Rasteriser Report

Aikaterini Baousi

University of Bristol, COMS30115
ab16651@my.bristol.ac.uk

May 11, 2017

Abstract

In this report the implementation and the extensions of the rasteriser project are thoroughly described. The extensions focus on providing solutions to decrease the time complexity of the project (eg. Breshenham, Parallel programming) and adding extra features in this implementation (e.g dynamic illumination). The code can be executed using the make command provided by the Makefile in the initial folder.

I. Implementation

This implementation takes a different approach in depicting 3D objects into 2D. Projections of the objects are used instead of raytracing.According to the distinct characteristics of the pinhole camera model , the 3D points of the triangles are represented in 2D using the theory behind similar triangles.VertexShader function is responsible for storing the projected 3D positions of the triangle in a 2D vector.

In order to create the lines between the projected points Interpolate function could be created and called inside DrawLineSDL function.Furthermore, the created surfaces should give a solid feeling and therefore it was necessary to color them row by row but it was very important to figure out the start and end point of all the created triangles before doing so. For that reason ComputePolygonRows function was implemented.DrawPolygon and DrawPolygonRows functions used the information produced by the previous function to draw the polygons(i.e set of triangles) row by row.A depth buffer was also constructed in order to prevent occlusion by drawing the pixels with depth smaller than the value stored in the depth buffer. Illumination was computed as a combination of direct and indirect light and was also affected by the other surfaces.Ultimately, as in raytracer a light source was added and could be controlled as the cameras position by the keyboard.

II. Extensions

The ways the aforementioned implementation got extended are the following:

i. Breshenham

The big advantage of this algorithm Breshenham's is that, it uses only integer calculations.The project implements Bresenham's Line Algorithm , replacing the usage of the naive interpolation used in the DrawPolygonRows method.

ii. Further illumination of the scene

Another extension of this project was to give the user the opportunity to dynamically increase and decrease the number of casted lights in the image.Function Illuminate() was created in that scope and is responsible for manipulating the light object created by class Light.This function increases the number of casted light by a small number every time the user presses "l" key. The number of lights decreases when the user presses "b" key.

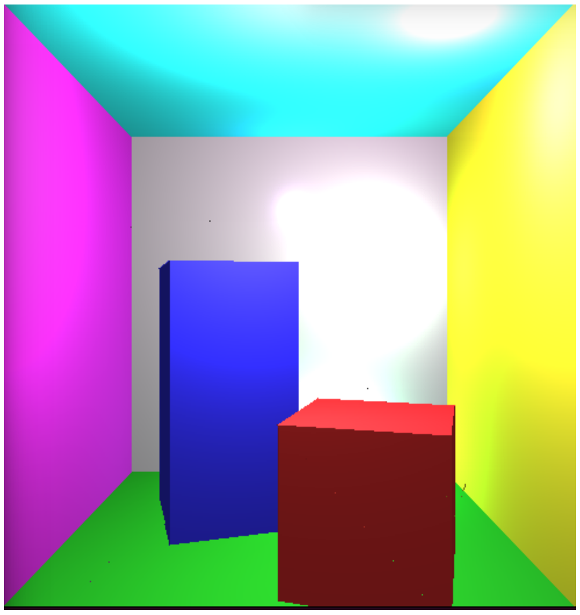

Figure 1: *Illuminated image.*

iii. Parallel Programing

OpenMP is an Application Program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism . It uses a thread-based parallelism model in which a master thread creates a team of threads in order to execute a specific region of a program. After the team threads complete the statements in the parallel region synchronize and terminate, only leaving the master thread active. For the purpose of this project, OpenMP was used to parallelize the intensive loops of the functions.

In the main function the maximum number of threads provided by the machine is used. All the workload created by the for loops of this implementation is spread across the number of the created threads. In addition the "nowait" clause is used because this implemention is embarassingly parallel as the procedures performed in each iteration are independent from each other.The overall time complexity of the implementation is decreased by this implementation.