Exercise 1

Classes: 9th March and 16th March

Introduction

You will implement part of the Turner Whitted Ray Tracing algorithm covered in the theoretical class. The algorithm consists in tracing primary and shadow rays to calculate local colors and shadows. Students will use the P3F (P3D file Format) scene files.
P3F language is designed as a minimal scene description language for the scenes used to benchmark the ray tracers developed by the students in P3D Course. It is based on the NFF (Neutral File Format) language developed by Eric Haines. **Read carefully the P3F format description in attachment.**

In order to implement the Ray Tracing algorithm, you should use the provided Visual Studio 2019 **P3D_Template.zip** template for Windows OS.

IMPORTANT: Download and install the free Visual Studio 2019 Community edition. The free version provides all that you need.

Installing the template for application development

Download the template P3D_Template.zip and copy it to: "C:\Users\MyUsername\Documents\Visual Studio 2019\Templates\ProjectTemplates". If Templates and ProjectTemplates directories don't exist, you should create them.

Now you can create your project. Open the Visual Studio and select select File => New => Project from the menu. Select the "P3D_Template" template (Figure 1).

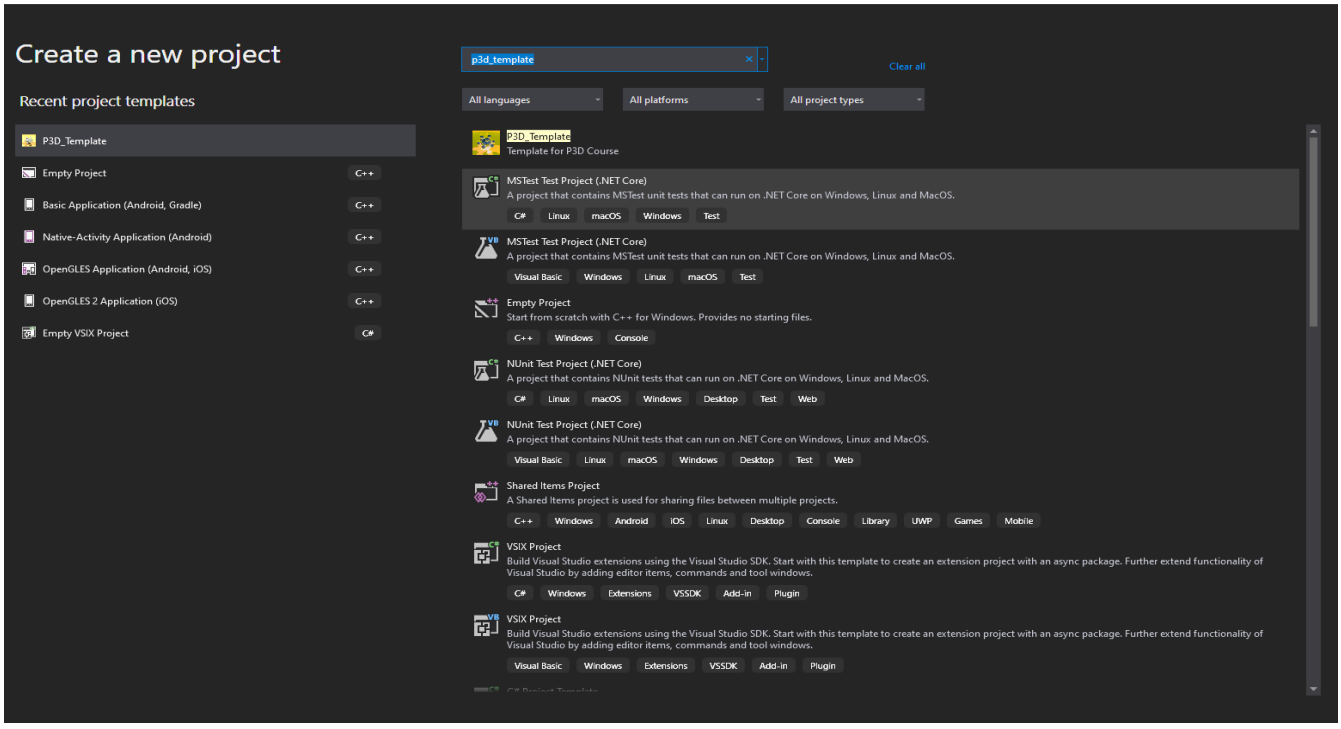

Figure 1 - Create a new project using the P3D_Template template.

Then click "Next" and call the project something like "MyRayTracer". After clicking "Create" it will create the new project.

Compiling and running

Select "x64" on the platform, and then right-click on the new "MyRayTracer" project in the Solution Explorer and select Build. With the first build it will set up Dependencies, which includes the **freeglut**, **glew** and **Devil** packages, within the project. The Dependencies files are in a self-extracting archive. It will pop up a window asking you where to put it (Figure 2). Simply click the Extract button.

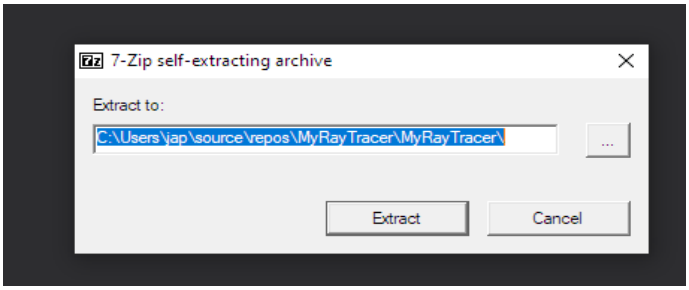

Figure 2: Click Extract to unpack the Angle files.

Then, select Debug => Start Debugging. The application will ask for a scene name. Provide any file from the P3D Scenes directory, for instance, **balls_low.p3f**. It will just draw blueish points in a window and it creates an output image file named RT_Output.png.

Now, you should change the renderscene() function in order to implement both scene->GetCamera()->PrimaryRay(pixel) and rayTracing (ray, 1, 1.0) functions.

The code of the template has already implemented several C++ classes like scene, camera, ray, geometric objects, material, lights, bounding box and more. It also provides some math functions, a P3F parser function, support for loading cubic textures (skybox) as well as getting the environment color from a light ray. You are free to use these features, to change and/or extend them.

I**MPORTANT**: this code comes with no guarantee, so use it at your own risk.

Using the balls_low.p3f scene test file, your algorithm should produce the following image: