TED UNIVERSITY

Project Report

CMPE 360

Project 4

AUTHORS:

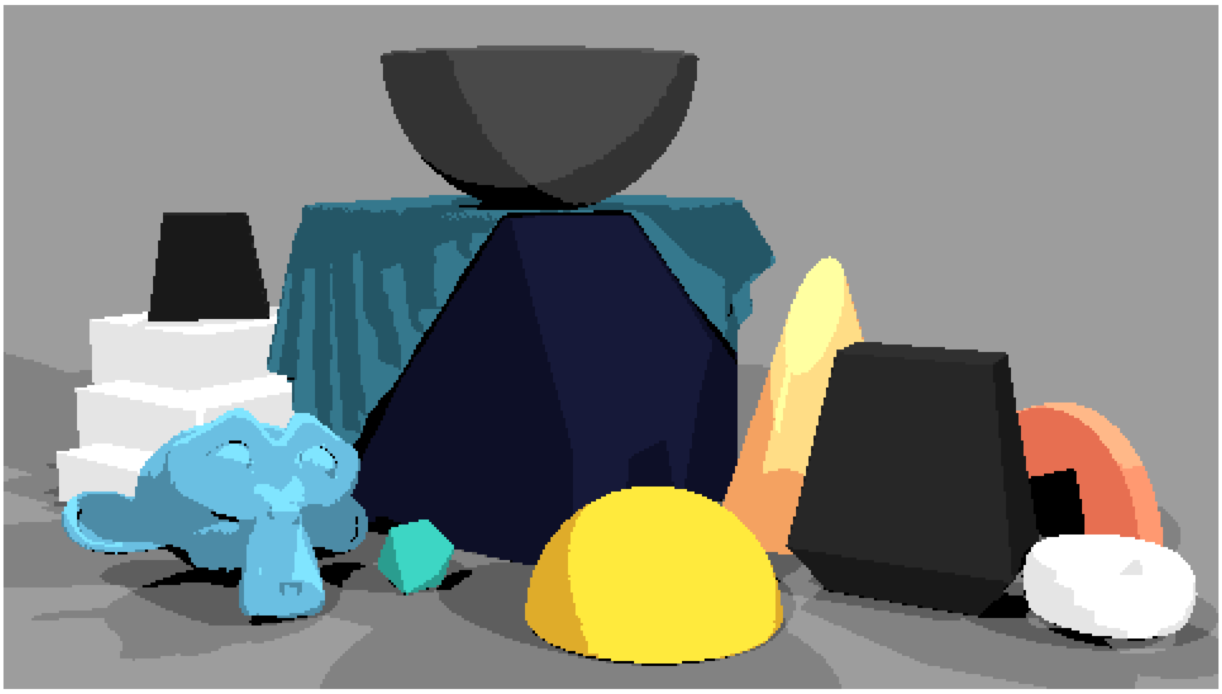
Kerem Güven	63556047646
Batuhan İşcan	11863132364

DATE: 11/12/2023

TABLE OF CONTEXT:

Part 1-

SHADOW RAY



In ray tracing, the image that is rendered is the light rays that reach the camera. So, in order to render the shadows, a shadow ray is sent from any point that can be seen to the light sources in the scene. While doing this, the point that the light hits is called hit_loc, meaning hit location and the light source is simply called light. The same calculation is done multiple times as there are multiple light sources.

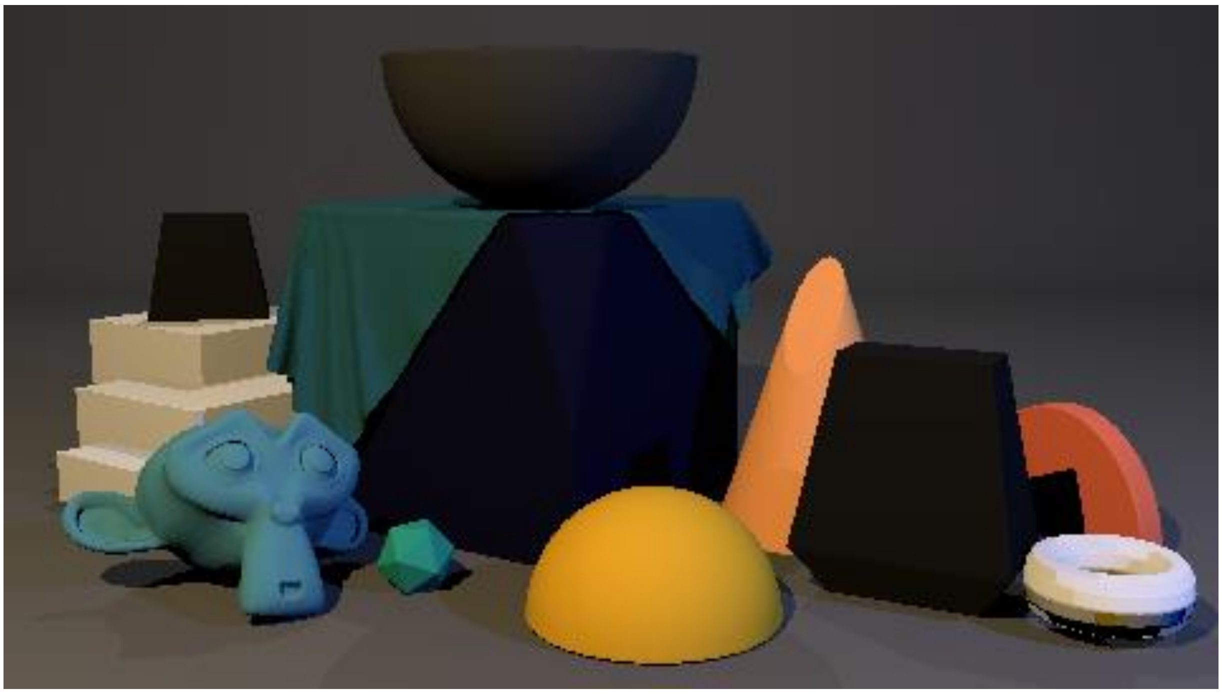
The hit_loc and the light.location are points in the 3D scene, each having x, y and z values. Since the light vector that we want goes from the hit_loc to the light, we need to substract their coordinates from one another and save it as a vector.

Afterwards, we used the normalized version of this vector in order to calculate a new origin point to disregard the shadow reflections on the same object, called a spurious self-occlusion. This was done by adding the normalized vector multiplied by an $\varepsilon > 0$ such that it matters to the initial hit point.

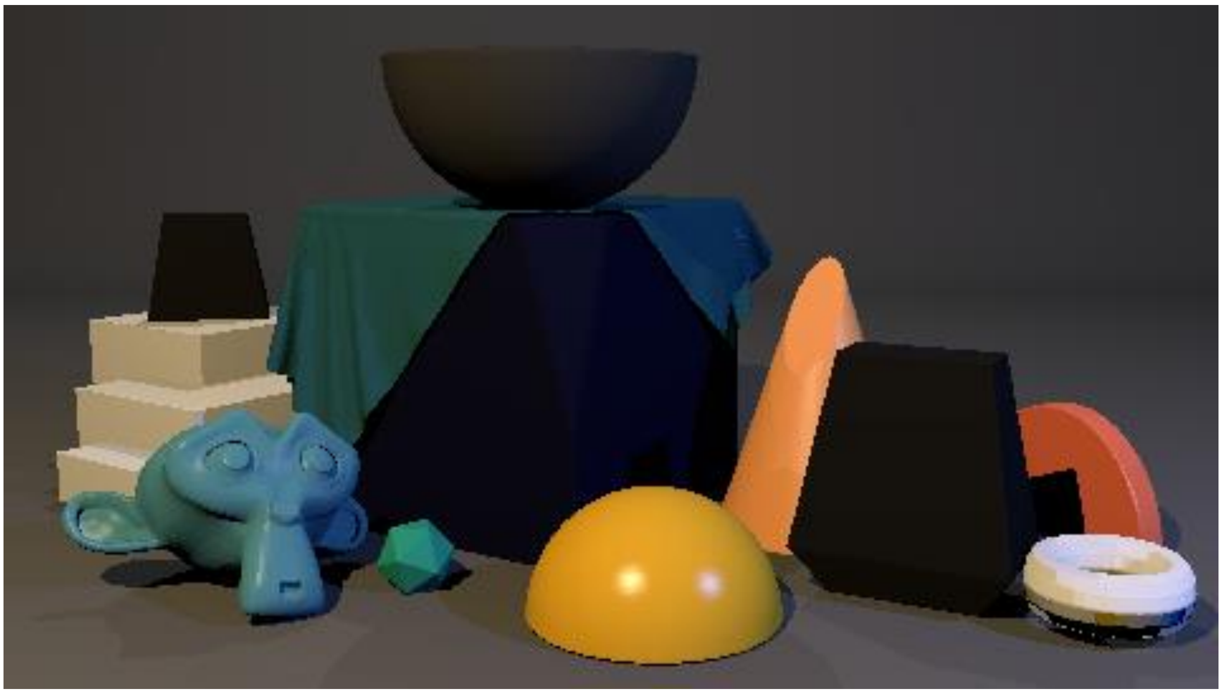
Finally, the bpy's ray_cast function is used again to recursively trace the rays with new parameters that were set in this part.

Part 2-

Blinn-Phong Diffuse Shading:



Blinn Phong Diffuse + Specular Shading



This time, the material of the object that hits the ray is considered. The materials have their diffuse_color and specular_color as vectors. These are given into the Blinn-Phong Shading part as vectors of x, y and z dimensions. These values are calculated in some operations with the light intensity and values such as the angle of the ray to calculate the colour of that pixel.

During the diffuse part, the vector diffuse_color is taken in and converted into an array from the numpy package. Afterwards, it is multiplied with the light intensity value that is calculated for each light source and their colour as well. The light's distance vector from the hit location that we calculated in the previous part is used to find the vector's length to calculate the light intensity.

The light direction vector that is the normalized light distance vector is then multiplied by the dot product with the hit_norm vector to find the angle between the different vectors. This angle is then used to multiply with the array diffuse_color and the light intensity, to be added to the original colour of the pixel.

As the name also suggests, this diffuses the shadows and creates a dim, yet a much more realistic lighting.

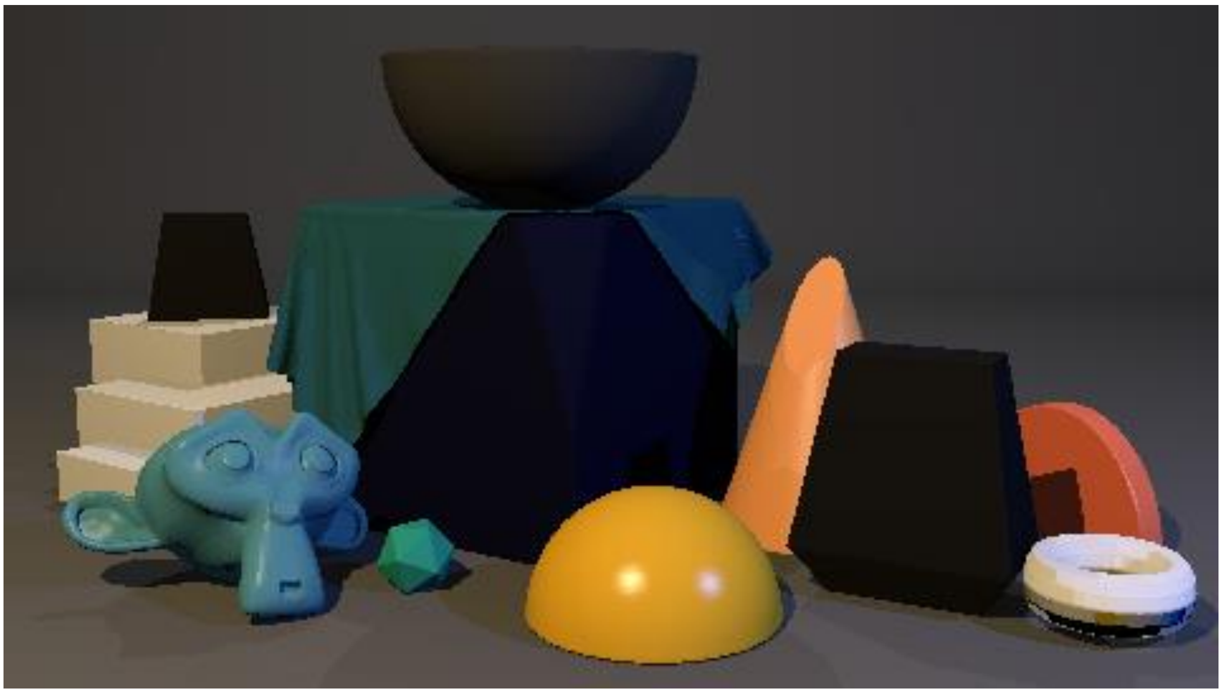
Following, the specular color is calculated and added on top. Again, the specular_color vector is made into a new array to use in the equation. The same value for the intensity of light is used and the dot product of the hit_norm vector with the normalized half_vector to find the angle this time.

The half_vector is the light direction vector, subtracted view vector, which is the vector that is from the pixel to the camera. The vector from the camera to the pixel is already parameterized so just multiplying it with the scalar -1 reverses this vector into the form we need. This value is taken to the power of the specular_hardness that's also a property of the object's material.

After finding the specular color value by multiplying the material's specular_color, light intensity and the angle, the image is rendered again, forming visible mirror-like brighter points and the full Blinn-Phong Shading.

Part 3-

Ambient Light:



In a room with no particular light source, the corner of the room that's supposed to be pitch black is realistically never that black. This is because light rays may come into the room from an unknown source and reflect on multiple objects, lighting even the corner. The Ambient Light is supposed to simulate this effect of light to reduce th edim parts on the image.

The material's diffuse_color property is used once again along with the ambient_color that is accessed from the scene, parameterized by bpy's built_in function ray_cast, as a value in the rendering engine's field. The ambient and the diffusion colours are both scalar numbers so they can just be multiplied and added to the colour of the pixel.

This results in the shadows being softer and having more variation between them. The image also generally acts as if there are more light sources.