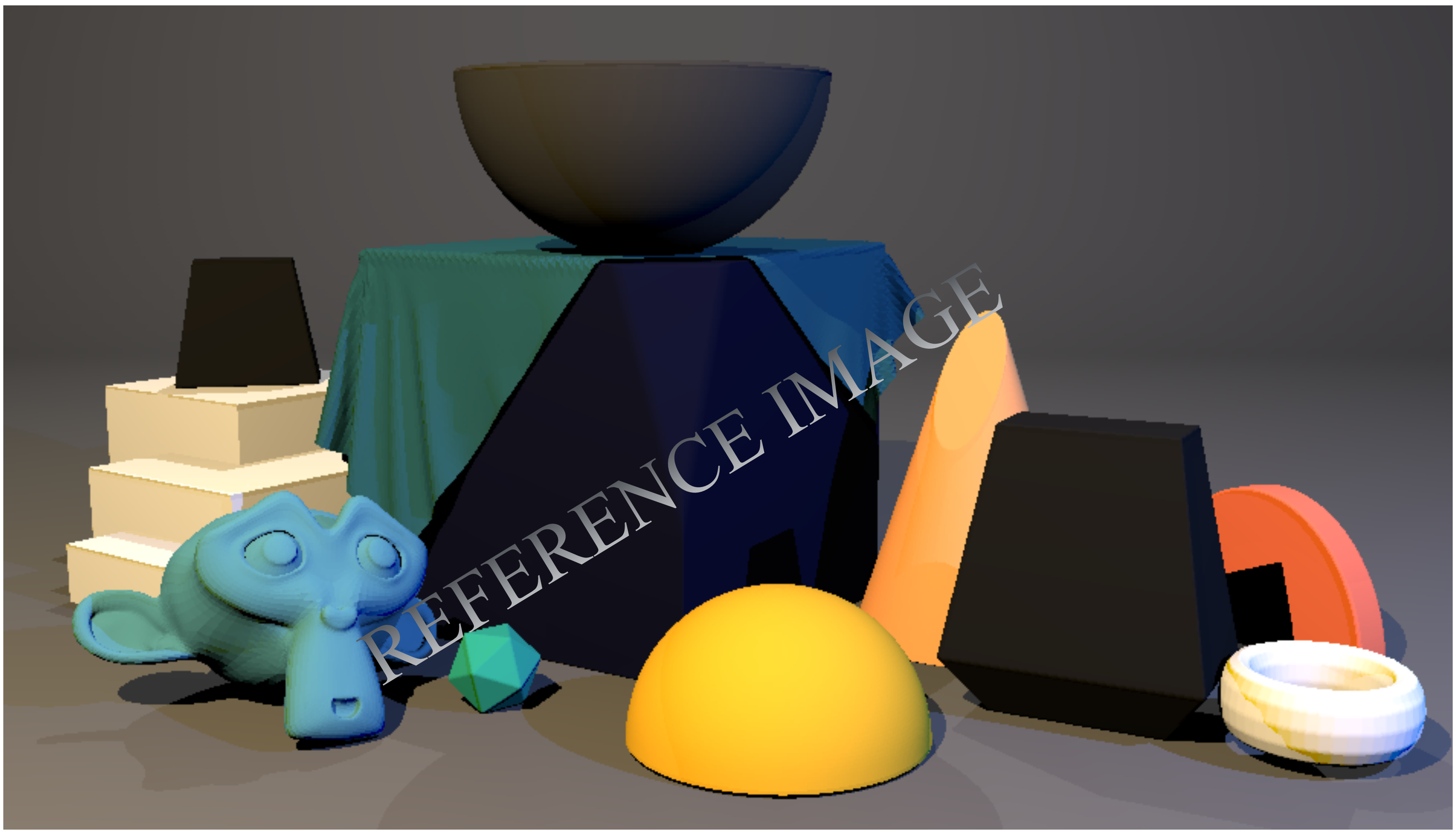
The ray-object intersection will be taken care of by the scene.ray_cast function from Blender. Our focus will be on how to use the intersection point and surface normal to figure out color calculations and light bouncing.

2.1. Shadow Ray

☐ **Checkpoint 1: Save the image with shadow ray.**

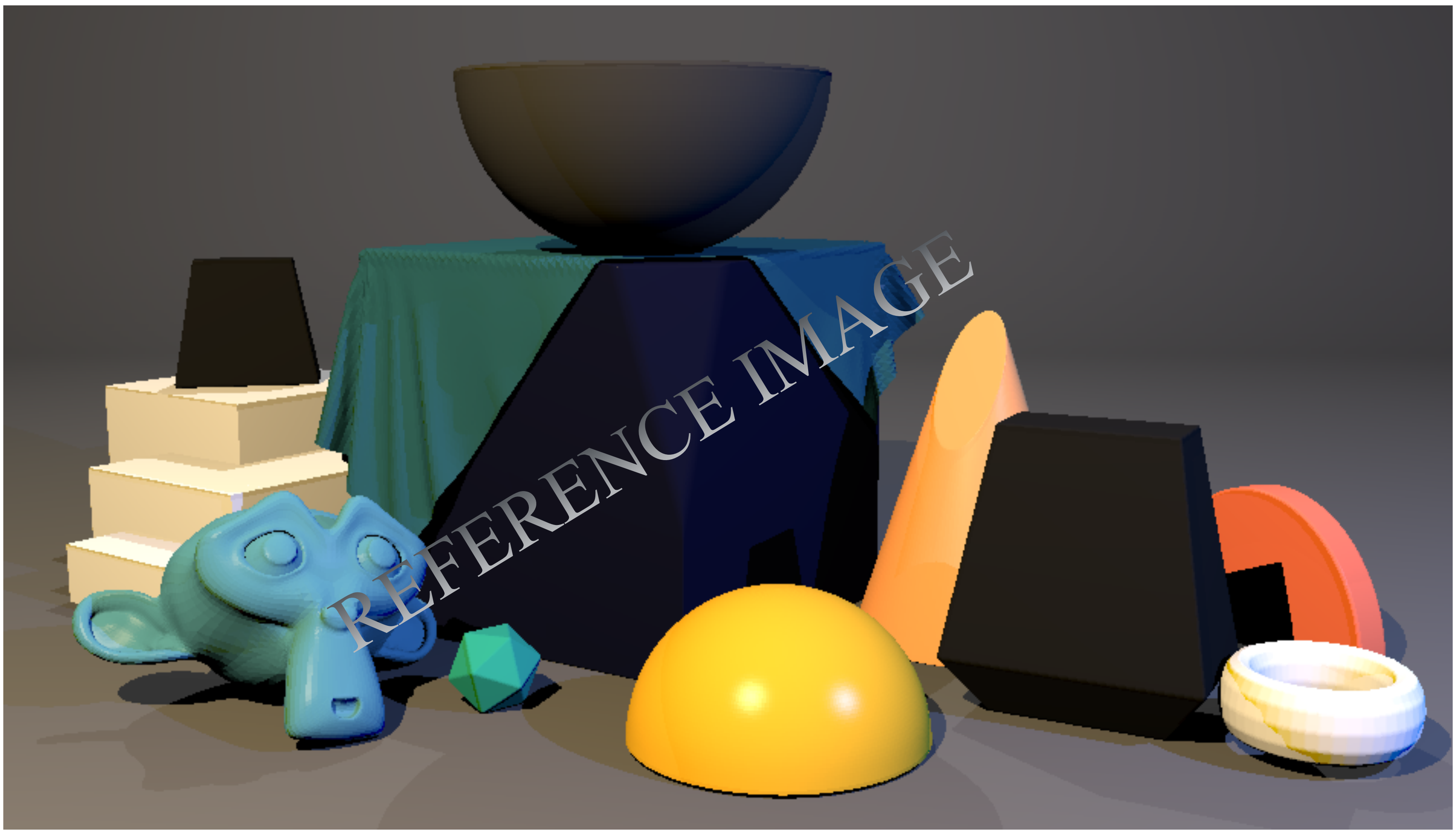
2.2. Blinn-Phong Model

With only the diffuse component, the image should look like this.



☐ **Checkpoint 2.1: Save the image with Blinn-Phong Shading (Diffuse).**

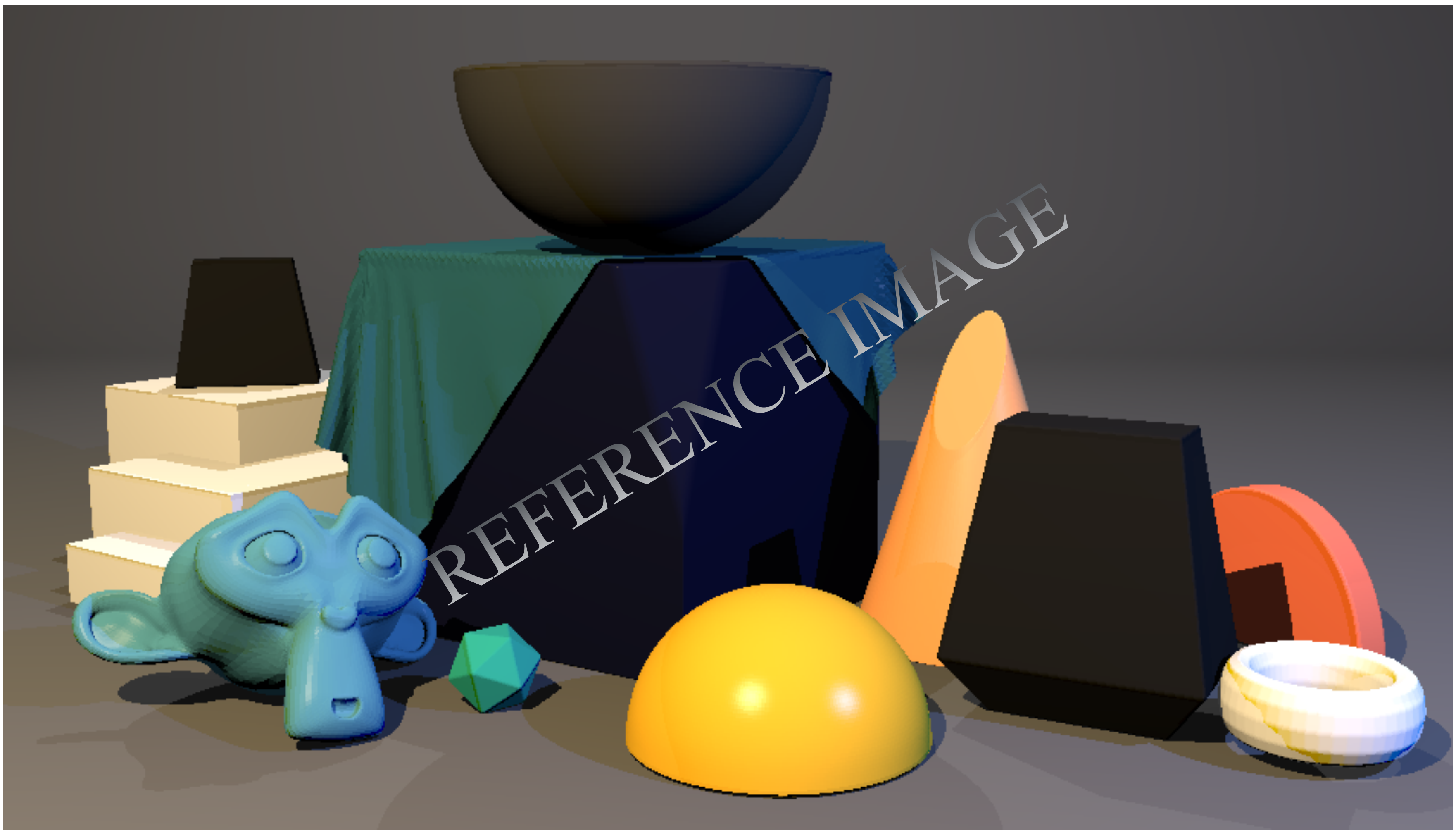
After adding the specular component, we have the full Blinn-Phong shading.



☐ **Checkpoint 2.2: Save the image with Blinn–Phong Shading (Specular).**

2.3.Ambient Light

The black area under the monkey and the shadow on the half-cylinder should brighten up. It is more obvious if you toggle between this and what you got from the last step.



☐ **Checkpoint 3: Save the image with ambient light.**

PROJECT REPORT SUBMISSION

!!IMPORTANT!!

This project report is due by 23:59 on Friday, 10 November 2023.

You can complete this project **individually** or groups of **two people**. Please don't forget to write the names of both people in the group in your report.

IMPORTANT!!

After finishing the assignment, please upload your pdf file and upload your simpleRT_plugin.py code (TODO1:, TODO2: and TODO3: parts are enough) to VPL on the LMS.

Follow the instruction and prepare a pdf file for uploading to the Moodle, your pdf file should include all parts. Please make sure your answers are numbered as below.

PART 1

- Save the image with shadow-ray and add the screenshot on your pdf file
- Explain your process in detailed and effect of shadow-ray.

PART 2

- Save the images with Blinn-Phong diffuse-shading add the screenshot on your pdf file.
- Save the images with Blinn-Phong specular-shading add the screenshot on your pdf file.
- Explain the Bling-Phong effect on your image and explain your process in detailed.

PART 3

- Save your image with ambient light and add the screenshot on your pdf file
- Explain your process and effect of ambient light.

GRADING RUBRIC

Project4	Points
PART 1	**15**
Save the image with shadow-ray and add the screenshot on your pdf file	5
Explain your process and effect of shadow-ray.	10
PART 2	**30**
Save the image with diffuse-shading add the screenshot on your pdf file.	5
Save the image with specular shading add the screenshot on your pdf file.	5
Explain the Bling-Phong effect on your image and explain your process.	20
PART 3	**55**
Save your image with ambient light and add the screenshot on your pdf file	5
Explain your process and effect of ambient light.	10
VPL Submission	5
Quiz	35
TOTAL	**100**