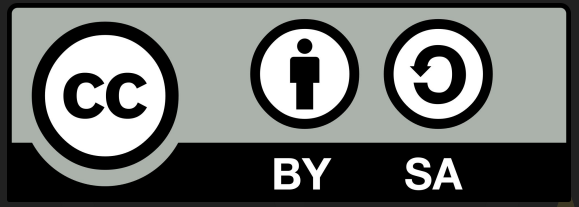
Computação Gráfica para Jogos Eletrônicos

Visão geral sobre o processo de renderização de jogos digitais

Slides por: Leonardo Tórtoro Pereira (leonardop@usp.br)
Adicionais 2020: Renata Vinhaga(renatavinhaga@usp.br)
Assistentes: Gustavo Ferreira Ceccon (gustavo.ceccon@usp.br),
Gabriel Simmel (gabriel.simmel.nascimento@usp.br) e Ítalo Tobler (italo.tobler.silva@usp.br)



Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos Eletrônicos (TEDJE)
Filiado ao grupo de cultura e extensão Fellowship of the Game (FoG), vinculado ao ICMC - USP

Objetivos

- Introduzir a área de Computação Gráfica (CG)
- Diferenças entre ambiente *offline* e *Real-time*
- CPU x GPU
- Introduzir o conceito de pipeline gráfico
- O que é uma API gráfica?
  - OpenGL
  - DirectX
  - Vulkan

Objetivos

- Como modelos são estruturados, representados e apresentados para os usuários
- Onde a matemática entra na CG e porque ela é tão essencial (transformações por exemplo)
- Apresentar os fenômenos e algoritmos básicos utilizados para trazer o mundo real para o virtual

Índice

1. Introdução

1. Introdução

- O que é Computação Gráfica (CG)?
  - Reproduzir o mundo real artificialmente
  - Geração de elementos != Processamento de Imagens
  - Processar inúmeras operações para decidir a cor de cada pixel em uma janela de visualização
  - Criar universos inimagináveis

Usuário x Engenheiro

- A computação gráfica possui diversos níveis
  - Usuário que vai utilizar o Blender
  - Engenheiro que vai criar as ferramentas pro usuário

Ainda assim, ambos estão trabalhando com CG !

1. Introdução

➔ Quais são os tópicos mais importantes da área?

- Modelagem 3D
- Animação
- Simulação
- *Shaders*
- Design de GPU
- Rendering
- Entre outros!

1. Introdução

➔ É responsável por

- Exibir dados de imagem e arte efetivamente e de maneira agradável ao usuário.
- Processar dados de imagem recebidos do mundo físico

1. Introdução

- ➔ Revolucionou
  - Animação
  - Filmes
  - Publicidade
  - Design gráfico
  - Jogos Eletrônicos

1. Introdução

- O que é uma API gráfica/ biblioteca?
  - OpenGL
  - DirectX
  - Vulkan

Real-Time x offline

https://www.techtudo.com.br/noticias/2020/04/valorant-riot-games-anuncia-inicio-de-beta-fechado-para-o-brasil.ghtml

https://www.blogmodernagem.com.br/2018/11/top-5-melhores-filmes-da-pixar.html

https://www.epicgames.com/store/pt-BR/product/detroit-become-human/home

https://www.epicgames.com/store/pt-BR/product/detroit-become-human/home

2. CPU vs GPU

2. CPU vs GPU

https://www.intel.com/content/www/us/en/homepage.html

https://www.nvidia.com/en-us/

2. CPU vs GPU

- ➔ No início da CG
  - ◆ Seu processamento era feito em CPU
    - Todo computador tem uma! :)
    - Poucos núcleos (antigamente só 1!) :(
    - Não é usada apenas para gráficos :(
    - Alto custo por núcleo :(

2. CPU vs GPU

- ➔ Atualmente
  - ◆ Seu processamento é feito (principalmente) em GPU
    - Deve ser comprada à parte :(
    - Muitos núcleos :)
      - 1152 - GTX 760
      - 2560 - GTX 1080
      - 5760 - Titan Z

2. CPU vs GPU

- ➔ Por que GPUs ficaram tão rápidas?
  - ◆ Intensidade aritmética
    - Mais transistores para computações
    - Menos para lógica de decisão
  - ◆ Economia
    - Demanda é alta devido à
      - ○ Indústria multibilionária dos jogos eletrônicos

3. Tipos de Imagens

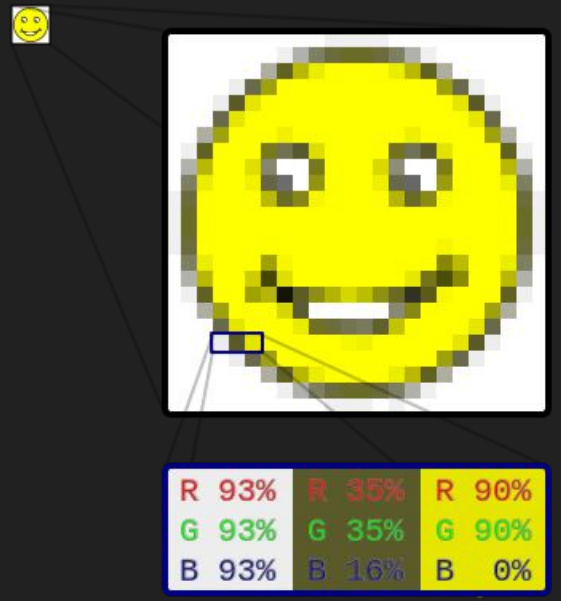
3. Tipos de Imagens

- Tipos de imagem 2D
  - Raster
  - Imagem vetorizada
  - Sprites

Imagens Rasterizadas

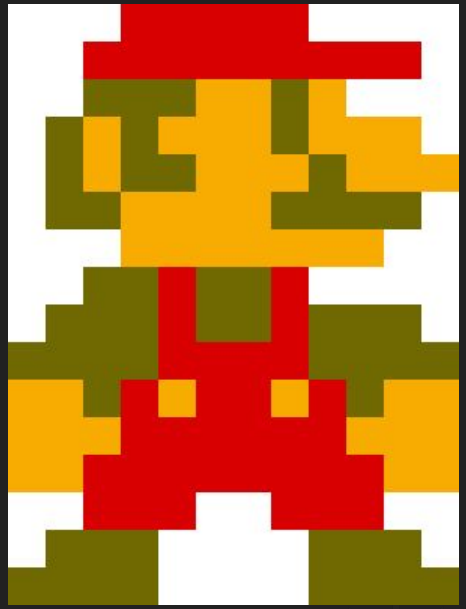
Raster

- Modo de representação de imagem
- Matriz de pixels
- Características do raster:
  - Altura
  - Largura
  - bits/pixel (define o alcance dos valores da matriz)



Raster

- Pixel Art
  - Arte a nível de pixel
  - Necessita baixa utilização de memória
    - Os primeiros consoles possuíam memória muito pequena
  - Ainda hoje utilizada em jogos
  - Baixo custo para exibição



Exemplos Pixel Art

Shameless self promotion

[Hyper Light Drifter]

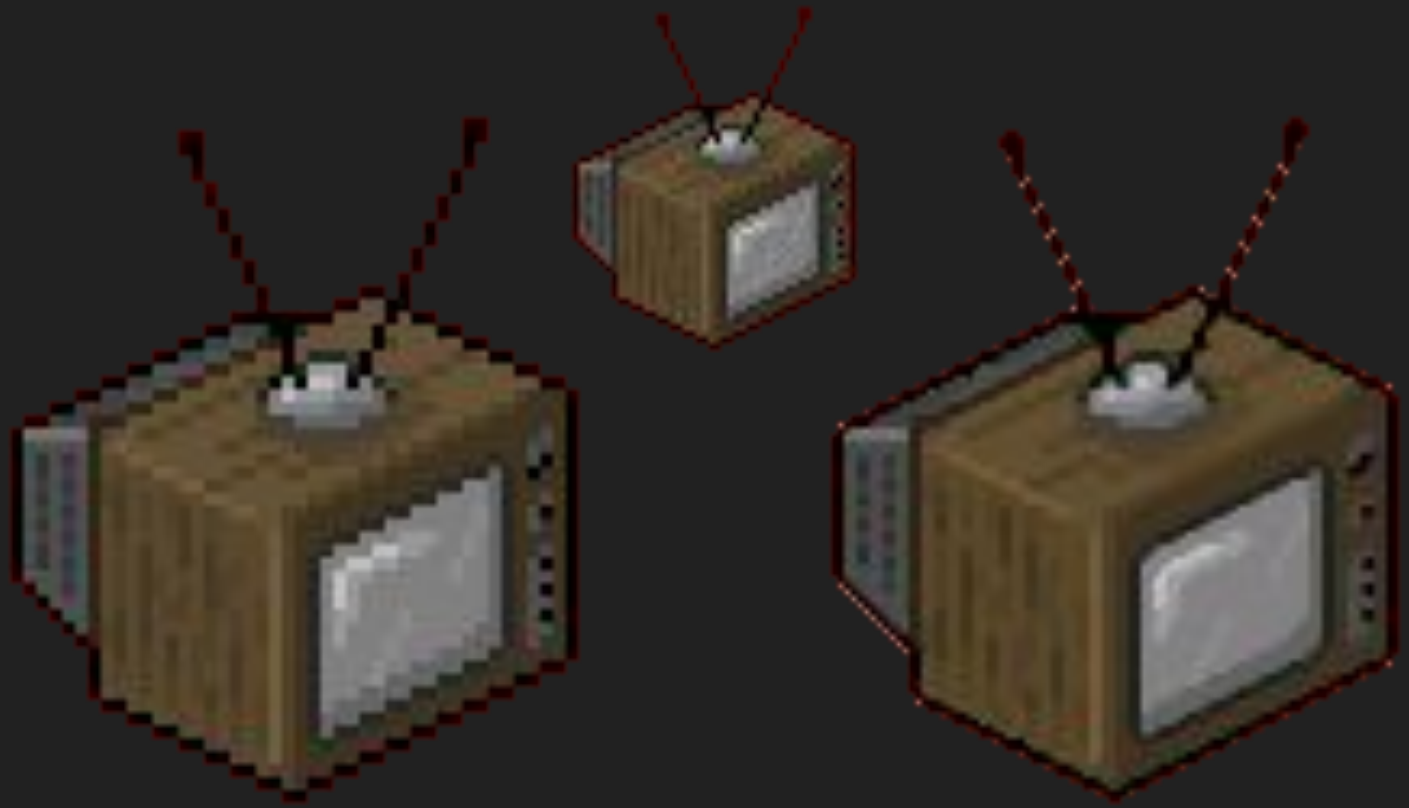
[Chrono Trigger]

[Blasphemous]

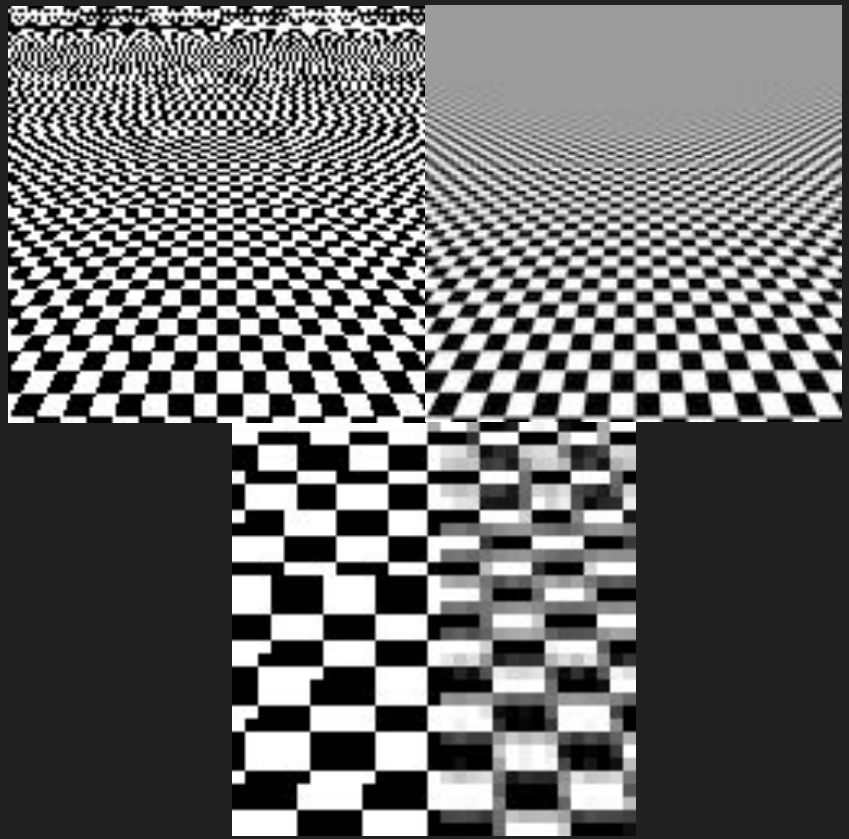
Raster

- Problema: reescalar imagens
  - Algoritmos de interpolação
- Aliasing
  - Solução: Anti-aliasing

Raster

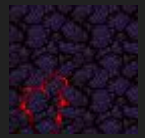


Raster

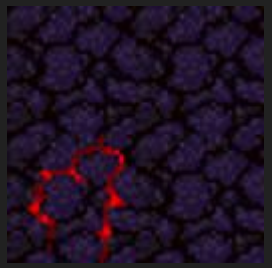


Raster

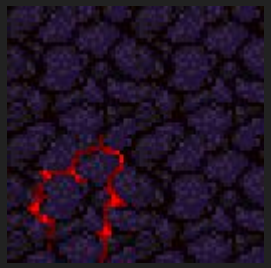
- Cuidado com anti-aliasing em pixel art!
- Antes de re-escalar use o método certo (Caixa)



64x64
Original



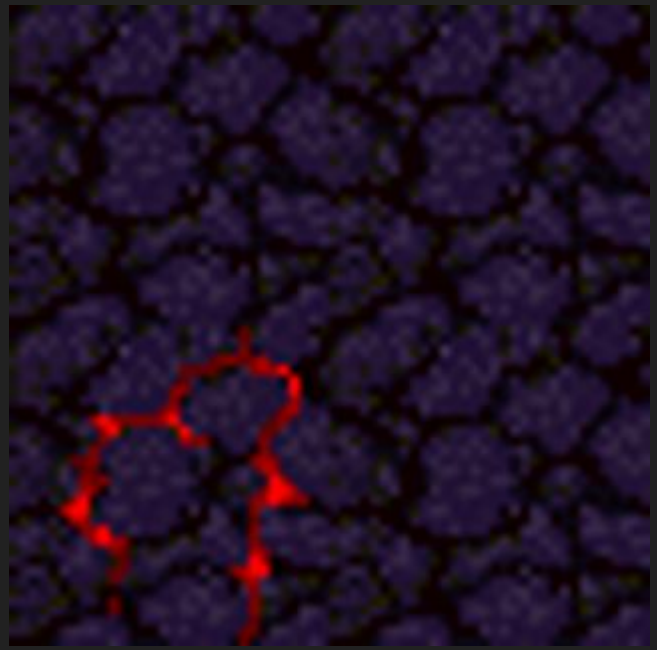
128x128
Filtro Bicúbico



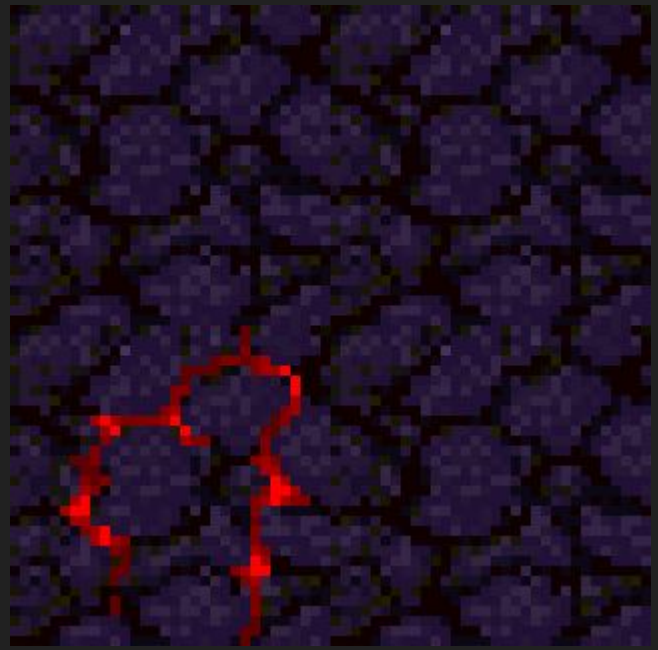
128x128
Filtro Caixa

Sprite feito por Leonardo Tórtoro Pereira. Não usar sem permissão :)

Raster



320x320 Bicúbica



320x320 Caixa

Sprite feito por Leonardo Tórtoro Pereira. Não usar sem permissão :)

Imagem Vetorizada

Imagem Vetorizada

- Representar Imagens por contornos e preenchimentos
  - Imagens compostas por “caminhos” e polígonos primitivos
- Softwares capazes de transformar imagens rasterizadas em vetorizadas
  - Grande perda de informação com imagens de tons contínuos
  - Processo inverso relativamente fácil

Exemplos Imagens Vetorizadas

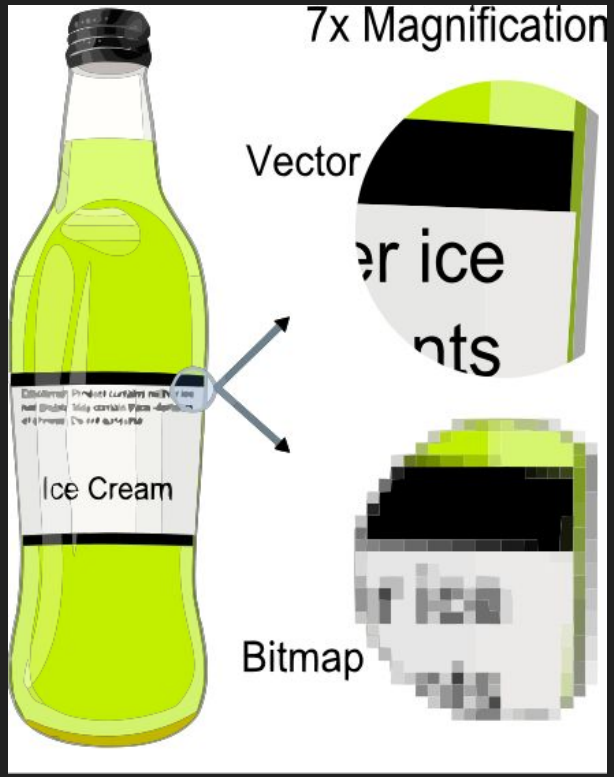
https://opengameart.org/forumtopic/2d-vector-artist-at-your-service

https://www.gamedevmarket.net/asset/desert-2d-game-vector-background/

Imagem Vetorizada

https://en.wikipedia.org/wiki/Vector_graphics

- Imagens vetorizadas não possuem problemas para re-escalar
  - Pode ser rasterizada em diferentes dimensões
- Em contrapartida, a imagem precisa ser rasterizada para ser exibida e a cada vez que for reescalada



Sprites

Sprites

- Bitmaps integrados a uma cena maior
  - Originalmente se referia a objetos independentes, processados separadamente e depois integrados a outros elementos
  - Esse método de organização facilitava detecção de colisões entre diferentes sprites

Quick Tips!

Ferramentas Interessantes para arte 2D

- Desenhos em geral:
  - Krita (Open Source)
  - Gimp (Open Source)
  - Photoshop
- Pixel art:
  - Aseprite (Tem na Steam)
  - Pyxel Edit
- Geração de níveis por tileset
  - Tiled (Com integração para Unity)

Ferramentas Interessantes para arte 2D

- Animação com Bones
  - DragonBones (Open Source)
  - Spine
  - Spriter

- Evite usar animação com Bones em pixel art!
  - É recomendado animar frame a frame :)

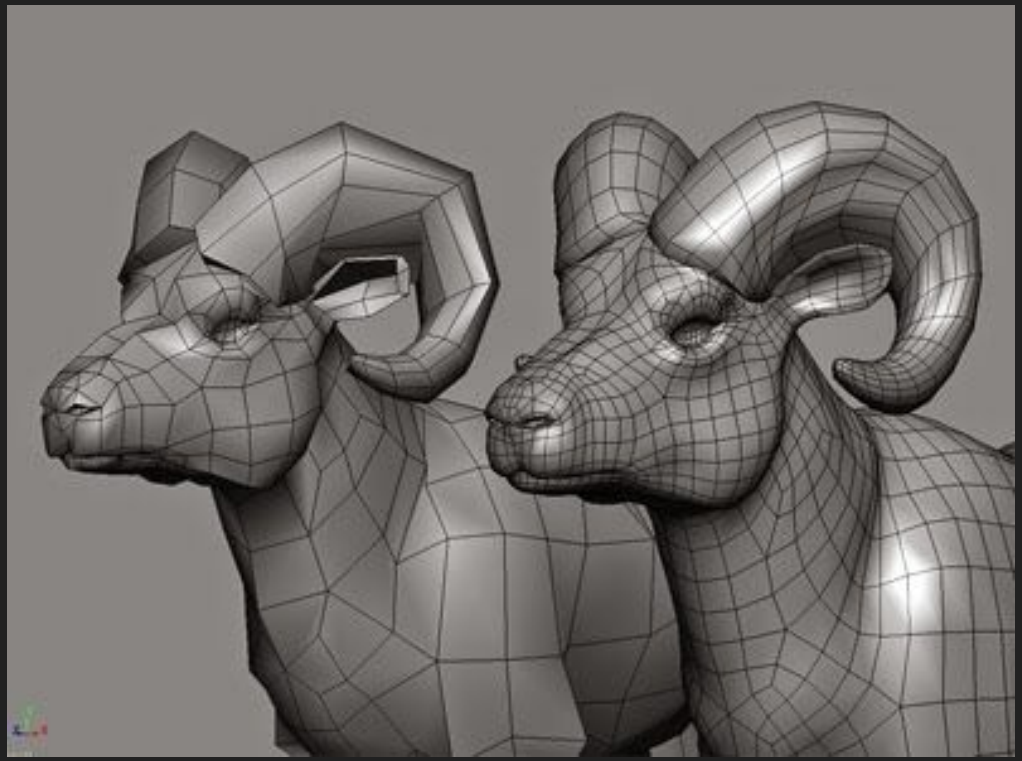
4. Modelos 3D

4. Modelos 3D

- Representados através de malhas
  - Superfícies representadas através de um conjunto de triângulos
- Por que usar triângulos? Por que não usar quadrados?
  - Simples, planares e geralmente não quebram em transformações

4. Modelos 3D

4. Modelos 3D



https://www.gfxtotal.com.br/tutoriais/modelar-abridor-de-latas-high-poly-3ds-max/

Transformações

- Como as coisas se movem?
  - Transladar (Soma)
  - Escalar (Multiplicação)
  - Rotacionar (Trigonometria)

- Transformações SÃO unidas e representadas com matrizes

Transformações

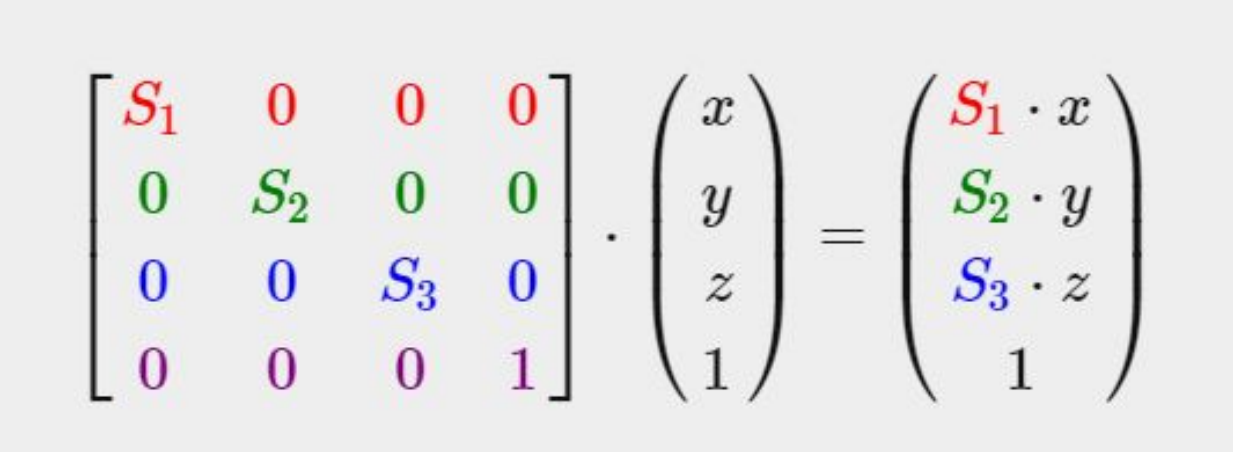
➔ Translação

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

https://learnopengl.com/Getting-started/Transformations

Transformações

➔ Escalar



$$\begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot y \\ S_3 \cdot z \\ 1 \end{pmatrix}$$

https://learnopengl.com/Getting-started/Transformations

Transformações

➔ Rotação

Rotation around the X-axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around the Y-axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around the Z-axis:

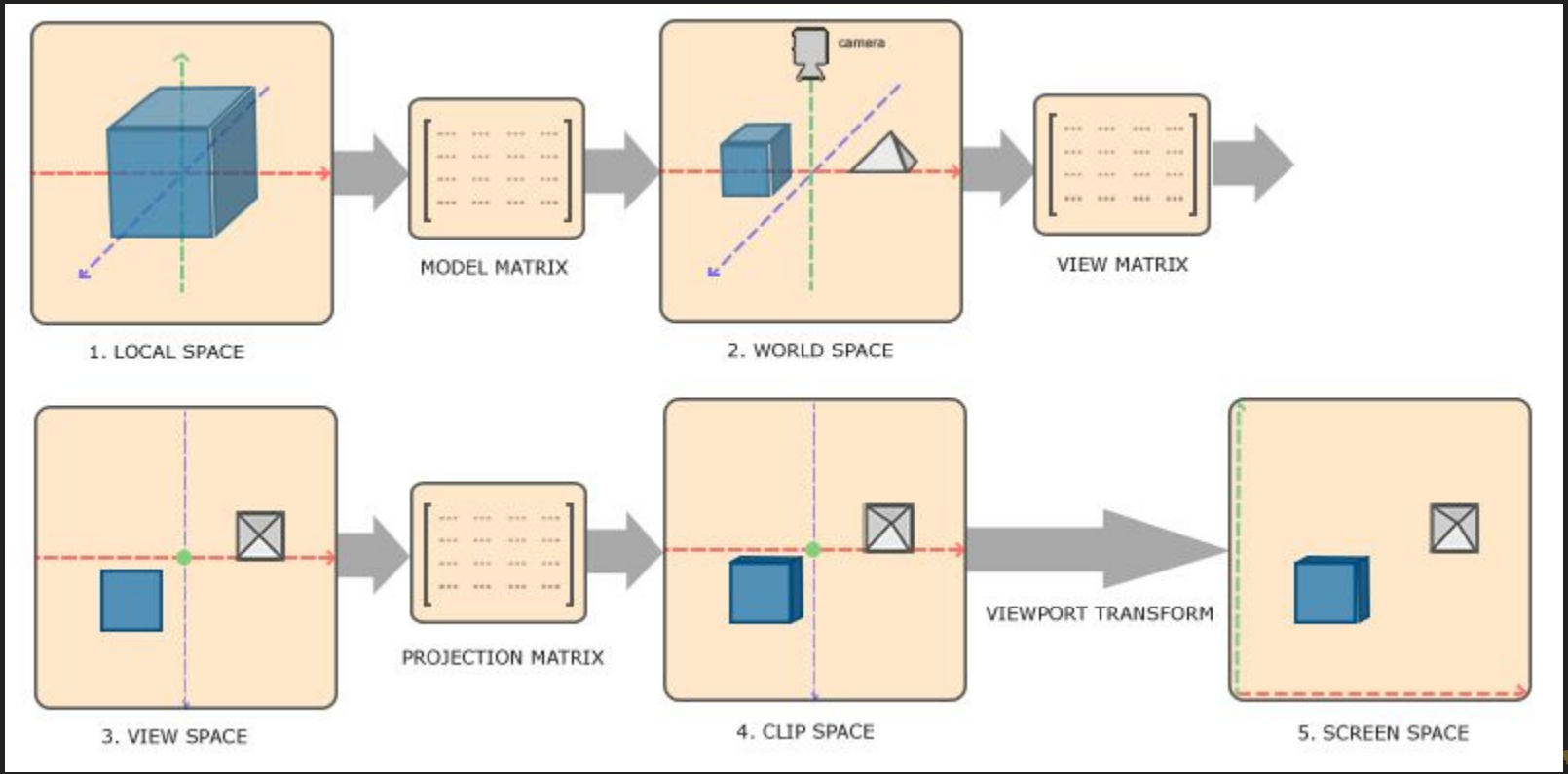
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

https://learnopengl.com/Getting-started/Transformations

Sistemas de Coordenadas

- Algumas matrizes vão nos ajudar a transformar os vértices de um objeto até o espaço de visão do monitor
- Pode parecer complicado mas é extremamente necessário.
- Pense que você está mudando a origem de referência, a base

Sistemas de Coordenadas



https://learnopengl.com/Getting-started/Coordinate-Systems

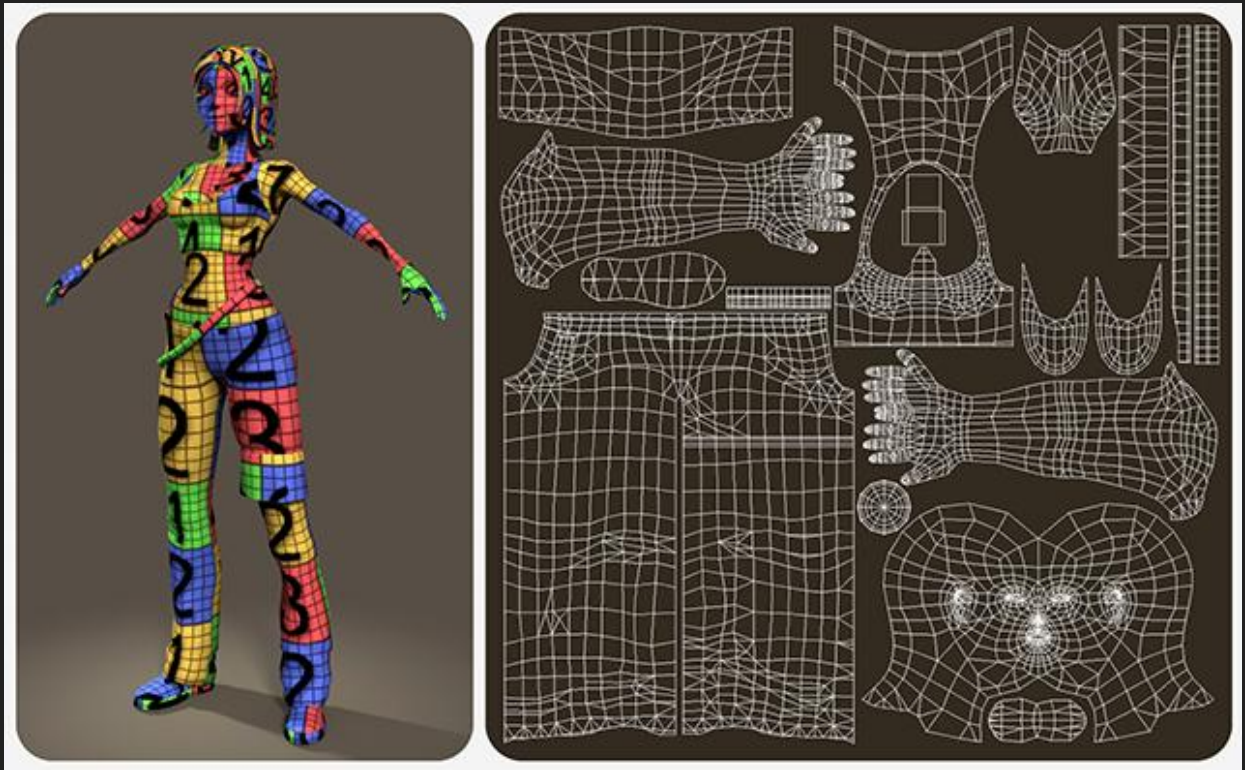
4. Modelos 3D

- Texturas
  - São imagens que vão para a memória da placa de vídeo
    - Toda imagem vira uma textura, até sprites
  - Mapeamento UV (S,T,U,V,W,X,Y,Z) é necessário para representar a superfície 3D numa imagem 2D
  - Filtro é uma interpolação para suavizar os pixels entre os vértices

Imagem (Textura)

https://www.pinterest.com/pin/429953095648140137/?from_navigate=true

Mapa UV



http://cgi.tutsplus.com/articles/game-character-creation-series-kila-chapter-3-uv-mapping--cg-26754

6. Renderização

- ➔ Filtros para escolher a melhor textura para dado pixel
- ➔ Dois modos comuns
  - ◆ Nearest (esquerda) seleciona o pixel que possui centro mais próximo da coordenada da textura
  - ◆ Filtro bilinear (direita) valor interpolado dos texels vizinhos da textura

http://learnopengl.com/#!Getting-started/Textures

Quick Tips!

Ferramentas interessantes para arte 3D

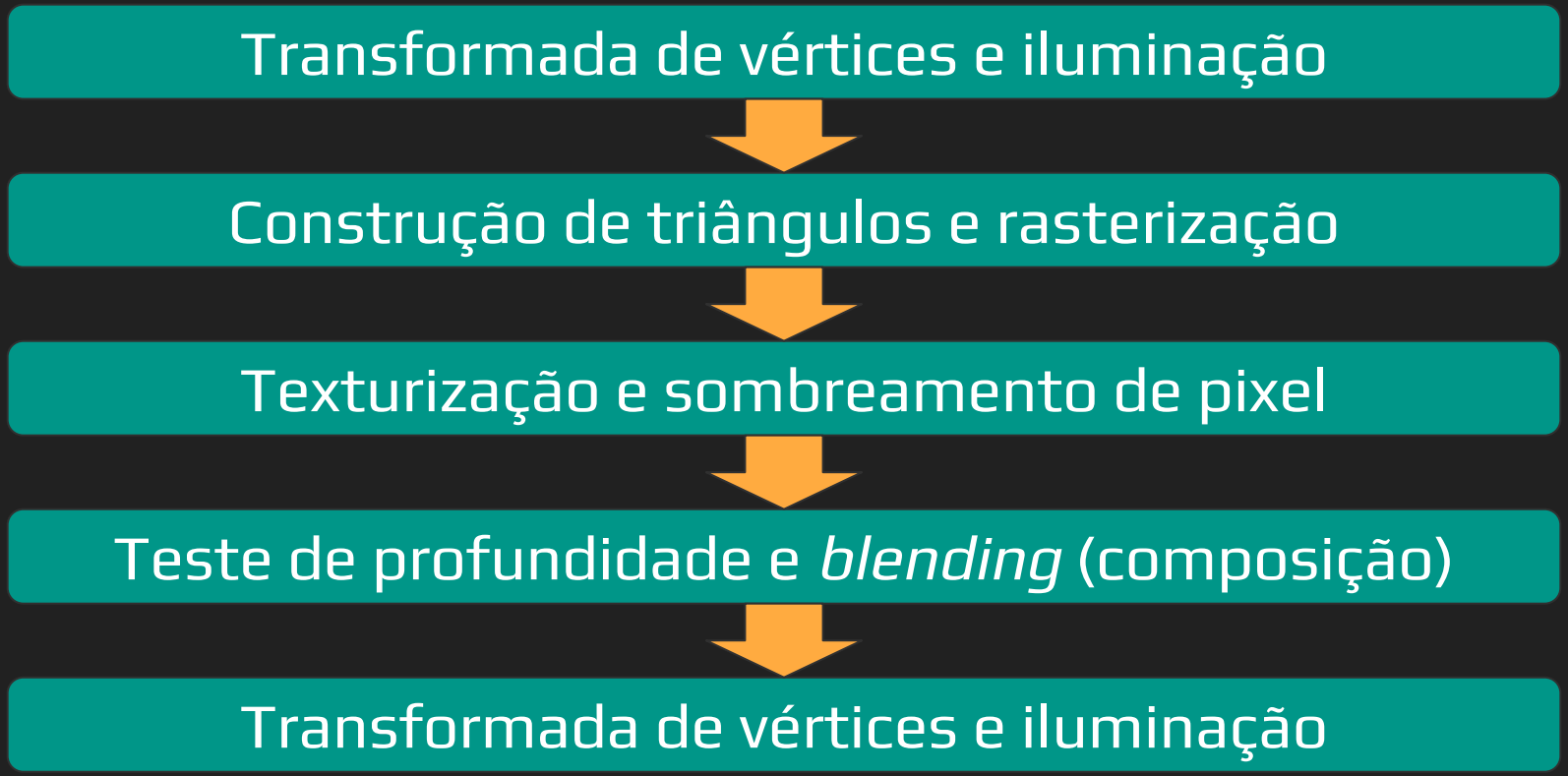
- Blender (Open Source)
  - Propósitos gerais (Até edição de vídeo)
- Autodesk Maya
  - Propósitos gerais
- Houdini
  - Animação, focada em geração procedural
- ZBrush
  - Foco em escultura, integração com várias ferramentas

5. Pipeline & Hardware

5. Pipeline & Hardware

- Pipeline:
  - Encadeamento de comandos
  - Ordem na qual os comandos são executados
- Antigamente:
  - Chips, placas e/ou unidades distintas por estágio
  - Fluxo de dados fixo pelo pipeline
  - Hardware costumava seguir isso:

5. Pipeline & Hardware



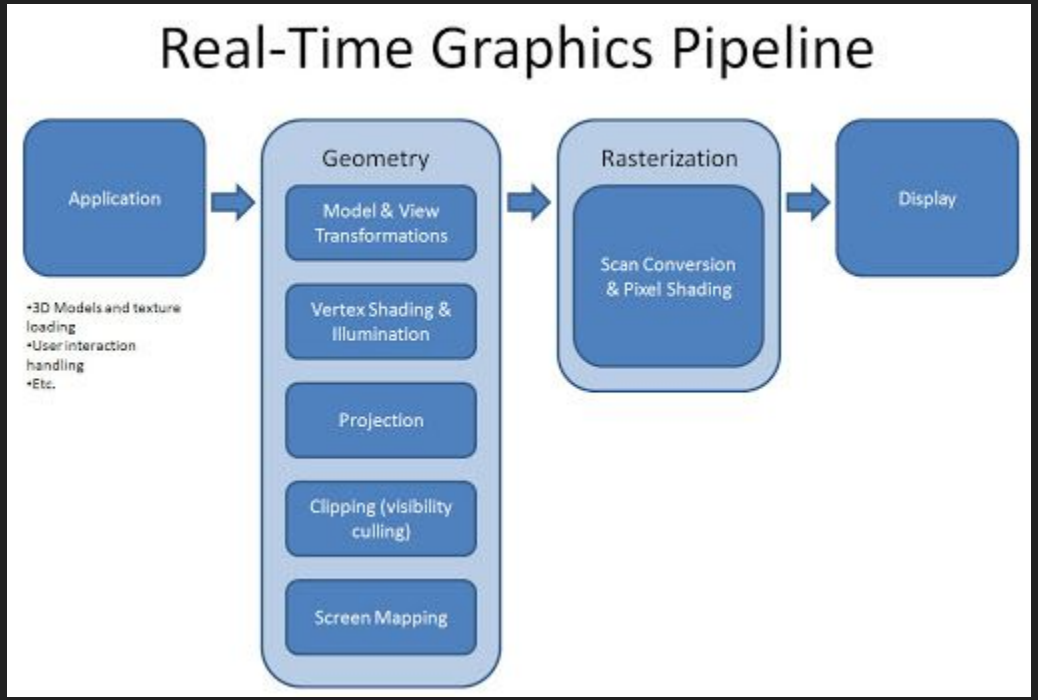
5. Pipeline & Hardware

- ➔ Atualmente
  - GPUs totalmente programáveis
  - Arquitetura unificada
  - Programável em C
  - Fluxo de dados arbitrário
  - Modelo de programação de múltiplos propósitos

5. Pipeline & Hardware

- ➔ Atualmente
  - ◆ Hardware de propósito específico
  - ◆ Threading e pipelining gerenciados por hardware

5. Pipeline & Hardware

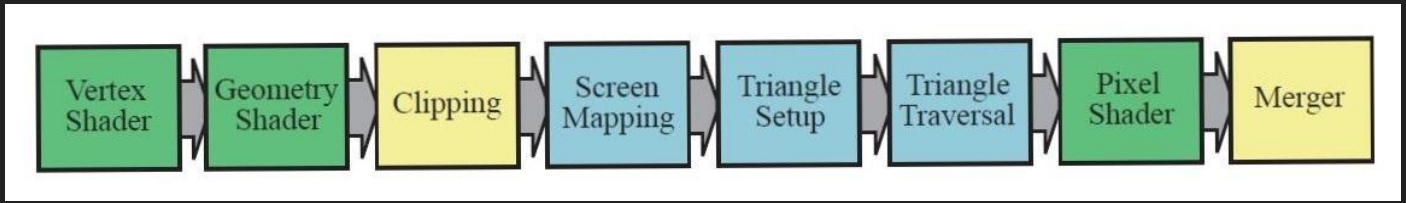


http://www.cgchannel.com/2010/11/cg-science-for-artists-part-2-the-real-time-rendering-pipeline/

6. Rendering Pipeline

6. Rendering Pipeline

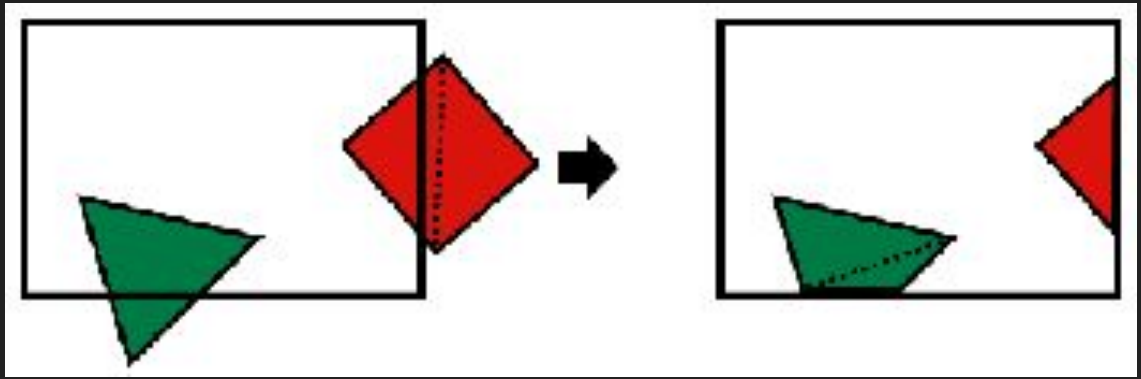
- Etapas fixas no hardware em amarelo
- Etapas configuráveis mas não programáveis em azul
- Etapas programáveis em verde (shaders)
  - Vertex Shader
  - Geometry Shader
  - Fragment (Pixel) Shader



https://www.realtimerendering.com/

6. Rendering Pipeline

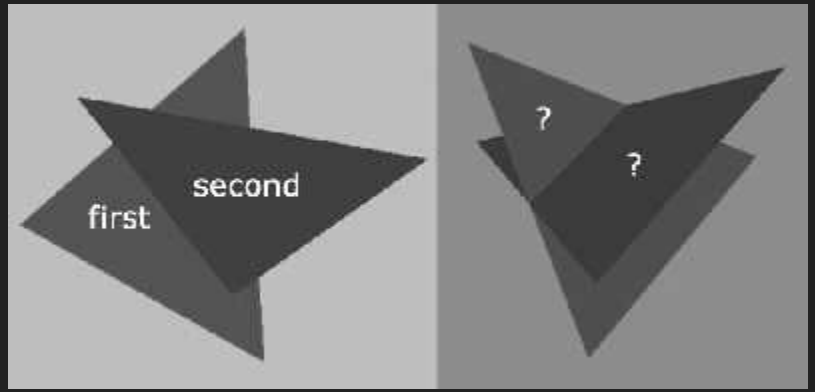
- Clipping (recorte)
  - Recorta polígonos ou pedaços fora da visão da tela



http://n63.icequake.net/doc/n64intro/kantan/images/1-2-3-5.gif

6. Rendering Pipeline

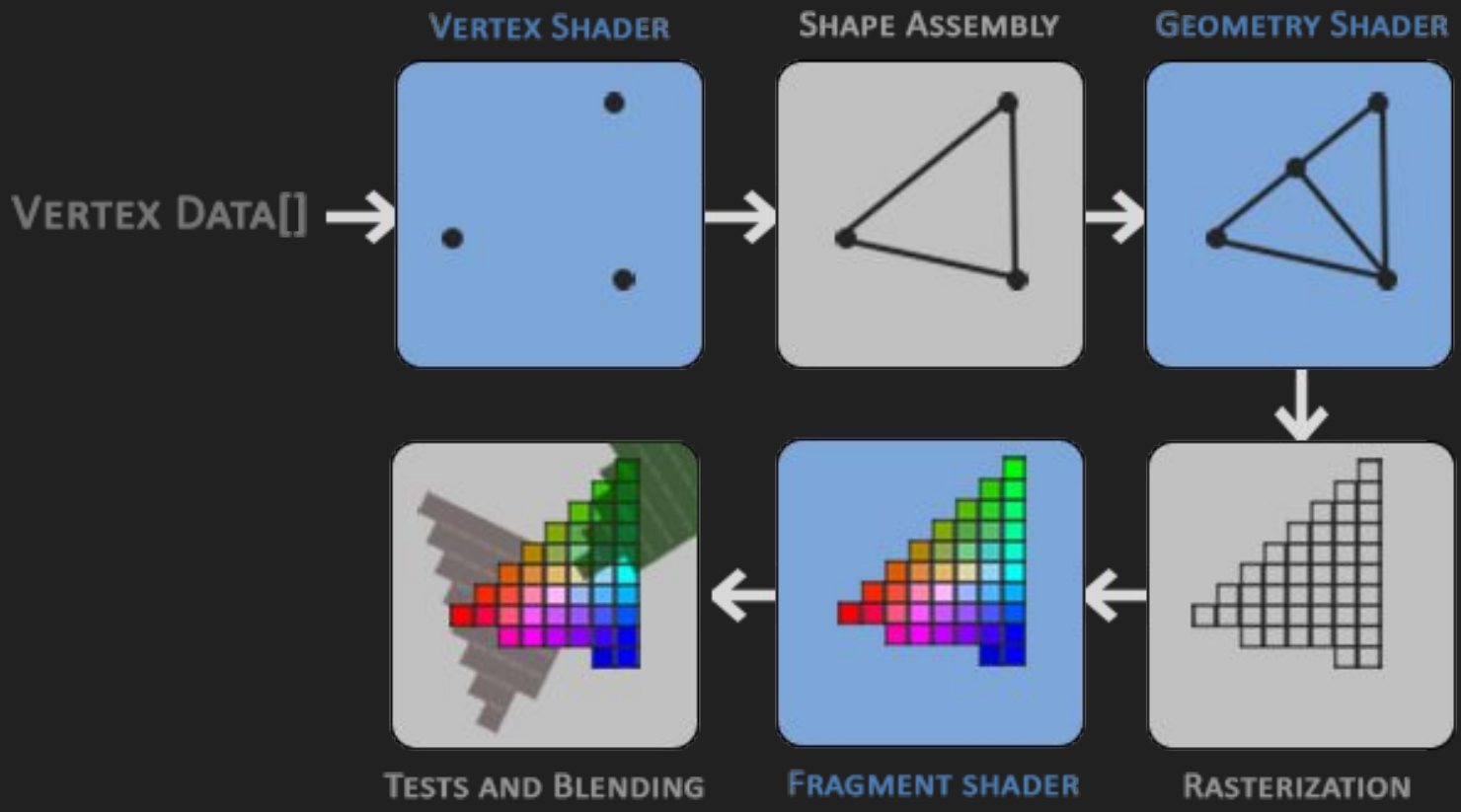
- Z-buffer e z-test
  - Literalmente a componente Z dos fragmentos
  - São úteis e configuráveis, importantes para interação entre fragmentos, principalmente alpha blending



6. Rendering Pipeline

- Shaders
  - São literalmente programas (instruções) que passamos para a placa de vídeo
  - GPU compila e guarda o programa na memória (limitações)
    - Por isso alguns jogos demoram para iniciar
  - Especificações DirectX e OpenGL tentam padronizar

6. Rendering Pipeline



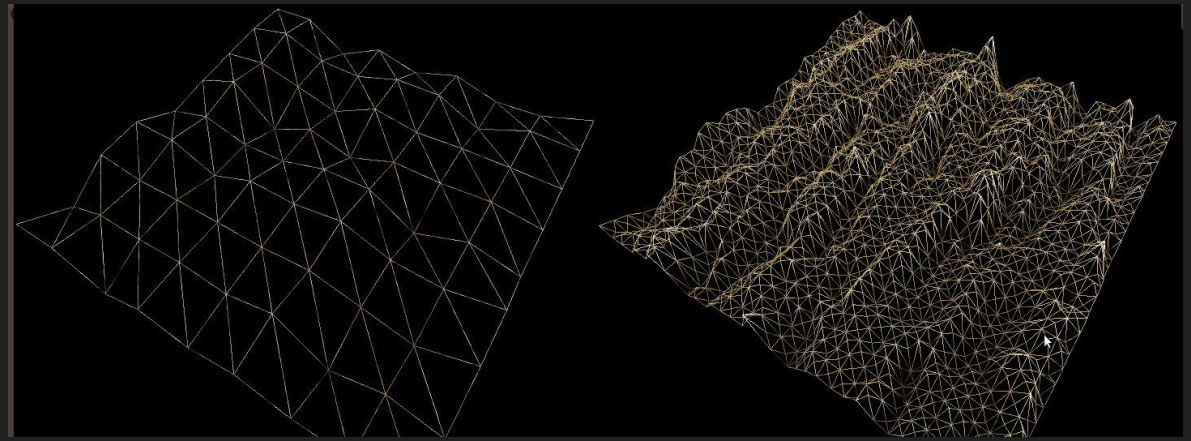
http://learnopengl.com/#!Getting-started/Hello-Triangle

6. Renderização

- Vertex Shader
  - Transformações de mundo
  - Matriz MVP (Model View Projection)
  - Mapeamento de coordenadas
- Fragment Shader
  - Nível de pixel/fragmento
  - Definir cor de saída

Geometry Shader

- Geometry Shader
  - Opcional no pipeline
  - Nova representação geométrica sobre a malha

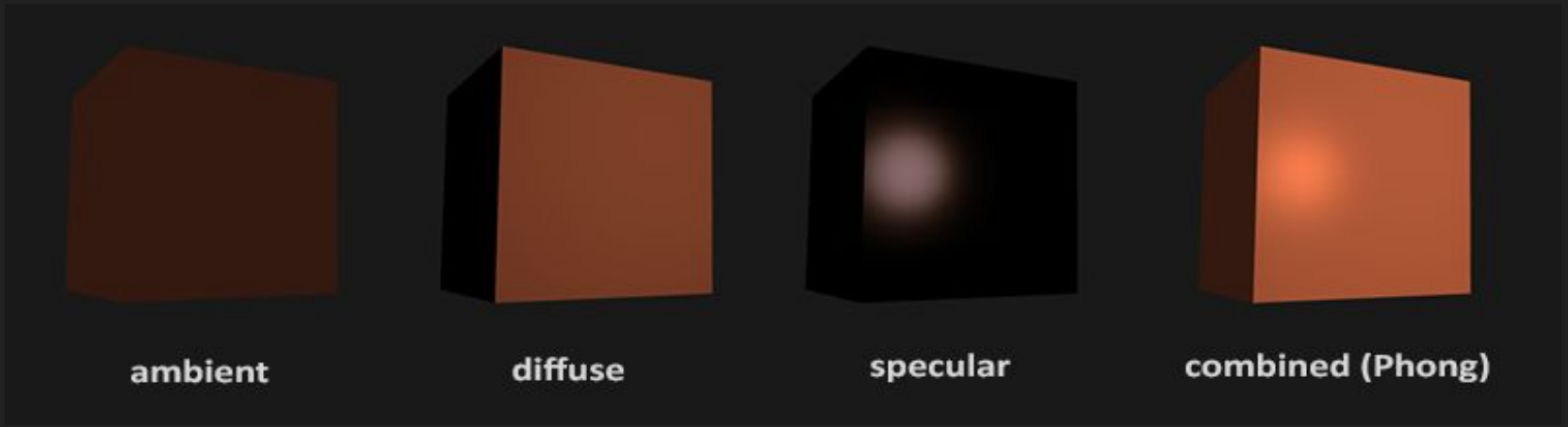


http://irrlicht.sourceforge.net/forum/viewtopic.php?t=35500&start=15

Iluminação Especular e Difusa

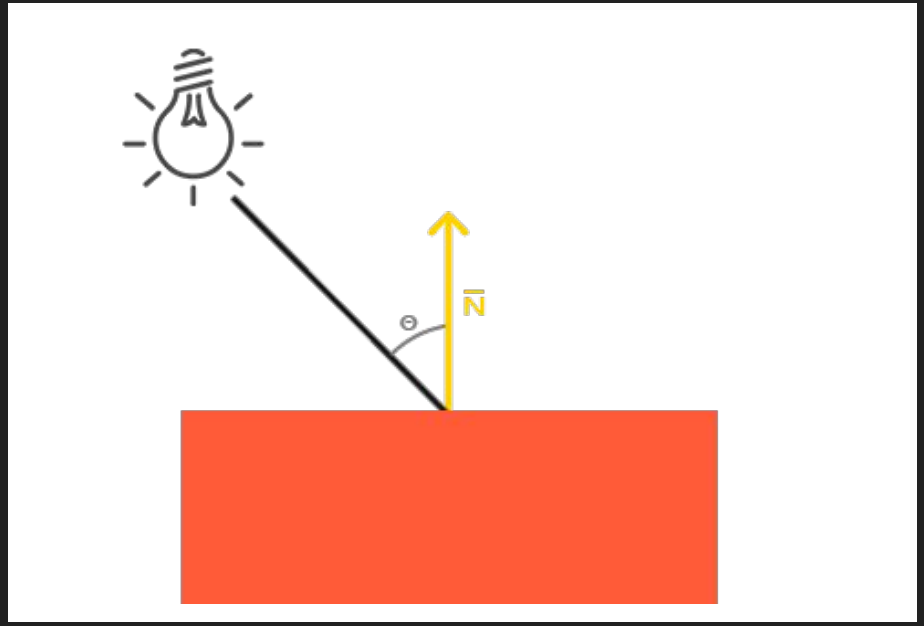
- Difusa
  - Cor que objeto recebe sob luz direta
  - Mais forte na direção da luz e esmaece conforme o ângulo da superfície aumenta
- Especular
  - Cor de destaque de um objeto.
  - Aparece como reflexão da luz na superfície
- https://clara.io/learn/user-guide/lighting_shading/materials/material_types/webgl_materials

Iluminação Especular e Difusa



https://learnopengl.com/Lighting/Basic-Lighting

Iluminação Especular e Difusa



https://learnopengl.com/Lighting/Basic-Lighting

6. Renderização

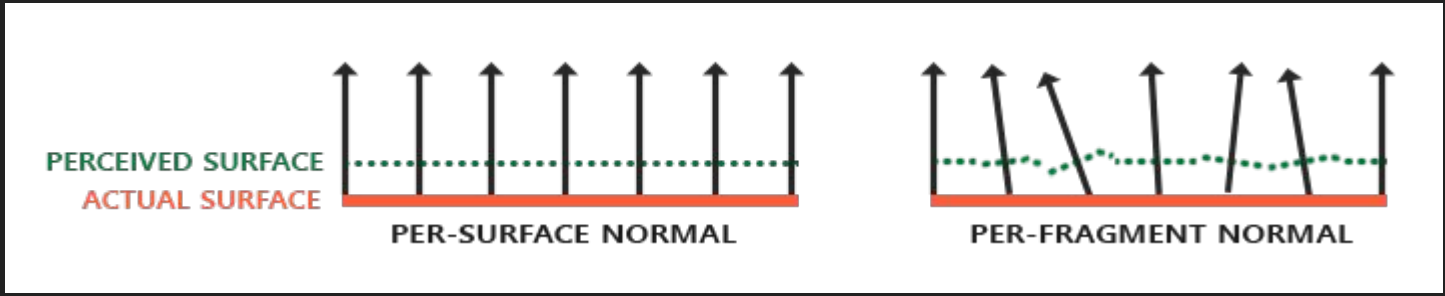
- Materiais
  - Descrevem como o objeto deve se comportar visualmente
  - Pode conter informações como reflexão, transparência, quão metálico, quão liso etc.
  - Depende diretamente dos shaders

6. Renderização

- ➔ Mapeamentos (Mapping)
  - ◆ Texturas
  - ◆ Bump
    - ● Displacement
    - ● Normal
    - ● Parallax
    - ● Height
  - ◆ Cube
  - ◆ Shadow

6. Renderização

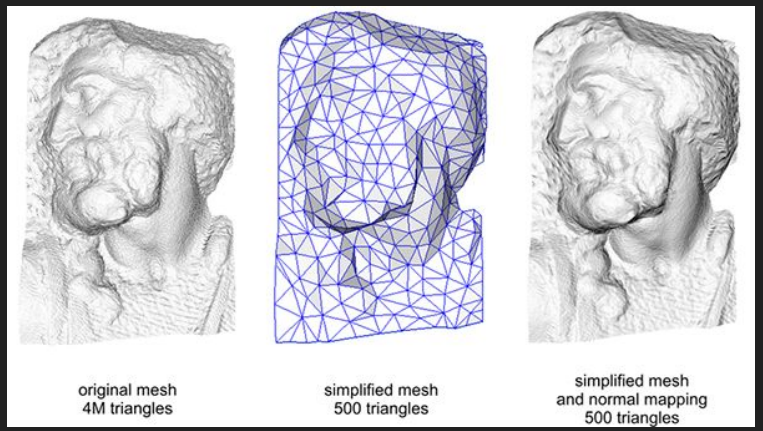
- Normal Map
  - Modifica a luz através da superfície da textura
  - Baseia-se no vetor normal à superfície



https://learnopengl.com/Lighting/Basic-Lighting

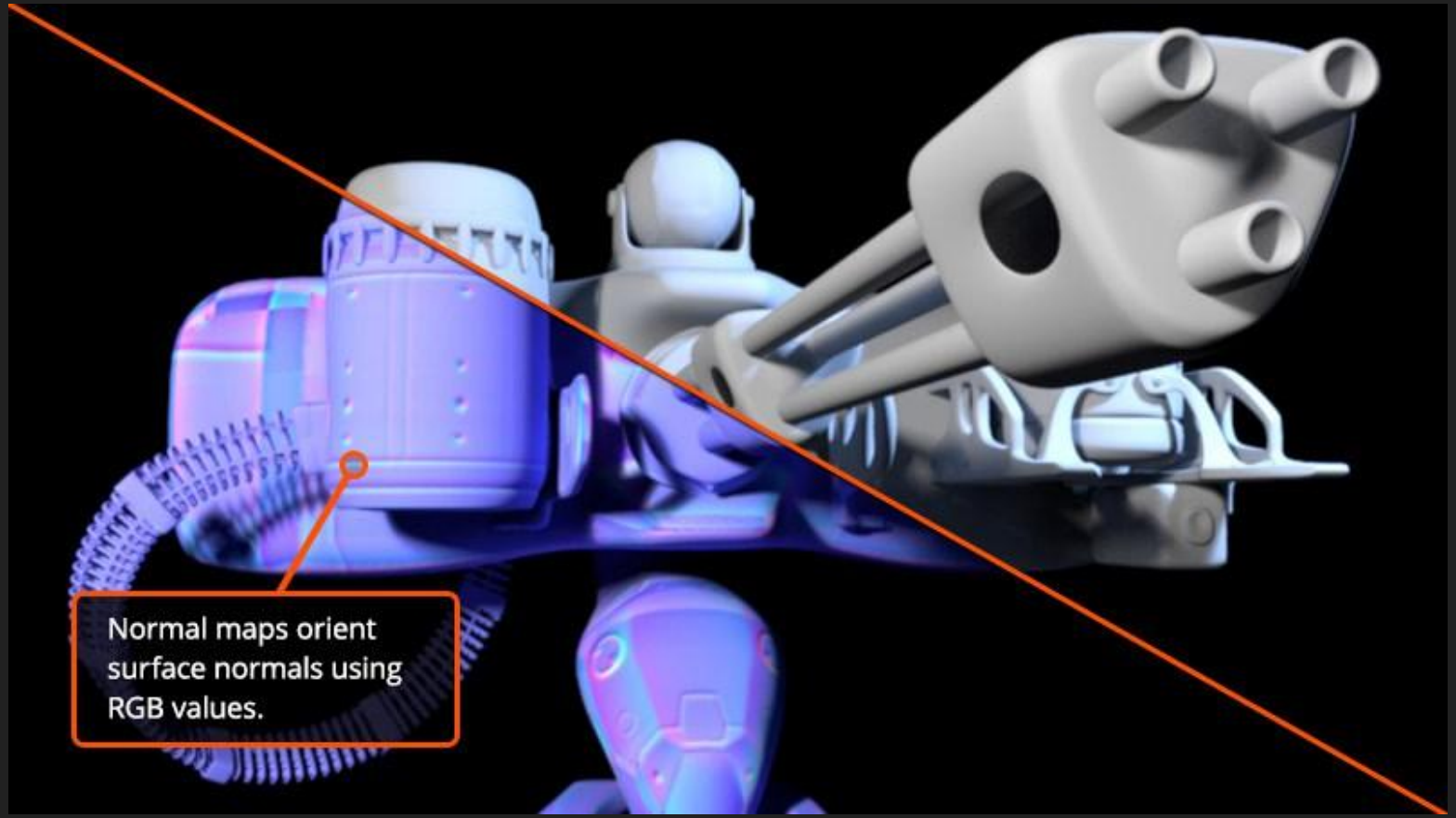
6. Renderização

➔ Normal Map



https://learnopengl.com/img/advanced-lighting/normal_mapping_comparison.png

Normal Map

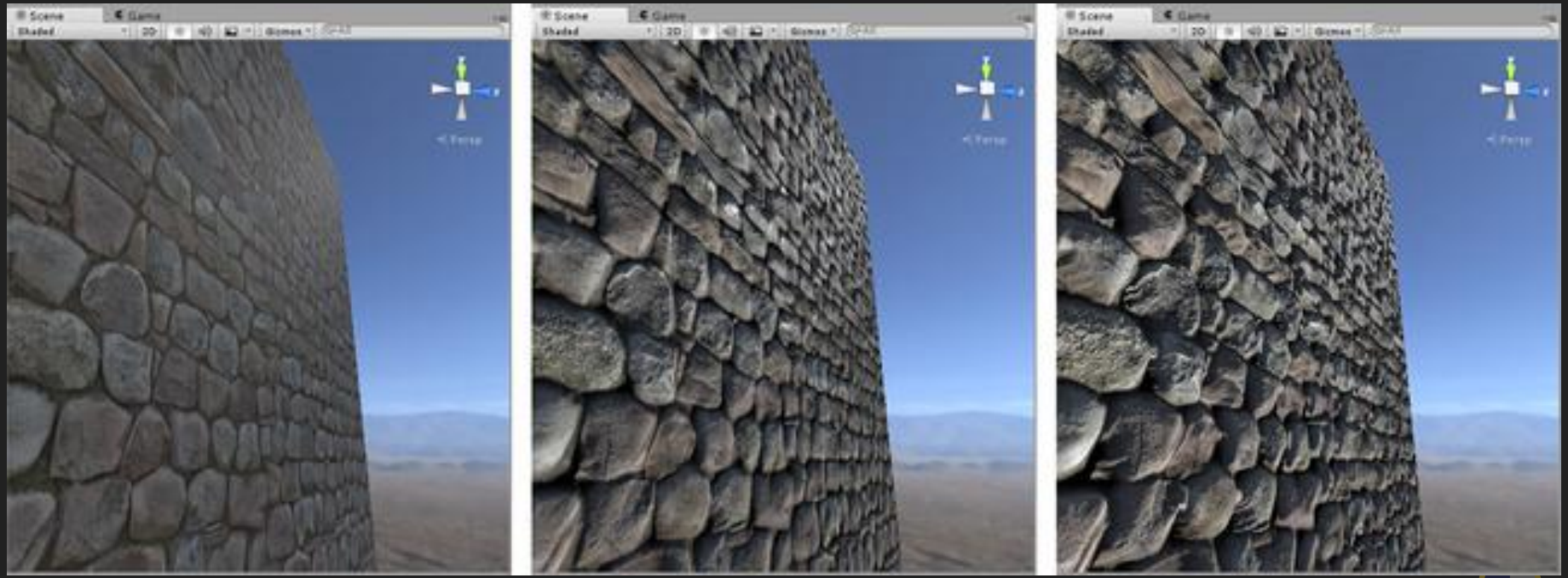


http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

6. Renderização

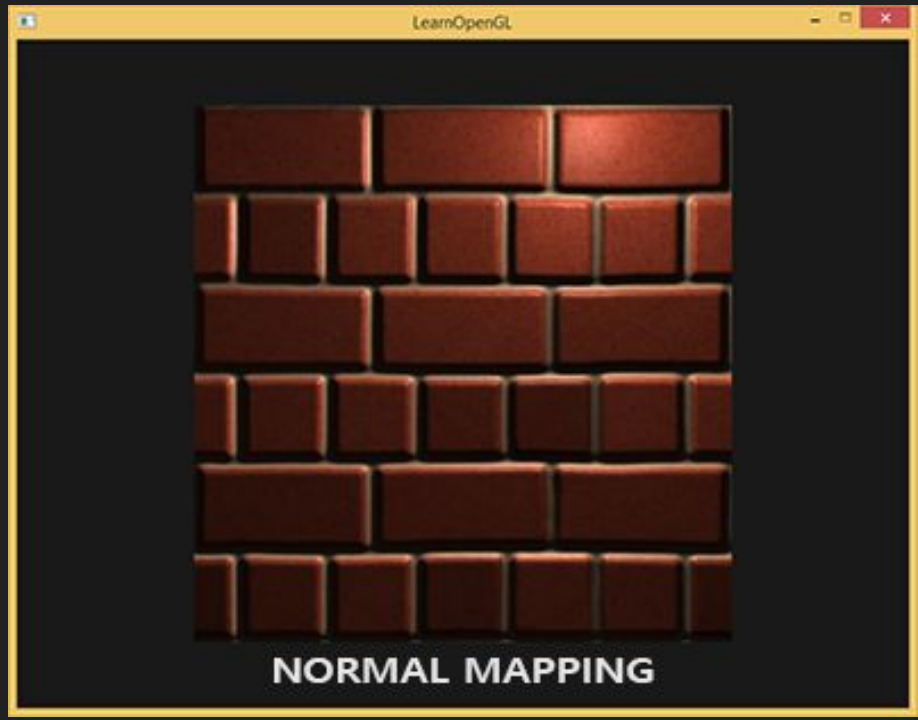
- Heigth ou Parallax map
  - Similar ao normal, mas mais complexo (e mais pesado)
  - Normalmente usado em conjunto com mapas normais
  - Move áreas da textura da superfície visível
    - Alcança um efeito a nível de superfície de oclusão
  - Protuberâncias terão suas partes próximas (frente à câmera) exagerados. E a outra parte reduzida

Heightmap ou Parallax Map



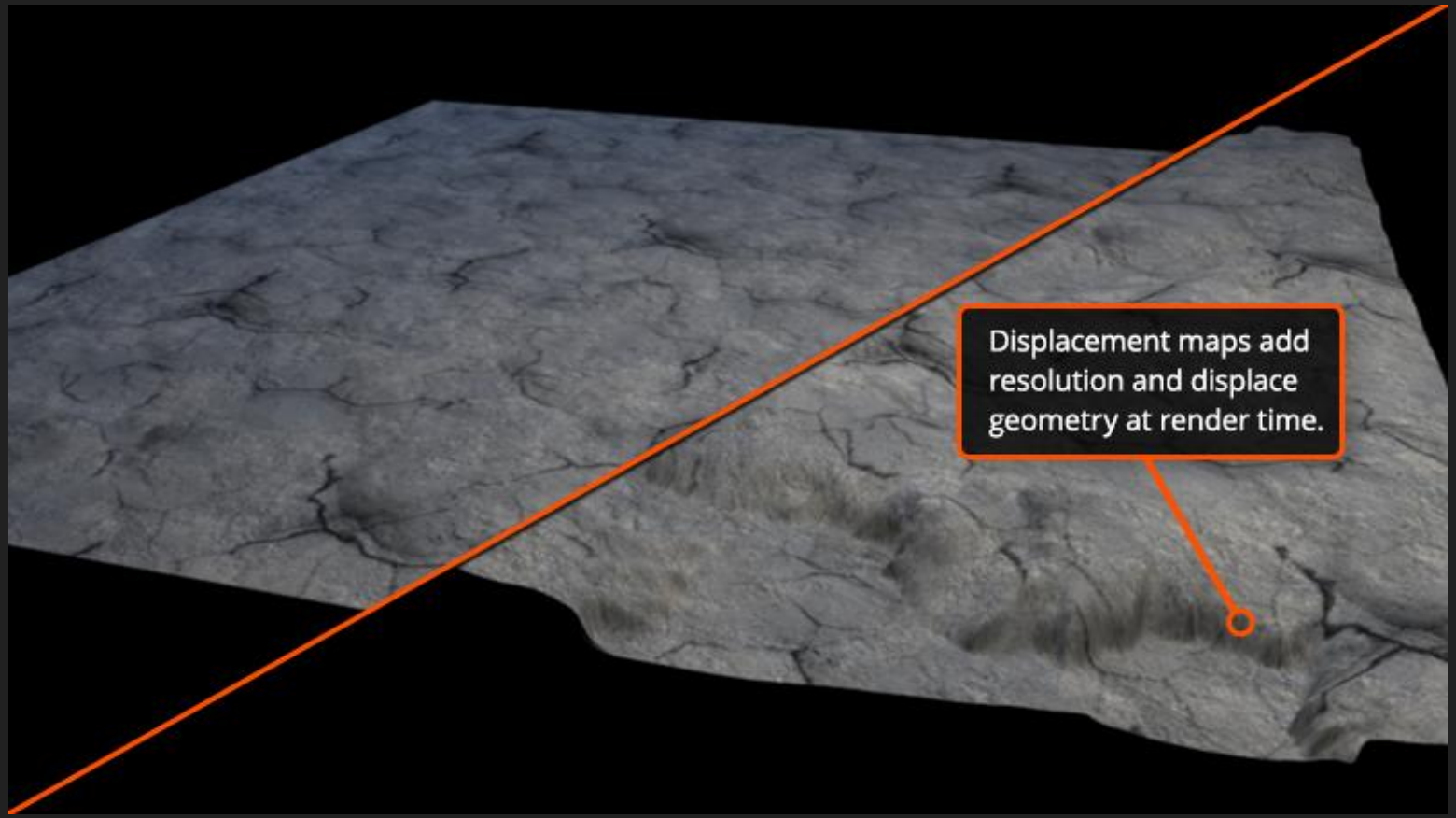
http://docs.unity3d.com/Manual/StandardShaderMaterialParameterHeightMap.html

Normal e Parallax Map



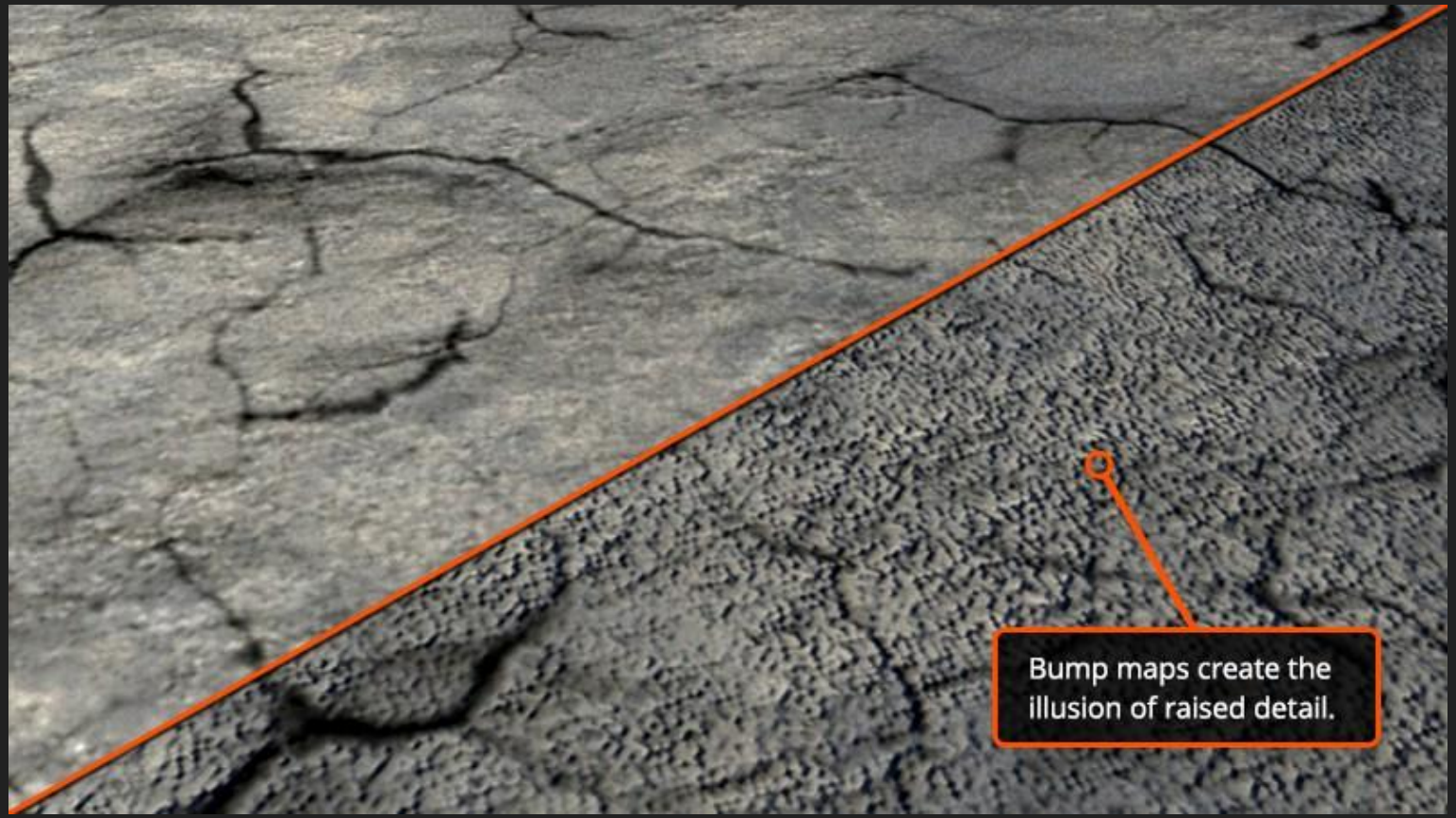
http://learnopengl.com/#!Advanced-Lighting/Parallax-Mapping

Displacement Map



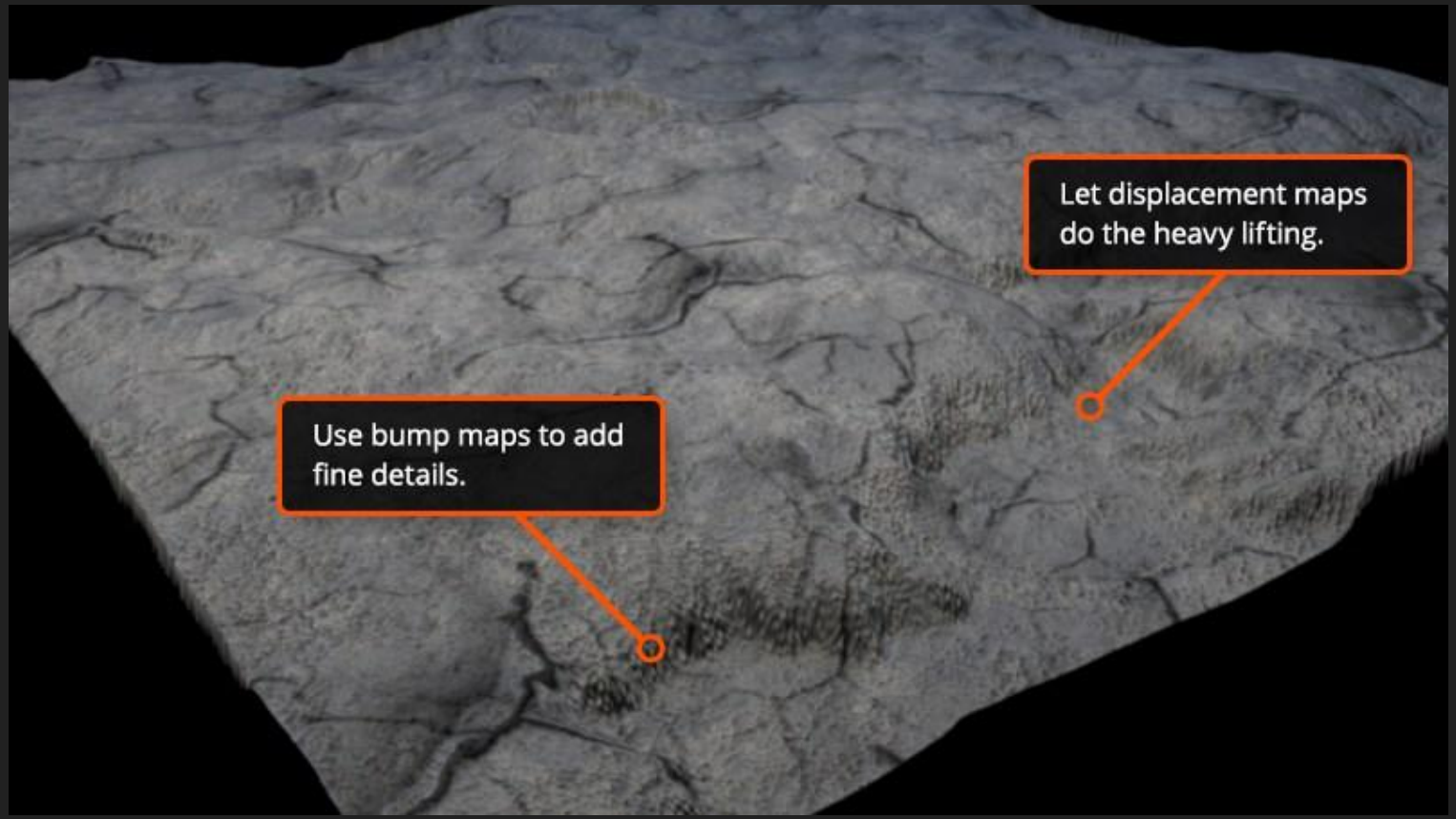
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Bump Map



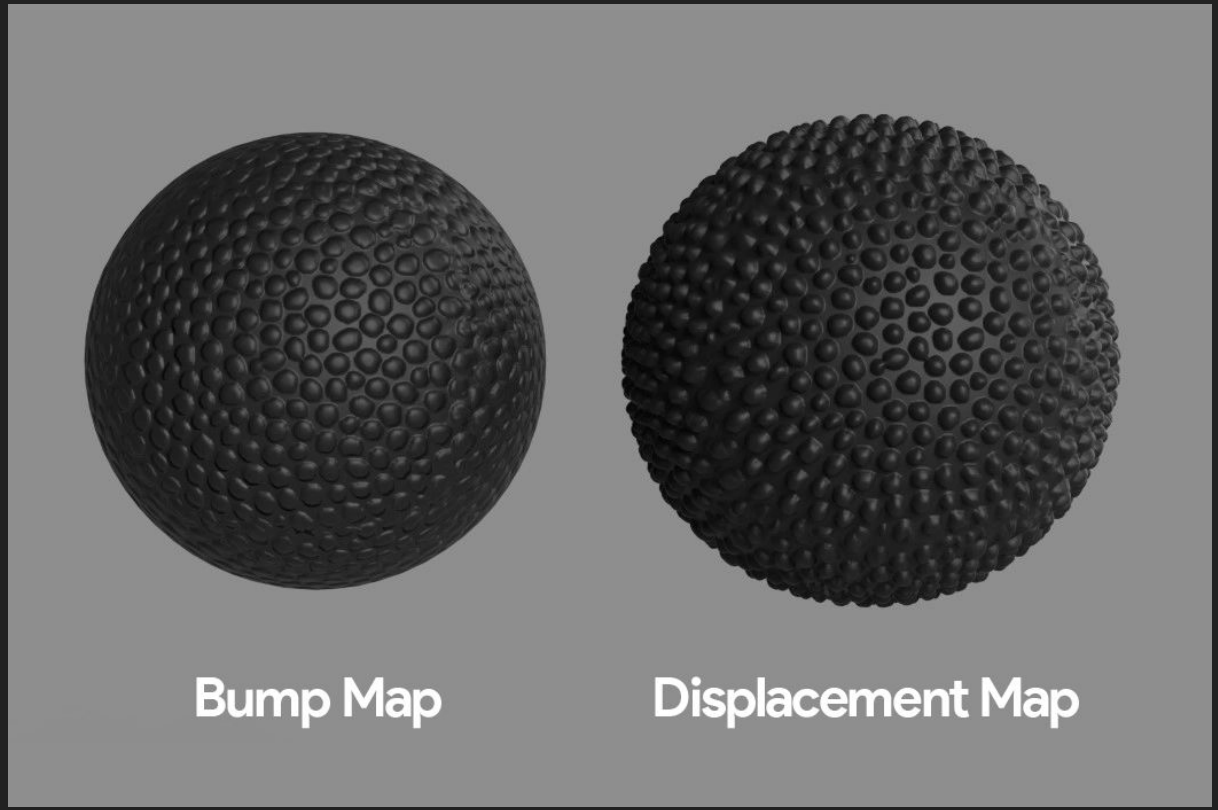
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Bump e Displacement Maps



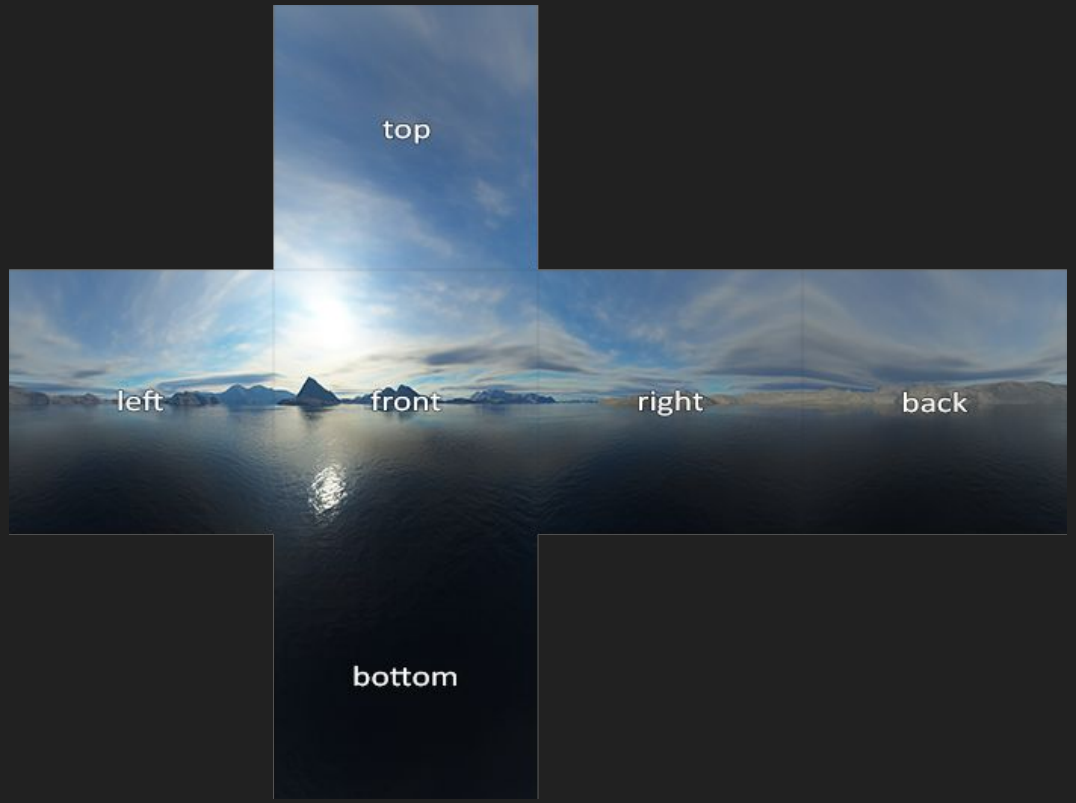
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Bump e Displacement Maps



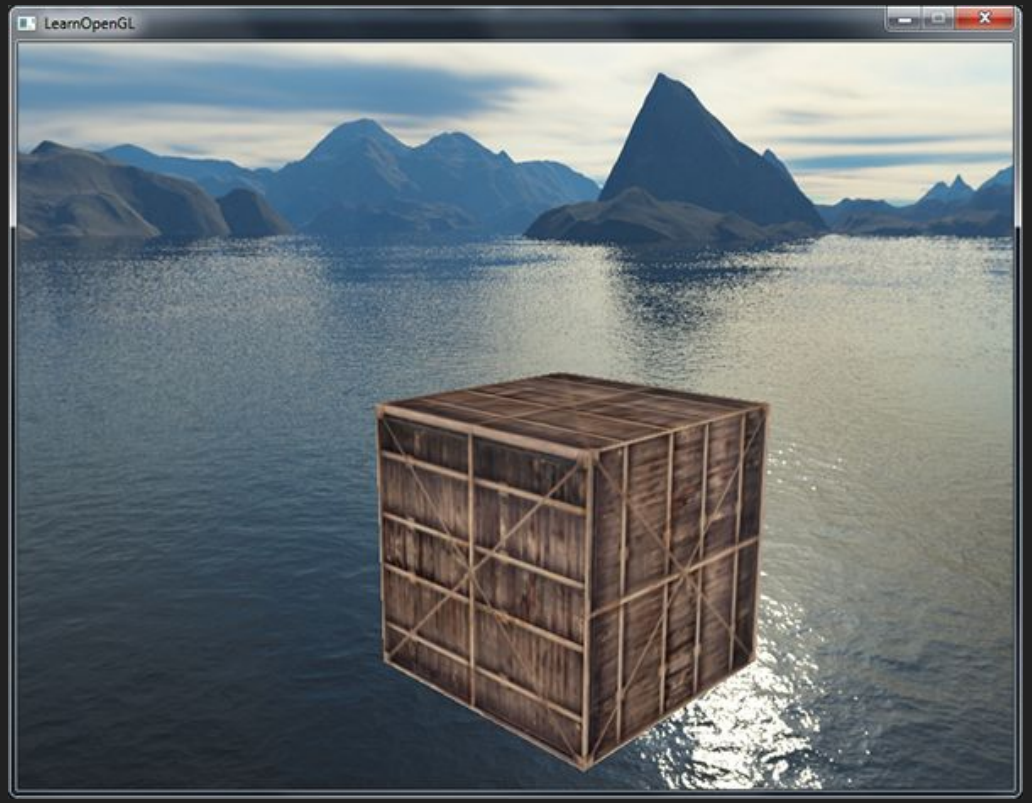
https://www.yankodesign.com/2019/04/13/creating-realistic-textures-with-displacement-maps-in-keyshot-8/

Cube Map



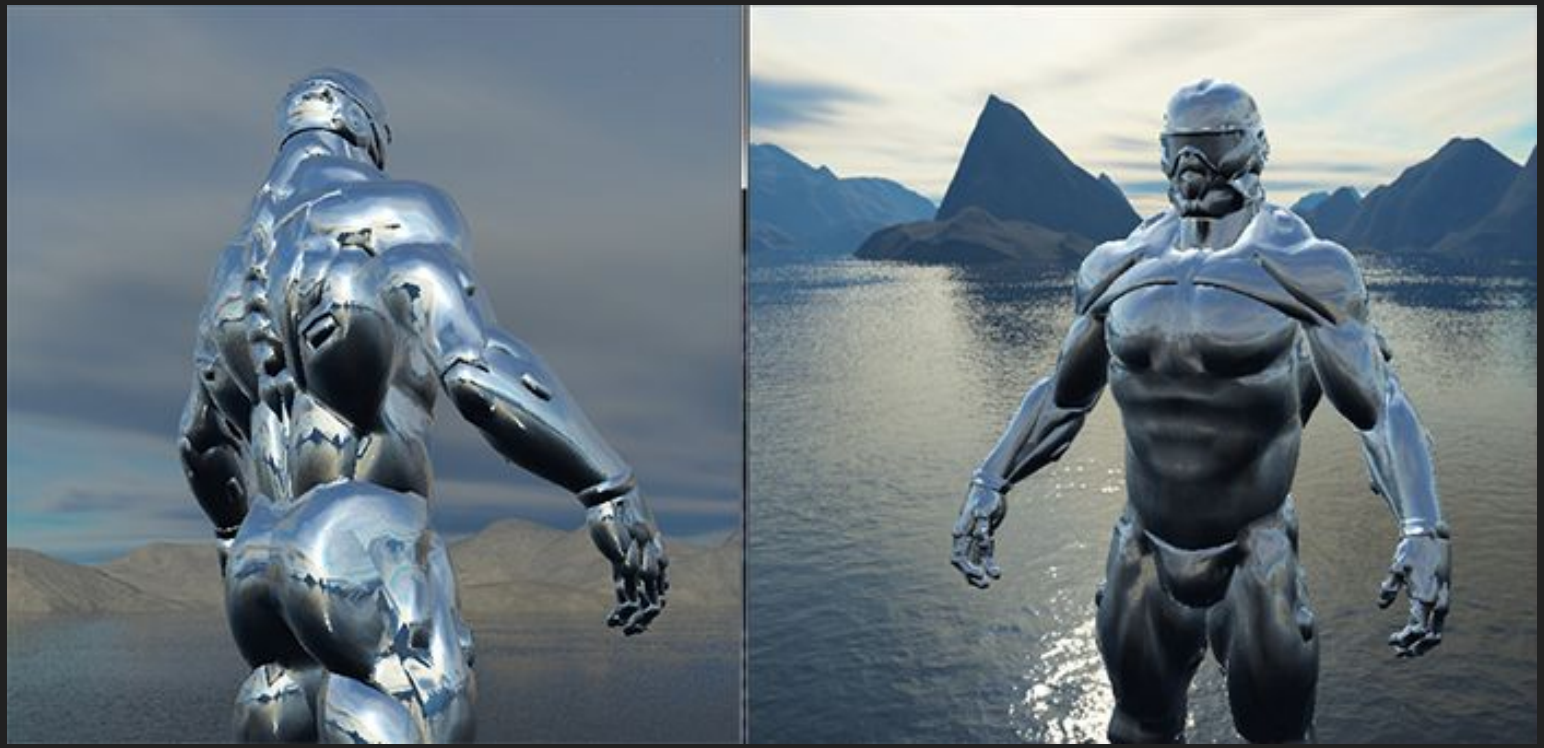
http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Skybox



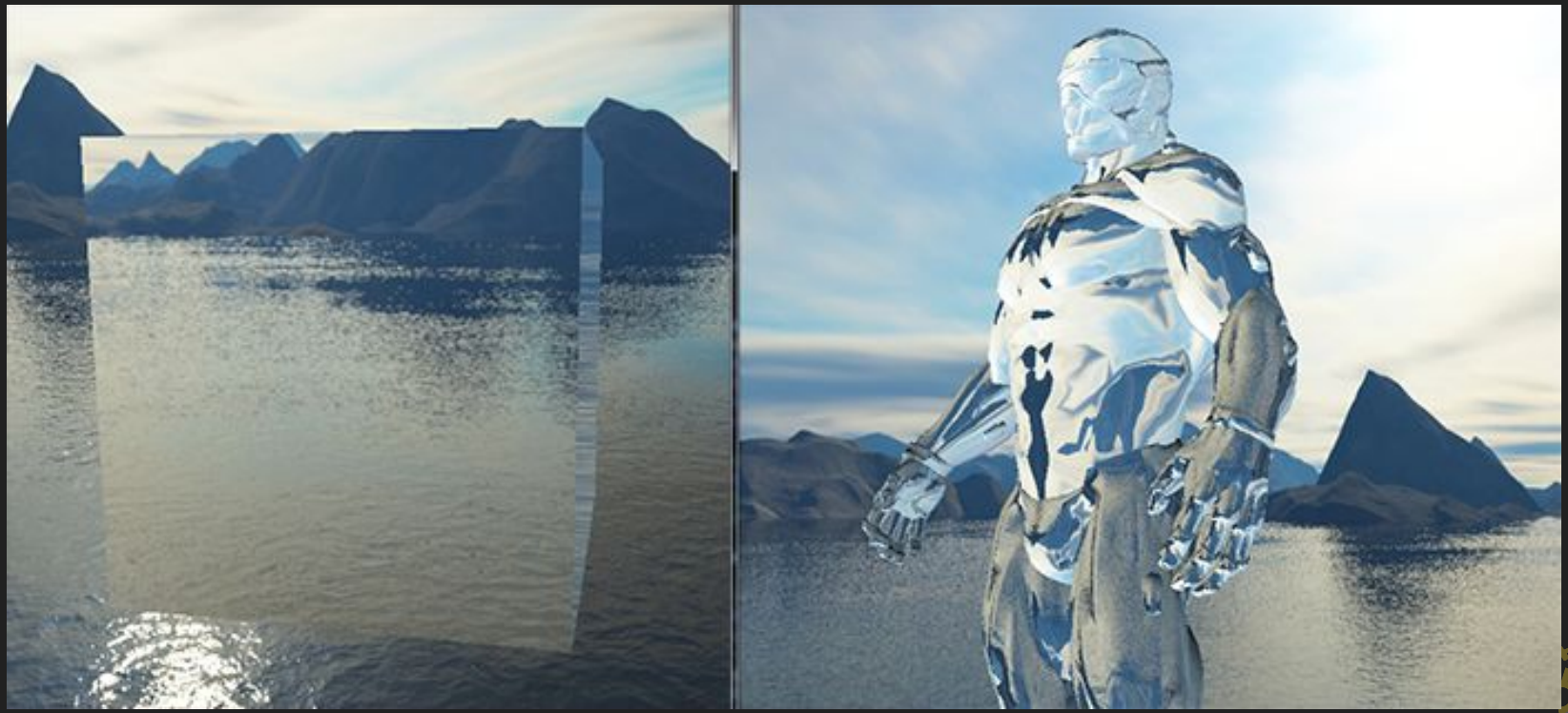
http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Reflexão



http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Refração

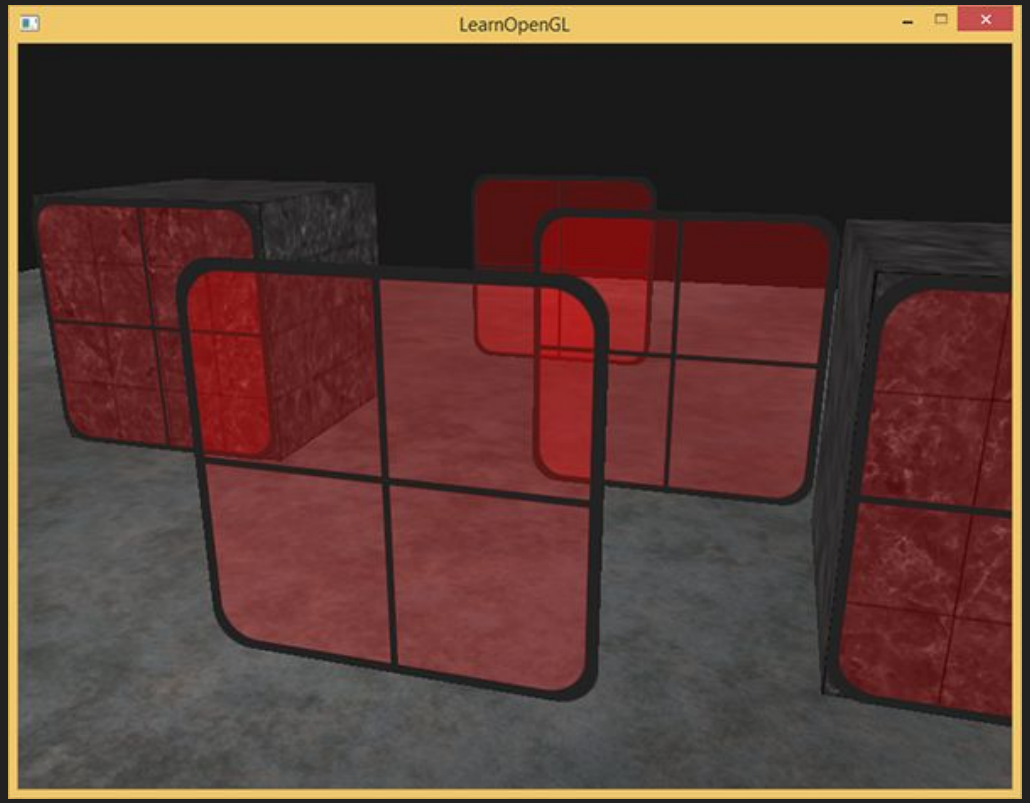


http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

6. Renderização

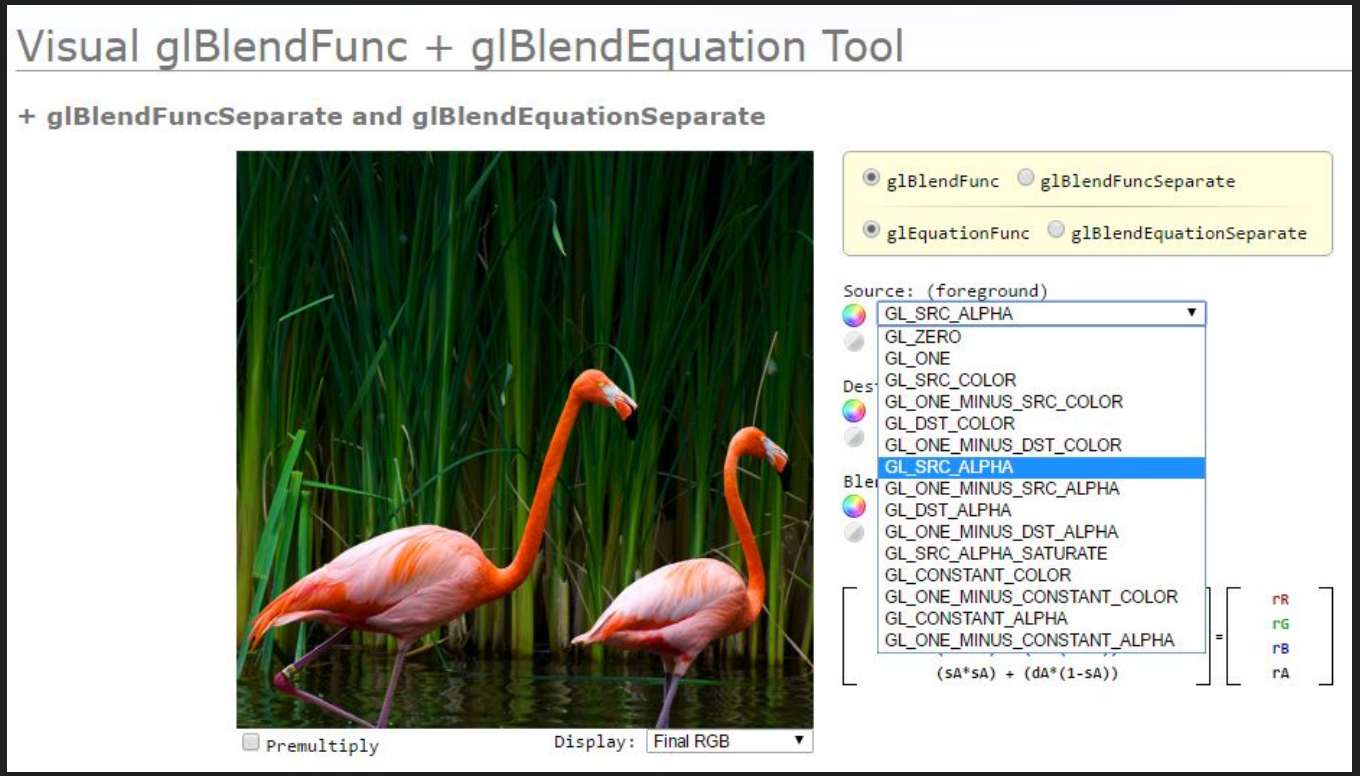
- Blending
  - Aritmética entre Alpha
  - Transparência
  - Translucência

Blending



http://learnopengl.com/#!Advanced-OpenGL/Blending

Blending

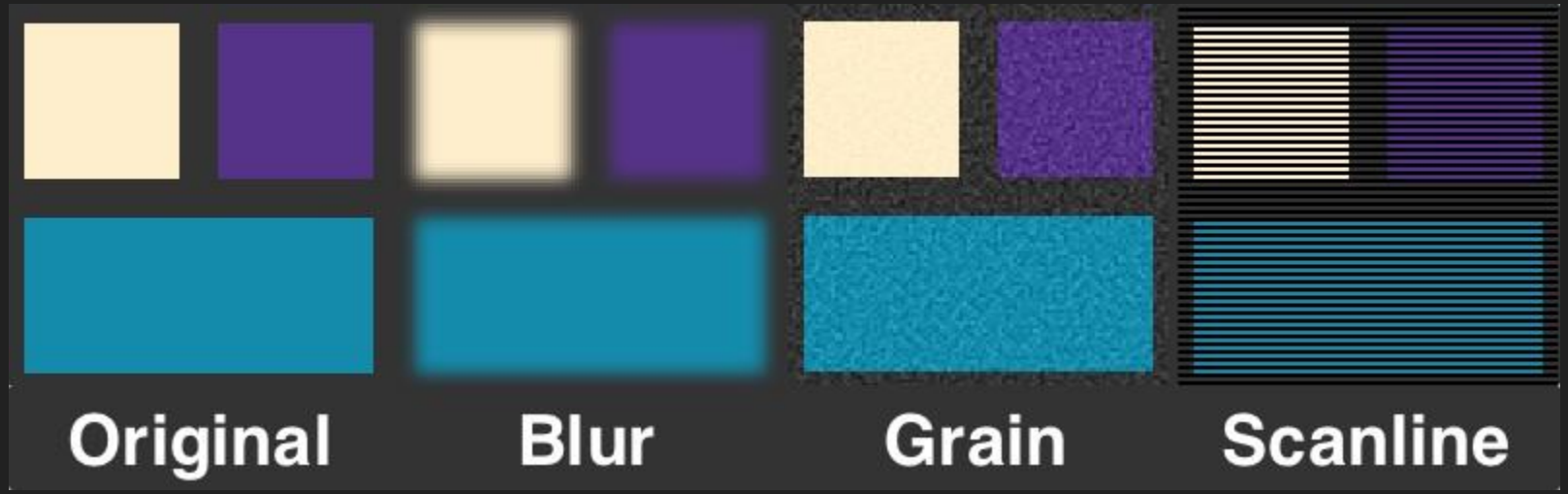


http://andersriggelsen.dk/glblendfunc.php

6. Renderização

- Post Processing
  - Fotografia
  - Aplicar shader no framebuffer
  - Médio custo
  - Realismo

Post Processing



http://haxepunk.com/documentation/tutorials/post-process/

Bloom

http://haxepunk.com/documentation/tutorials/post-process/

HDR

http://gamesetwatch.com/

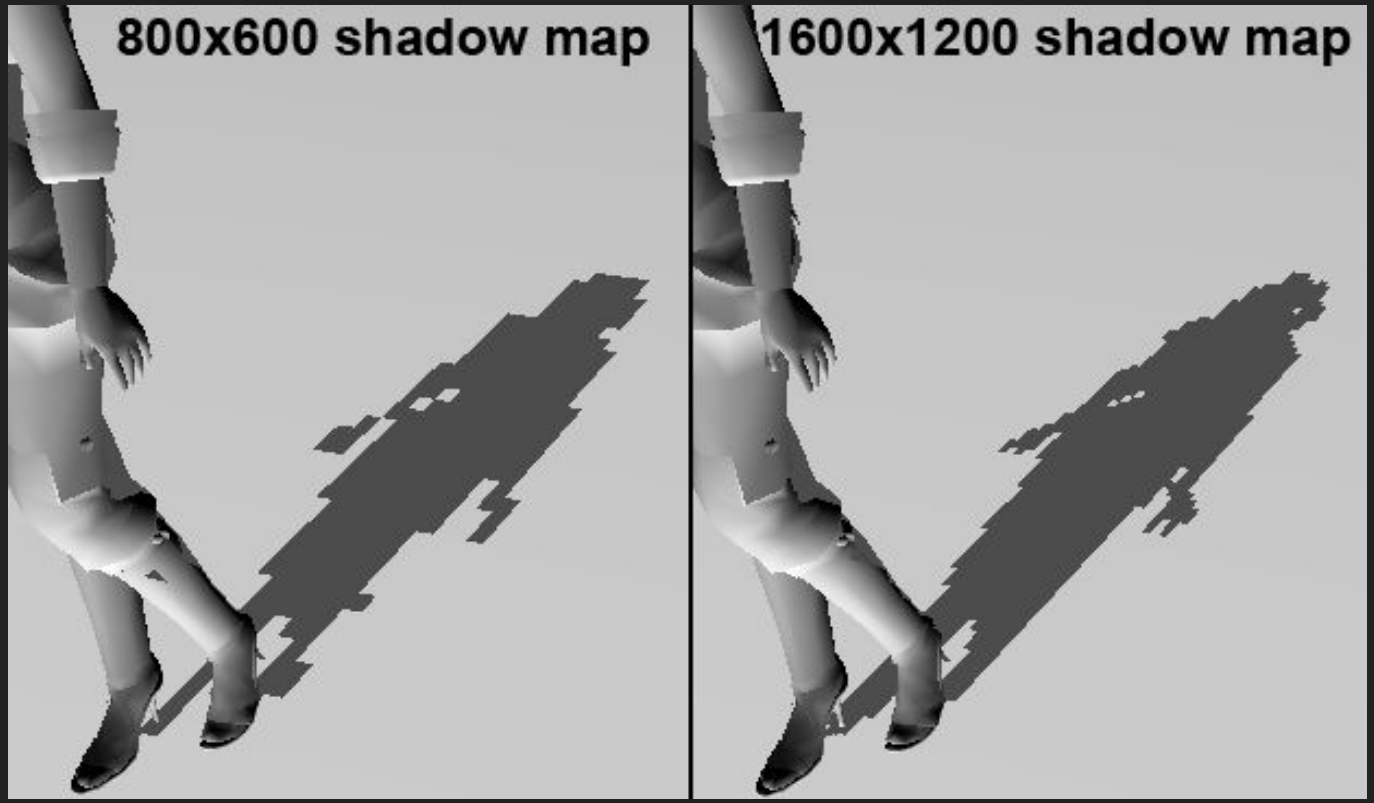
Depth of Field

https://steamcommunity.com/sharedfiles/filedetails/?id=134522361

6. Renderização

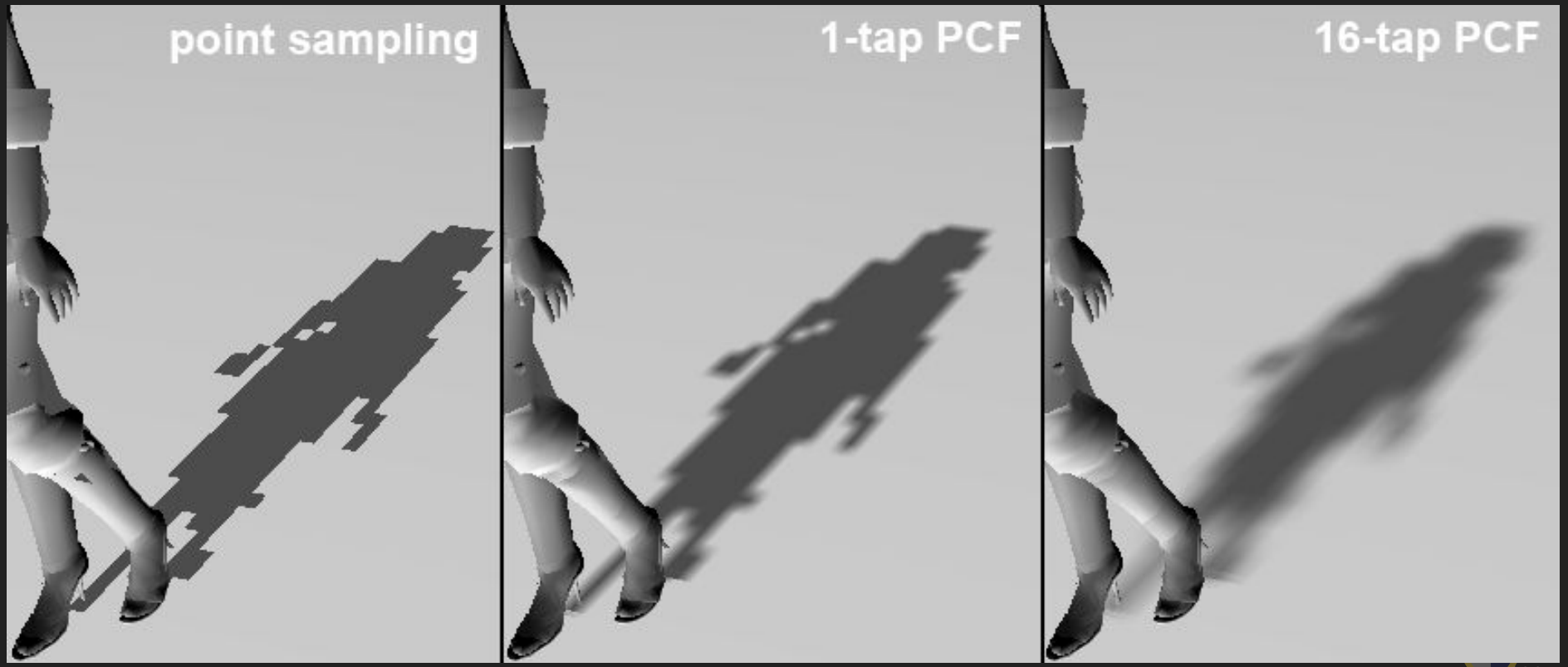
- Sombra
  - Alto custo
  - Mapeamento
  - Aproximação
  - Anti-aliasing

Shadow Map



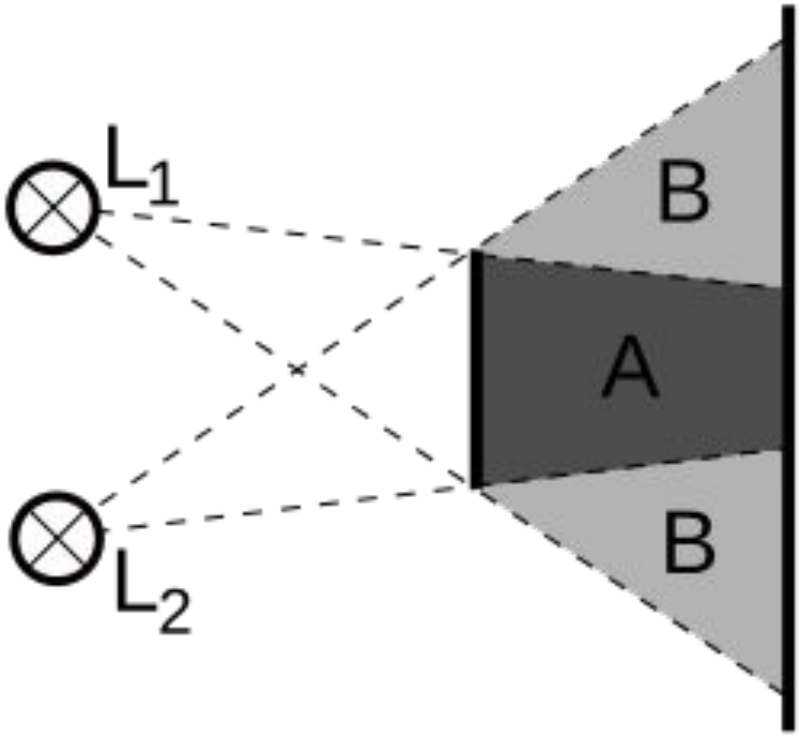
https://takinginitiative.wordpress.com/2011/05/25/directx10-tutorial-10-shadow-mapping-part-2/

Shadow Map



https://takinginitiative.wordpress.com/2011/05/25/directx10-tutorial-10-shadow-mapping-part-2/

Soft Shadow e Penumbra



https://en.wikipedia.org/wiki/Umbra,_penumbra_and_antumbra

6. Renderização

- http://acko.net/files/fullfrontal/fullfrontal/webglmath/online.html

Ray tracing

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

Ray tracing

- Simulação do trajeto feito pelos raios de luz dos objetos até olhos do jogador
- https://www.youtube.com/watch?v=25N7l8vqepQ&ab_channel=Imperoland
- https://www.youtube.com/watch?v=6owqvwty_UQ&ab_channel=KhanAcademyLabs

Dúvidas?

Referências

Referências

[1]http://www.nvidia.com/content/nvision2008/tech_presentations/Technology_Keynotes/NVISION08-Tech_Keynote-GPU.pdf
[2]http://n64.icequake.net/doc/n64intro/kantan/
[3]https://en.wikipedia.org/wiki/Microcode
[4]https://en.wikipedia.org/wiki/Texel_(graphics)
[5]http://www.nvidia.com/object/cuda_home_new.html
[6]https://llpanorama.wordpress.com/2008/05/21/my-first-cuda-program/
[7]http://www.nvidia.com/content/gtc/documents/1055_gtc09.pdf
[8]https://en.wikipedia.org/wiki/Computer_graphics
[9]http://www.graphics.cornell.edu/online/tutorial/
[10]https://en.wikipedia.org/wiki/2D_computer_graphics
[11]https://en.wikipedia.org/wiki/Pixel_art
[12]https://en.wikipedia.org/wiki/Font_rasterization
[13]http://acko.net/files/fullfrontal/fullfrontal/webglmath/online.html
[14]http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Referências

[12]https://en.wikipedia.org/wiki/Sprite_(computer_graphics)
[13]https://en.wikipedia.org/wiki/Vector_graphics
[14]https://en.wikipedia.org/wiki/3D_computer_graphics
[15]https://en.wikipedia.org/wiki/Computer_animation
[16]https://en.wikipedia.org/wiki/Uncanny_valley
[17]https://www.youtube.com/watch?v=eN3PsU_iA80
[18]https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
[19]http://meseec.ce.rit.edu/551-projects/fall2014/3-1.pdf
[20]https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/
[21]https://help.poliigon.com/en/articles/1712652-what-are-the-different-texture-maps-for

Referências Complementares

[1]http://nesdev.com/NESDoc.pdf
[2]http://www.vasulka.org/archive/Writings/VideogameImpact.pdf#page=24
[3]https://en.wikipedia.org/wiki/Real-time_computer_graphics
[4]http://dkc-forever.blogspot.com.br/2015/11/curiosidades-designer-da-rare-revela.html
[5]http://level42.ca/projects/ultra64/Documentation/man/
[6]http://www.nintendoblast.com.br/2014/01/revisitando-os-tempos-aureos-do-mode-7.html
[7] Asset usado na Unity: https://assetstore.unity.com/packages/3d/vegetation/trees/hd-dry-tree-125878