Estudo e implementação de um acelerador gráfico para apresentação de uma imagem 3D projetada em uma tela 2D

Ruan Petrus Alves Leite [1], **Prof. Dr. Marcus Vinicius Lamar** [1]

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Brasília – DF – Brasil

leite.ruan@aluno.unb.br, lamar@unb.br

Resumo. *O processo de desenho e projeção de objetos em 3D é bastante custoso quando feito em software. Este artigo tem como objetivo estudar, projetar e implementar um acelerador gráfico em hardware para o desenho de objetos 3D. O acelerador foi criado para ser utilizado em conjunto com um processador RISC-V. Foi decidido uma arquitetura de três módulos para o acelerador, um módulo para comunicação com o processador, um módulo de transformações geométricas e projeção, e um módulo de rasterização. O sistema de comunicação utiliza uma região de memória compartilhada com o processador, as transformações geométricas são feitas utilizando multiplicação de matrizes e coordenadas homogêneas, e o circuito de rasterização utiliza um algoritmo de Bounding Box para o desenho e Zbuffer para a correta renderização em profundidade. Os testes demonstram uma melhoria significativa comparada a antiga versão em software.*
Palavras-Chave *RISC-V, SystemVerilog, FPGA, Acelerador, Rasterização, Zbuffer.*

1. Introdução

O processo de desenho e projeção de objetos em 3D é bastante custoso quando feito em software. Este projeto tem como objetivo estudar, projetar e implementar um acelerador gráfico, em hardware, para o desenho de objetos 3D. O acelerador foi criado para ser utilizado em conjunto com um processador RISC-V. O acelerador será escrito em SystemVerilog e implementado em uma FPGA.

2. Metodologia

A arquitetura do acelerador foi baseada no artigo [G. et al. 2016]. É uma arquitetura de três módulos, um módulo para a comunicação com o processador, um módulo de transformações geométricas e projeção, e um módulo de rasterização.

Objetos 3D serão tratados como um conjunto de triângulos, todas as operações serão feitas individualmente em cada um dos seus triângulos.

2.1. Módulo de comunicação

O sistema de comunicação funcionaria através de uma memória compartilhada onde são passados triângulos, matrizes de transformação e flags de processamento entre os módulos.

2.2. Transformações geométricas

Utilizando coordenadas homogêneas e matrizes 4x4, consegue-se aplicar qualquer conjunto de transformações lineares no triângulo. Para isso multiplica-se a matriz de transformação por cada um dos pontos do triângulo. Por exemplo seja A a matriz de transformação e P um ponto a do triângulo temos:

$$A = \begin{Bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

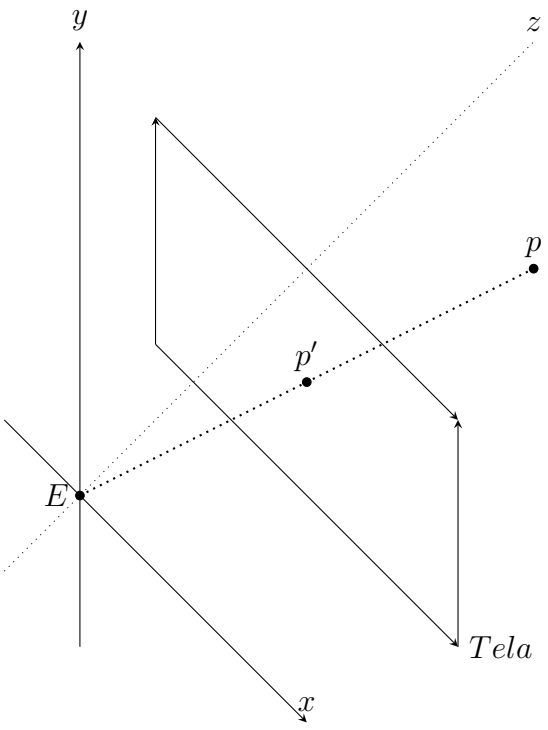
$$P = \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$A \cdot P = \begin{Bmatrix} x \cdot a_x + y \cdot b_x + z \cdot c_x + d_x \\ x \cdot a_y + y \cdot b_y + z \cdot c_y + d_y \\ x \cdot a_z + y \cdot b_z + z \cdot c_z + d_z \\ 1 \end{Bmatrix}$$

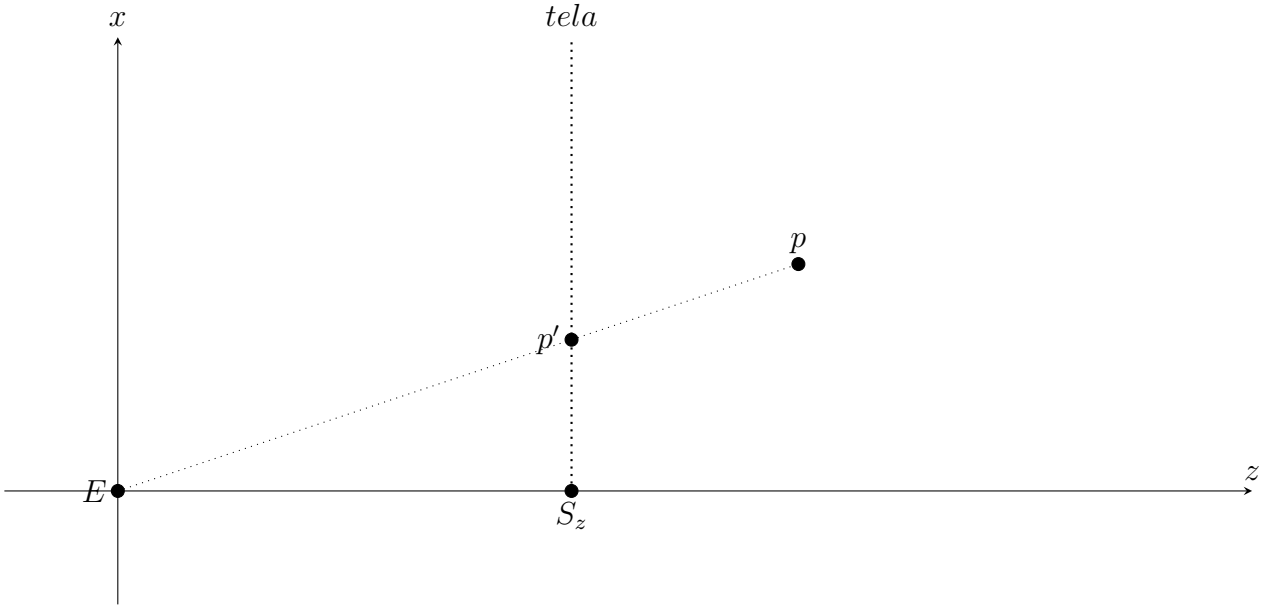
Dessa forma é possível aplicar operações como rotação, translação e escalamento.

2.3. Projeção

Podemos projetar um triângulo projetando os seus pontos individualmente. A projeção de um ponto pode ser visto no diagrama abaixo, onde E são os olhos do observador, p o ponto a ser projetado e p' a projeção do ponto na tela.



Observando o diagrama de cima temos:



Sendo S_z como a coordenada z da tela, temos:

$$
\begin{aligned}
p &= (p_x, p_y, p_z) \\
E &= (E_x, E_y, E_z) \\
p' &= (p'_x, p'_y, p'_z) \\
p'_y &= \frac{(p_y - E_y)(S_z - E_z)}{(p_z - E_z)} + E_y \\
p'_x &= \frac{(p_x - E_x)(S_z - E_z)}{(p_z - E_z)} + E_x
\end{aligned}
$$

Considerando $E = (0, 0, 0)$ e $S_z = 1$

$$
\begin{aligned}
p'_y &= \frac{p_y}{p_z} \\
p'_x &= \frac{p_x}{p_z}
\end{aligned}
$$

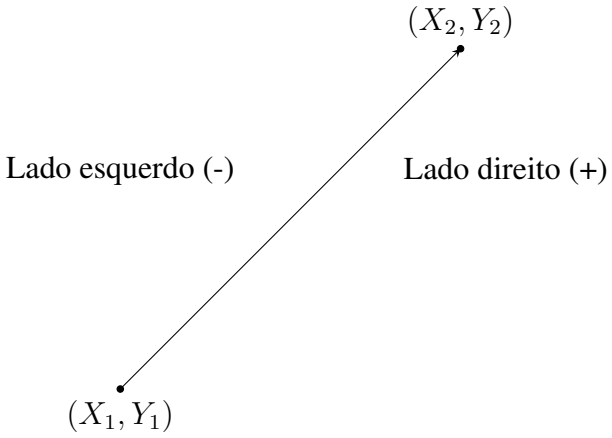
2.4. Rasterizador

O algoritmo utilizado para a rasterização é o algoritmo proposto por [Pineda 1988] "A parallel algorithm for polygon rasterization". Nele, Juan define a função da aresta $E(x, y)$.

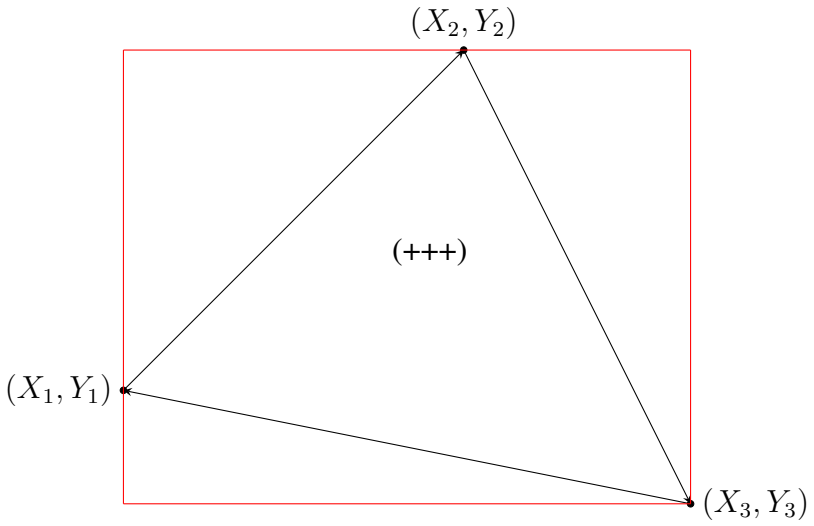
$$E(x, y) = (x - X_1)(Y_2 - Y_1) - (y - Y_1)(X_2 - X_1)$$

Considerando os vértices do triângulo (X_1, Y_1), (X_2, Y_2) no sentido horário.

$$\begin{cases} E(x,y) > 0 \text{ se } (x,y) \text{ está à direita da aresta} \\ E(x,y) = 0 \text{ se (x, y) está exatamente na aresta} \\ E(x,y) < 0 \text{ se (x, y) está à esquerda da aresta} \end{cases}$$



Dessa forma é criado um retângulo ao redor do triângulo, chamado Bounding Box, e para cada ponto (x, y) do retângulo é testado se $E(x, y) >= 0$ na função das três arestas, caso seja verdade então o ponto é desenhado.



Podemos calcular o valor da equação para um ponto inicial e as seguintes taxas de variação:

$$E(x+1, y) = E(x, y) + (Y2 - Y1)$$
$$E(x, y+1) = E(x, y) - (X2 - X1)$$

Utilizando as taxas de variação podemos fazer apenas três somas durante o loop da Bounding Box, uma para cada função da aresta.

2.4.1. Zbuffer

O Zbuffer é necessário para que os triângulos possam ser desenhados de forma correta em relação à profundidade.

O Zbuffer consiste em guardar para cada pixel da tela o valor do Z do ponto mais próximo que foi desenhado. Desta forma, mesmo que um pixel esteja dentro de um triângulo, se um pixel nas mesmas coordenadas já foi desenhado com um Z menor então o pixel não é desenhado.

É necessário para cada ponto (x_p, y_p) na tela, que o triângulo cobre, descobrir a coordenada z_p.

Será utilizado as coordenadas baricêntricas w_0, w_l, w_2, de tal forma que:

$$\frac{1}{z_p} = \frac{1}{z_0} \cdot w_0 + \frac{1}{z_1} \cdot w_1 + \frac{1}{z_2} \cdot w_2$$

As coordenadas baricêntricas podem ser vistas como a taxa de contribuição que cada vértice tem em um ponto.

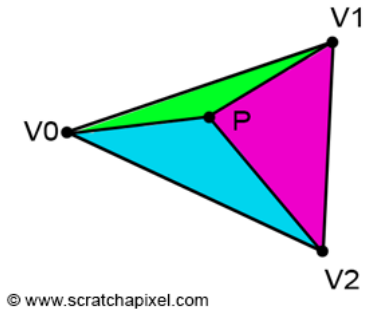


Figura 1. Coordenadas baricêntricas, https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/rasterization-stage.html, (Licenciado sobre CC BY-NC-ND 4.0) creativecommons.org/licenses/by-nc-nd/4.0/

Como descrito em [scr], em um triângulo com vértices v_0 v_1 e v_2, a coordenada baricêntrica de um vértice v dado um ponto p, é a área do triângulo gerada por p e os vértices opostos a v sobre a área do triângulo formada pelos 3 vértices v_0, v_1, v_2. Na imagem a cima w_0 seria a área do triângulo roxo sobre a área total. Sendo E_{AB} a função da aresta AB, tem-se que:

$$
\begin{aligned}
w_0(p_x, p_y) &= \frac{E_{12}(p_x, p_y)}{E_{01}(X_2, Y_2)} \\
w_1(p_x, p_y) &= \frac{E_{20}(p_x, p_y)}{E_{01}(X_2, Y_2)} \\
w_2(p_x, p_y) &= \frac{E_{01}(p_x, p_y)}{E_{01}(X_2, Y_2)} \\
w_0(p_x, p_y) &+ w_1(p_x, p_y) + w_2(p_x, p_y) = 1
\end{aligned}
$$

Podemos calcular o valor da $\frac{1}{z}$ para um ponto inicial e as seguintes taxas de variação:

$$
\begin{aligned}
dz_{10} &= \frac{1}{z_1} - \frac{1}{z_0} \\
dz_{20} &= \frac{1}{z_2} - \frac{1}{z_0} \\
\frac{1}{z_p}(p_x, p_y) &= \frac{1}{z_0} w_0(p_x, p_y) + \frac{1}{z_1} w_1(p_x, p_y) + \frac{1}{z_2} w_2(p_x, p_y) \\
\frac{1}{z_p}(p_x, p_y) &= \frac{1}{z_0}(1 - w_1(p_x, p_y) - w_2(p_x, p_y)) + \frac{1}{z_1} w_1(p_x, p_y) + \frac{1}{z_2} w_2(p_x, p_y) \\
\frac{1}{z_p}(p_x, p_y) &= \frac{1}{z_0} + dz_{10} w_1(p_x, p_y) + dz_{20} w_2(p_x, p_y) \\
\frac{1}{z_p}(p_x, p_y) &= \frac{1}{z_0} + dz_{10} \frac{E_{20}(p_x, p_y)}{E_{01}(X_2, Y_2)} + dz_{20} \frac{E_{01}(p_x, p_y)}{E_{01}(X_2, Y_2)} \\
\frac{1}{z_p}(p_x + 1, p_y) &= \frac{1}{z_0} + dz_{10} \frac{E_{20}(p_x, p_y) + Y_2 - Y_0}{E_{01}(X_2, Y_2)} + dz_{20} \frac{E_{01}(p_x, p_y) + Y_0 - Y_1}{E_{01}(X_2, Y_2)} \\
\frac{1}{z_p}(p_x + 1, p_y) &= \frac{1}{z_p}(p_x, p_y) + \frac{dz_{10}(Y_2 - Y_0)}{E_{01}(X_2, Y_2)} + \frac{dz_{20}(Y_0 - Y_1)}{E_{01}(X_2, Y_2)} \\
\frac{1}{z_p}(p_x, p_y + 1) &= \frac{1}{z_p}(p_x, p_y) + \frac{-dz_{10}(X_2 - X_0)}{E_{01}(X_2, Y_2)} + \frac{-dz_{20}(X_0 - X_1)}{E_{01}(X_2, Y_2)}
\end{aligned}
$$

Utilizando as taxas de variação podemos fazer apenas seis somas durante o loop da Bounding Box, duas para cada coordenada baricêntrica.

3. Resultados

Códigos do circuito e de suas simulações em software podem ser encontrados no link https://github.com/RuanPetrus/gpu.

3.1. Módulo de comunicação

O módulo de comunicação não foi implementado em hardware, sendo apenas feitas simulações utilizando threads em C.

3.2. Módulo de transformação geométrica e projeção

O módulo de transformações geométricas foi implementado como uma multiplicação de uma matriz 4x4 por um vetor 4x1. A multiplicação é dividida em 3 operações lineares executadas em paralelo, que por sua vez utiliza 1 multiplicação e 2 somas de pontos flutuantes sequencialmente. No caso da FPGA escolhida o circuito requer 5 ciclos.

3.3. Módulo de rasterização

Para o módulo de rasterização de triângulos, precalculou-se as funções das arestas para o ponto mais acima à esquerda da bounding box, e as variações das funções quando a coordenada x do ponto aumenta em 1 e o mesmo para a coordenada y. Com isso, o circuito requer $5 + 2 * A$ ciclos, onde A é a área do triângulo a ser desenhado.

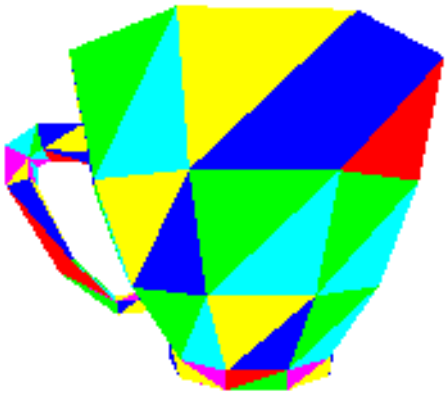


Figura 2. Circuito simulado por software renderizando uma xícara

4. Conclusão

O projeto se propôs a implementar um acelerador gráfico entregando o circuito que desenha o triângulo, e o módulo de transformações geométricas. Os dois são uma melhoria significativa em velocidade comparada à sua versão em software, e aumentariam a velocidade de renderização significativamente.

Para a continuação do projeto é necessária a implementação do módulo de comunicação, texturas e coordenadas normais para que seja possível renderizar elementos como luz e sombra.

O método de rasterização utilizado possui uma propriedade de que qualquer característica baseada nas coordenadas (x, y) do ponto e que pode ser calculada como um valor inicial e uma taxa de variação pode ser adicionada ao ao processo de renderização sem nenhuma penalidade de performance. Dessa forma, diversas propriedades futuras podem ser implementadas sem penalidades de performance.

Referências

Rasterization. https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/rasterization-stage.html. Accessed: 2024-10-30.

G., R. S., B., P. K., Ananda, C. M., e P., J. E. (2016). Design of graphics processing framework on fpga. *IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016, India*.

Pineda, J. (1988). A parallel algorithm for polygon rasterization. *ACM SIGGRAPH Computer Graphics*.