Computer Graphics Coursework Report

Zl16533 yp16505

Rasterizer

Different from ray tracer, rasterizer is much quicker since it did not simulate the real world, but used the mathematical method to calculate the relationship between objects and their projection on the screen.

The vertices of all the triangles were projected onto the screen in the first step. Then the edges of the triangles were drawn by linking the vertices.

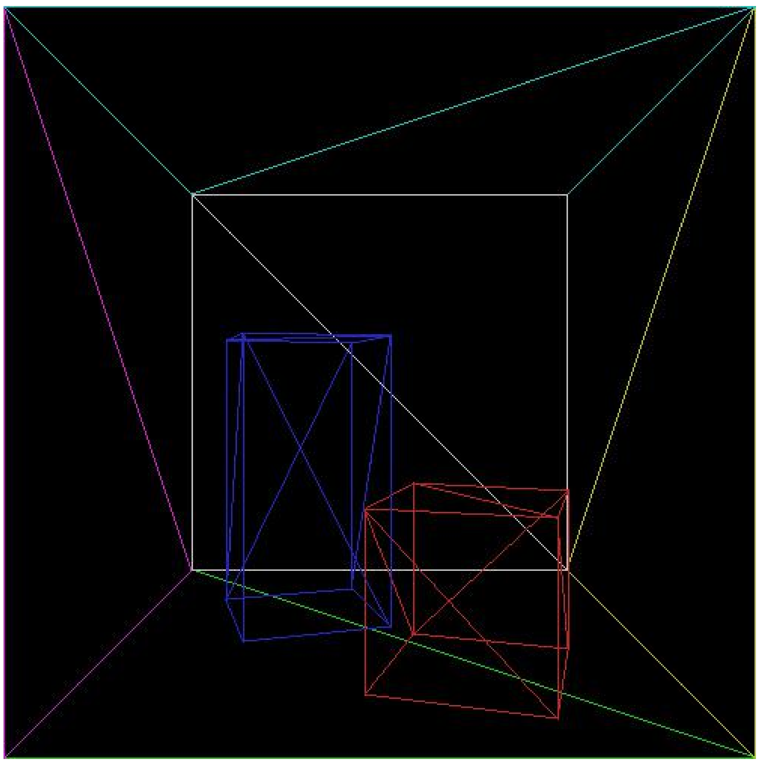

Figure 1

The pixels between each two edges would be filled with the color of the triangle.

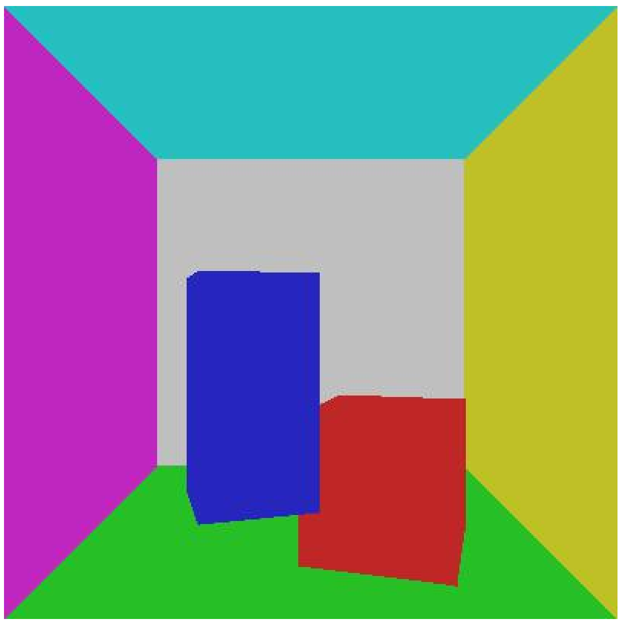
Figure 2

As shown in Figure 2, there is no depth information, so what we did next is to use the value of vertices on z-axis to calculate the depth of its projection and decide which triangle is in the front.

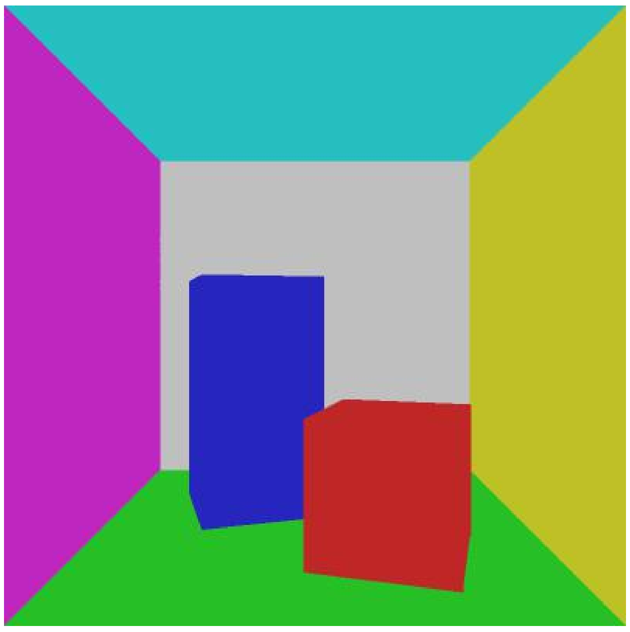
Figure 3

Then the light was added into the scene. Also different from the ray tracer, each of the pixel was projected back to the 3D-space, the power of light was calculate in 3D space and the value was added onto the pixel. The effect was shown in Figure 4.

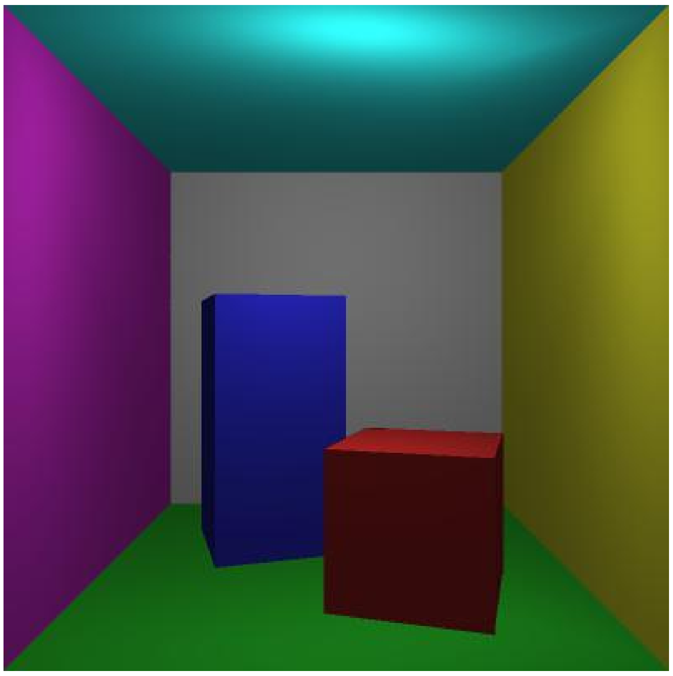

Figure 4

However, the light looks a bit skew cause the continuous pixels on the screen were not the projection of continuous points in 3D space. But the light will affect only continuous points in 3D space, which means that light cannot be calculated simply on the screen. The pixels need to be projected back to the 3D space and calculate the light power in the space. After we applied this process, the light now looks not skew anymore.

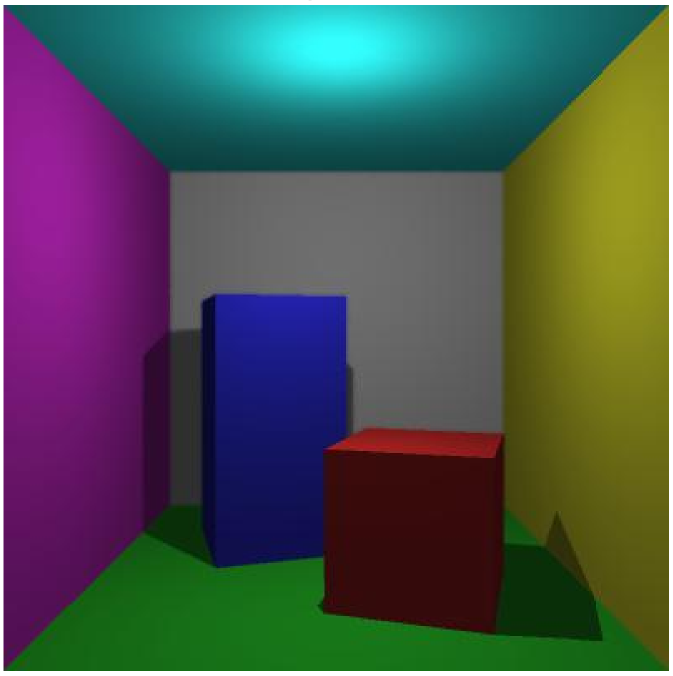

Figure 5

The control of camera and light were designed, which enabled the camera and light to move in the scene.

Extension

Shadow

The way how shadow was added was very like to the way we did in the ray tracer part. The information of shadow (whether there is shadow or not, the value of the shadow) was calculated in the 3D space and then projected onto the screen

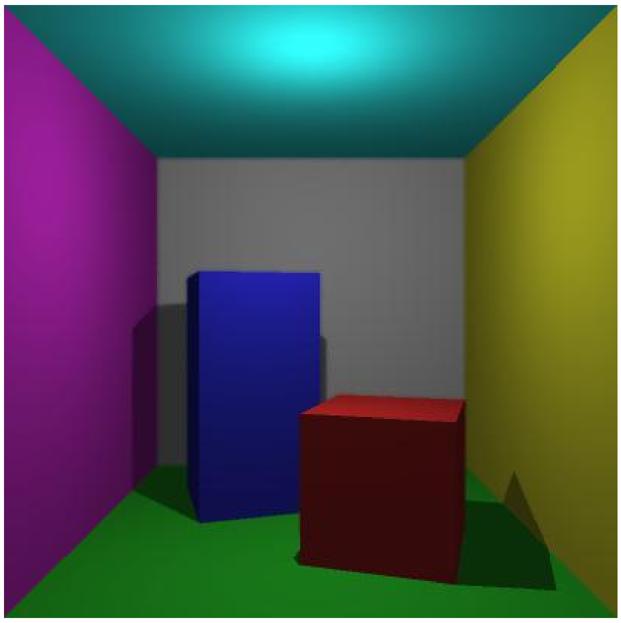

Figure 6

Anti-aliasing

Just like how soft edge worked, we still double the screen and combine four neighborhood pixels into one pixel.

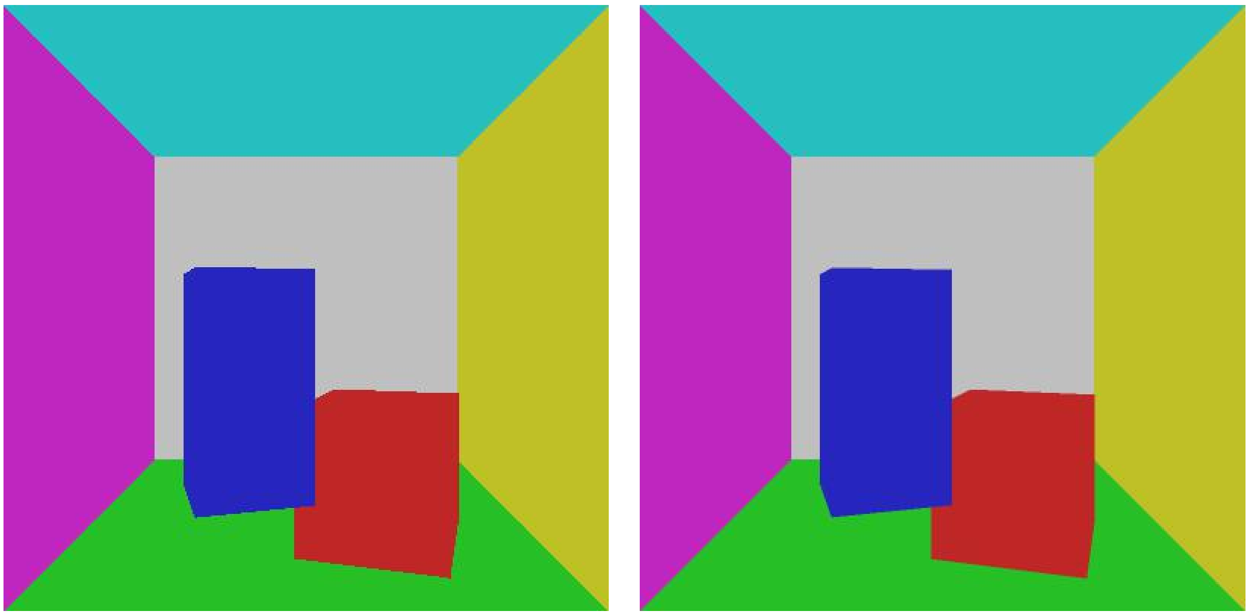

Figure 7

Depth of field

Since the value of the depth of each pixel had been stored, we could add the effect of depth just on the screen. A focus depth was set, any pixel which depth is larger than the focus depth will be combined by several neighborhood pixels. Pixels with large depth will combine far neighborhood pixels, which makes them more blurry. Pixels with small depth will combine close pixels and therefore they are clear.

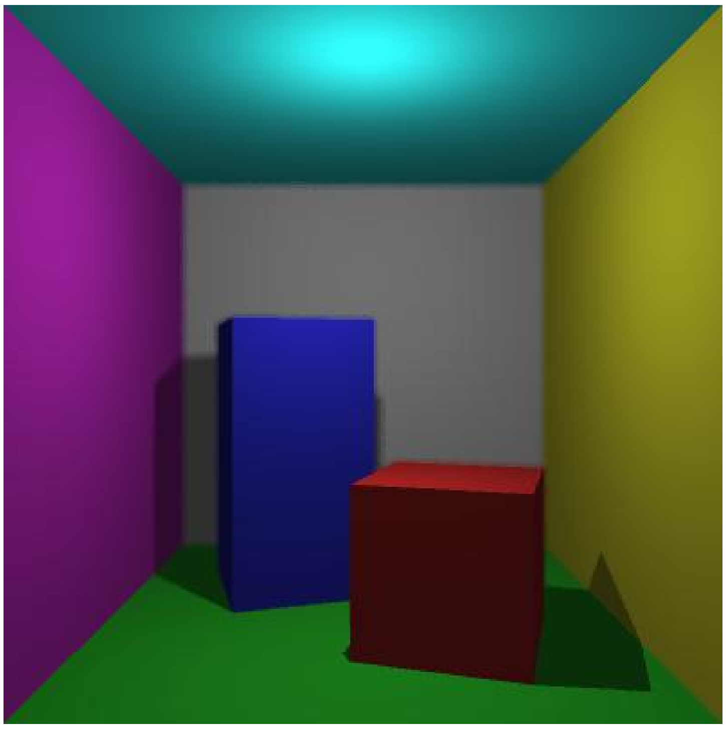

Figure 8

Clipping

When the camera moved, the objects may move out of the screen, and these pixels which is not on the screen should not be displayed. So, we added a function to check all the projection of the objects, if the projection is beyond the boundary of the screen, it should not be calculated or rendered.

Parallel Rendering

Threads were also used for parallel rendering. This time, objects were divided into six group and each thread will deal one of the group. Rendering a 400x400 image without parallel thread would cost 10 seconds and rendering it with parallel threads only take 5 seconds. The effect of increasing efficiency is distinct.

Apart from that, another measure was used to increase the render speed. When we are rendering shadow, we notice that when the position of light source is higher than the tall block, the position of shadow will be lower than the tall block. Therefore, in this case we do not need to compute shadow for pixels that are higher than the tall block, given that there is no shadow at there. after using this measure, the render time has been decreased from 5 seconds to about 3 seconds.