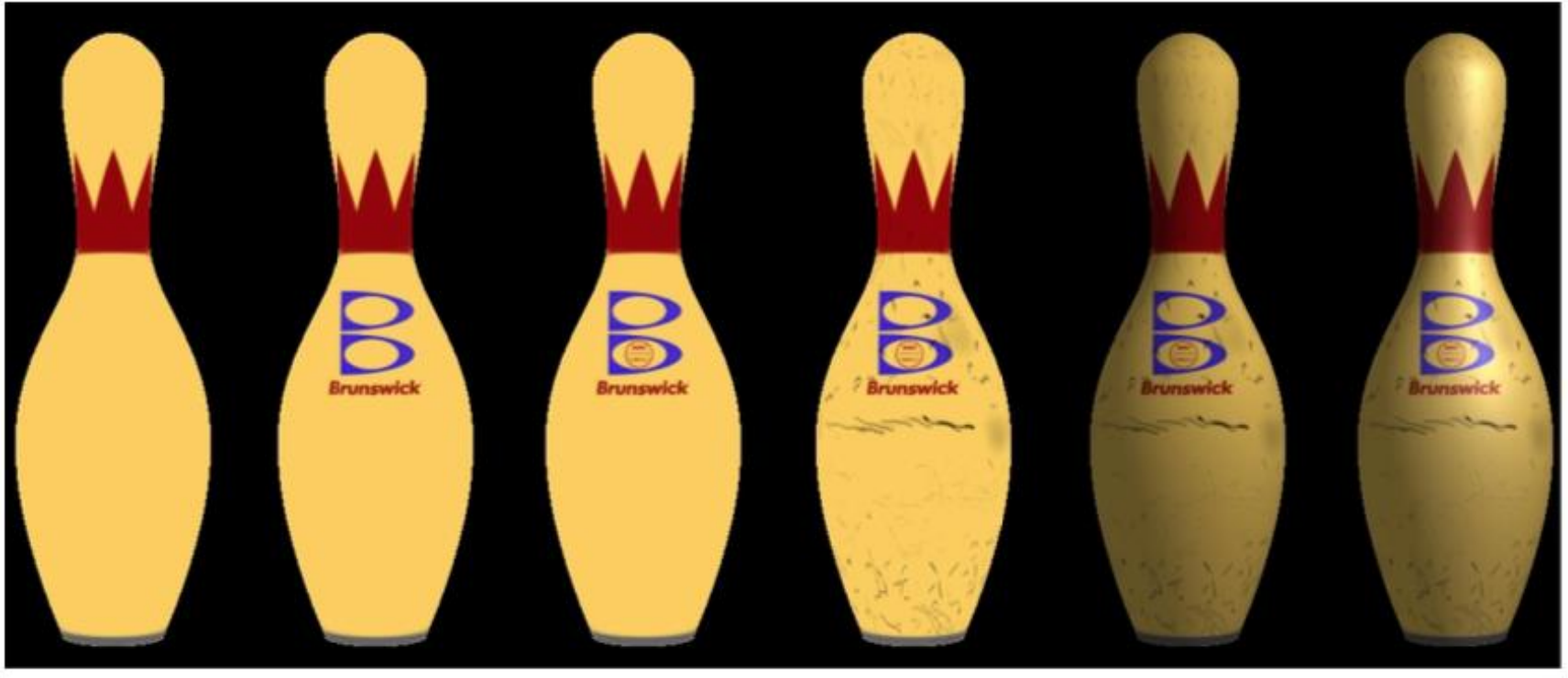
Insper

Computação Gráfica

Aula 09: Mapeamento de Texturas

Mapeamento de Texturas

Realismos das Superfícies



- Adiciona detalhes sem aumentar a complexidade geométrica
- Se aplica uma imagem na geometria ou se pinta proceduralmente

Texturas são só um dos efeitos possíveis

Texel

Um **texel** (texture element) é a unidade fundamental de uma textura.

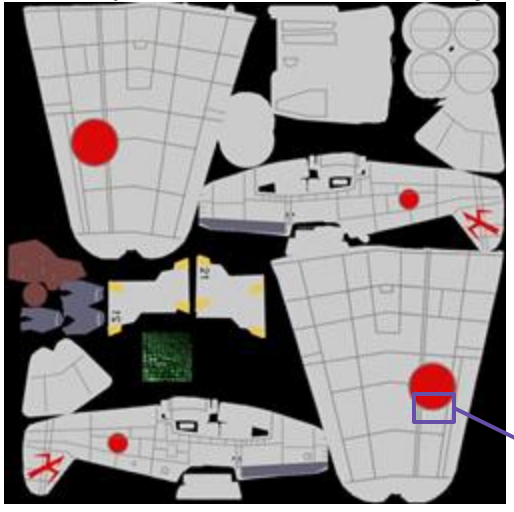
No processo de renderização existe um mapeamento do **texel** para o pixel apropriado na imagem final.

Fonte: Ultima Underworld: The Stygian Abyss

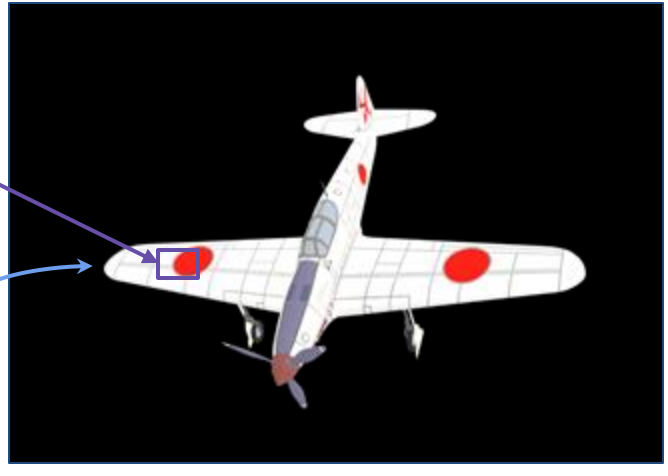
O que é texturizar

Geometria (composta por triângulos)

Textura (originária por exemplo de um JPG ou PNG)



vértices possuem suas coordenadas no espaço, mas também coordenadas de mapeamento de textura



Na renderização, para definir a cor do pixel de um objeto texturizado é necessário identificar e calcular a cor pela textura

Identificando o Texel Correspondente

Usualmente temos as coordenadas de Textura normalizadas (variando de 0 a 1).
Para encontrar a coordenada real no bitmap temos de multiplicar pela resolução.

$U_{real} = U_{norm} \times LarguraImage$

$V_{real} = V_{norm} \times AlturaImage$

Exemplo:

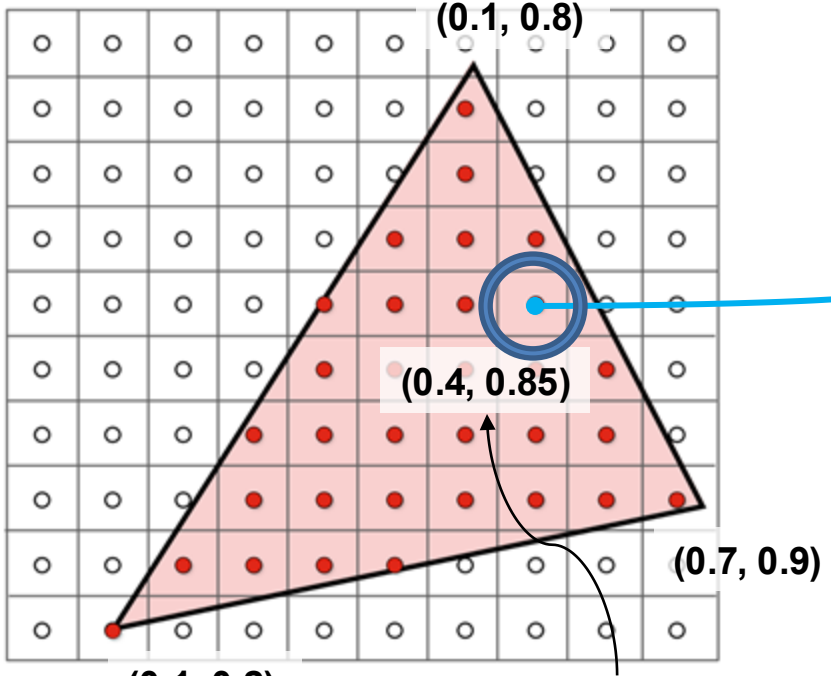
$Coord_{norm} = (0.34; 0.71)$

$U_{real} = 0.34 \times 20 = 6,8$
$V_{real} = 0.72 \times 15 = 10,8$

Arredondando (*floor*) = (6; 10)

Amostrando Texturas (pontuais)

Na rasterização amostramos o pixel



Cada vértice possui uma coordenada de textura (u,v)

Quando amostramos o pixel descobrimos suas coordenadas baricêntricas

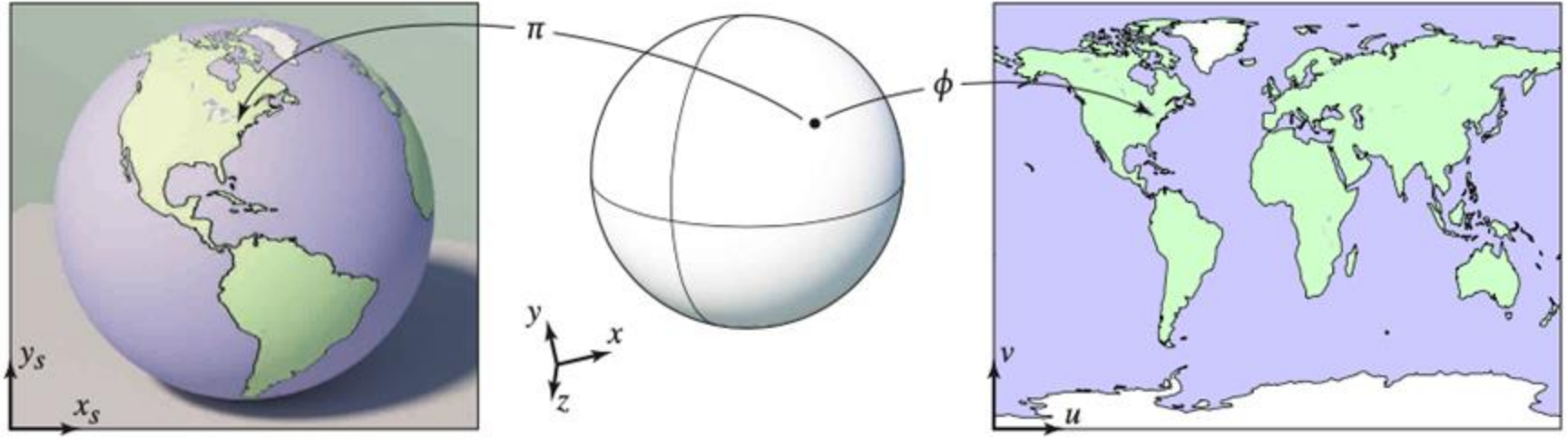
Podemos então descobrir as coordenadas de textura (u, v), selecionar a cor e aplicar no pixel

Insper

Três Espaços

As superfícies habitam o espaço 3D do mundo

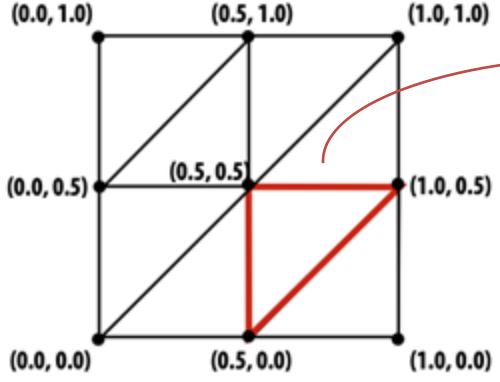
Cada ponto das superfícies 3D também tem uma posição onde fica na tela 2D e na textura 2D.



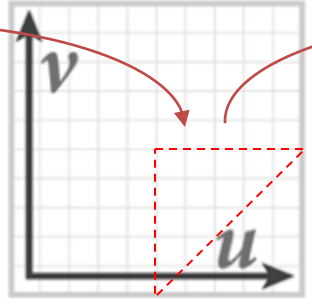
Espaço da Tela | **Espaço do Mundo** | **Espaço da Textura**

Coordenadas de Textura

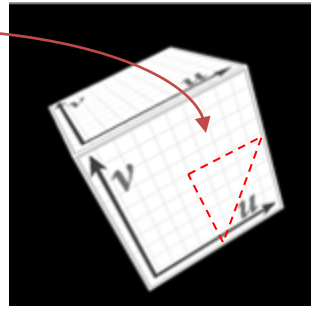
As coordenadas de Textura definem um mapeamento das coordenadas nas superfícies (pontos 3D no triângulo) para coordenadas de textura (pontos 2D na imagem).



Oito triângulos (uma face de um cubo) com parâmetros de superfície fornecida como coordenadas de textura por vértice.



As coordenadas de textura são definidas num espaço normalizado $[0,1]^2$. Localização do triângulo destacado no espaço de textura mostrado em vermelho.



Resultado final renderizado (cubo inteiro mostrado). Localização do triângulo após a projeção na tela mostrada em vermelho.

Insper

Mapeamento das coordenadas de textura.

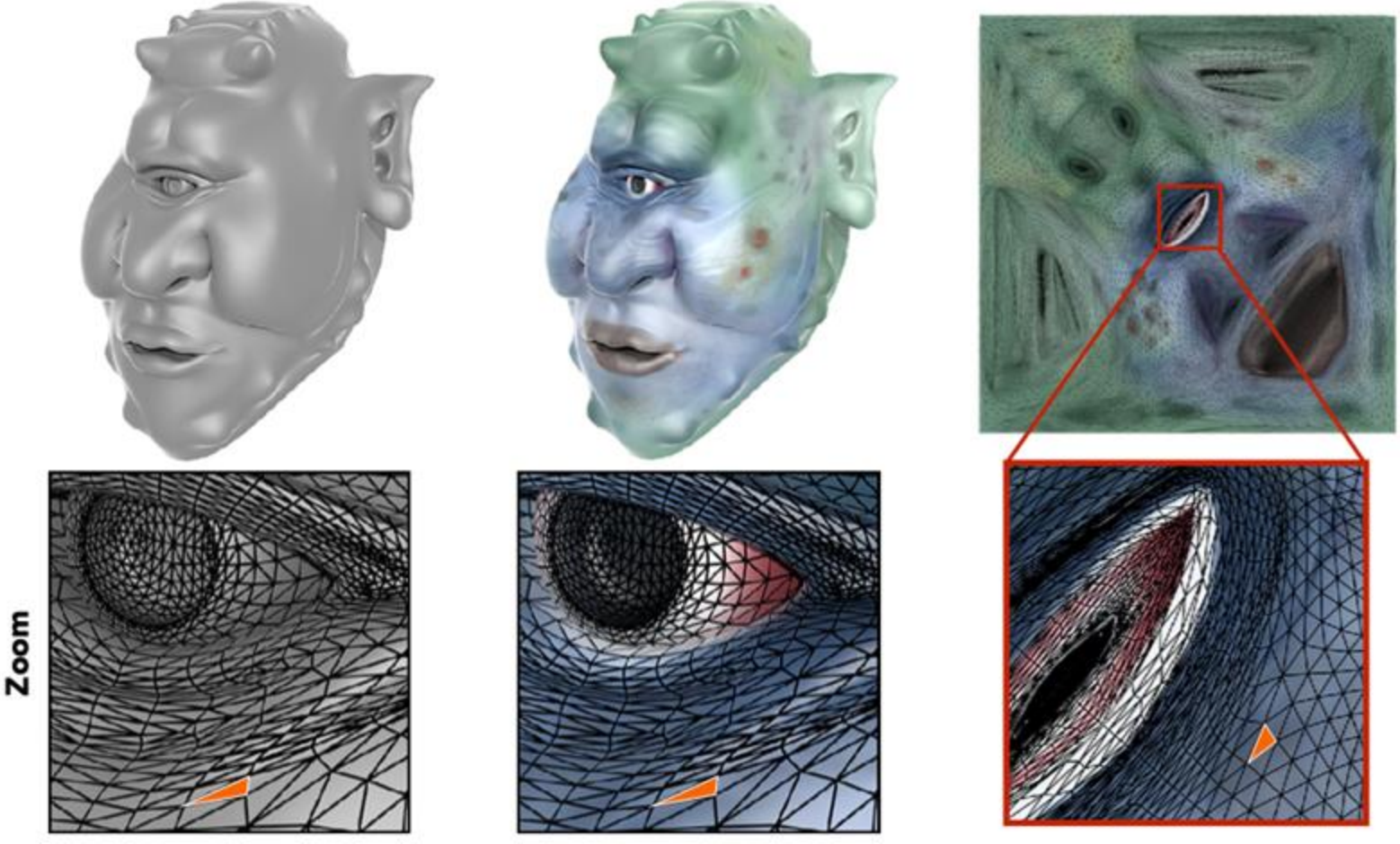
https://www.wikiwand.com/en/Texture_mapping

Textura Aplicada na Superfície

Renderização sem textura

Renderização com textura

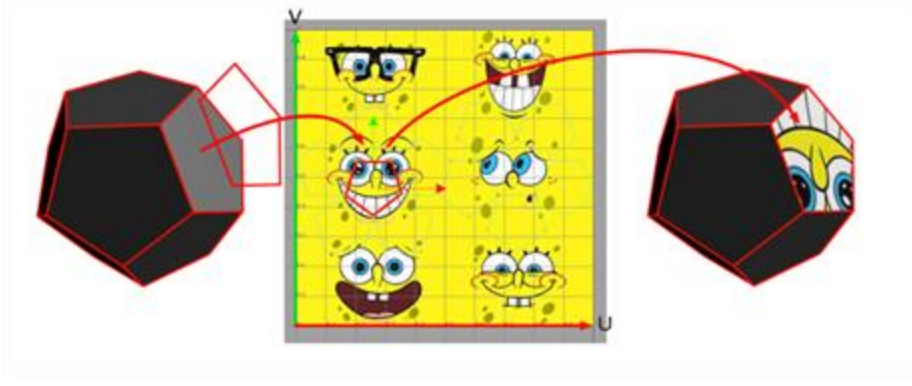
Textura

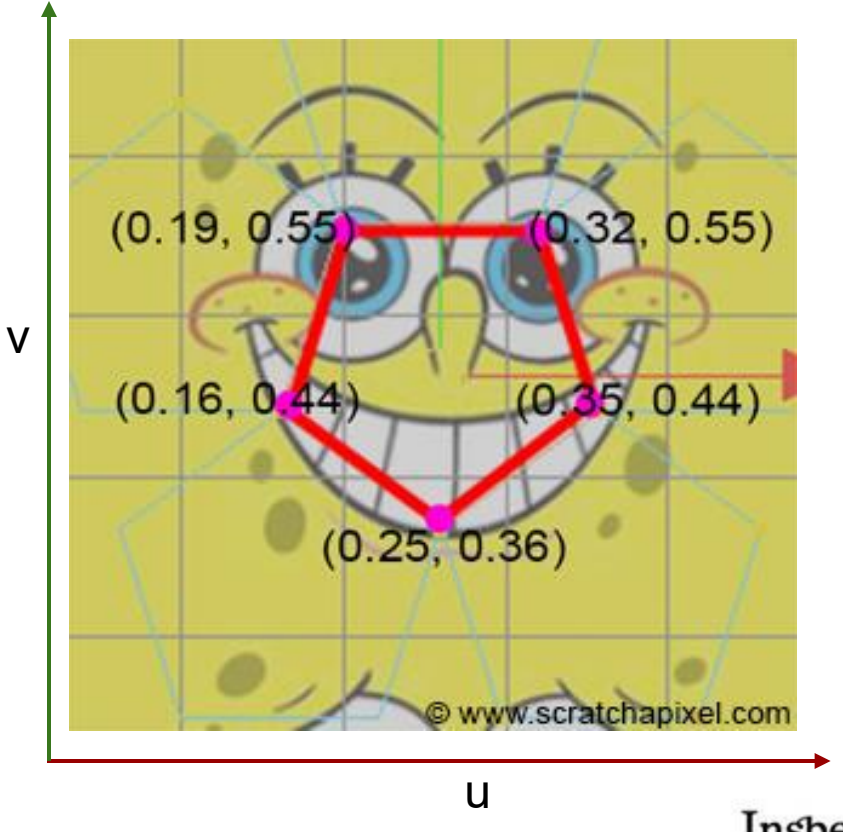


Cada triângulo "copia" um pedaço da textura para a superfície.

UV Layout

O processo pelo qual as faces são dispostas na superfície da textura é chamado de layout UV.

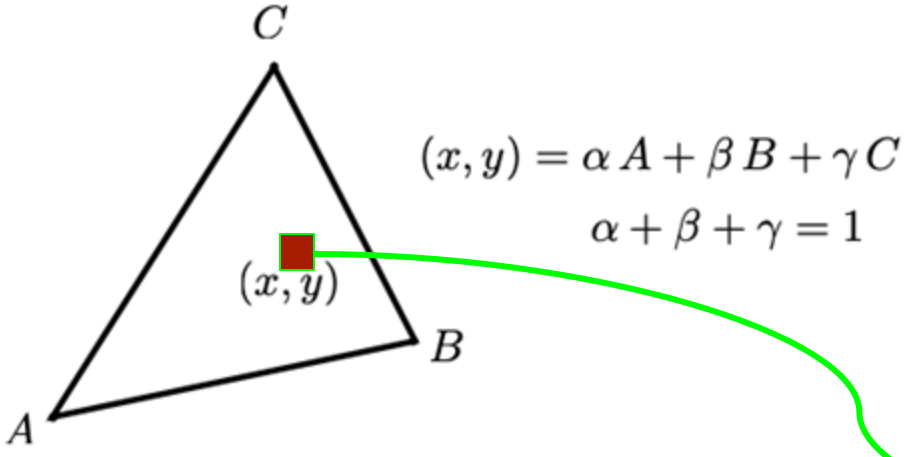




Insper

Operação de Mapeamento de Textura

Ao rasterizar a tela no ponto(x, y) identificar o peso das coordenadas:



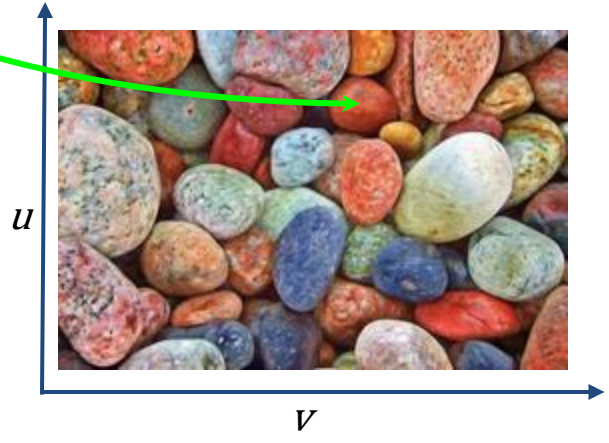
$$(x, y) = \alpha A + \beta B + \gamma C$$

$$\alpha + \beta + \gamma = 1$$

Os valores das coordenadas de textura são dois (u, v) que são definidos em cada vértice do triângulo.

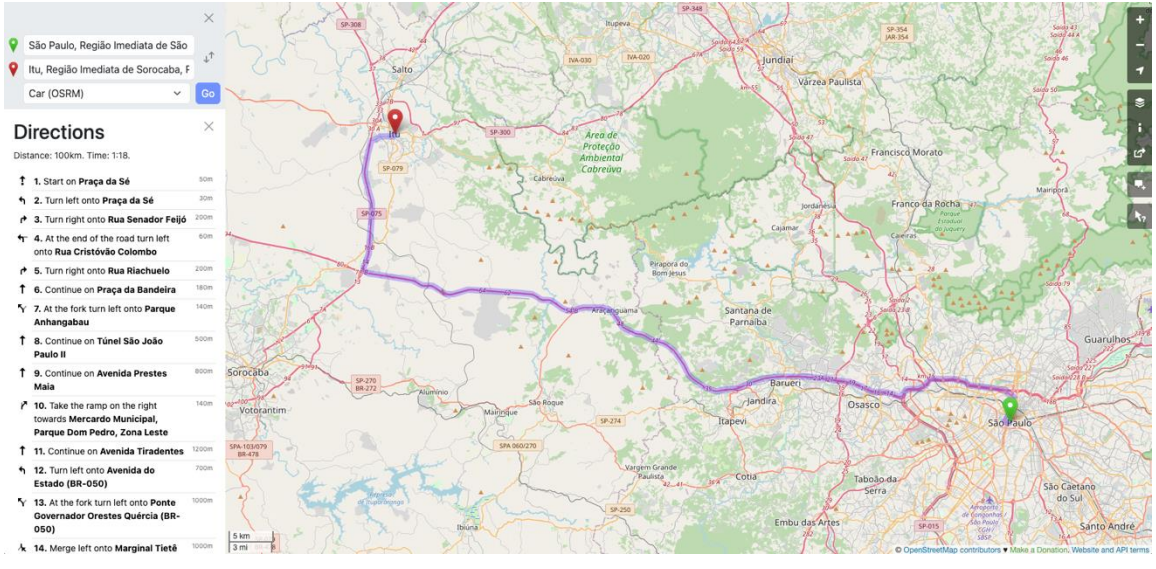
Use as mesmas interpolações para identificar o valor da coordenada de textura em (x, y) para valores (u, v)

Avaliar o valor da cor na textura nas coordenadas (u, v) da textura.



Médias

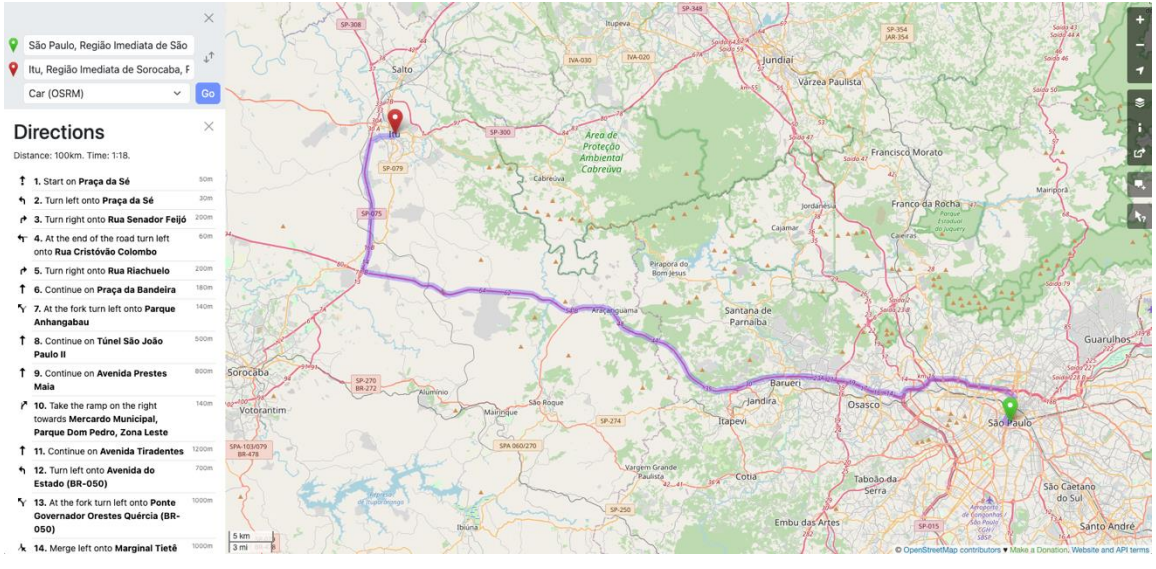
Prof. Luciano e Fabio Orfali foram até Itú ver o grande orelhão da cidade. A distância é de ~100 km, assim eles dividiram ½ a ½ a condução para a ida.



Insper

Médias

O prof. Luciano dirigiu a 100 km/h até Araçariguama que fica a 50 km de São Paulo, já o prof. Fabio dirigiu o restante a 125 km/h.



Insper

Fazendo as contas: São Paulo -> Itú

São Paulo até Araçariguama
- 50 km de São Paulo
- 100 km/h

Araçariguama até Itú
- 50 km
- 125 km/h

Qual foi a velocidade média desse percurso?

$$v = \frac{100 + 125}{2} = 112{,}5 \text{ km/h ?}$$

Refazendo as contas: São Paulo -> Itú

Foram:

- 50 km a 100 km/h = 0.5h
- 50 km a 125 km/h = 0.4h

Tempo total = 0.9h

Usando a fórmula de velocidade:

$$v = \frac{\Delta s}{\Delta t}$$

100 / 0.9 ≈ 111,1 km/h

Problema: Como a distância está fixa, o que realmente influenciou a velocidade média é o tempo gasto em cada parte do trajeto, que está no denominador, não a distância.

Média Harmônica

Necessária quando lidamos com taxas ou razões.

$$\bar{h} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Restrições:

- **Valores Não Nulos**: A média harmônica não pode ser calculada se um dos valores da amostra for zero;
- **Valores Positivos**: Embora a média harmônica possa tecnicamente ser calculada com valores negativos, seu uso faz mais sentido com valores positivos.

Conferindo as contas: São Paulo -> Itú

São Paulo até Araçariguama
- 50 km de São Paulo
- 100 km/h

Araçariguama até Itú
- 50 km
- 125 km/h

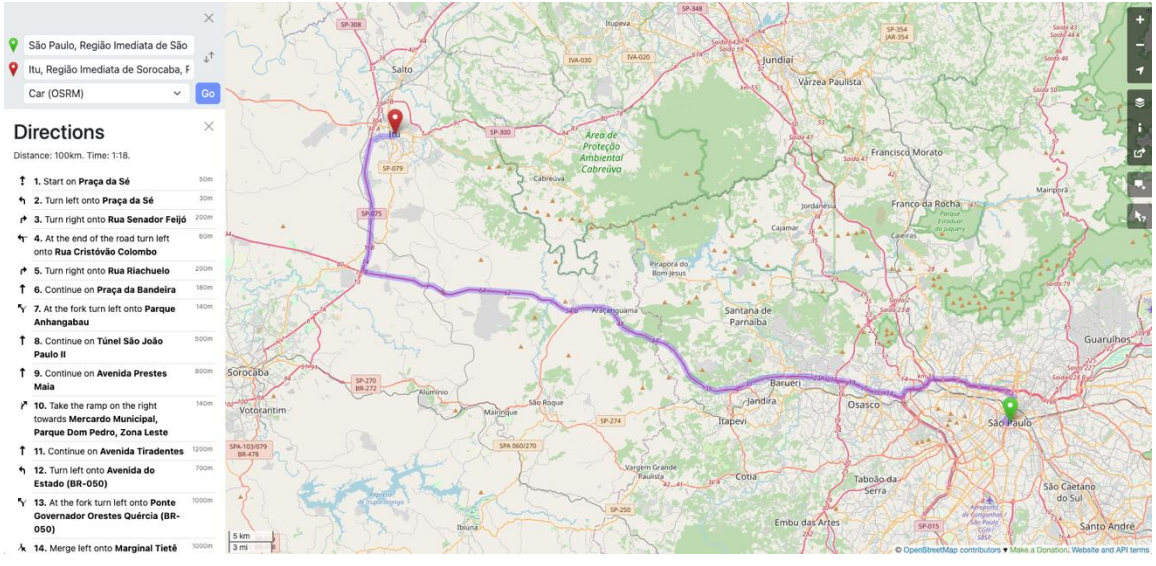
Qual foi a velocidade média desse percurso?

$$v = \frac{2}{\frac{1}{100} + \frac{1}{125}} = \frac{2}{0.01 + 0.008} = \frac{2}{0.018} \approx 111.1$$

A velocidade média da ida foi de 111.1 km/h

Médias

Na volta o prof. Luciano dirigiu novamente a 100 km/h até Osasco (80 km) e o prof. Fabio dirigiu o restante (20km) a 125 km/h. Qual foi a velocidade média nesse caso?



Insper

Média Harmônica Ponderada

Com um conjunto de pesos w_i associados ao conjunto de valores x_i a média harmônica ponderada é definida por:

$$\frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}}$$

Fazendo as contas: Itú -> São Paulo

$$v = \frac{100}{\frac{80}{100} + \frac{20}{125}} \qquad v = \frac{1}{\frac{0.8}{100} + \frac{0.2}{125}}$$

$$v = \frac{1}{0.008 + 0.0016} = \frac{1}{0.0096} \approx 104.167$$

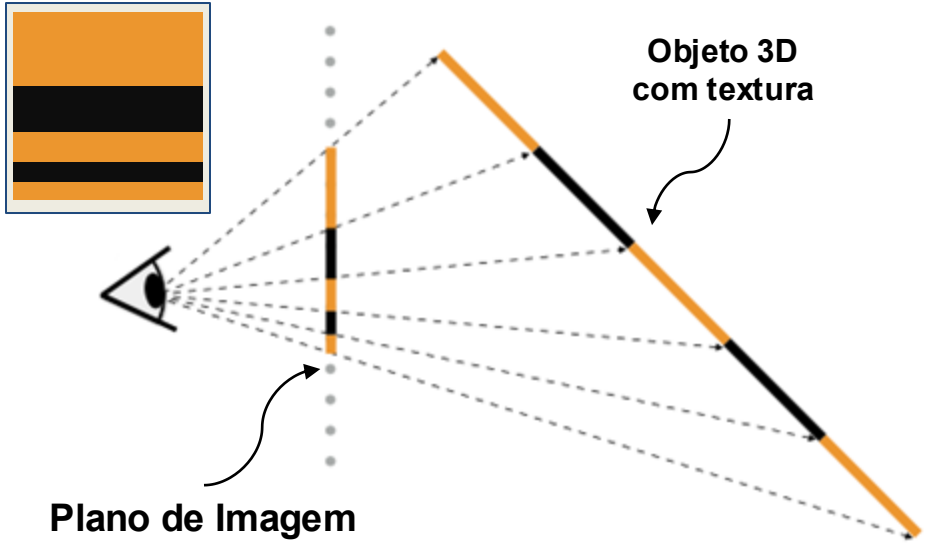
A velocidade média da volta foi de 104.167 km/h

Já vamos usar essas fórmulas para resolver um problema em Computação Gráfica

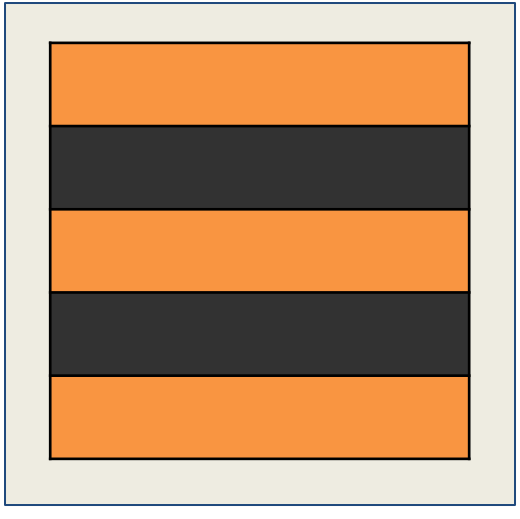
Projeções Perspectivas Cria Não Linearidades

Uma interpolação no espaço da tela (2D) não é a mesma de no espaço da câmera (3D).

A imagem deveria ser:



Porém se não tratarmos corretamente a imagem fica:

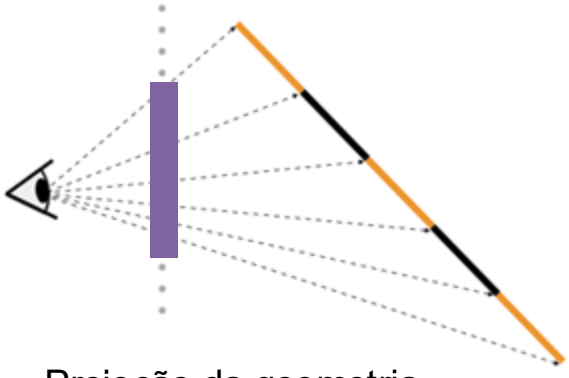


Uma interpolação linear nas coordenadas do mundo precisa de uma interpolação não linear quando transferida para as coordenadas da tela!

Projeções Perspectivas Cria Não Linearidades

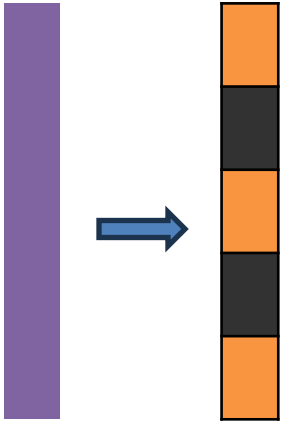
O problema está na simplificação que podemos ter com a projeção, onde projetamos e tratamos o triângulo da tela como se fosse paralelo ao plano da imagem.

1ª fase



Projeção da geometria

2ª fase

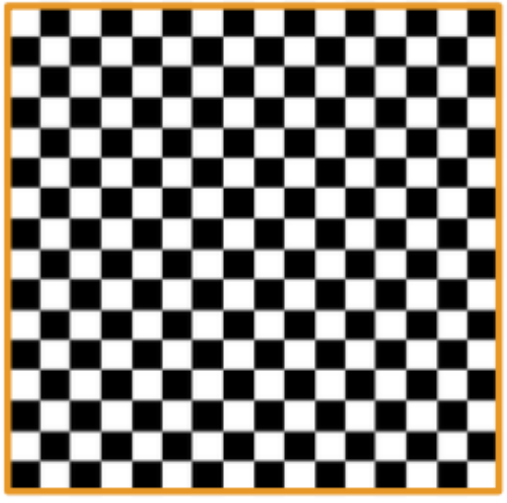


Calcular textura pela geometria projetada

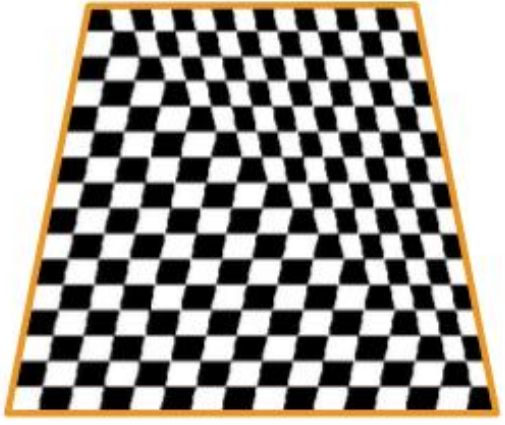
ou seja, precisamos usar o X, Y e Z para fazer as contas de interpolação, mesmo em 2D.

Mas mesmo assim outros problemas podem acontecer!

Projeção Perspectiva e Interpolação



Textura



Plano inclinado com uma projeção perspectiva

O que está errado?

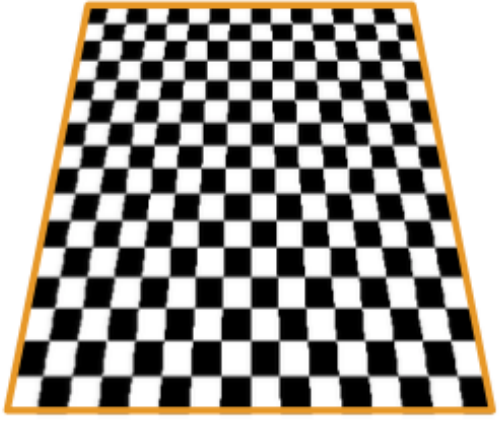
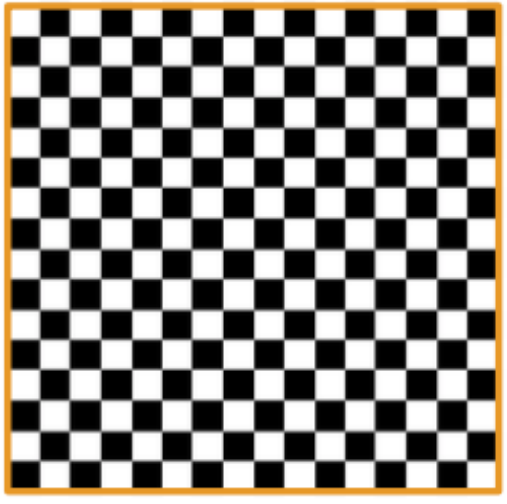
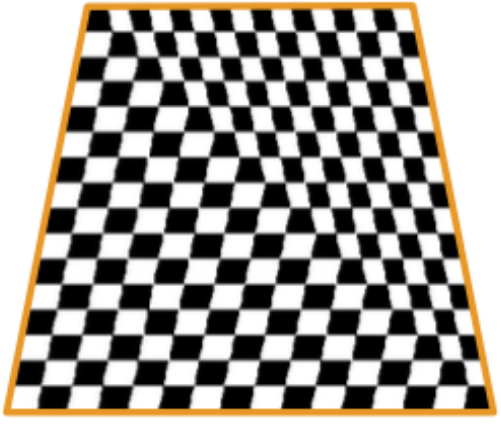


Imagem correta

Projeção Perspectiva e Interpolação



Textura



Interpolação baricêntrica de coordenadas de textura diretamente no espaço da tela (2D) sem a coordenada Z ajustada

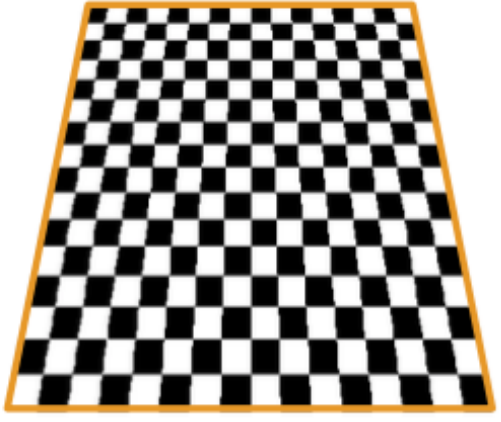
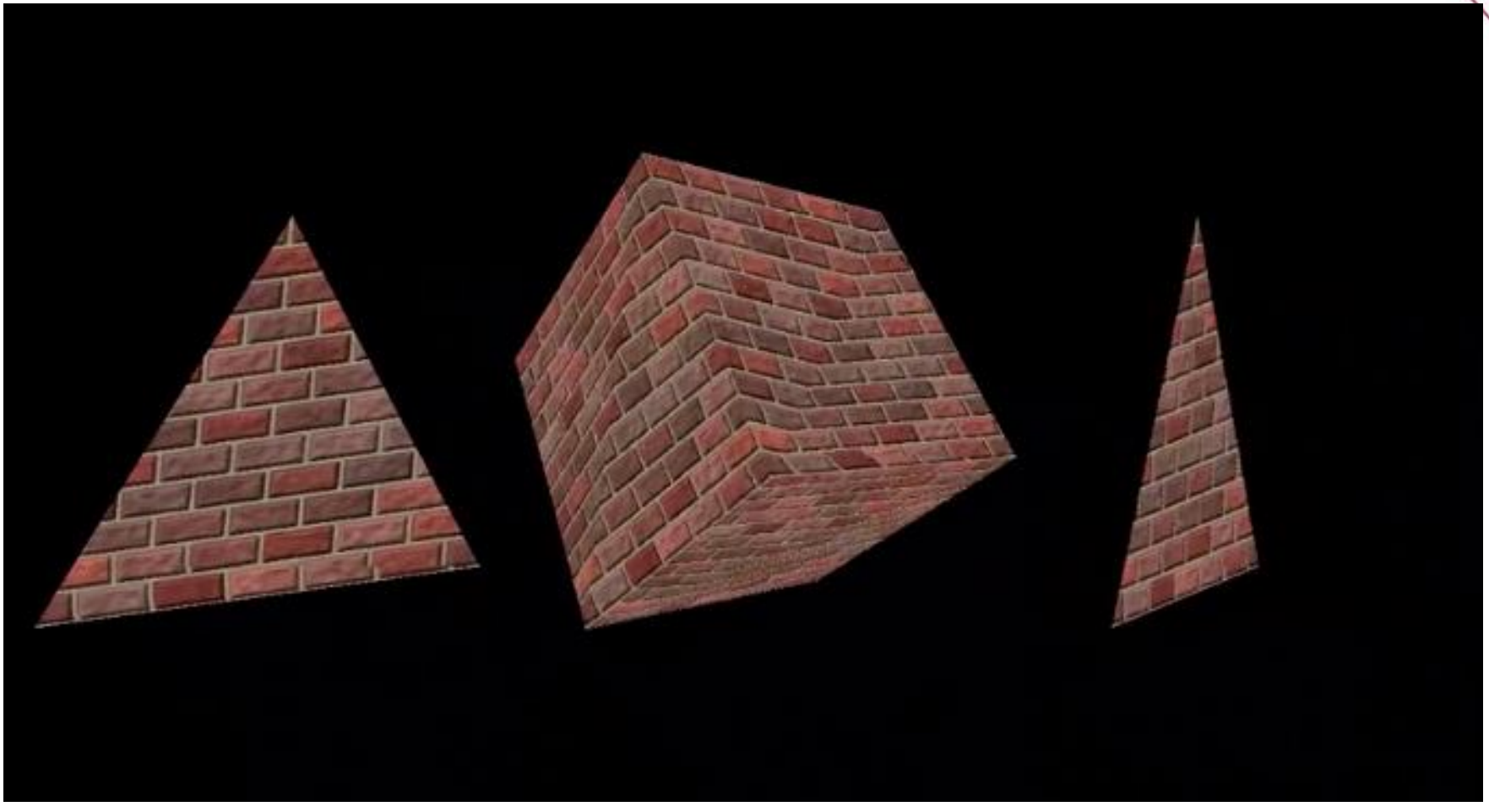


Imagem correta

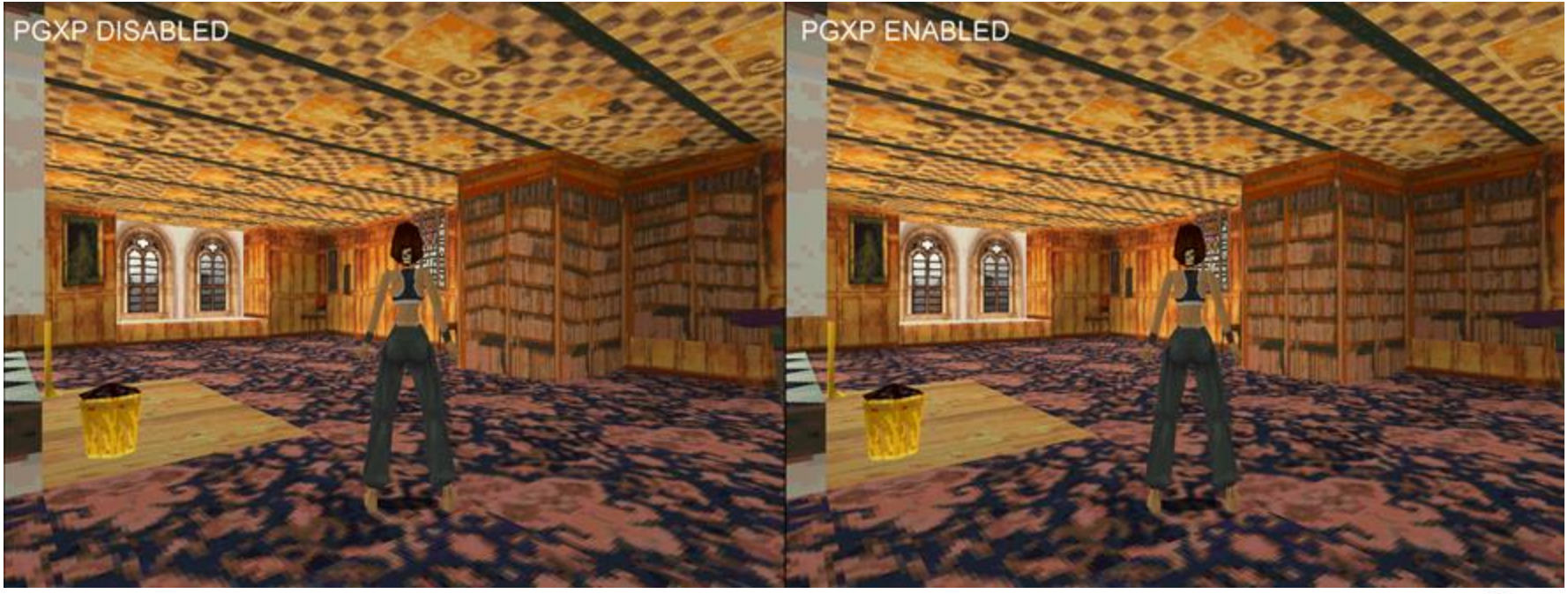
Mapeamento de texturas Afim



https://www.reddit.com/r/GraphicsProgramming/comments/fhvrhl/learning_3d_graphics_implemented_affine_texture/

PlayStation 1

O PS1 não tinha texturas corrigidas pela perspectiva. Hoje o PGXP (Parallel / Precision Geometry Transform Pipeline) é um aprimoramento para a emulação do PlayStation que produz texturas em geometrias 3D com a correção necessária.



Insper

Matriz de Transformação Perspectiva (Z)

Vamos entender o que acontece com o um ponto Z (3D) quando ele é projetado no espaço 2D.

$$\mathbf{P} = \begin{bmatrix} \frac{near}{right} & 0 & 0 & 0 \\ 0 & \frac{near}{top} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{-2far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ Z \\ 1 \end{bmatrix}$$

troquei near e far para inverter sinal

$$Z' = Z \cdot \frac{\text{far} + \text{near}}{\text{near} - \text{far}} + \frac{2 \cdot \text{far} \cdot \text{near}}{\text{near} - \text{far}}$$

$$w' = -Z$$

$$Z_{\text{normalizado}} = \frac{Z'}{w'} = \frac{Z \cdot \frac{\text{far}+\text{near}}{\text{near}-\text{far}} + \frac{2 \cdot \text{far} \cdot \text{near}}{\text{near}-\text{far}}}{-Z} = \frac{\text{far} + \text{near}}{\text{near} - \text{far}} + \frac{2 \cdot \text{far} \cdot \text{near}}{\text{near} - \text{far}} \cdot \frac{1}{-Z}$$

Matriz de Transformação Perspectiva (Z)

$$Z_{\text{normalizado}} = \frac{\text{far} + \text{near}}{\text{near} - \text{far}} + \frac{2 \cdot \text{far} \cdot \text{near}}{\text{near} - \text{far}} \cdot \frac{1}{-Z}$$

Por exemplo: Se, *near* = 1 (-1) e *far* = 10 (-10)

$$Z_n = -\frac{11}{9} - \frac{20}{9 \cdot Z}$$

Ou seja, embora a função de projeção preserve a ordenação da coordenada de profundidade (Z), essa transformação não é linear.

Isso é uma função racional que gera uma hipérbole

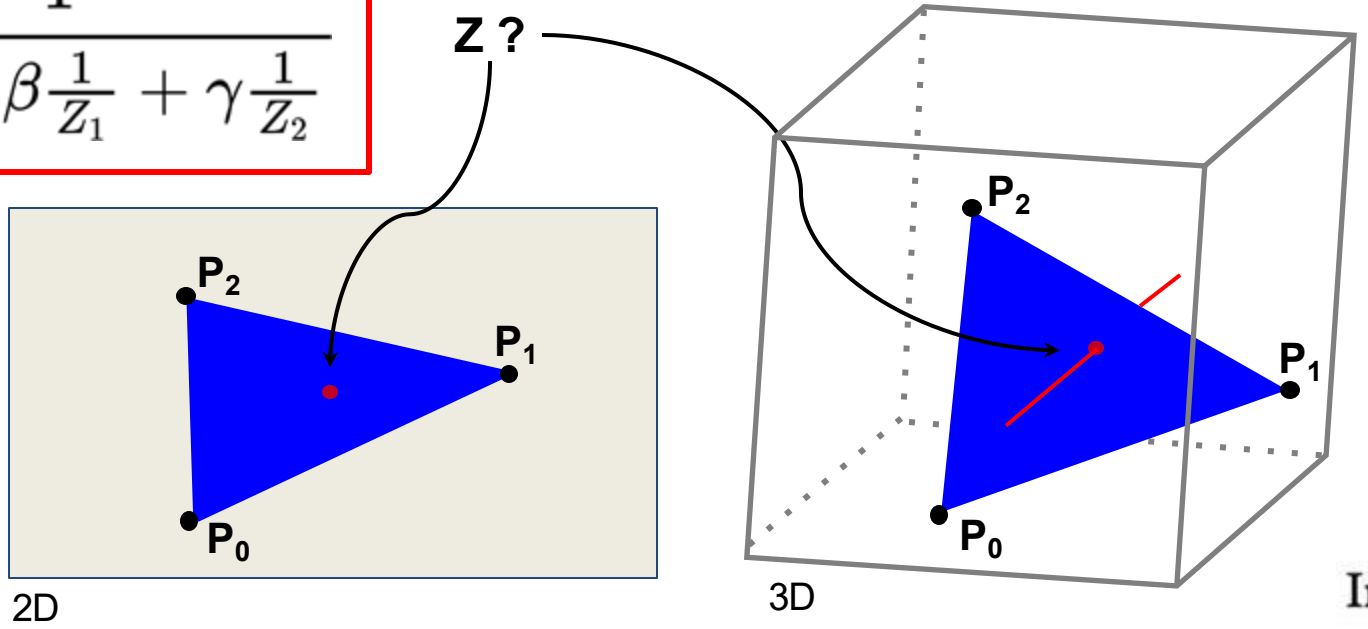
Insper

Calculando o Z no interior do triângulo

Assim, utilizamos o valor Z dos vértices no espaço da câmera para calcular o Z projetado* do ponto amostrado, por meio da média harmônica ponderada das coordenadas baricêntricas com pesos α, β e γ.

*Embora a projeção envolva apenas X e Y, o Z permanece como uma dimensão válida.

$$Z = \frac{1}{\alpha\frac{1}{Z_0} + \beta\frac{1}{Z_1} + \gamma\frac{1}{Z_2}}$$



Insper

Média Harmônica Ponderada

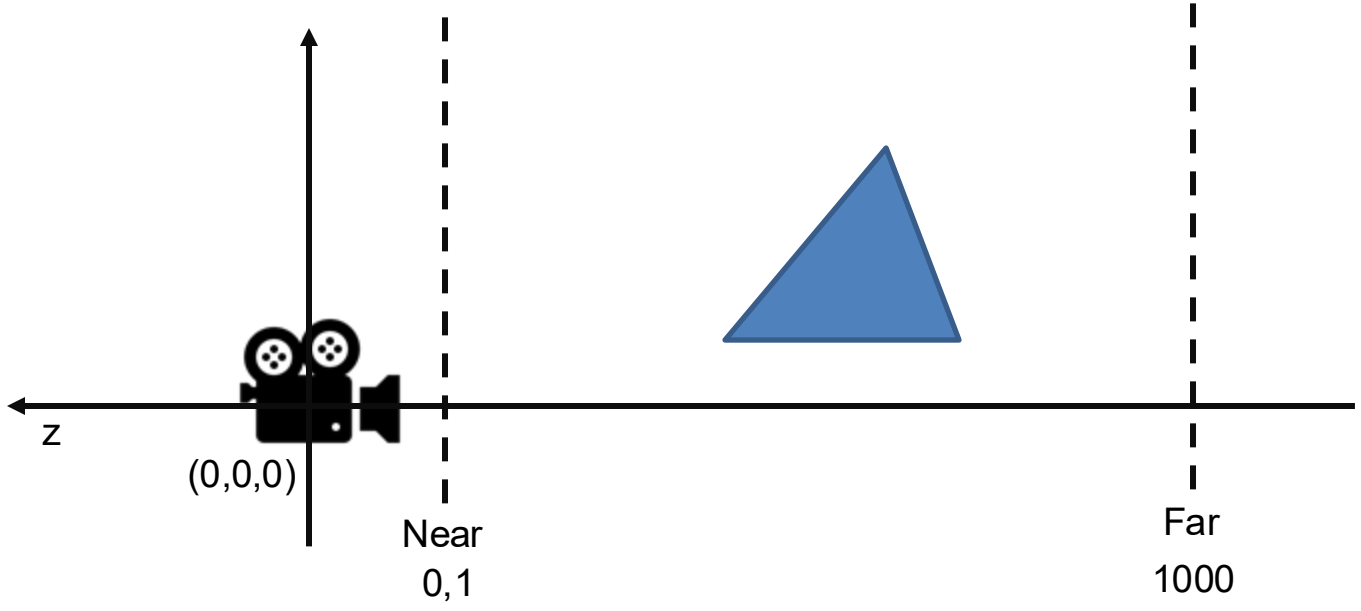
Faça os cálculos com os valores de **Z no espaço da câmera (view)**

não use NDC

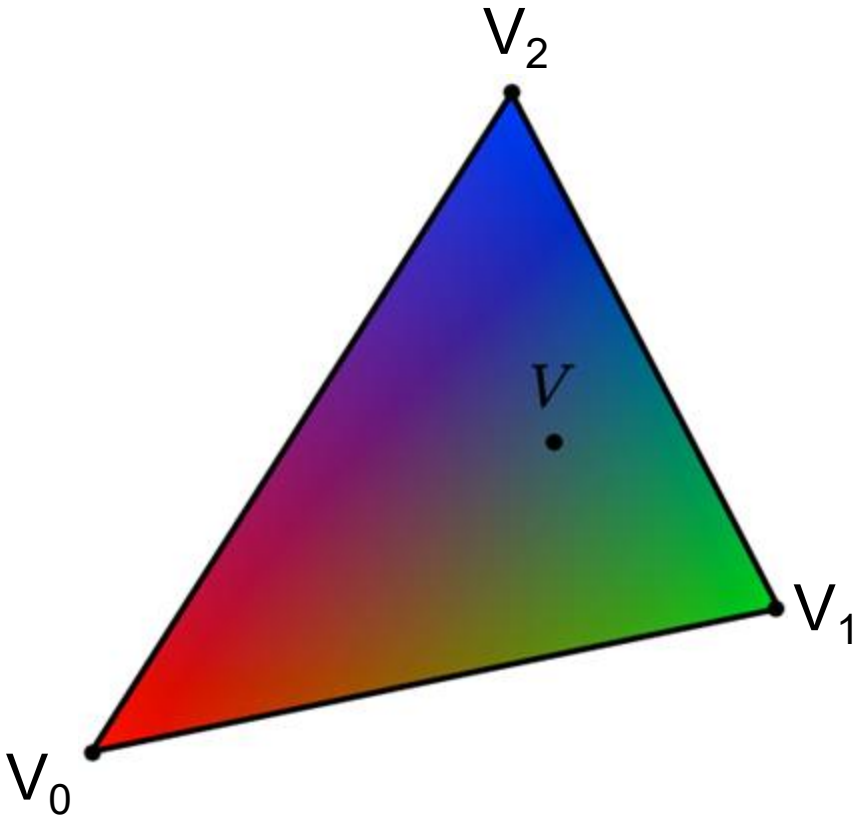
Use os valores positivos (inverta o sinal da coordenada Z)

Por causa do *Near* os valores serão sempre maiores que zero

podemos usar tranquilamente média harmônica



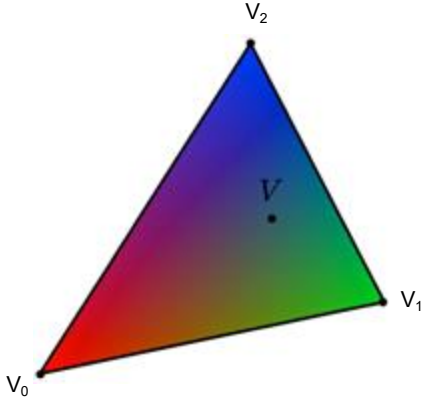
Interpolação não linear em Z pelo Triângulo



V = ?

Representação Conceitual da Interpolação

Usar agora o Z dividindo o parâmetro que se deseja interpolar.
E depois multiplica o Z do ponto sendo amostrado.



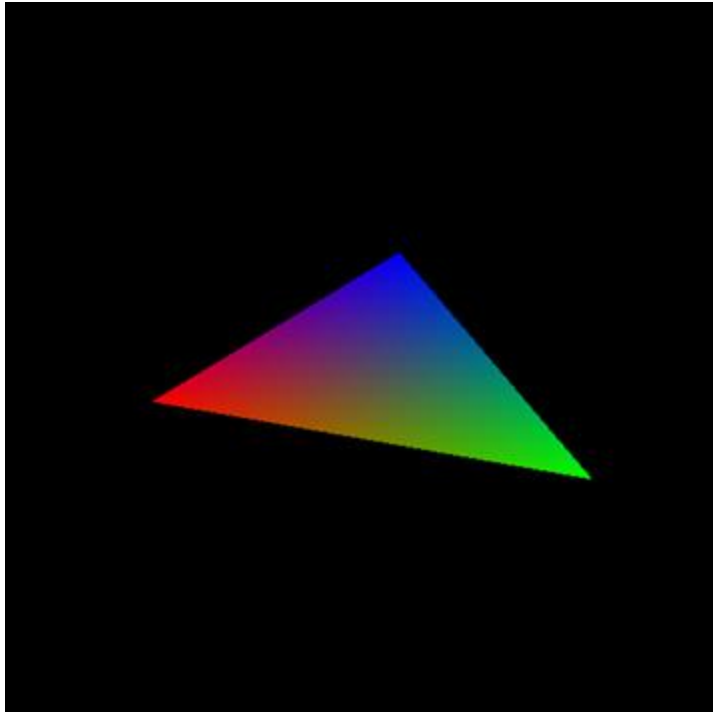
$$V = Z \cdot \left(\alpha \frac{V_0}{Z_0} + \beta \frac{V_1}{Z_1} + \gamma \frac{V_2}{Z_2} \right)$$

As coordenadas baricêntricas projetadas no espaço da tela não são transformações afim. Já se forem interpoladas com a divisão do valor de Z são transformações afim.

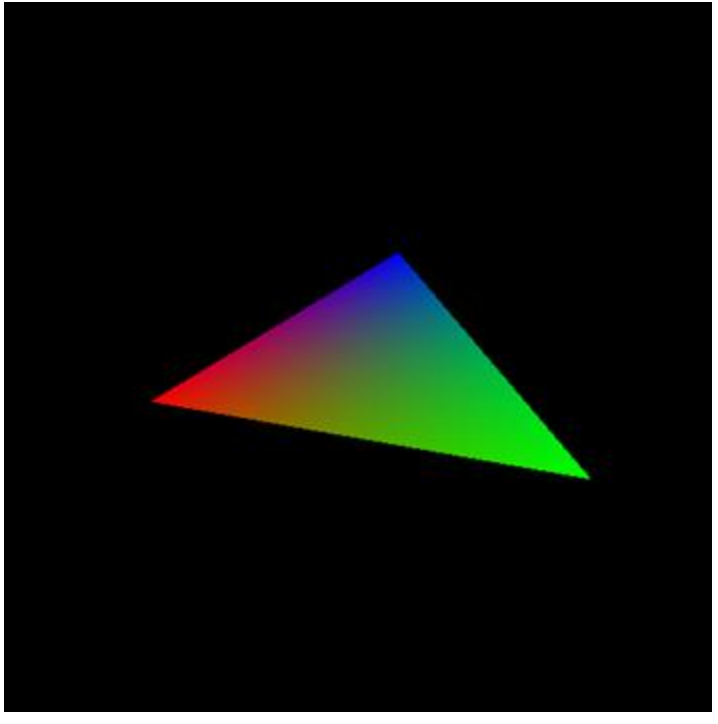
Cuidado: o Z a ser usado é do espaço da câmera, não use o Z do NDC!

Diferença no resultado quando usada o Z

Interpolação Baricêntrica 2D



Interpolação Baricêntrica 3D



Mais informações em: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/visibility-problem-depth-buffer-depth-interpolation.html

https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/perspective-correct-interpolation-vertex-attributes

Coordenadas Baricêntricas para (u, v)

Mapear as texturas para os triângulos é a mesma proposta, só que ao invés de interpolar (r, g, b) usaremos as coordenadas (u, v) de textura.

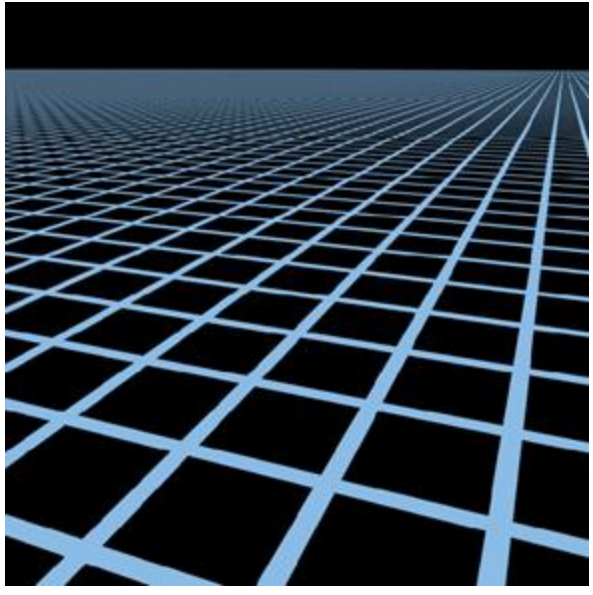
Uma equação mais simples (que pode ser optimizada) é:

$$u = \frac{\alpha \frac{u_0}{z_0} + \beta \frac{u_1}{z_1} + \gamma \frac{u_2}{z_2}}{\alpha \frac{1}{z_0} + \beta \frac{1}{z_1} + \gamma \frac{1}{z_2}}$$

$$v = \frac{\alpha \frac{v_0}{z_0} + \beta \frac{v_1}{z_1} + \gamma \frac{v_2}{z_2}}{\alpha \frac{1}{z_0} + \beta \frac{1}{z_1} + \gamma \frac{1}{z_2}}$$

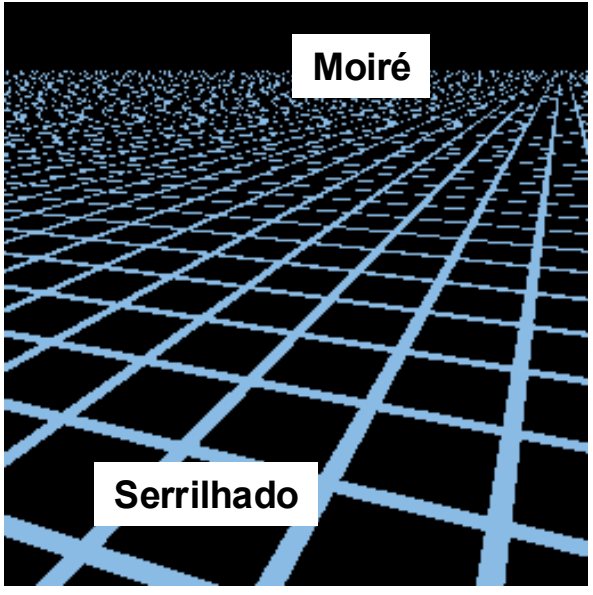
Equação que a maioria das placas de vídeo e APIs gráficas (como OpenGL e DirectX) resolvem o problema para renderizar texturas de forma realista.

Mais problemas, amostrando texturas



Referência em Alta-resolução

imagem original em 1280x1280 pixels



Por amostragem (pontuais)

256x256 pixels

Para resolver esse problema temos de usar alguma técnica como Mipmap, que está como atividade extra não obrigatória

Color e TextureCoordinate

Os nós **Color** e **TextureCoordinate** define um conjunto de pontos que podem ser usados para armazenar cores e coordenadas de textura respectivamente.

```
Color : X3DColorNode {
  MFColor [in,out] color    [NULL] [0,1]
  SFNode  [in,out] metadata NULL   [X3DMetadataObject]
}
```

```
TextureCoordinate : X3DTextureCoordinateNode {
  SFNode  [in,out] metadata NULL [X3DMetadataObject]
  MFVec2f [in,out] point    []   (-∞,∞)
}
```

https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/rendering.html#Color

ImageTexture

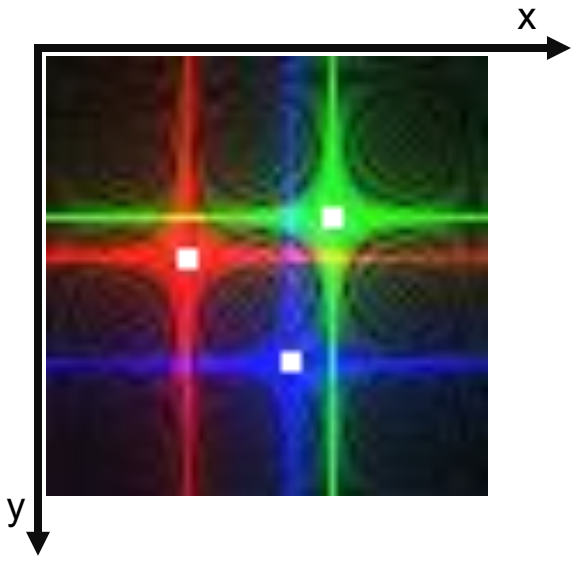
O nó **ImageTexture** serve para ler uma imagem, o atributo **url** é usado para ter uma lista de pontos de acesso da imagem.

```
ImageTexture : X3DTexture2DNode, X3DUrlObject {
  SFNode   [in,out] metadata          NULL [X3DMetadataObject]
  MFString [in,out] url               []   [URI]
  SFBool   []       repeatS           TRUE
  SFBool   []       repeatT           TRUE
  SFNode   []       textureProperties NULL [TextureProperties]
}
```


https://www.web3d.org/documents/specifications/19775-

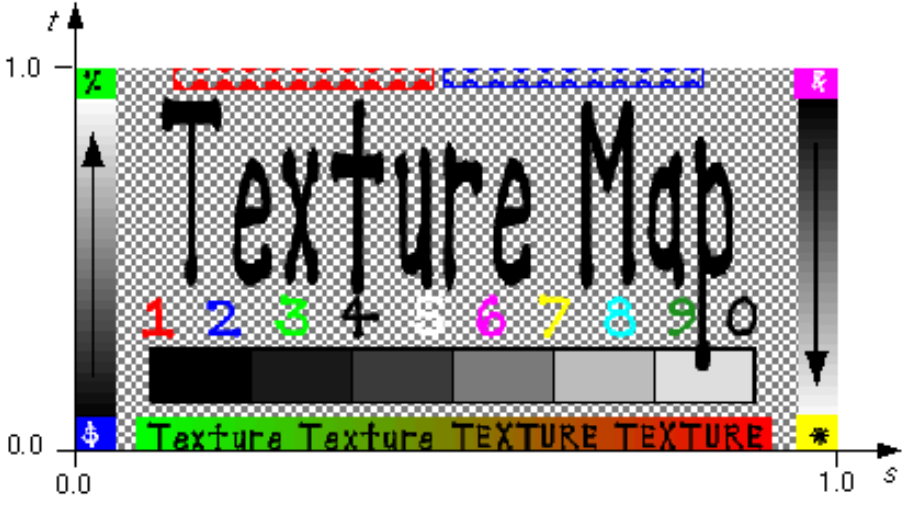
Atenção

Como as imagens são organizadas em PNG



http://www.libpng.org/pub/png/png-textures.html

Como as texturas são mapeadas



Em geral chamamos de *u* e *v*, mas no X3D temos *t* e *s*.

https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/texturing.html

Insper

Computação Gráfica

Luciano Soares
<lpsoares@insper.edu.br>

Fabio Orfali
<fabioo1@insper.edu.br>