Ray Tracing

ESTRUTURA DE DADOS – UFPB – 2014.2

Abraão Állysson dos Santos Honório - Graduando em Engenharia de computação

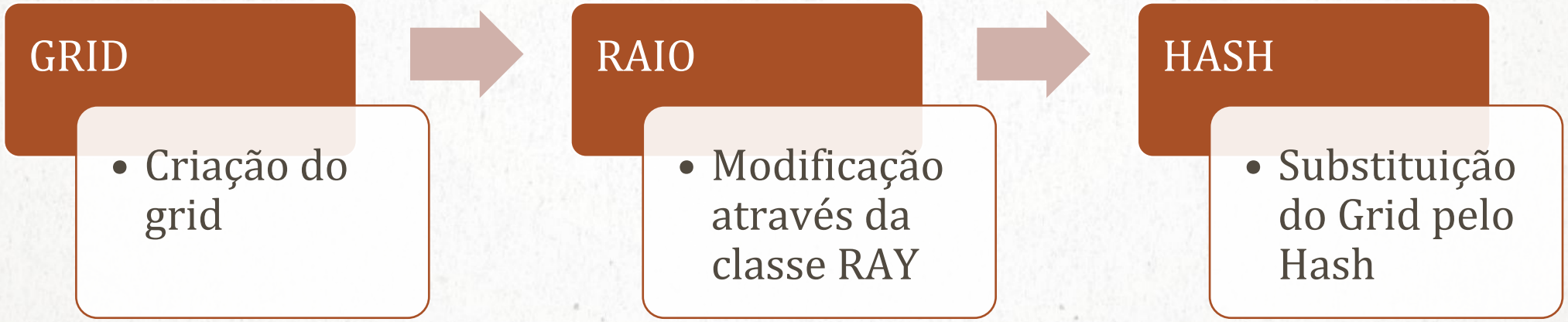
OBJETIVOS

- Realizar o processor de criação e traçamento de raios (Raytracer) usando a linguagem C++;
- Realizar o processor 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 dinamificar os raios e assim melhorar a performance do nosso sistema.
  3. Substituição do Grid por uma tabela hash
     - Através de uma função hash, trocaremos o grid por uma tabela hash.

APLICAÇÃO DAS ETAPAS



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 grande 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:

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

w* = qualquer variável no plano do GRID

RAIO

- Para fazer o raio deslocar o raio igual ao deslocamento dos triângulos, para não deslocar a imagem.
- Logo após, calculamos os pontos onde o raio sai do plano
- pegando seus pontos de saída já calculados e fazendo a área do bounding box utilizando as coordenadas em x , y e z.

Fórmula para o tratamento do raio

$$Ray_{max} = (\max[w*] - R_{origin}) \,/\, R\_direction$$
$$Ray_{min} = (min[w*] - R_{origin}) \,/\, R\_direction$$

HASH

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

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

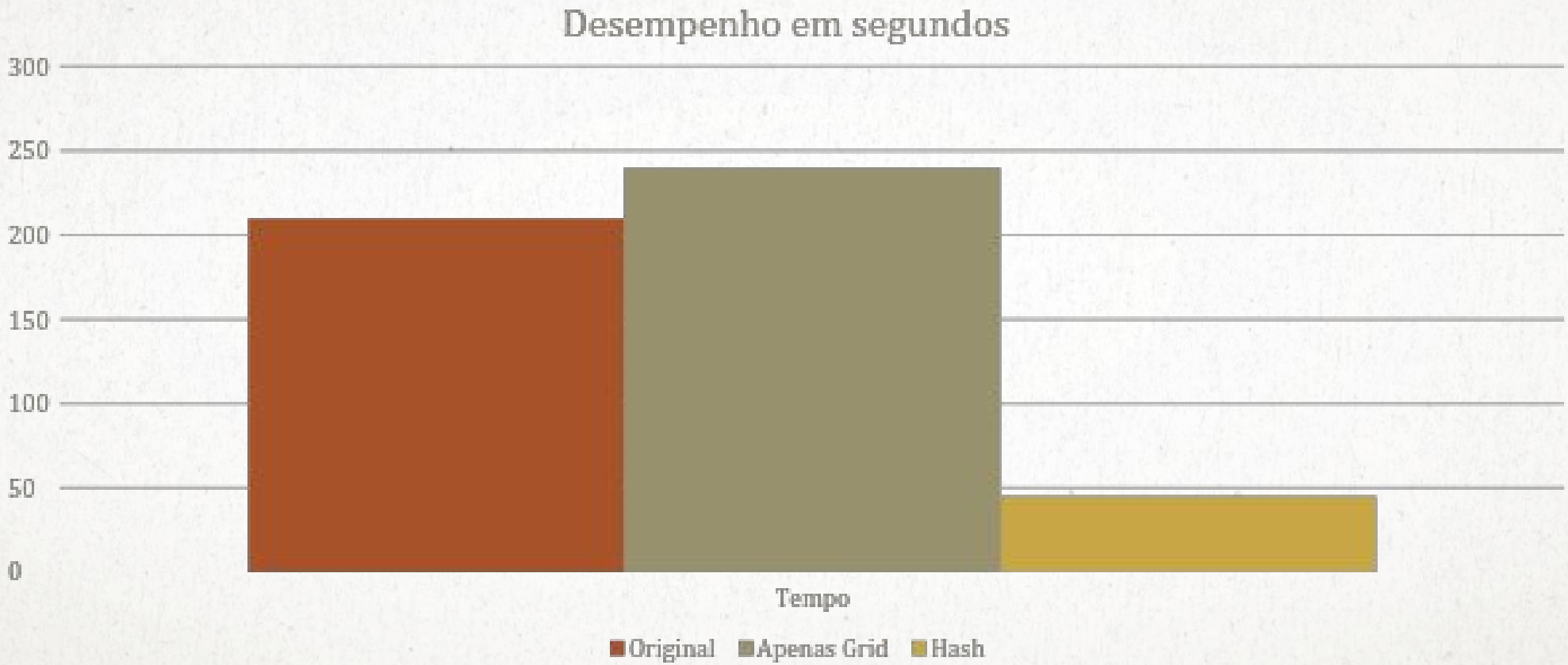
Onde:

N = 1297;

A ≅ 0.6108

X,Y,Z = coordenadas da bounding box

COMPARAÇÃO


Desempenho em segundos
300
250
200
150
100
50
0
Tempo
Original
Apenas Grid
Hash