CS 488 Final Project

Term: Fall 2023

Name: Devin Leamy

UW ID: 20872933

UW UserID: dleamy

Eva

A WebGPU Real-time Ray Tracer

Contents

1. Overview

Eva is a real-time ray tracer built in Rust using WebGPU, with an integrated scripting API.

2. Features

2.1. Texture Mapping

Any material can be assigned a texture. Textures are sourced from: `./eva/assets/textures`.

```
# Load a texture.
texture_handle = Eva.add_texture("texture.png")

# Add the texture to a material.
textured_material = Material(
  1.0,
  0.0,
  (1.0, 1.0, 1.0),
  texture=texture_handle
)

# Add the material to some geometry.
box = Box()
box.set_material(textured_material)
```

2.2. Skyboxes

Scenes can optionally set a skybox. Skyboxes are sourced from: `./eva/assets/skybox`. Skyboxes are defined by six images, listed in the order: [“x”, “-x”, “y”, “-y”, “z”, “-z”], defining the six faces of a cube.

```
Eva.add_skybox([
    "clouds/x.png",
    "clouds/-x.png",
    "clouds/y.png",
    "clouds/-y.png",
    "clouds/z.png",
    "clouds/-z.png",
])
```

2.3. Phong Shading

Eva can render `.obj` meshes with triangular faces. If the mesh has vertex normals, Phong Shading is applied.

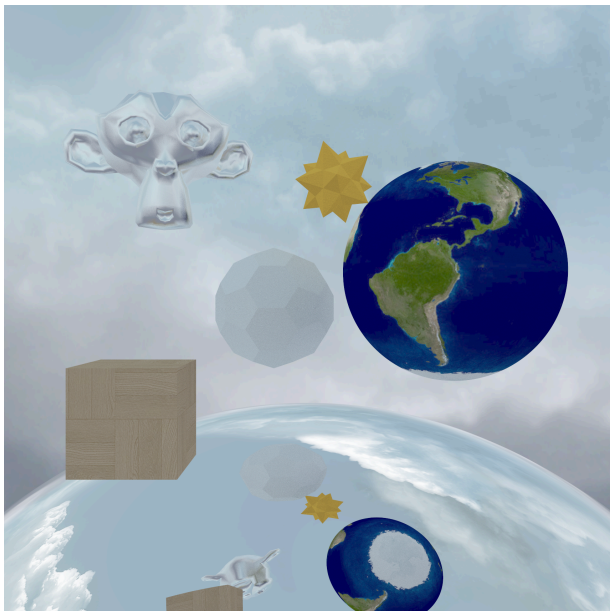

Figure 1: Suzanne Phong Shading

2.4. Real-time Ray Tracing

Eva supports two render modes `RenderStatic` and `RenderDynamic`. Implementing `RenderDynamic` makes your application real-time, and provides `update` and `handle_input` methods.

```
class Realtime(RenderDynamic):
  def __init__(self):
    super().__init__()

    self.cube = Box()
    self.add_geometry(cube)

  def update(self):
    self.cube.rotate_x(1)

  def handle_input(self, key, state):
    # Move the camera left and right in response to input.
    if state == "Pressed" and key == "A":
      self.camera.translate(-1, 0, 0)
    if state == "Pressed" and key == "D":
      self.camera.translate(1, 0, 0)
```

2.5. Reflections

2.6. Python Scripting

Eva is divided into two core components: `/eva` and `/eva-py`. `/eva-py` defines a scripting API for the `/eva` renderer. Scripts are sources from `/scripts`.

Scripts can be run using the utilities `run.sh` and `debug.sh`. `debug.sh` will display build logs.

To run a script, `my-scene.py` execute:

```
./debug.sh my-scene
```

2.7. TODO: Photon mapping

2.8. TODO: PBR Materials

3. Technical Overview

3.1. Ray Tracer

3.2. Scripting Bindings

3.3. Scripting API

4. Development Process

4.1. Lighting

4.2. Web

5. Post Mortem

5.1. Porting t

6. Resources