Raytracer

Jason Haciepiri, Maria Marinova

Raytracing accurately computes a 2D representation of a 3D scene. It models the physics behind light interaction with surfaces by tracing rays of lights from a light source to the camera. However, this is not as efficient as rasterizing and hence not commonly used in real-time applications.

1. Compiling and running the code

The project includes a Makefile, which compiles the code. It contains the `-O3` flag in `CC_Opts` so that the code is compiled in the most optimised way. The `-lX11` flag is added to the linker options `LL_Opts` to allow communication with the X11 Windows Manager. The flag `-fopenmp` is added to the `CC` flags. The last two flags allow for parallelism in the code, which will be elaborated on in Section 3.

The project is compiled and run using the command:

```
make && .\Build\skeleton
```

2. Parts 1 and 2

Some summary of the parts up to 70%.

3. Extensions

Cramer's rule for inverting matrices

Using Cramer's rule instead of `glm::inverse` to invert matrices sped up the render time significantly. The rule is described below:

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \qquad A^{-1} = \frac{1}{det(A)} \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix}$$

$$A_{00} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \quad A_{01} = -\begin{vmatrix} a_{10} & a_{12} \\ a_{20} & a_{22} \end{vmatrix} \quad A_{02} = \begin{vmatrix} a_{10} & a_{11} \\ a_{20} & a_{21} \end{vmatrix}$$

$$A_{10} = -\begin{vmatrix} a_{01} & a_{02} \\ a_{21} & a_{22} \end{vmatrix} \quad A_{11} = \begin{vmatrix} a_{00} & a_{02} \\ a_{20} & a_{22} \end{vmatrix} \quad A_{12} = -\begin{vmatrix} a_{00} & a_{01} \\ a_{20} & a_{21} \end{vmatrix}$$

$$A_{20} = \begin{vmatrix} a_{01} & a_{02} \\ a_{11} & a_{12} \end{vmatrix} \quad A_{21} = -\begin{vmatrix} a_{00} & a_{02} \\ a_{10} & a_{12} \end{vmatrix} \quad A_{22} = \begin{vmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{vmatrix}$$

A check for $det(A) = 0$ is also added, in which case the inversion step is skipped.

Parallelisation via OpenMP

In addition to Cramer's rule, the render time was improved by parallelising loops with no data dependencies. This was done with means of OpenMP – `pragma omp parallel for` was used above all but one of the outermost `for` loops in the `Draw()` function.

As mentioned above, the `-lX11` is necessary to allow communication with the X11 Windows Manager. The flag `-fopenmp` and the header `#include <X11/Xlib.h>` were also needed to allow usage of OpenMP.

The current render time allows for the camera and light movement to be responsive even with a 500×500 screen.

Anti-aliasing

In order for an anti-aliasing effect to be achieved multiple rays – instead of just one – are shot for each pixel. The colours for each ray are averaged, and the result is the current pixel's colour.

As expected, the more rays per pixel, the slower the render time gets. After experimenting with different values, 9 ray per pixel seemed to give the best results in terms of gain vs. time efficiency.

Depth of field and Robert's Operator for edge detection

The next extension implemented is depth of field. A global variable `focus` was created. For each pixel in the 2D scene, the 3D depth of its respective intersection was preserved in the 2D array `float intersections[SCREEN_WIDTH][SCREEN_HEIGHT]`. Once the whole image is rendered, the intersection different from `focus`$\pm$`slack` are blurred. In the submitted version $focus = 0.004f$ `and` $slack = 0.001f$. The blurring is achieved via a 5×5 Gaussian filter. If the camera is moved, the pixel in focus change respectively.

As the objects in the scene lack texture, the effect is not always easily noticeable. To ensure it is functioning, Robert's Operator for edge detection was implemented. The resulting image is produced side by side with the original and also responds to camera movements. The scene is first represented in grayscale. The Operator is then applied as follows: $\nabla f = |f(x,y) - f(x+1,y+1)| + |f(x,y+1) - f(x+1,y)|$, which is derived from:

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Multiple light sources

References