

# Ray Tracing

**ESTRUTURA DE DADOS – UFPB – 2014.2**

---

# OBJETIVOS

- Realizar o processo de criação e traçamento de raios (Raytracer) usando a linguagem C++;
- Realizar o processo de raytracing em tempo menor ao arquivo original;
- Utilizar o sistema de tabela hash no programa;



# ETAPAS

- Para a criação do trabalho foi necessário três etapas:
    1. Criação de um Grid (Tela) para adicionar os triângulos da imagem
      - Criar cálculos para implementar uma bounding box de cada triângulo na imagem.
      - Adicionar valores máximos e mínimos em x , y e z em cada célula da Tela.
      - Testar impacto dos raios com cada célula do Grid.
    2. Tratamento dos raios através da classe “Ray.h”
      - Utilizar a origem e a direção dos raios, para desmistificar os raios e assim melhorar a performance do nosso sistema.
    3. Substituição do Grid por uma tabela hash
      - Por meio de uma função hash, o grid foi substituído por uma tabela hash.
-

# ETAPAS DE CRIAÇÃO





# TENTATIVAS

- Nas primeiras tentativas, tentamos passar os valores dos vértices para uma função onde a mesma transformaria o valor do vértice, mas foi notado que após realizar esse processo com todos os vértices, o código se tornaria muito extenso e confuso.

```
int calcula_xmin (float max, float min, float h)
{
    int x = (int) (((h - min) * (res) / (max - min))) / res;
    return x;
}
int calcula_ymin (float max, float min, float h)
{
    int y = (int) (((h - min) * (res) / (max - min))) / res;
    return y;
}
int calcula_zmin (float max, float min, float h)
{
    int z = (int) (((h - min) * (res) / (max - min))) / res;
    return z;
}
int calcula_xmax (float min, float max, float h)
{
    int x = (int) (((h - min) * (res) / (max - min))) / res;
    return x;
}
int calcula_ymax (float min, float max, float h)
{
    int y = (int) (((h - min) * (res) / (max - min))) / res;
    return y;
}
int calcula_zmax (float min, float max, float h)
{
    int z = (int) (((h - min) * (res) / (max - min))) / res;
    return z;
}
```

# GRID

- Na etapa do grid foi necessário a criação do próprio, utilizando um array tridimensional de Triângulos.
- Após isso calculamos os máximos e mínimos de cada triângulo
- Adicionar do mínimo valor até o máximo valor no grid.

Utilizamos a formula:

$$Final\ Triangles = abs\left(\frac{valor_{vertice[w*]} - min[w*] * (resolution - 1)}{max[w*] - min[w*]}\right)$$

$w^*$  = qualquer variável no plano do GRID



# RAIO

- Para produzir o deslocamento do raio, foi necessário igualar ao deslocamento dos triângulos para não deslocar a imagem.
- Logo após, calculamos os pontos onde o raio sai do plano.
- Com os pontos de saída calculados e com área do bounding box fazendo uso das coordenadas x, y e z.

Fórmulas usada para o tratamento dos raios

$$\text{Ray}_{\max} = (\max[w^*] - R_{\text{origin}}) / R_{\text{direction}}$$

$$\text{Ray}_{\min} = (\min[w^*] - R_{\text{origin}}) / R_{\text{direction}}$$

# HASH

- Na etapa final, substituímos o grid (tridimensional) por uma tabela hash (unidimensional), e adicionamos a seguinte função de hash

$$hash = N * (((X + Y + Z) * A) \% 1)$$

Onde:

$$N = 1297;$$

$$A \cong 0.6108$$

X,Y,Z = coordenadas da bounding box



# COMPARAÇÃO

Desempenho em segundos

