# COMS30115 – Computer Graphics

Linqing Fu(lf17866), Aday Muhajier(U1458584)

April 26, 2018

## 1 Compile

To generate the file, please execute the following commands.

**cd raytracer (or cd rasteriser)**
**make**
**./Build/skeleton**

Please notice that when including the external library SDL we use:

**#include<SDL2/SDL.h>**

Because in my computer the library I download is SDL2.

## 2 Raytracer

### 2.1 Basic Rendering

Our implementation of the raytracer use a flat lighting model, rendering each surface with a single color. You can move the camera pressing the arrow keys ‘up’, ‘down’, ‘left’ and ‘right’. You can also move the light position pressing ‘w’ and ‘s’.

### 2.2 Parallel Optimization

Optimisation made is by taking advantage multithreading on multicore system using openMP with its most aggressive O3 flag. This is done on tracing path which is part of the Raytracer. We examined a significant improvement by executing this in parallel because the majority of the process on Raytracer takes place on calculating collisions. The challenge in paralelising this is is in making sure that each loop can execute independent of each other. We record an improvement of 10x when compare to the original time.

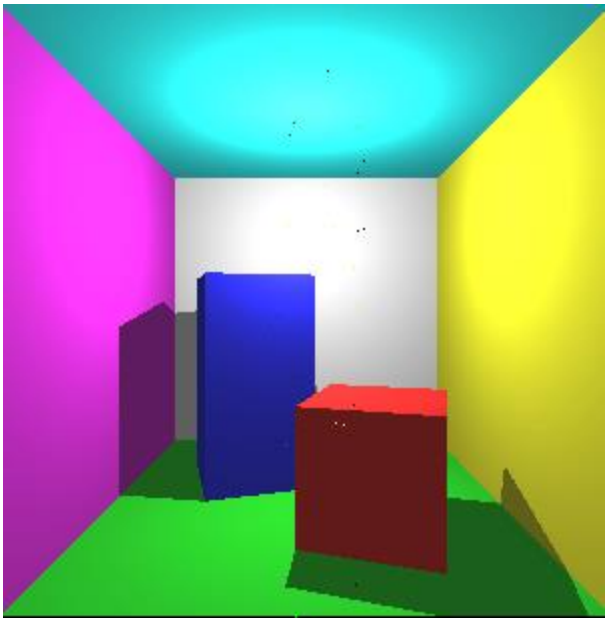

Figure 1: raytracer

# 3 Rasteriser

## 2.1 Basic Rasteriser

The basic raytracer will render the cornell box using perspective projection with flat colors. We use a depth buffer to ensure the box is correctly rendered. You can move the camera pressing the arrow keys ‘left’ and ‘right’ to see the rotation. You can also move the light position pressing ‘w’, ‘s’, ‘a’ and ‘d’.

## 2.2 Texture Mapping

We implement texture mapping in the rasterizer so the surface colors are determined by a image. We add a field in Pixel-struct to save the coordinate of the pixel in loaded image.

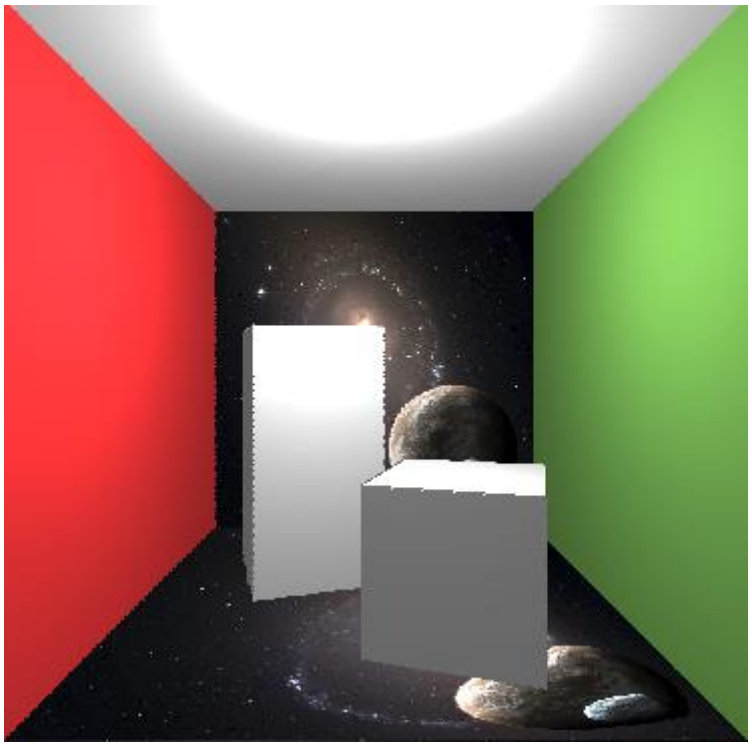

Figure 2: rasteriser