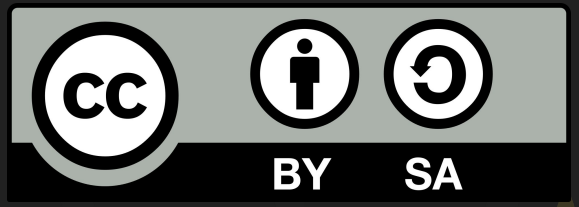
Computação Gráfica para Jogos Eletrônicos

Visão geral sobre o processo de renderização de jogos digitais

Slides por: Leonardo Tórtoro Pereira
Assistentes: Gustavo Ferreira Ceccon, Gabriel Simmel e Ítalo Tobler
TEDJE - FoG - ICMC



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)
- Mostrar contribuições dos jogos eletrônicos para a área
- Mostrar a evolução dos hardwares da área
  - De CPU a GPU
- Introduzir o conceito de pipeline e mostrar o pipeline da GPU e OpenGL

Objetivos

- Como modelos são estruturados, representados e apresentados para os usuários
- Como transformações são feitas em 3D e como isso é representado na câmera
- Mostrar os conceitos e algoritmos básicos por trás das principais técnicas utilizadas na área, além de exemplos de utilização

Índice

1. Introdução
2. CPU vs GPU
3. Tipos de imagens
4. Modelos 3D
5. Pipeline & Hardware
6. Renderização

1. Introdução

1. Introdução

- O que é Computação Gráfica (CG)?
  - Imagens e filmes criados usando computadores
  - Dados de imagem criados por computador
    - Principalmente com ajuda de softwares e hardwares gráficos especializados

1. Introdução

➔ Quais são os tópicos mais importantes da área?

- Design de interface de usuário
- Gráficos de *sprites* e de vetor
- Modelagem 3D
- *Shaders*
- Design de GPU
- Visão computacional
- Entre outros!

1. Introdução

- A CG baseia-se fortemente em 3 ramos da ciência
  - Geometria
  - Óptica
  - Física

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

- Bibliotecas gráficas mais usadas:
  - OpenGL
  - DirectX
  - Vulkan

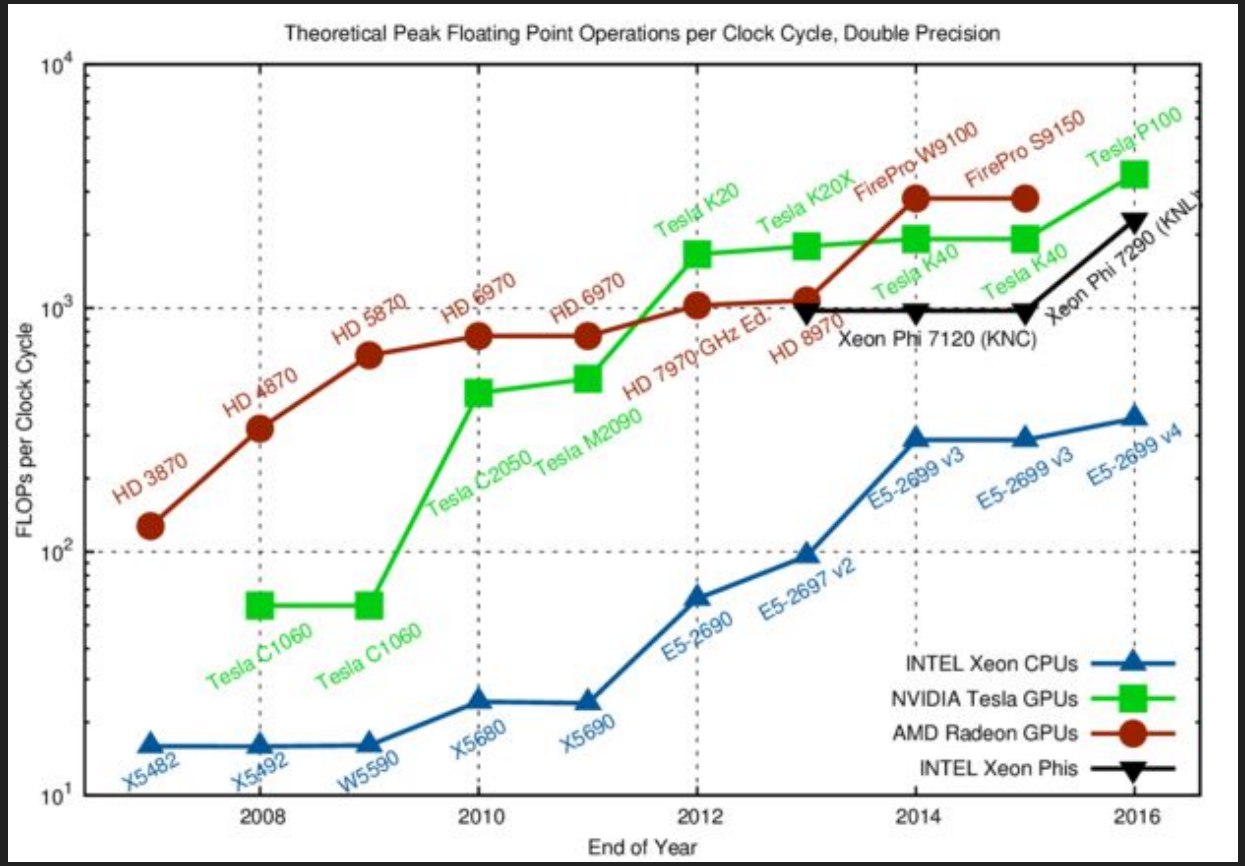
2. CPU vs GPU

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



https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

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

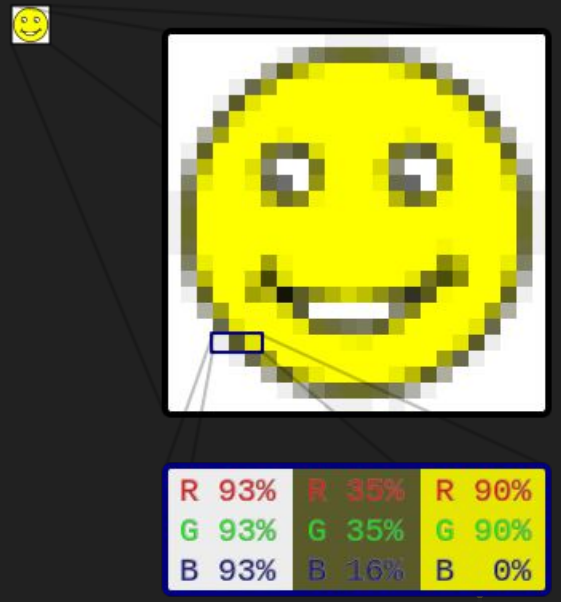
3. Tipos de Imagens

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

Imagens Rasterizadas

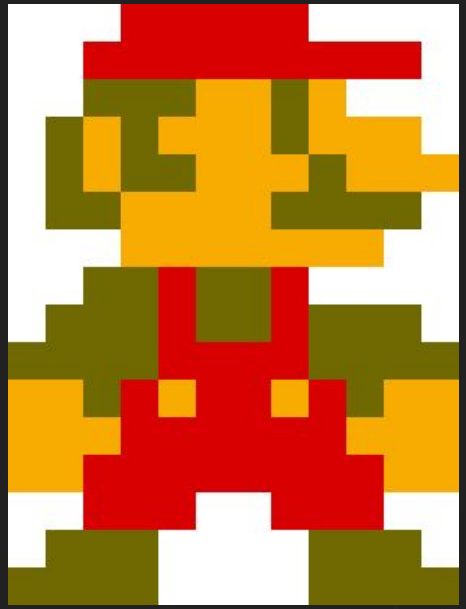
Raster

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



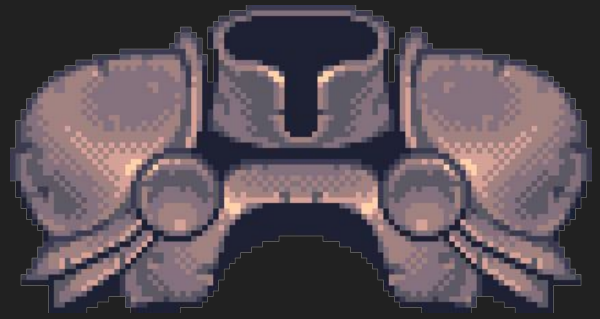
Raster

- 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



Raster

Exemplos Pixel Art

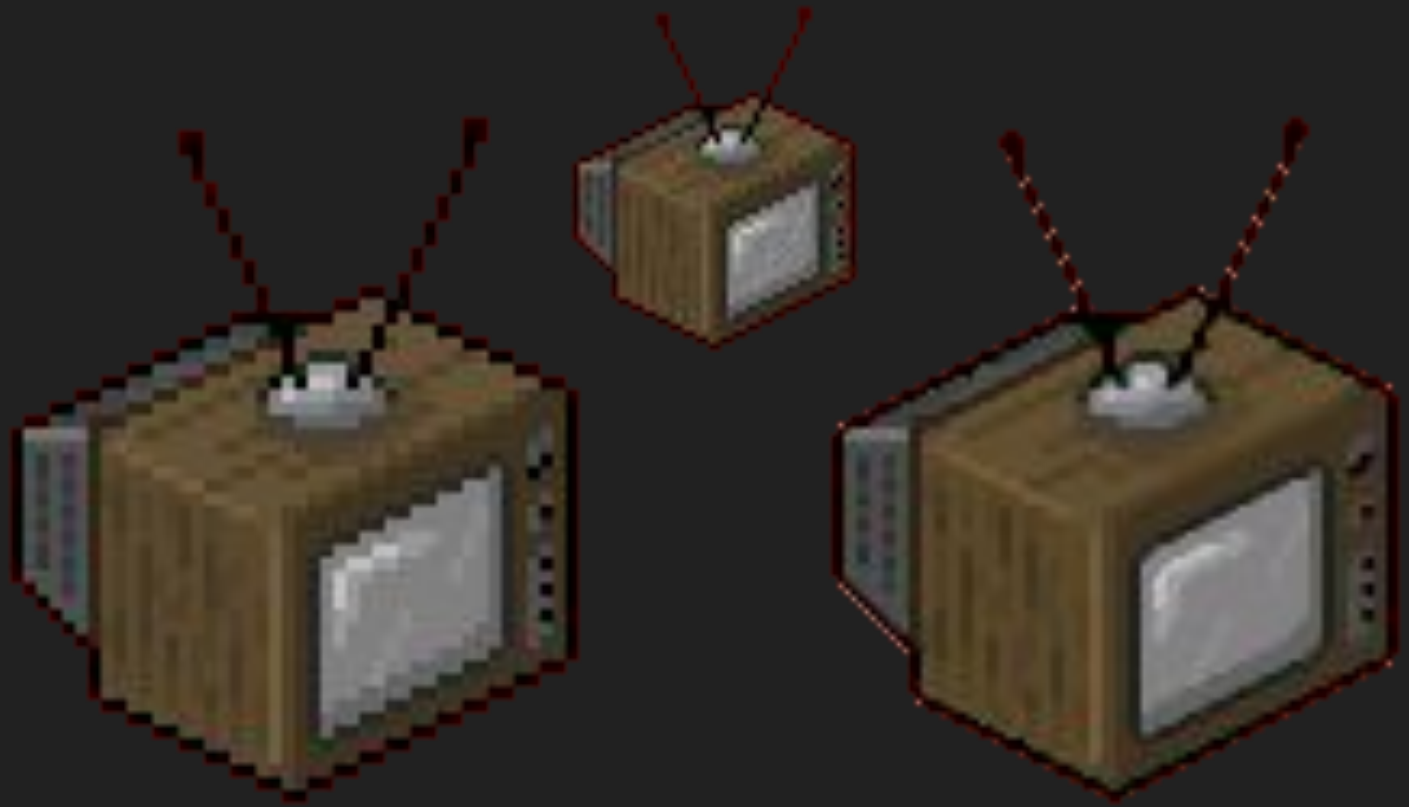


Shameless self promotion

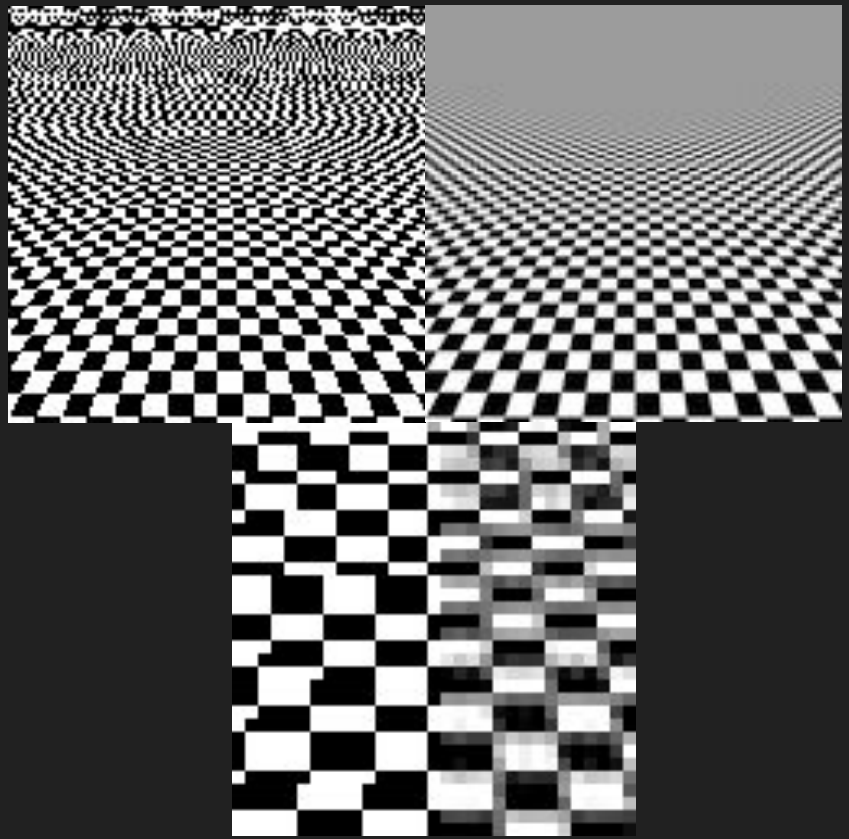
Raster

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

Raster

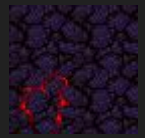


Raster

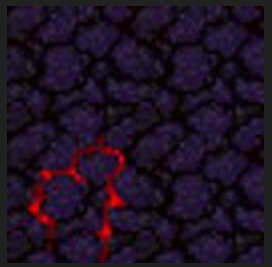


Raster

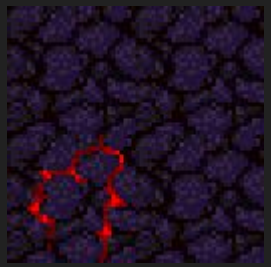
- Cuidado com anti-aliasing em pixel art!
- Antes de re-escalar use o método certo (Caixa)



64x64
Original



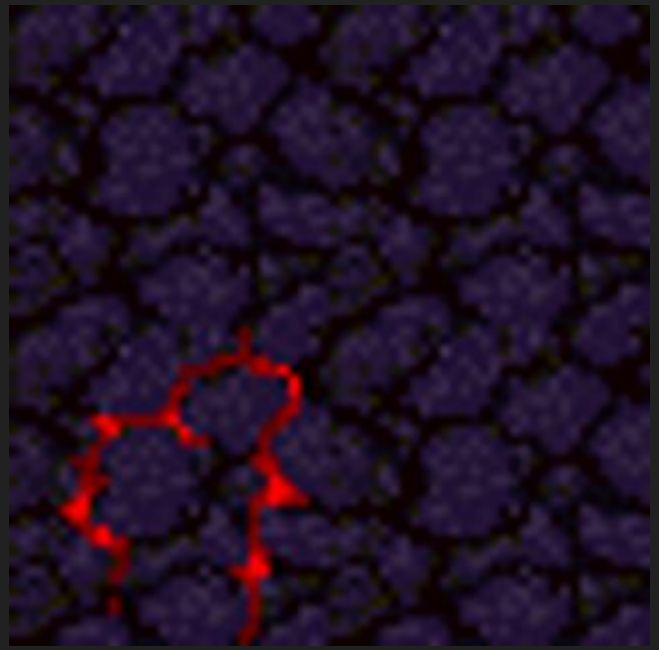
128x128
Filtro Bicúbico



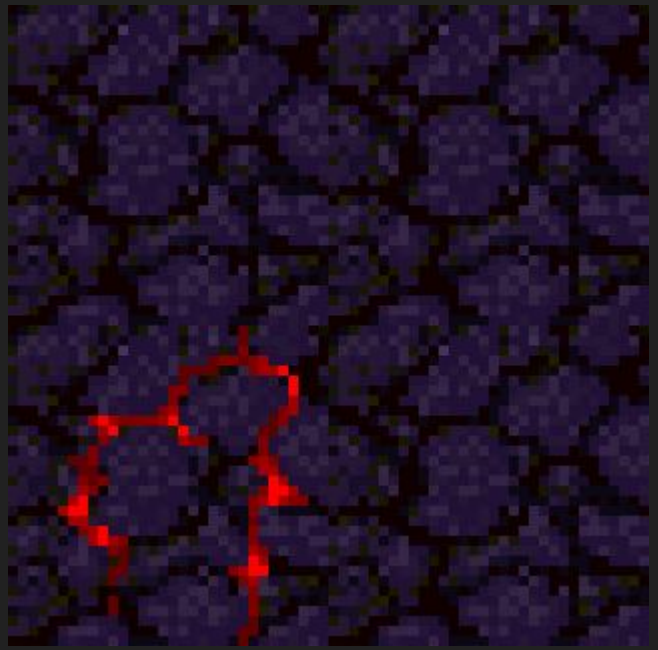
128x128
Filtro Caixa

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

Imagem Vetorizada

Exemplos

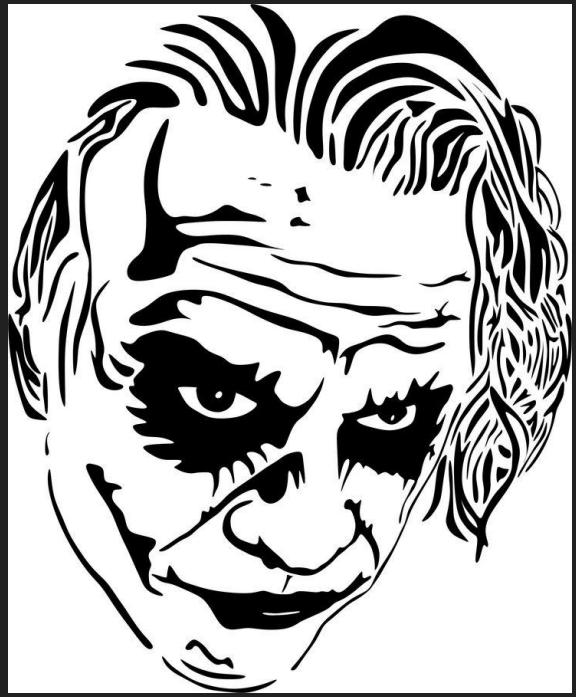
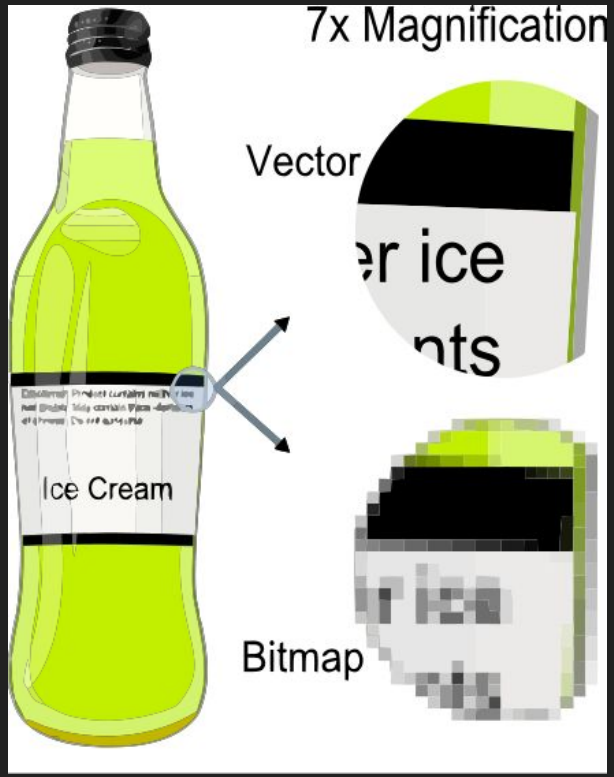


Imagem Vetorizada

- 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

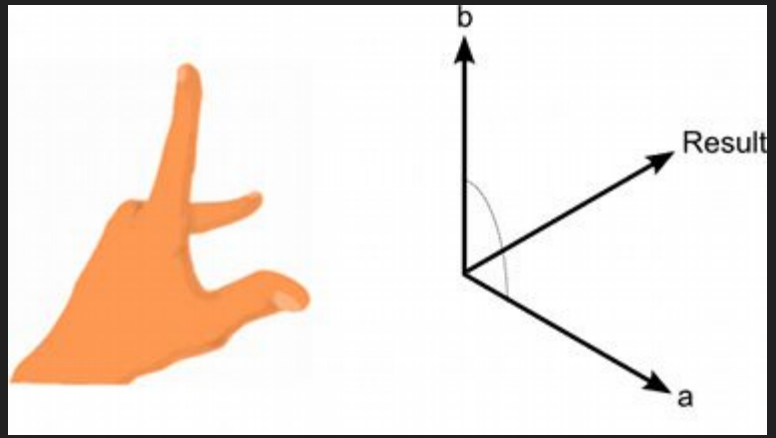
- Representados através de malhas
  - Superfícies representadas através de um conjunto de triângulos
- Por que usar triângulos? Por que não usar quadrados?
  - Simples, planares e geralmente não quebram em transformações
- Tudo isso no sistema de coordenadas euclidiana

4. Modelos 3D

- Sistema euclidiano
  - Ortogonal e Ortonormal
  - Vetores e operações entre vetores nos componentes
- Formas de multiplicação
  - Dot product = produto interno
  - Cross product = produto vetorial

4. Modelos 3D

- Left hand rule vs. Right hand
  - Cross(X, Y) = Z
- Normais e Binormais
  - Cada vértice guarda um valor, que pode ser calculado através da malha



4. Modelos 3D

- Transformações
  - TRS: Translate, Rotate, Scale
- Operações apenas de matrizes
  - Usamos 4x4 invés de 3x3, para conseguimos representar translate como multiplicação
- Multiplicação é altamente otimizada pela GPU
  - Multiplicação de matriz não é comutativa!
  - Boa notícia: multiplicação retorna uma matriz (Model)

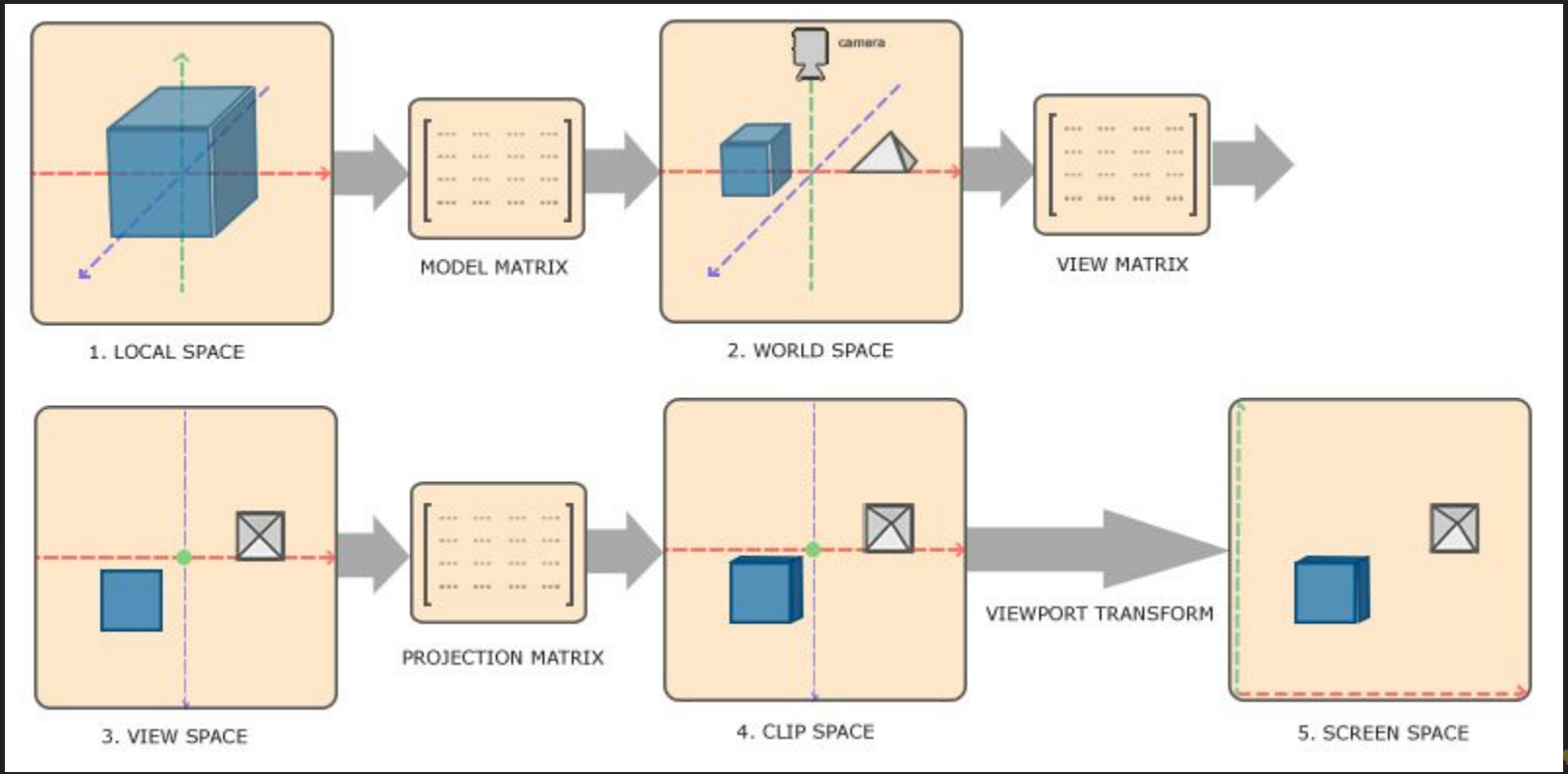
4. Modelos 3D

- Camera
  - É simplesmente uma matriz, na verdade, duas
- Visão (View)
  - Posiciona a camera na origem, na verdade acontece o contrário...
- Projeção (Projection)
  - Ortogonal: para jogos 2D
  - Perspectiva: para jogos 3D (geralmente)

4. Modelos 3D

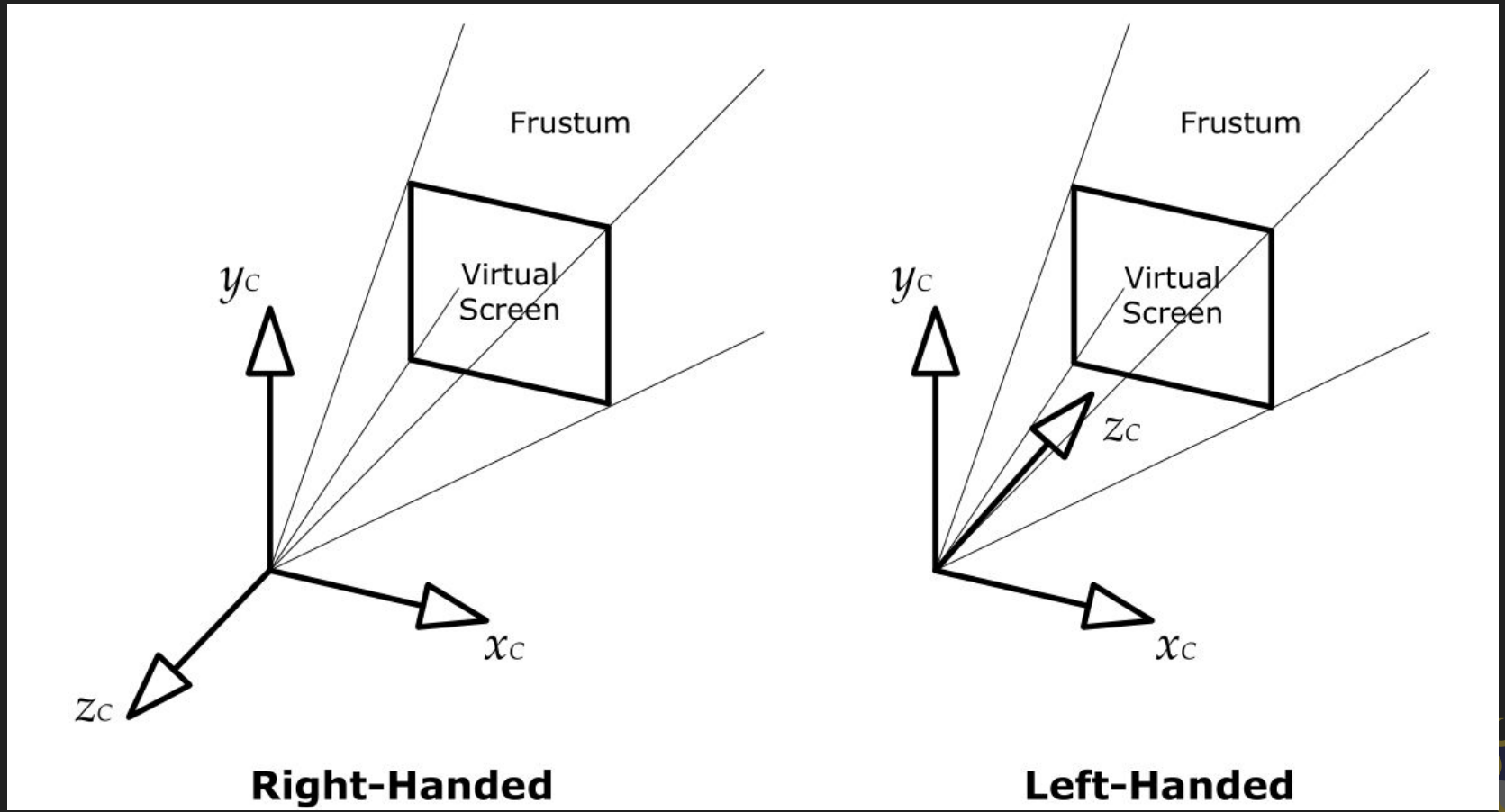
- Camera
  - É simplesmente uma matriz, na verdade, duas
- Visão (View)
  - Posiciona a camera na origem, na verdade acontece o contrário...
- Projeção (Projection)
  - Ortogonal: para jogos 2D
  - Perspectiva: para jogos 3D (geralmente)

MVP

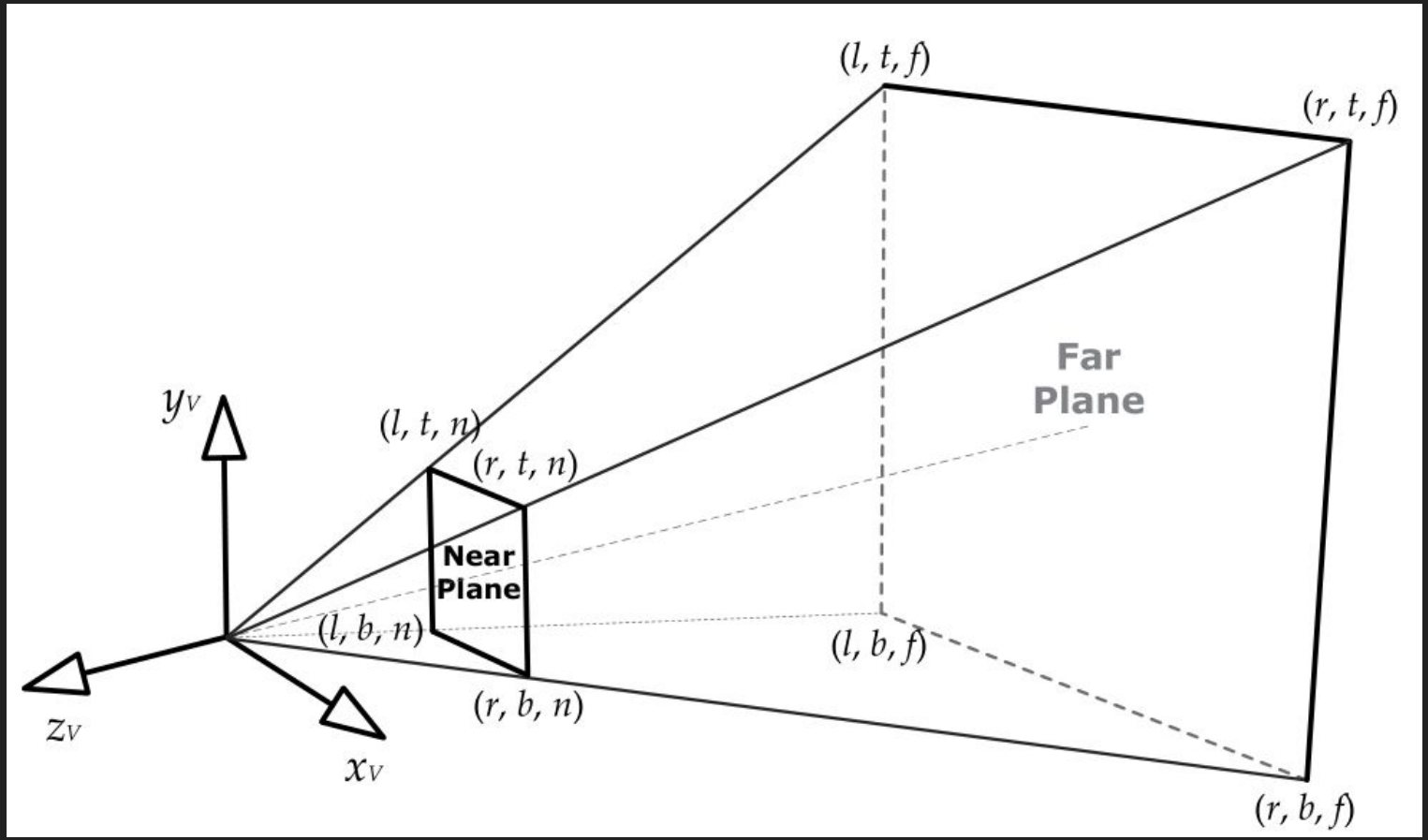


https://learnopengl.com/Getting-started/Coordinate-Systems

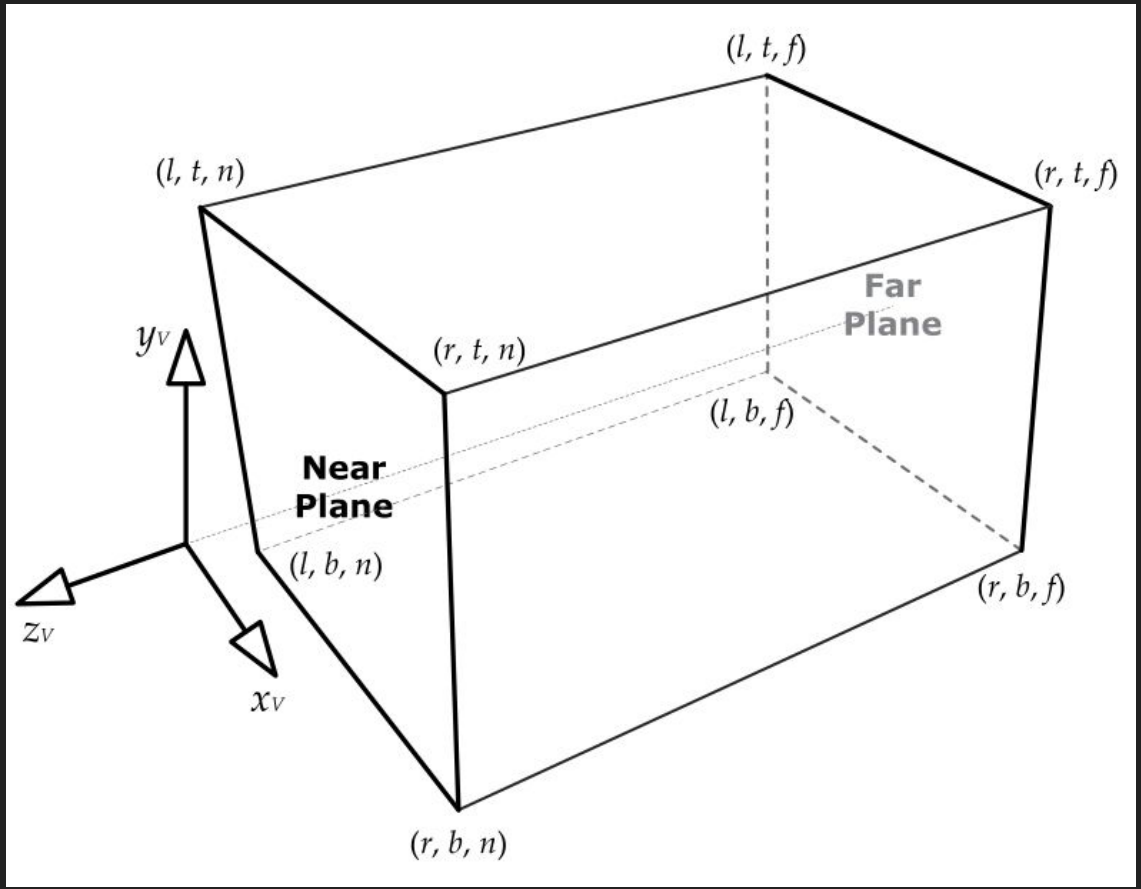
MVP


Frustum
Virtual Screen
y_C
x_C
z_C
Right-Handed
Frustum
Virtual Screen
y_C
z_C
x_C
Left-Handed

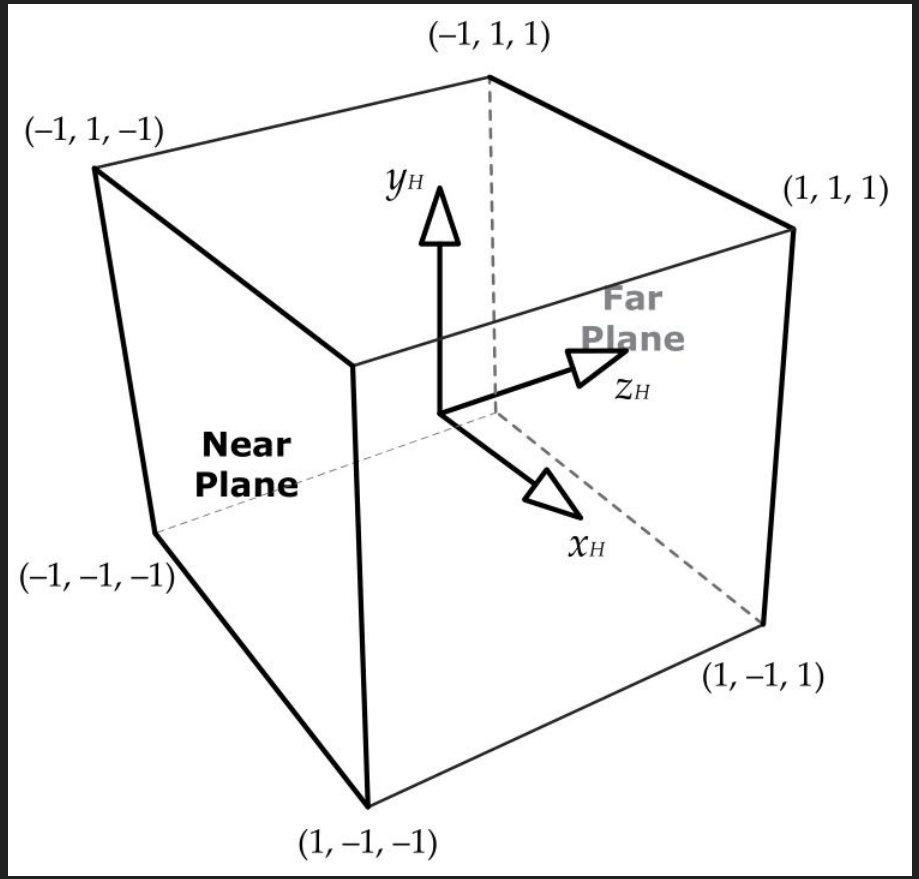
MVP



MVP



MVP



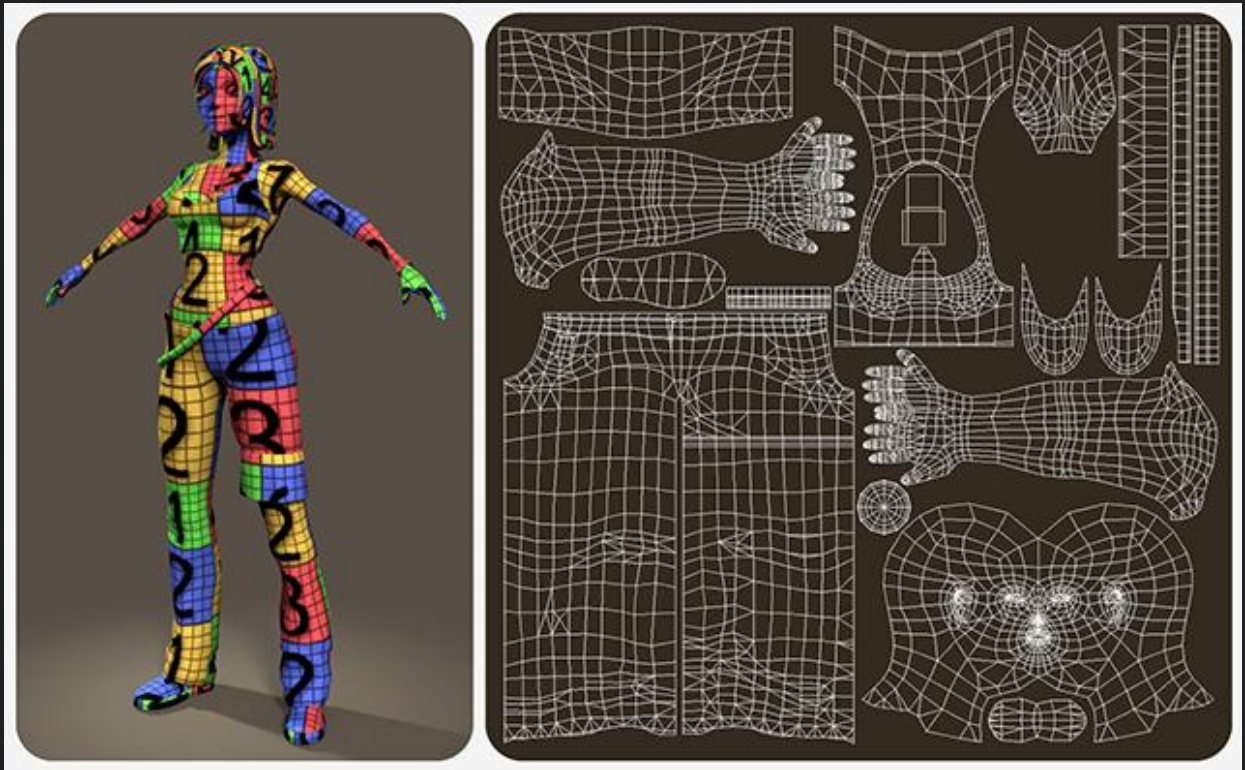
4. Modelos 3D

- 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

BLENDER TIME!!! - Cortando um Cubo

Quick Tips!

Ferramentas interessantes para arte 3D

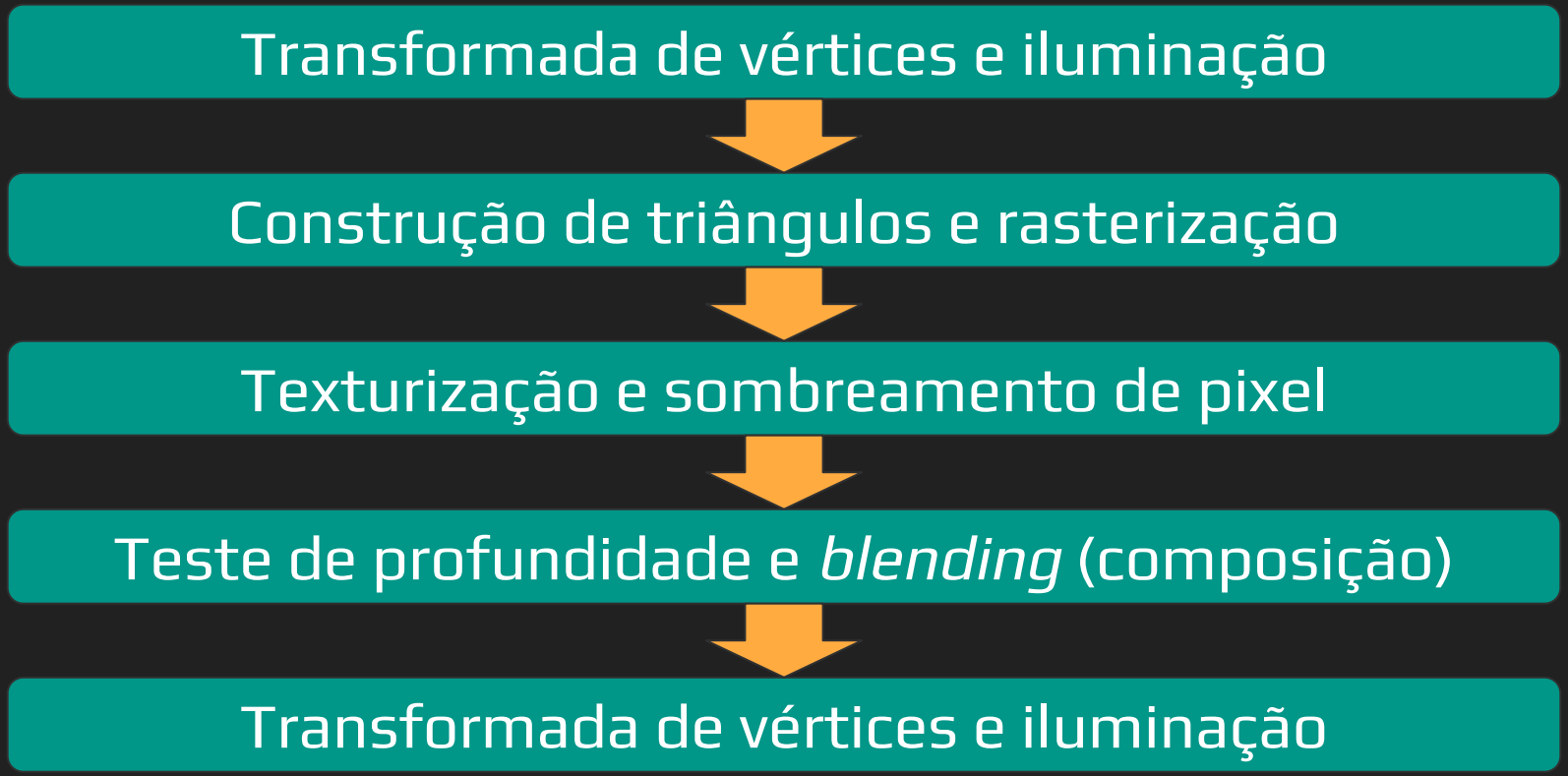
- Blender (Open Source)
  - Propósitos gerais (Até edição de vídeo)
- Autodesk Maya
  - Propósitos gerais
- Houdini
  - Animação, focada em geração procedural
- ZBrush
  - Foco em escultura, integração com várias ferramentas

5. Pipeline & Hardware

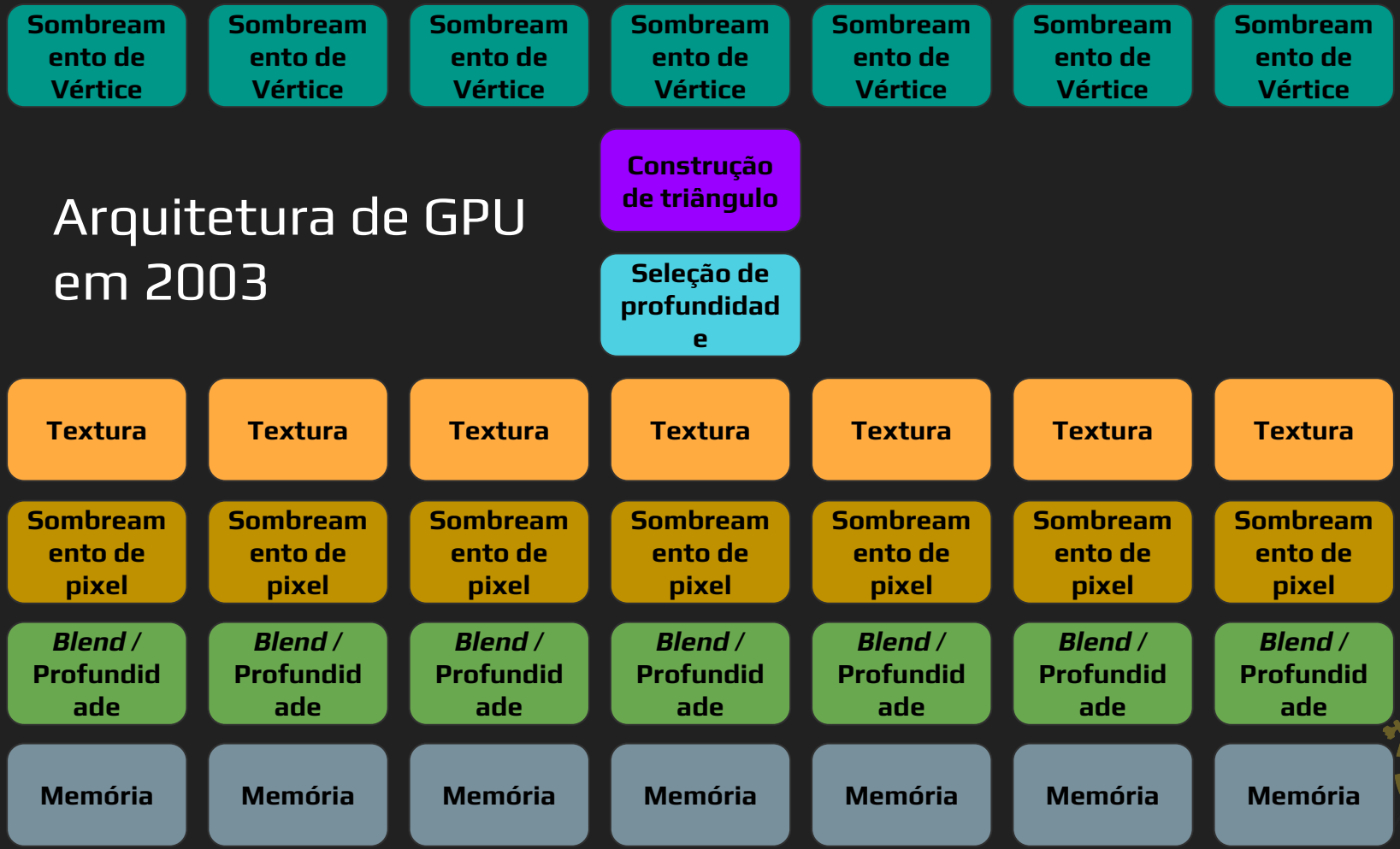
5. Pipeline & Hardware

- 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



Arquitetura de GPU em 2003



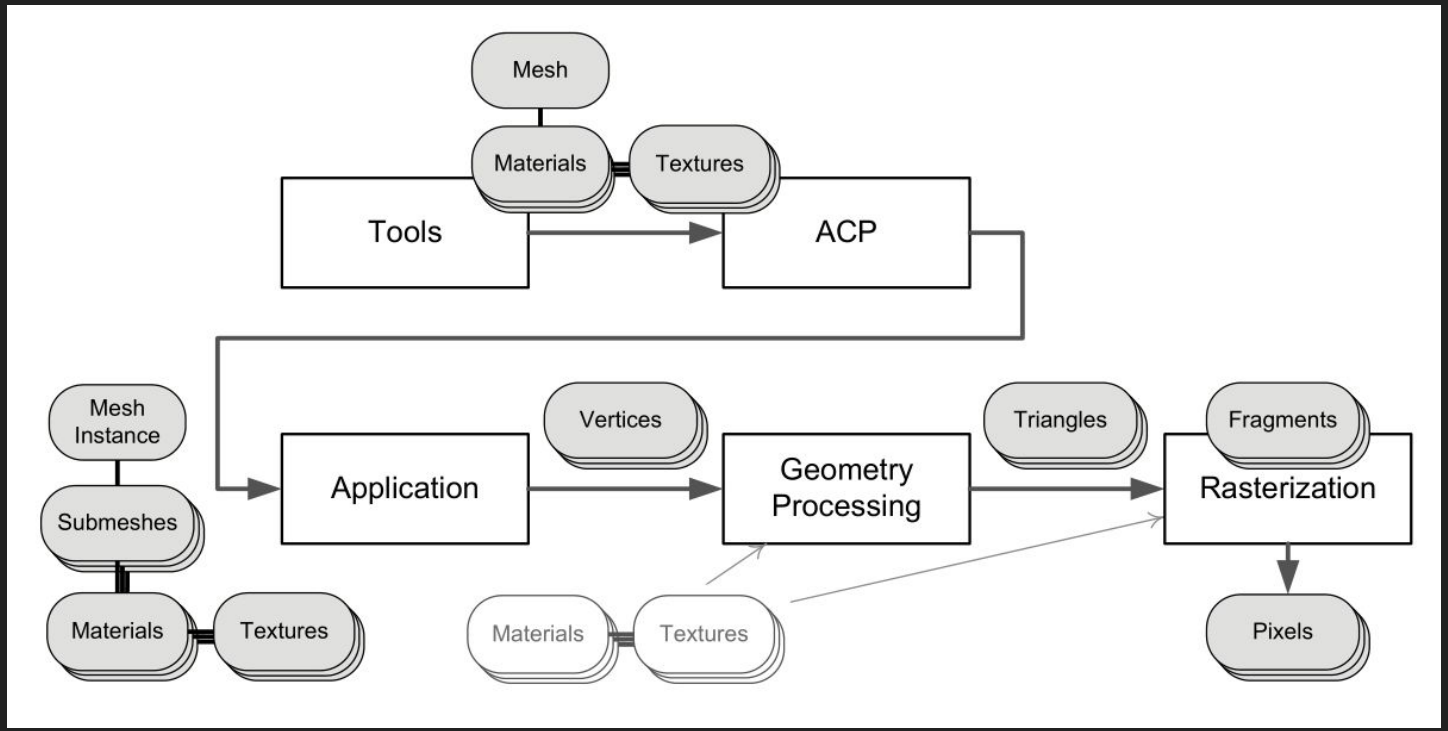
5. Pipeline & Hardware

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

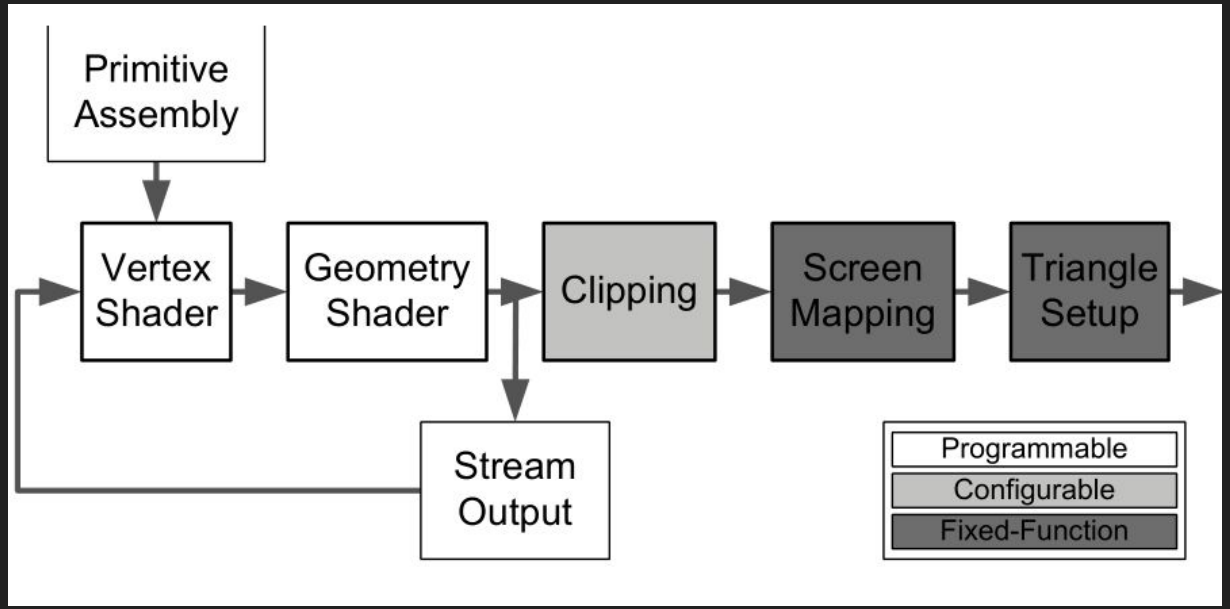
5. Pipeline & Hardware

- Atualmente
  - Hardware de propósito específico
  - Threading e pipelining gerenciados por hardware
  - “Gráficos” e computação livremente misturados

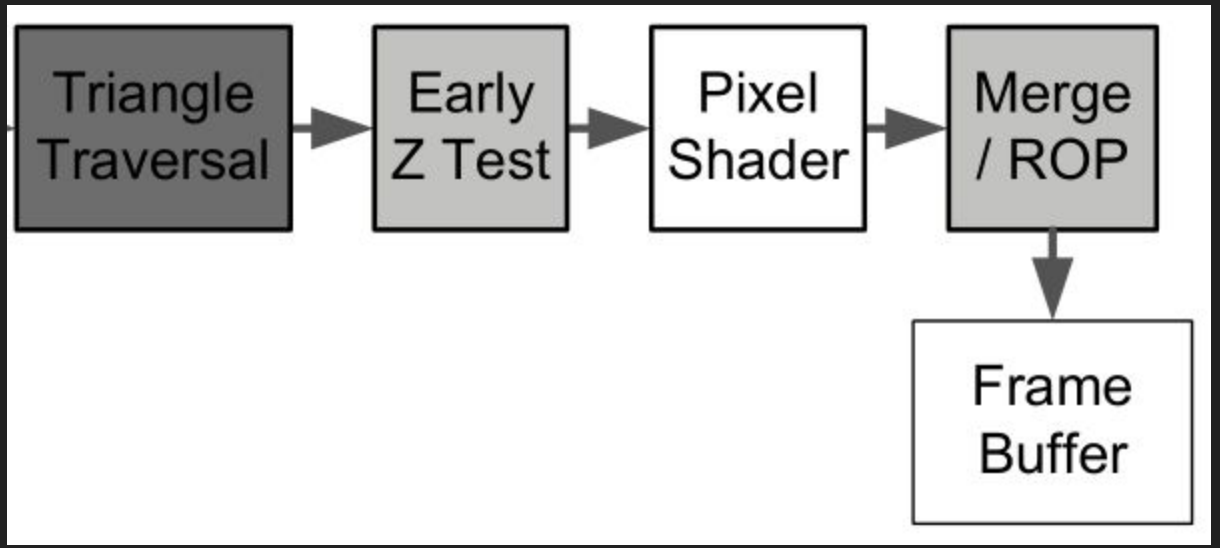
Transformações dos Dados



Pipeline na GPU

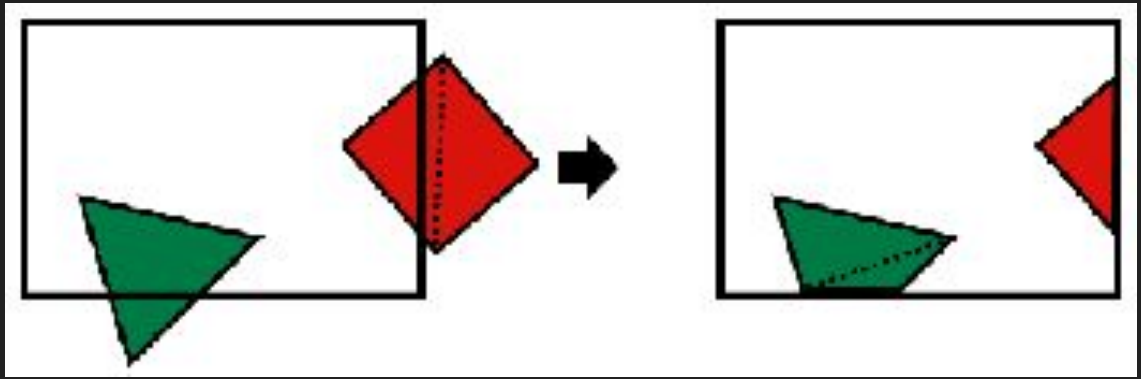


Pipeline na GPU



5. Pipeline & Hardware

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



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

5. Pipeline & Hardware

- ➔ Culling (Seleção)
  - ◆ Seleciona dados desnecessários para o pipeline gráficos
  - ◆ Exemplo: dados para desenhar a parte “de trás” de um objeto
  - ◆ N64 suporta dois tipos de culling:
    - Back-face e de volume

5. Pipeline & Hardware

- ➔ Back-face culling
  - ◆ Retira a parte de trás de objetos

http://n63.icequake.net/doc/n64intro/kantan/images/1-2-3-6.gif

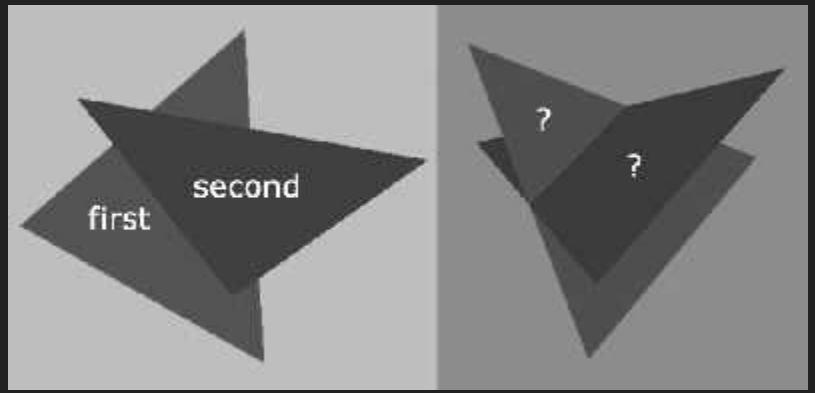
5. Pipeline & Hardware

- Volume culling
  - Retira itens que estão completamente fora do campo visual da lista de exibição (display list) que desenha objetos

http://n63.icequake.net/doc/n64intro/kantan/images/1-2-3-7.gif

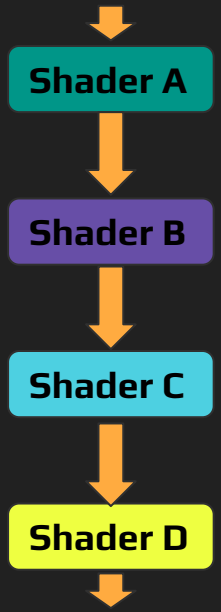
5. Pipeline & Hardware

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

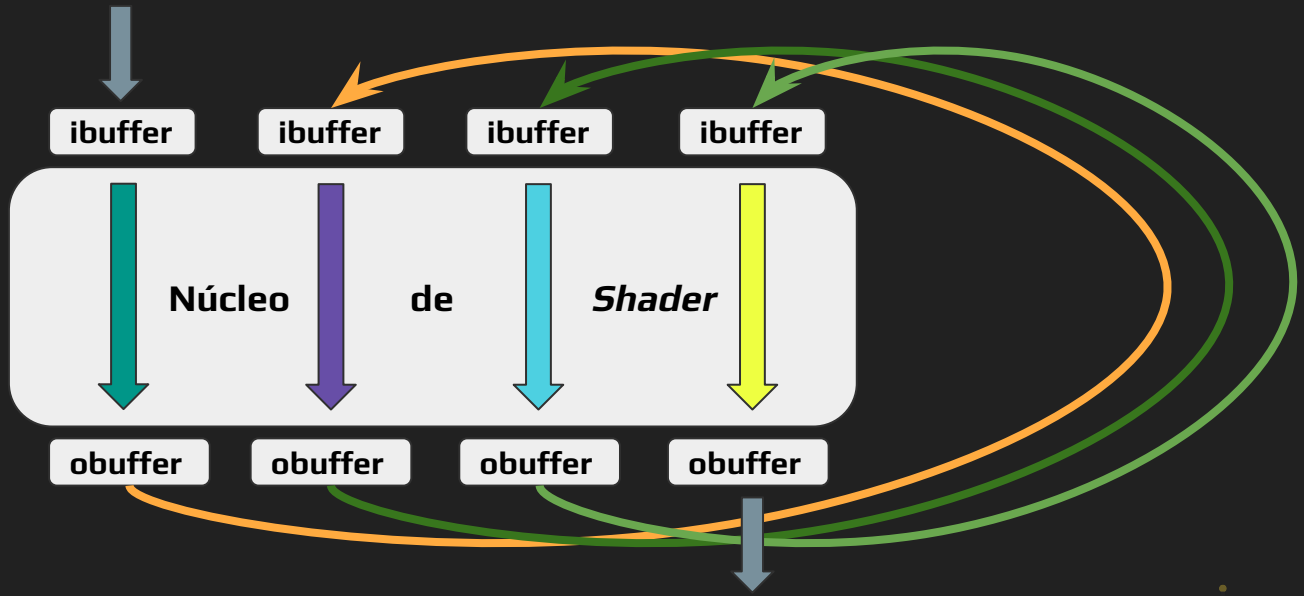


Design discreto X unificado

Design discreto

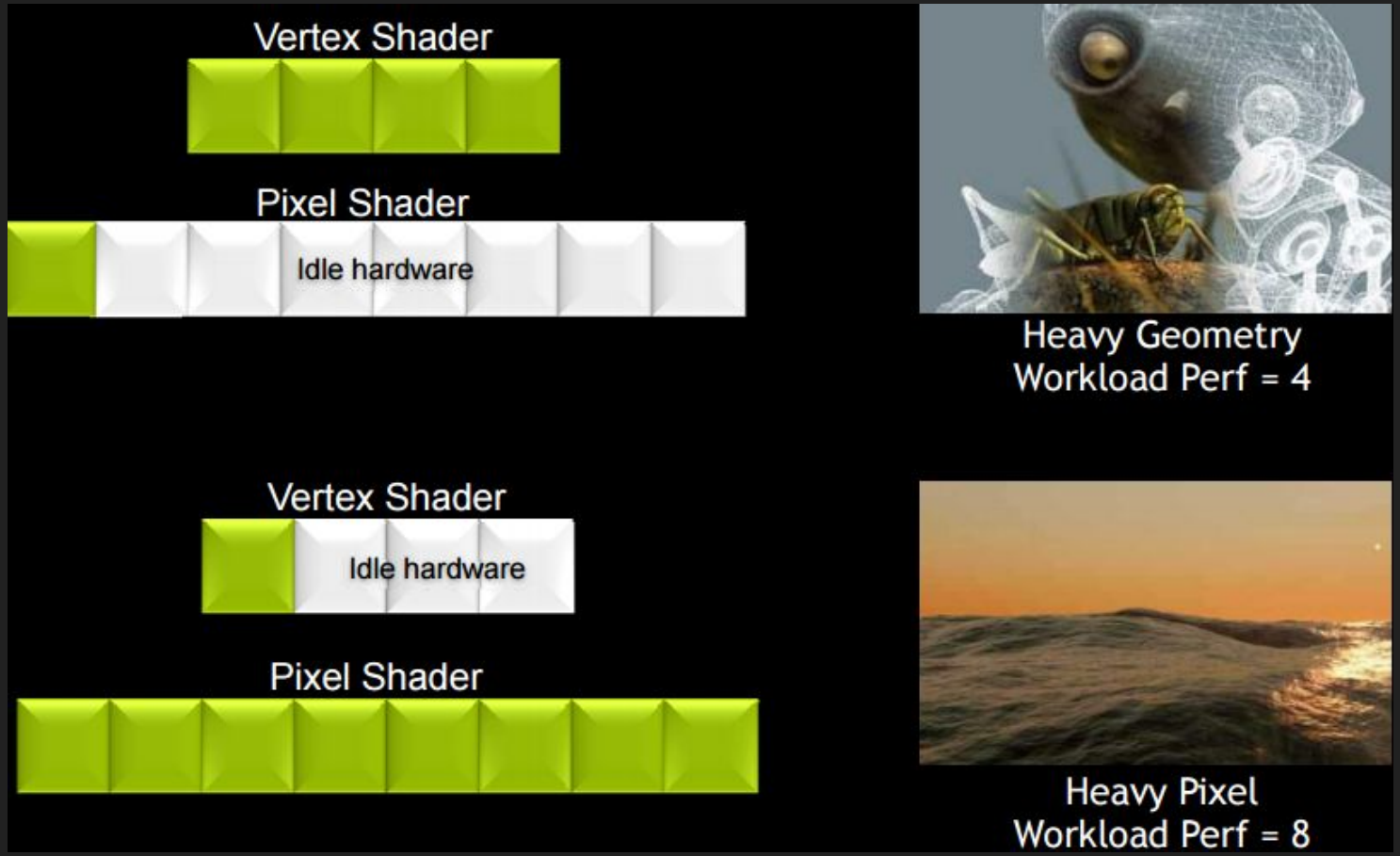


Design unificado



Arquitetura unificada: Sombreamentos de vértices, pixels, etc. tornam-se threads rodando em diferentes programas em núcleos flexíveis

Por que unificar?



http://www.nvidia.com/content/nvision2008/tech_presentations/Technology_Keynotes/NVISION08-Tech_Keynote-GPU.pdf

Por que unificar?

Heavy Geometry
Workload Perf = 11

Heavy Pixel
Workload Perf = 11

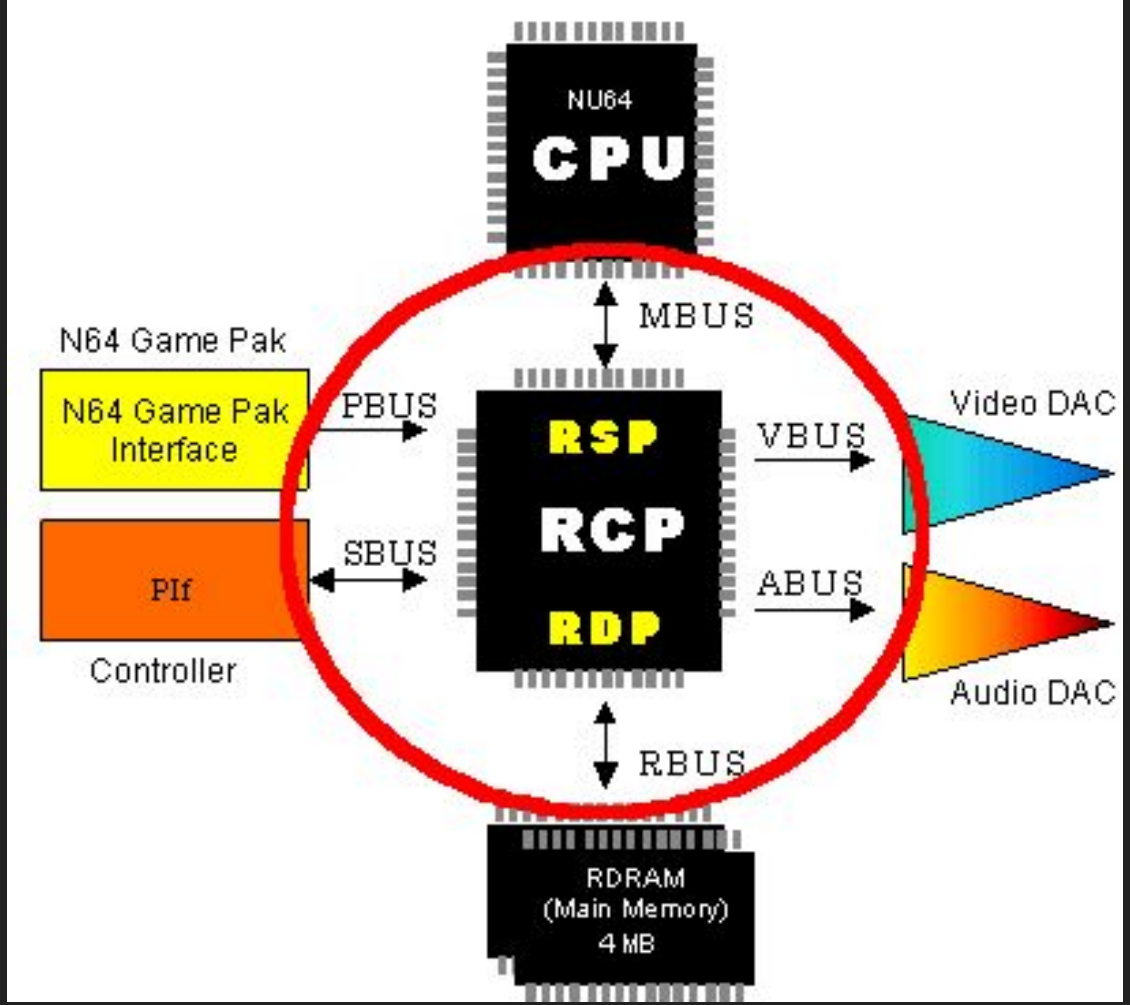
http://www.nvidia.com/content/nvision2008/tech_presentations/Technology_Keynotes/NVISION08-Tech_Keynote-GPU.pdf

Bonus Stage: Arquitetura do N64

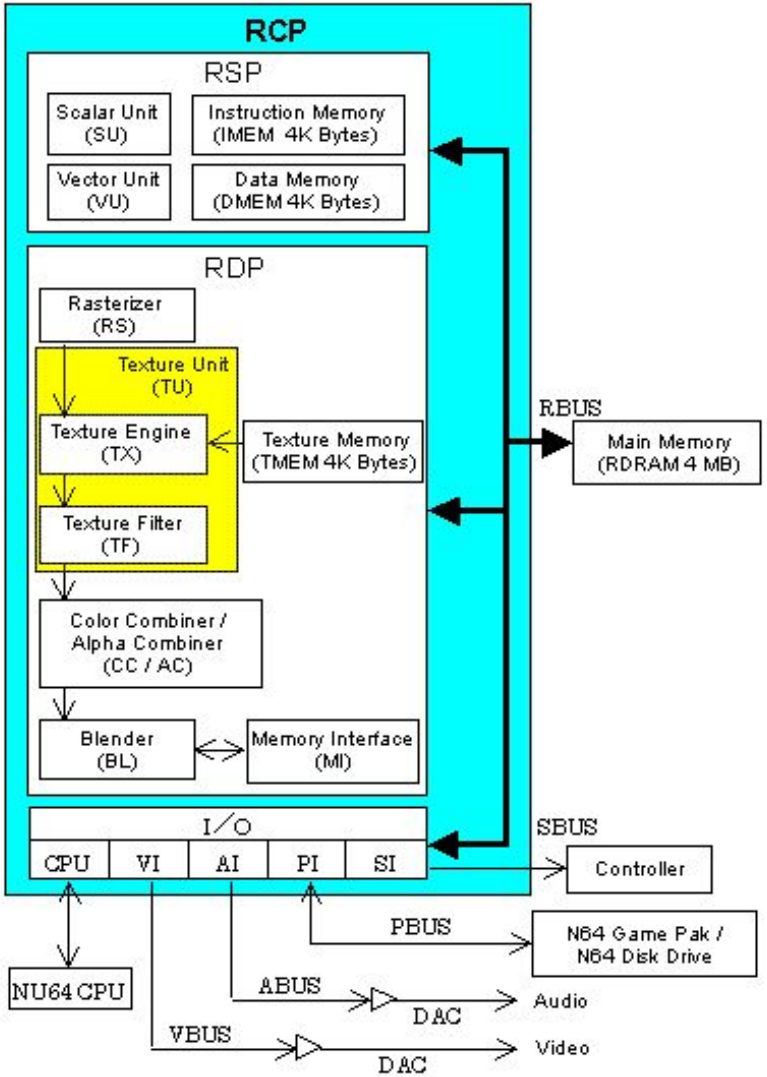
https://media0dk-a.akamaihd.net/24/12/e7f4ff8d93dd9182292772a7815e8305.jpg

http://vignette1.wikia.nocookie.net/banjokazooie/images/a/ab/Ss-banjo-kazooie-xbla-007.jpg/revision/latest?cb=20100413020934

https://www.serebii.net/stadium/mewtwo.jpg

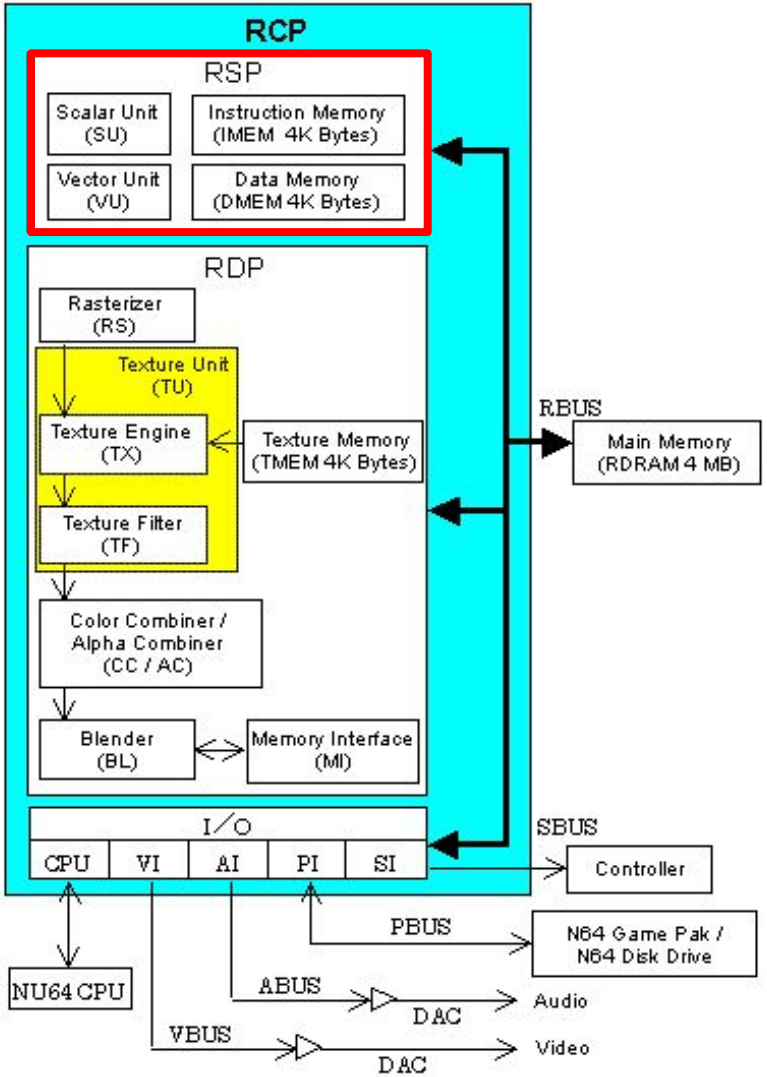


http://n63.icequake.net/doc/n64intro/kantan/images/1-2-3-1.gif



Blocos de processos do RCP

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



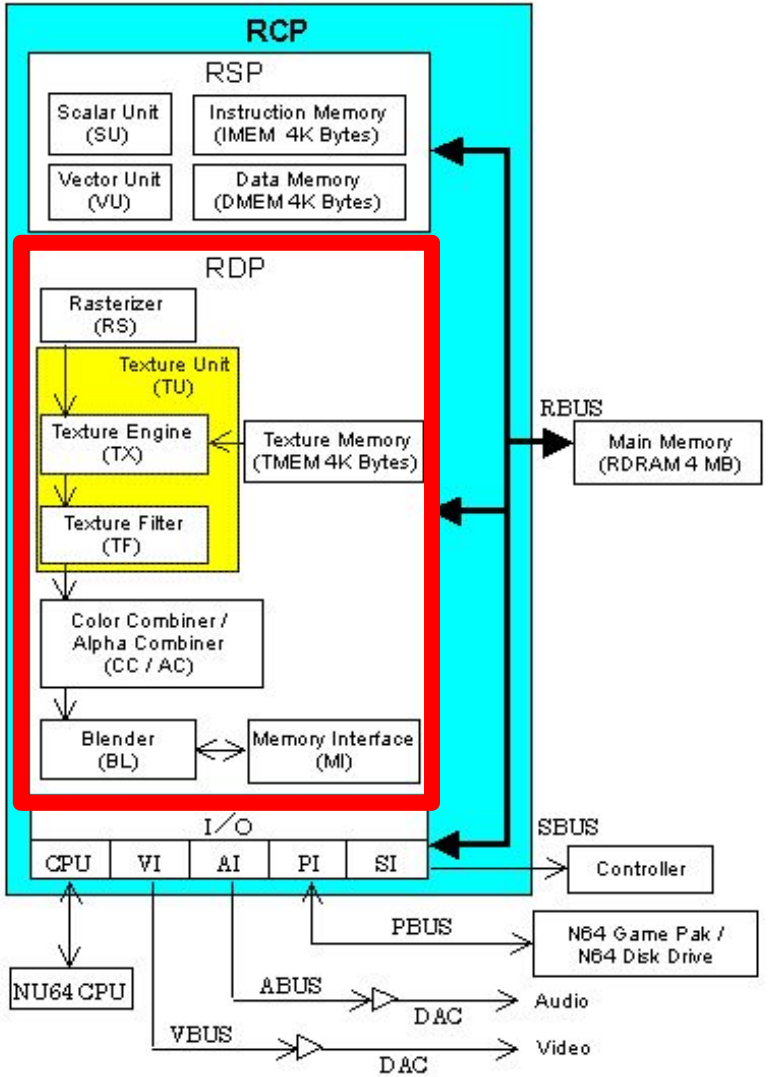
Blocos de processos do RSP

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

Arquitetura do N64

- ➔ Processos gráficos executados no RSP
  - ◆ A maioria dos processos do RSP são executados quando os dados de vértices são carregados no *cache* de vértices
  - ◆ Os processos principais são:
    - Transformadas geométricas
      - Necessárias para mover ou escalar objetos 3D

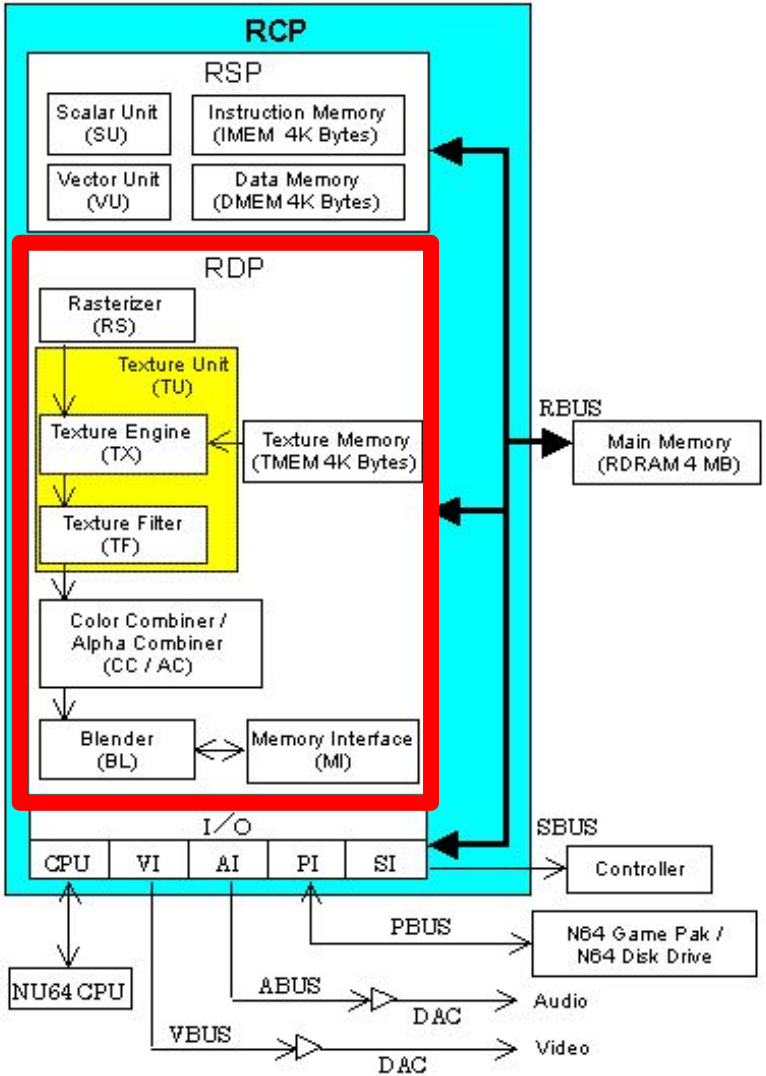
Exemplo de Volume Culling do N64



Unidades do RDP

- Rasterizador (RS) transforma triângulos e retângulos em pixels
- Motor de Textura (TX) provê amostragem para texels (elementos de imagem) através do uso de TMEM (Memória de Textura)
- Filtro de Textura (TF) provê filtragem para texels criados pelo TX
- Combinador de Cor/Combinador Alpha (CC/AC) combina duas cores de pixels criadas pelo RS e texels criados pelo TF e interpola entre essas duas cores.

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



Unidades do RDP

- O Misturador (Blender) (BL) mistura a cor do pixel determinada em CC, a cor no *frame buffer*, a cor de *fog*, e assim em diante. Desenha a cor resultante no *frame buffer*. Nestemomento, também provê o Z-*buffering* para a primeira parte do processo de *anti-aliasing*.
- Interface de Memória (MI) processa informação de pixel no *frame buffer* incluindo operações de ler, modificar e escrever

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

Bonus Stage: Arquitetura do SNES

https://upload.wikimedia.org/wikipedia/en/thumb/f/f4/Supermarioworld.jpg/250px-Supermarioworld.jpg

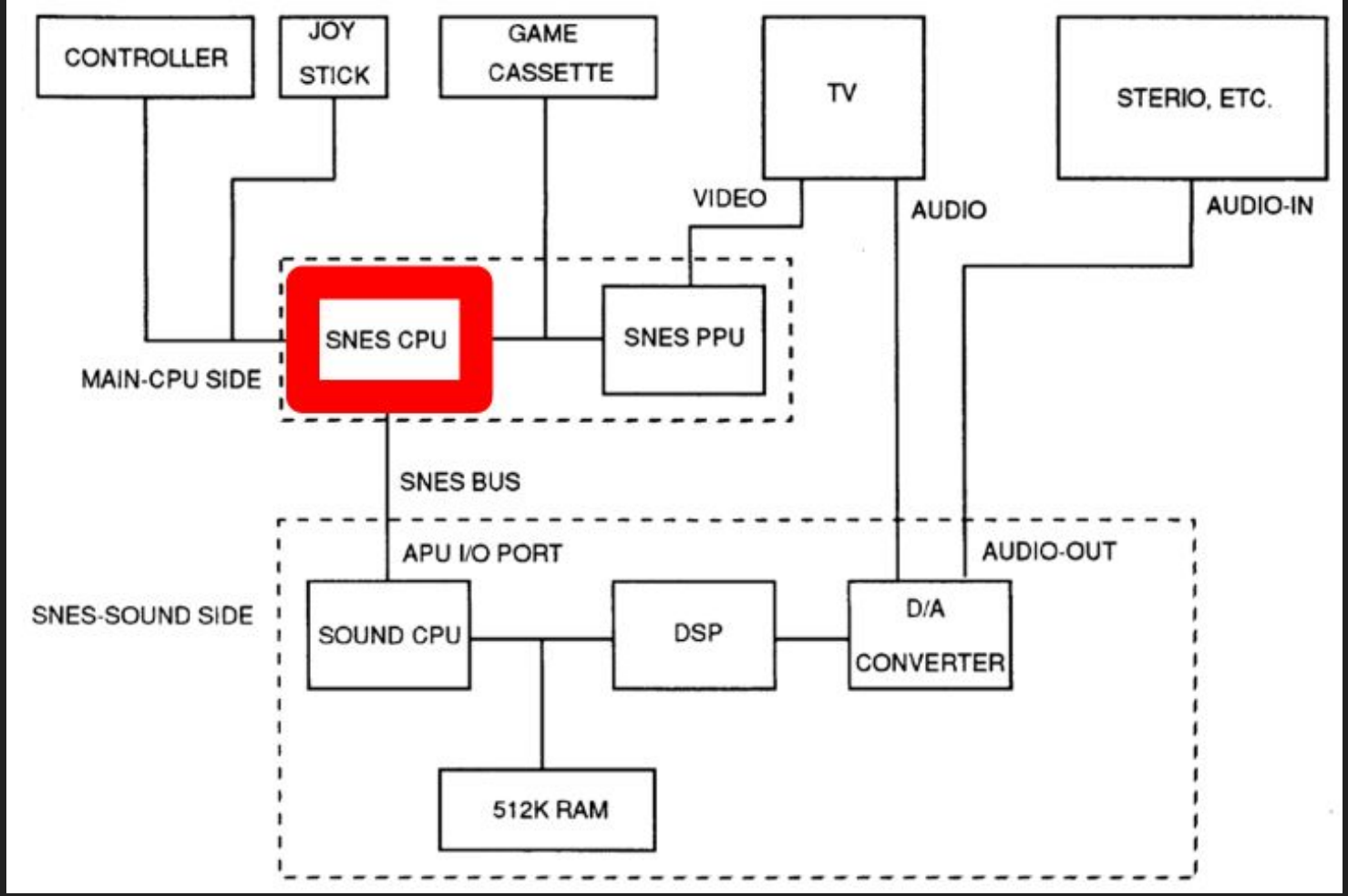
http://cdn.gamer-network.net/2017/usgamer/street_fighter_II_turbo_snes_classic_02.png

https://tctechcrunch2011.files.wordpress.com/2017/09/ffvi.jpg

http://4.bp.blogspot.com/_9MOywHCf_4Y/S_bAab1OJEI/AAAAAAAAFB4/amu6EOdDsd4/s400/gfs_4582_2_2.jpg

Arquitetura do SNES

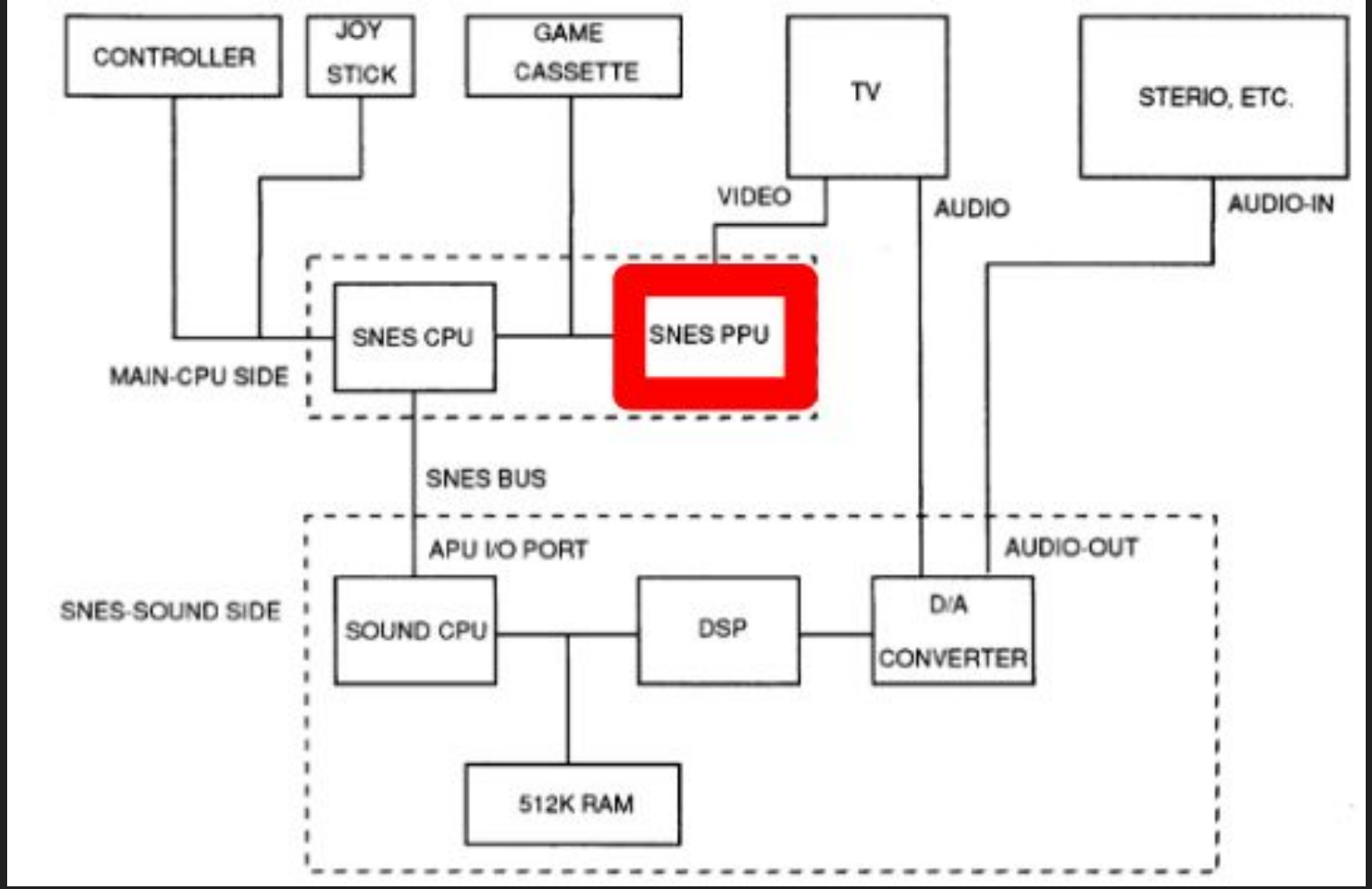
- SNES
  - Velocidade da CPU 2.86MHz (até 10.74 MHz)
  - 16bits
  - 24 bit bus - usado para acessos gerais
  - 8 bit bus - usado para acesos de registro APU e PPU
  - Instruções por segundo: 1.79 MIPS



http://meseec.ce.rit.edu/551-projects/fall2014/3-1.pdf

Arquitetura do SNES

- SNES
  - Velocidade da CPU 2.86MHz (até 10.74 MHz)
  - 16bits
  - 24 bit bus - usado para acessos gerais
  - 8 bit bus - usado para acessos de registro APU e PPU
  - Instruções por segundo: 1.79 MIPS



http://meseec.ce.rit.edu/551-projects/fall2014/3-1.pdf

Arquitetura do SNES

- Picture Processing Unit (PPU)
  - 2 unidades: PPU1 e PPU2
  - PPU1 gera dados, rotação e escala de caracteres de background
  - PPU2 realiza efeitos especiais
  - 32.768 cores
  - Mesmo sinal de clock da CPU
  - 7 modos de vídeo diferente

Arquitetura do SNES

➔ Picture Processing Unit (PPU): 64kB de RAM

- 256x224
- 512x224
- 256x239
- 512x239
- 512x448
- 512x478

Arquitetura do SNES

- SNES pode mostrar 256 cores de uma vez
  - Divididas em 16 sub-paletas, com 16 cores (1 sempre transparente)
  - Tiles de BG usam qualquer uma das 8 primeiras sub-paletas, enquanto os sprites usam as outras 8

Arquitetura do SNES

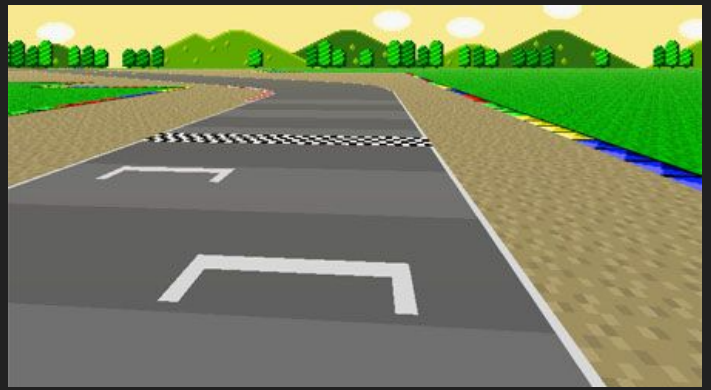
- Modos de vídeo
  - Alteram entre si quantidade de camadas e cores nas paletas
  - Especificação e decodificação de elementos varia

Arquitetura do SNES

- Mode 7
  - Camada única que pode ser rotacionada e escalada usando transformações de matrizes
  - HDMA (Horizontal Direct Memory Access) é normalmente usado para mudar os parâmetros da matriz para cada scanline para gerar efeitos de perspectiva

https://www.coranac.com/tonc/img/mode7/m7_ex_00.png

http://www.nintendoblast.com.br/2014/01/revisitando-os-tempos-aureos-do-mode-7.html



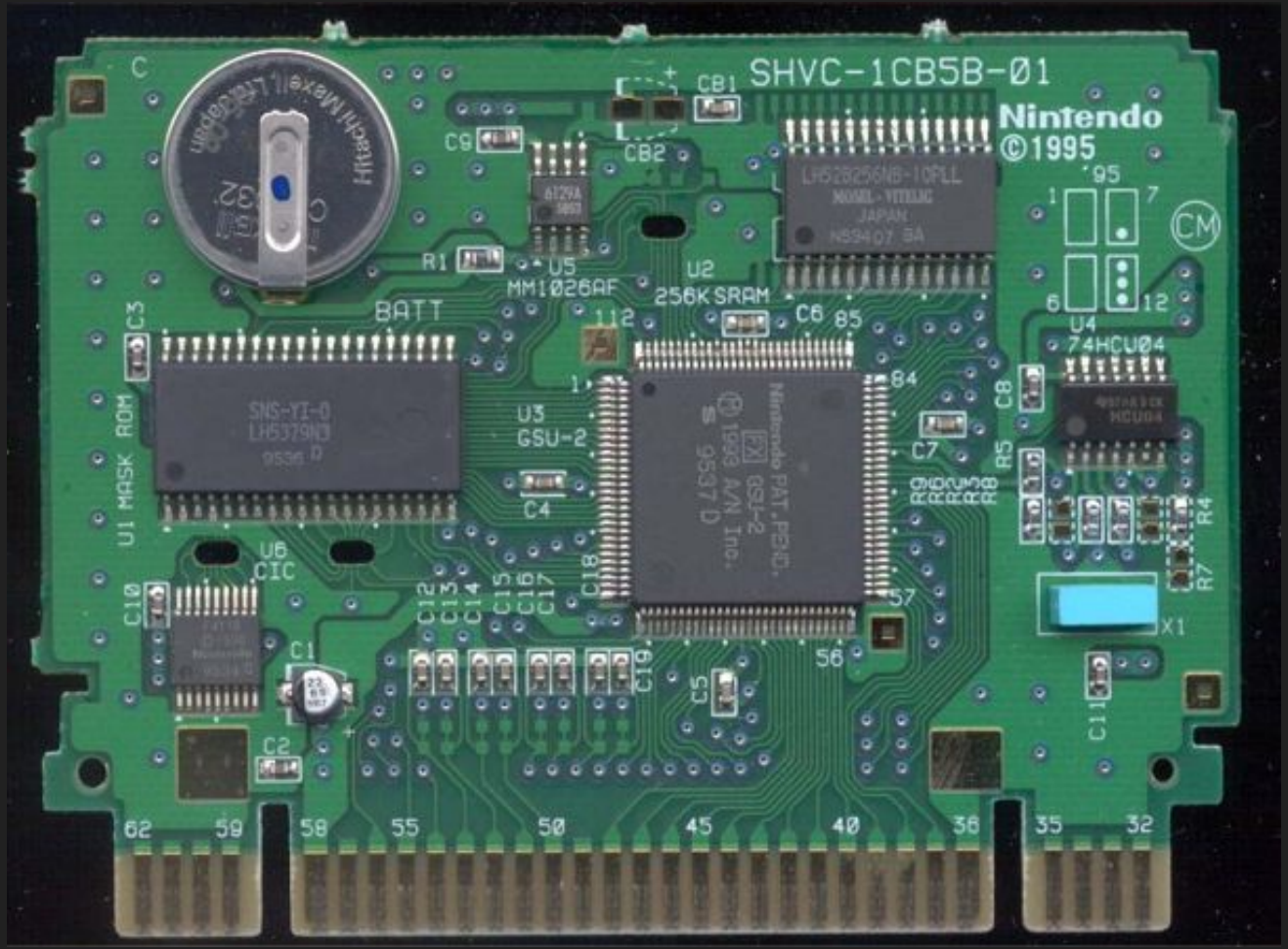
https://www.heyuguys.com/images/2012/05/mode7_example.jpeg

http://www.nintendoblast.com.br/2014/01/revisitando-os-tempos-aureos-do-mode-7.html

Arquitetura do SNES

- Cartuchos
  - Super FX chip
    - CPU RISC
    - Renderiza gráficos que a CPU normal não consegue
    - Processa principalmente polígonos 3D
    - 10.5 MHz de clock
  - Super Accelerator System

https://en.wikipedia.org/wiki/File:SNES_Star_Fox.png



http://meseec.ce.rit.edu/551-projects/fall2014/3-1.pdf

Cx4 - Co-processador matemático

http://www.vgmuseum.com/end/snes/a/mmx2-3.gif

Engine do DKC 2 (feita em Assembly)

Bonus Stage: Uncanny Valley

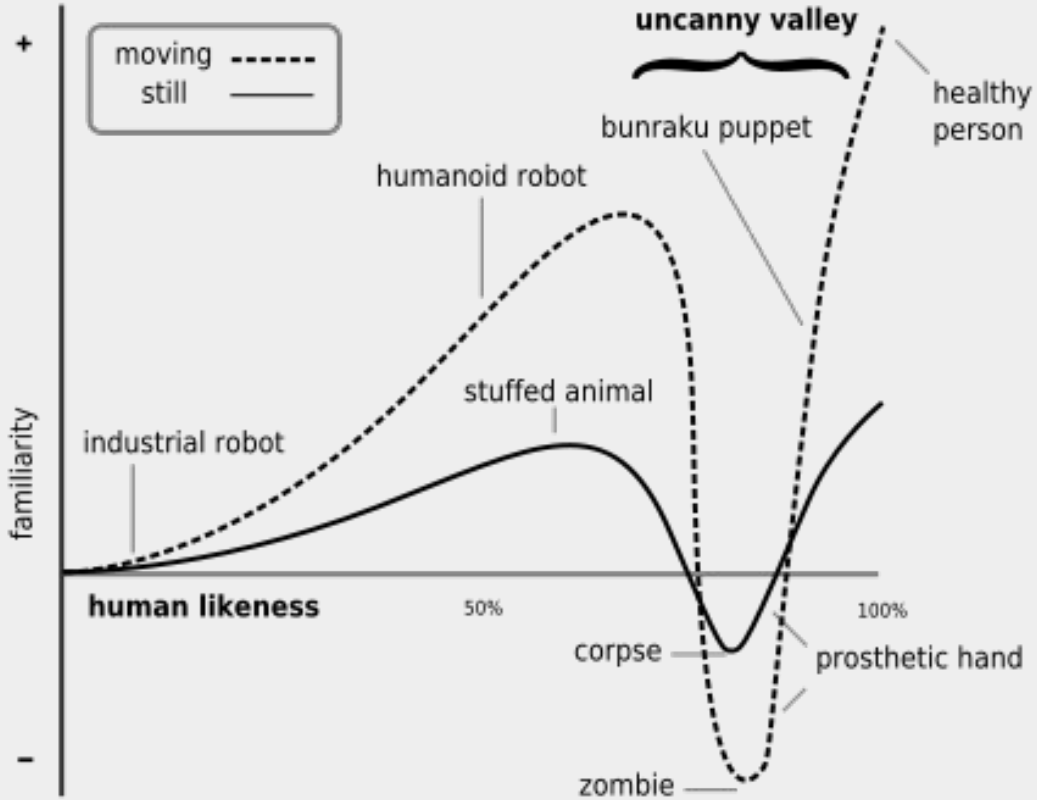
Uncanny Valley

Mass Effect: Andromeda

Bonus Stage: *Uncanny Valley*

- ➔ Gráficos atuais ultra realistas
  - ◆ *Uncanny Valley*
    - Parece e move-se quase igual um ser real
      - Mas não totalmente
    - Causa asco em algumas pessoas

Uncanny Valley



https://upload.wikimedia.org/wikipedia/commons/f/f0/Mori_Uncanny_Valley.svg

Uncanny Valley

https://www.japan-zone.com/culture/bunraku.shtml

6. Renderização

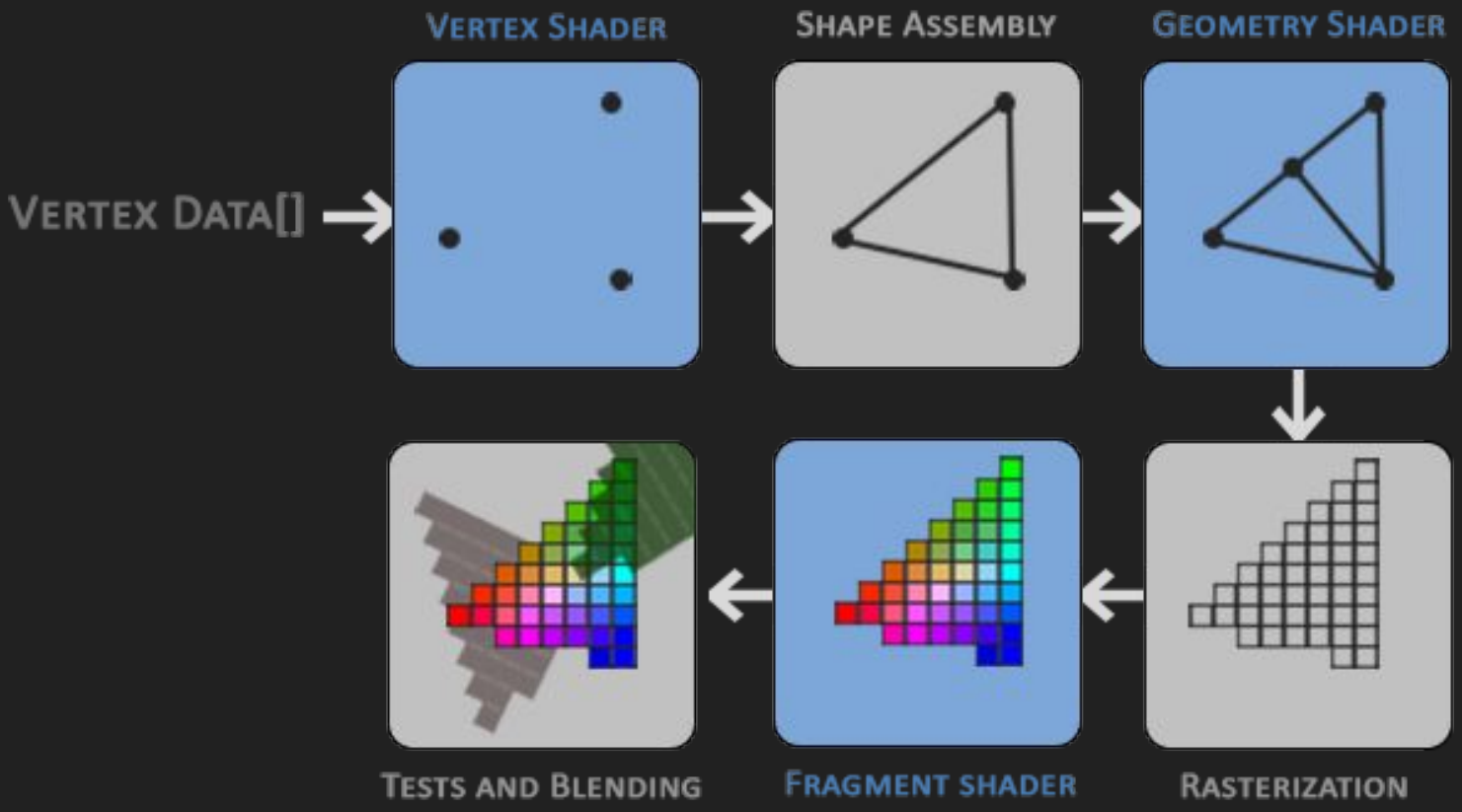
6. Renderização

- Requer
  - Geometria (modelo)
  - Instruções de como desenhar (*shader*)
- Shader
  - Vertex Shader
  - Geometry Shader
  - Fragment (Pixel) Shader

6. Renderização

- 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

Pipeline da renderização

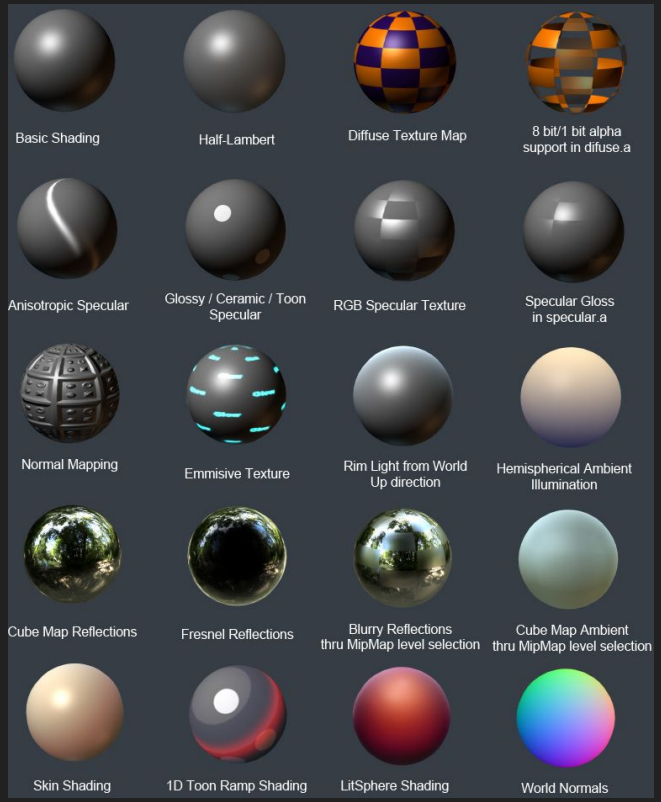


http://learnopengl.com/#!Getting-started/Hello-Triangle

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

Fragment (Pixel) Shader



http://polycount.com/discussion/70739/lcubershader-cgfx-a-maya-viewport-shader-for-fun-and-learning

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

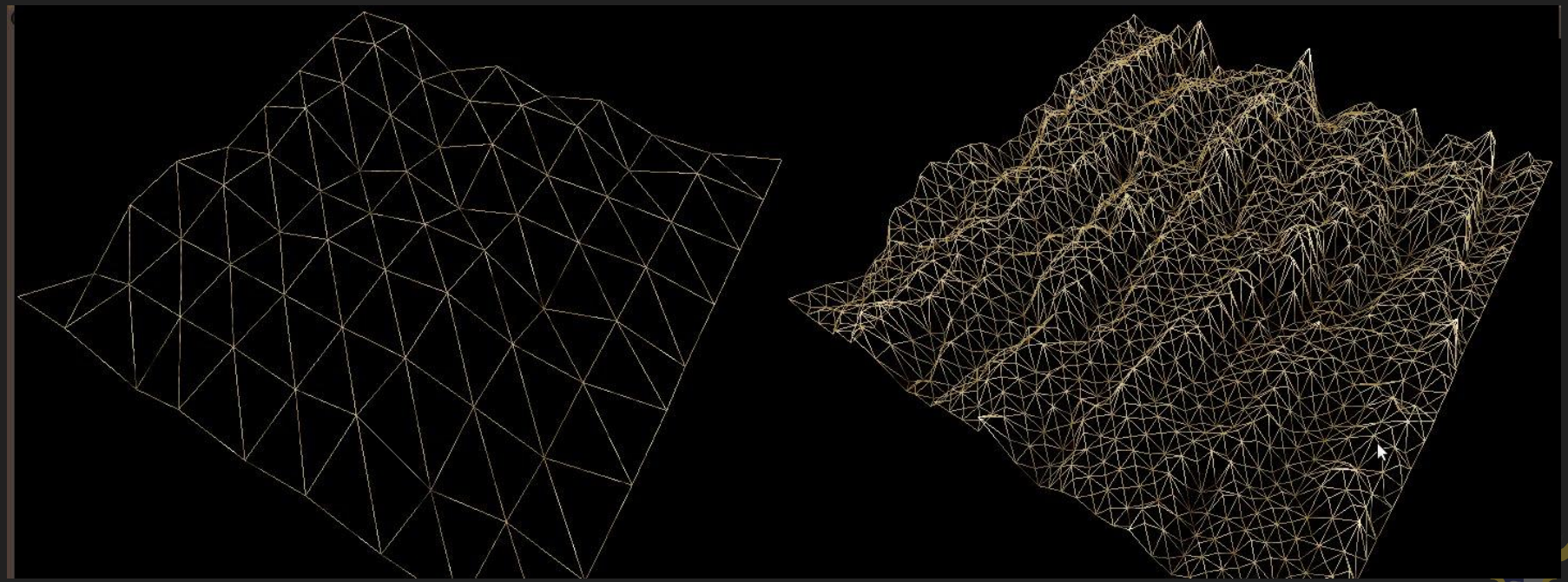
6. Renderização

- Vertex Shader
  - Transformações de mundo
  - Matriz MVP (Model View Projection)
  - Mapeamento de coordenadas
- Geometry Shader
  - Altera os vértices
  - Cria novos vértices

6. Renderização

- Tesselation Shader
  - Quebra a geometria
  - Adiciona detalhes
- Fragment Shader
  - Processamento paralelo
  - Interpolação dos vértices

Geometry Shader



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

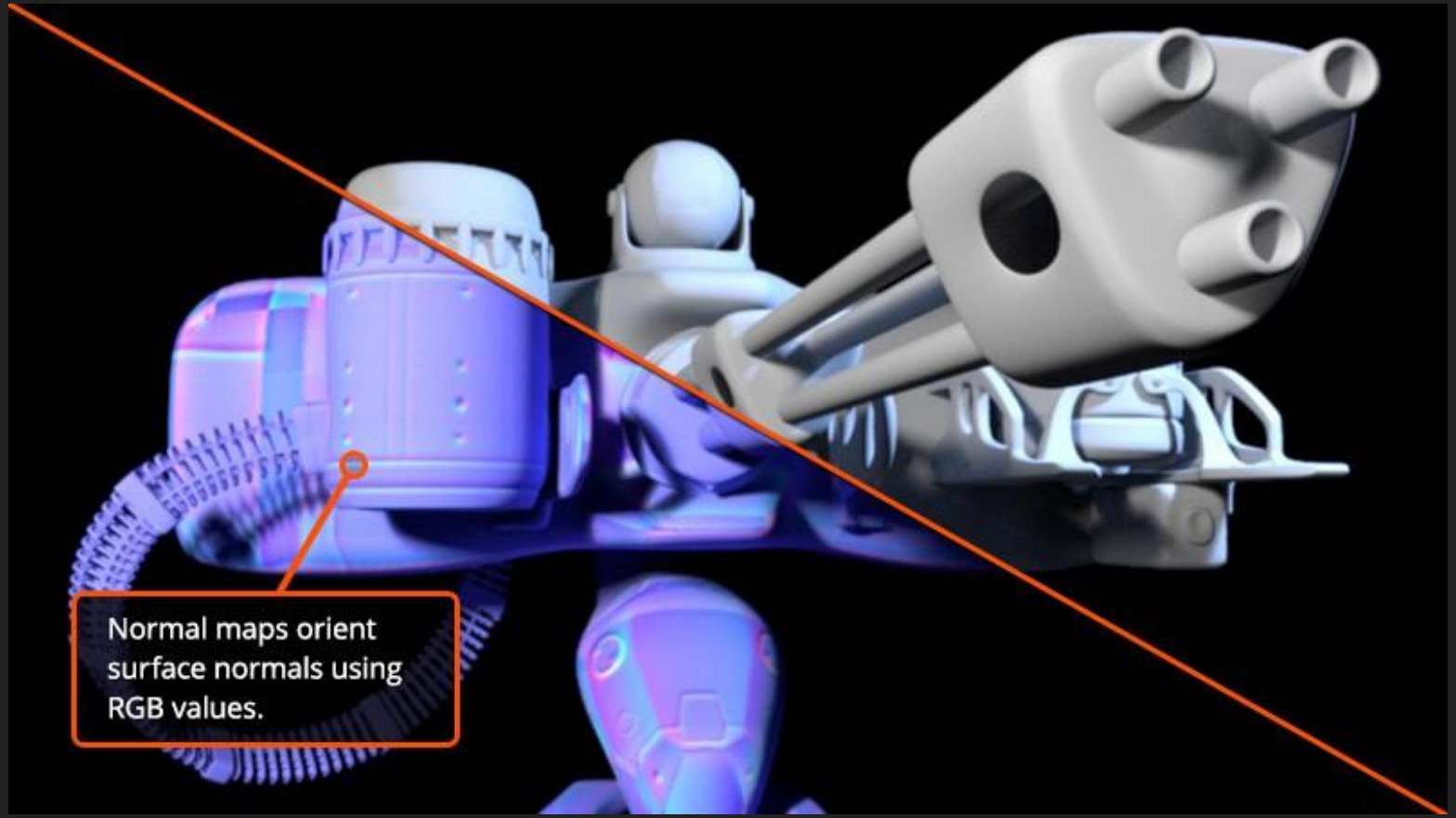
6. Renderização

- 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

Normal Map

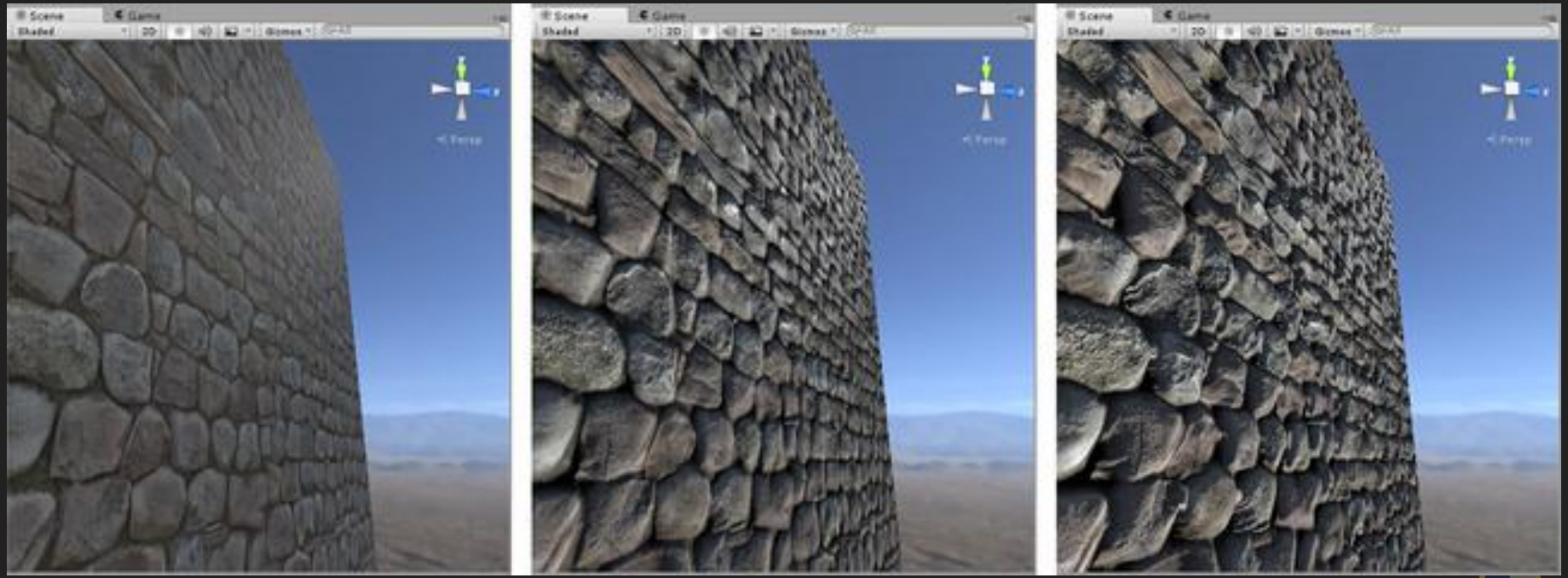


http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

6. Renderização

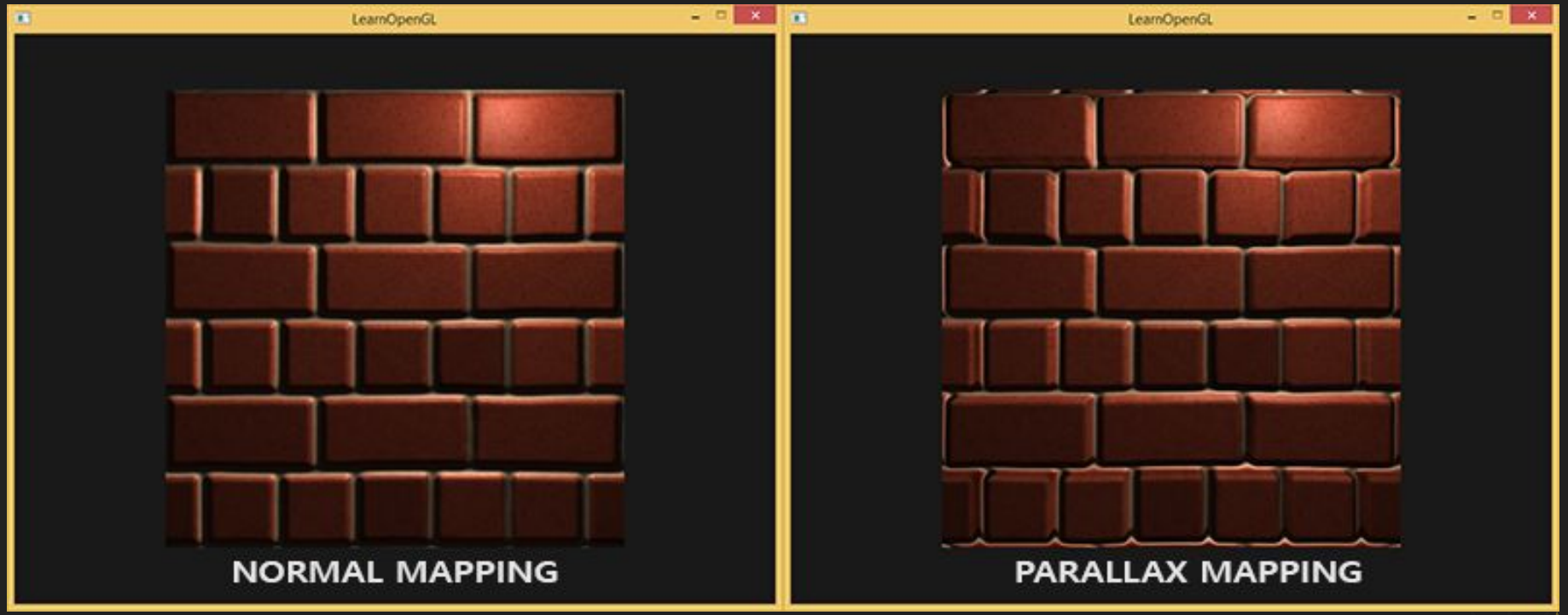
- Heigth ou Parallax map
  - Similar ao normal, mas mais complexo (e mais pesado)
  - Normalmente usado em conjunto com mapas normais
  - Move áreas da textura da superfície visível
    - Alcança um efeito a nível de superfície de oclusão
  - Protuberâncias terão suas partes próximas (frente à camera)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

Heightmap ou Parallax Map



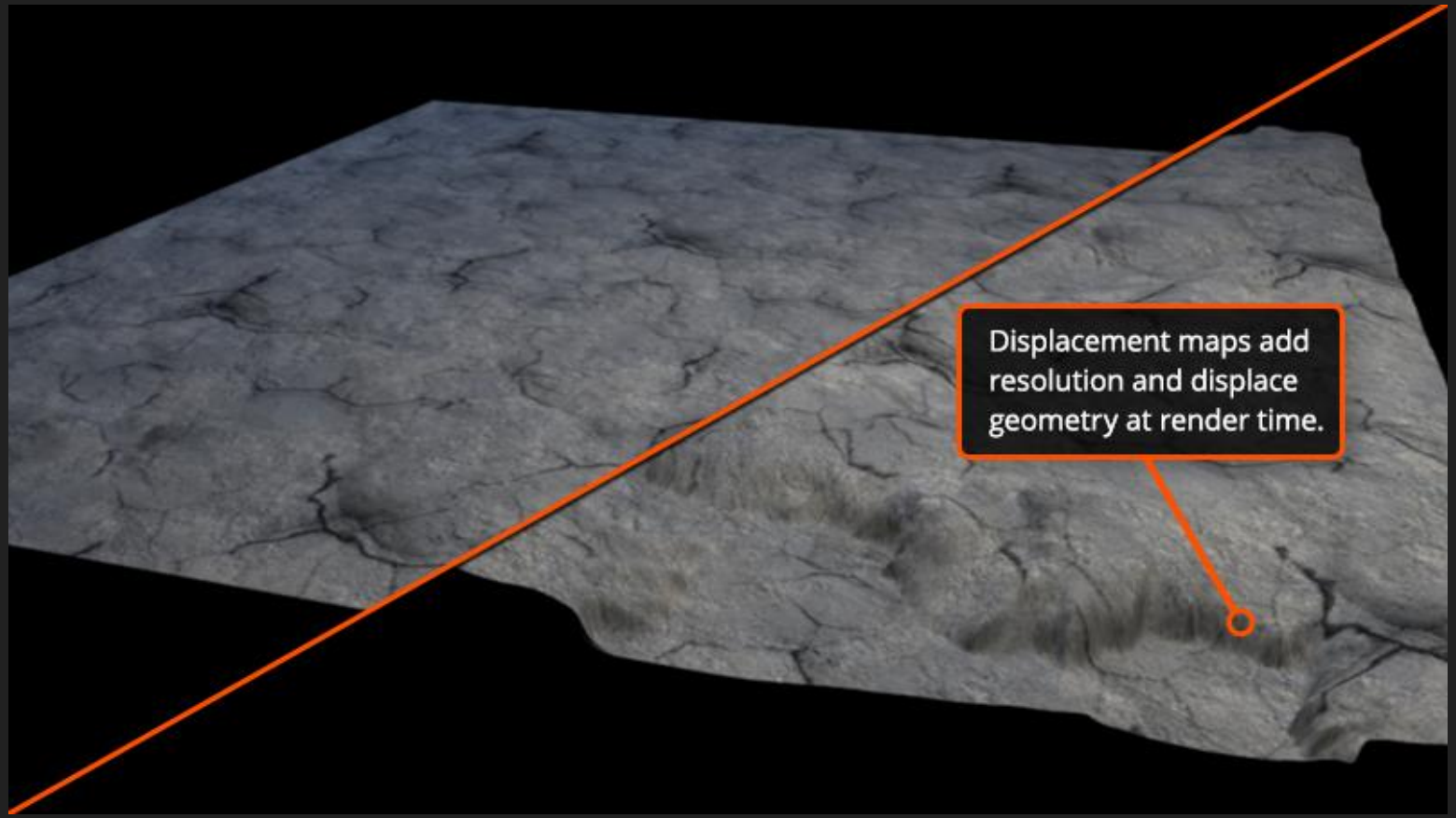
http://docs.unity3d.com/Manual/StandardShaderMaterialParameterHeightMap.html

Normal e Parallax Map



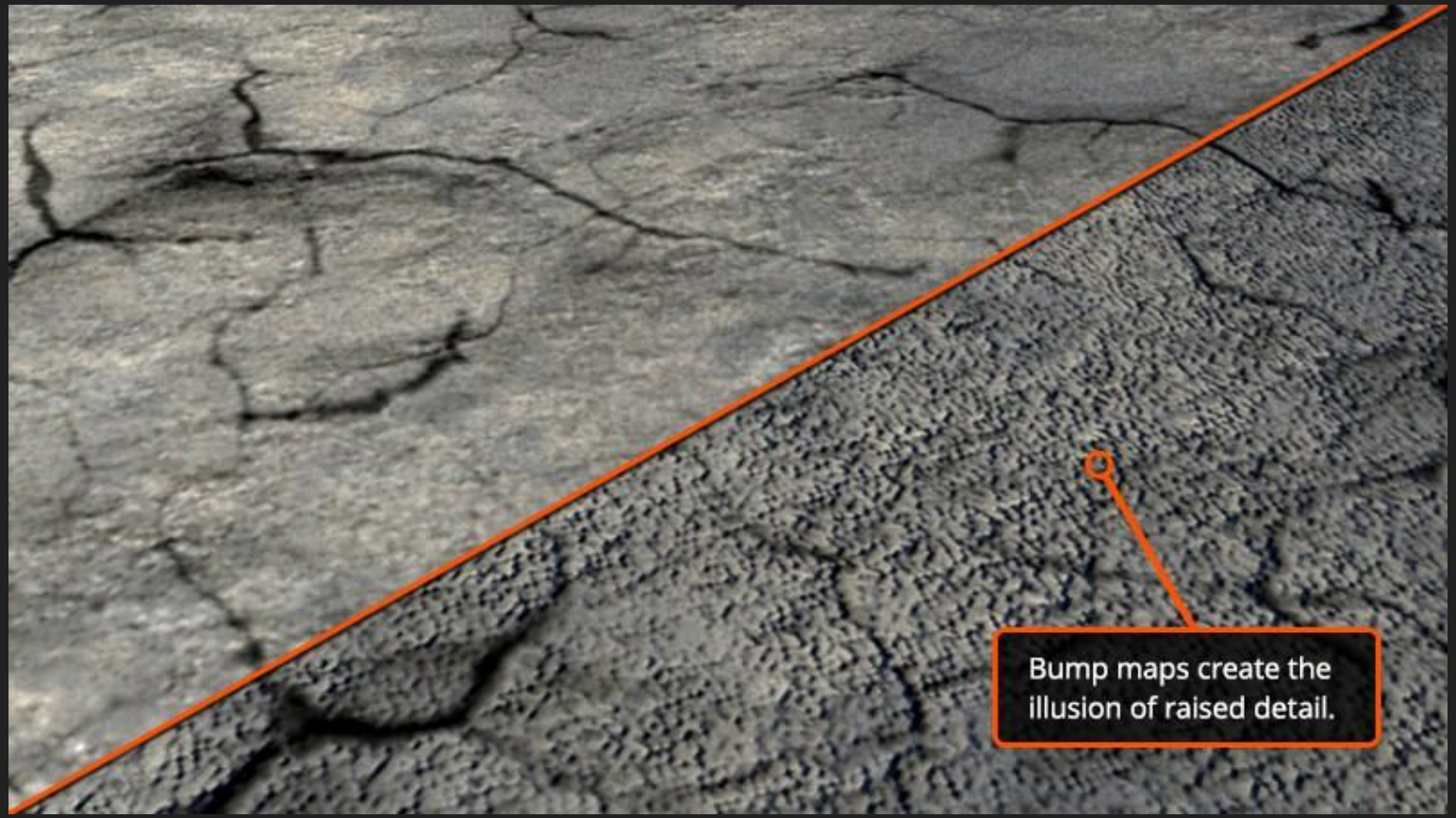
http://learnopengl.com/#!Advanced-Lighting/Parallax-Mapping

Displacement Map



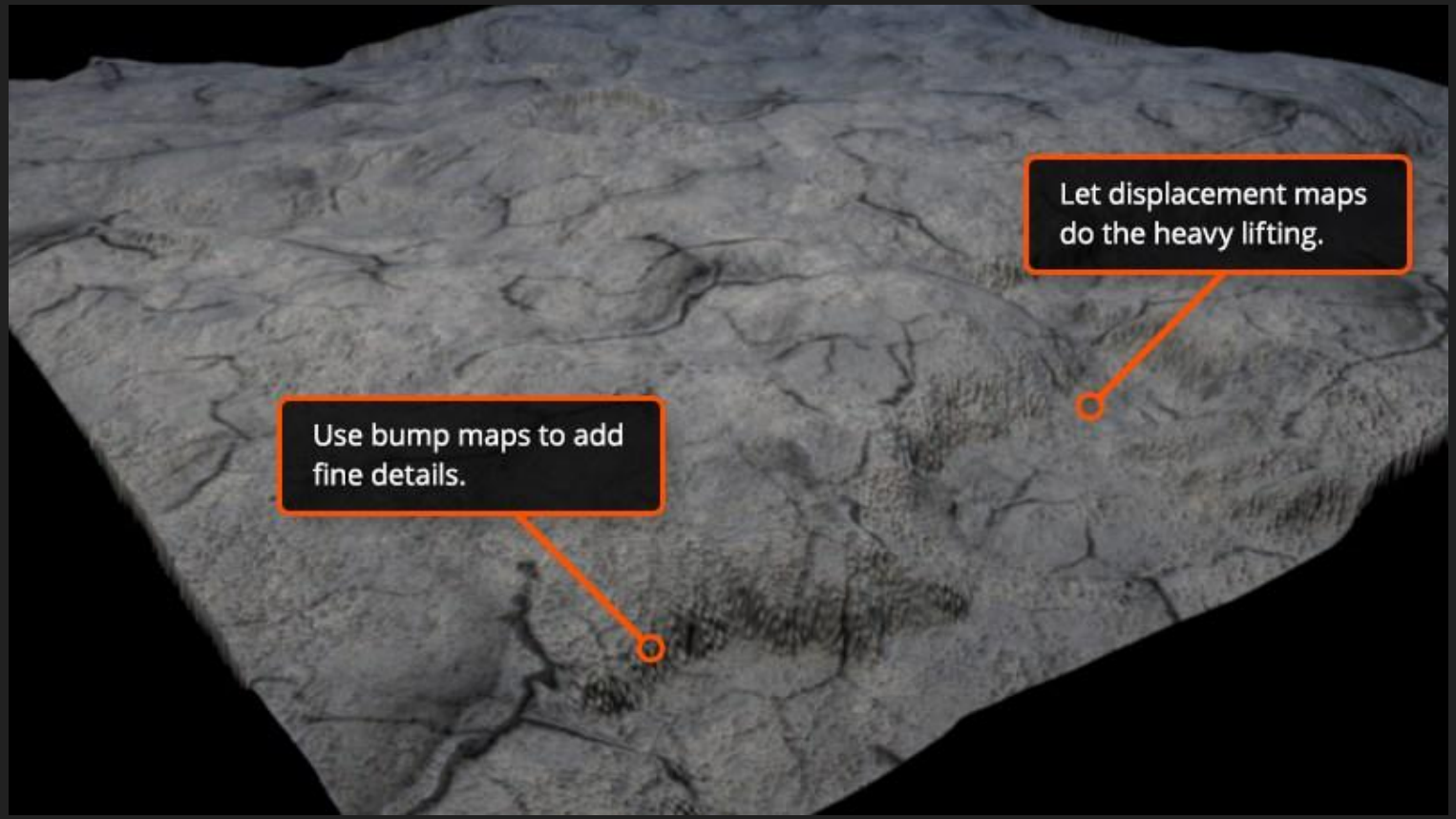
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Bump Map



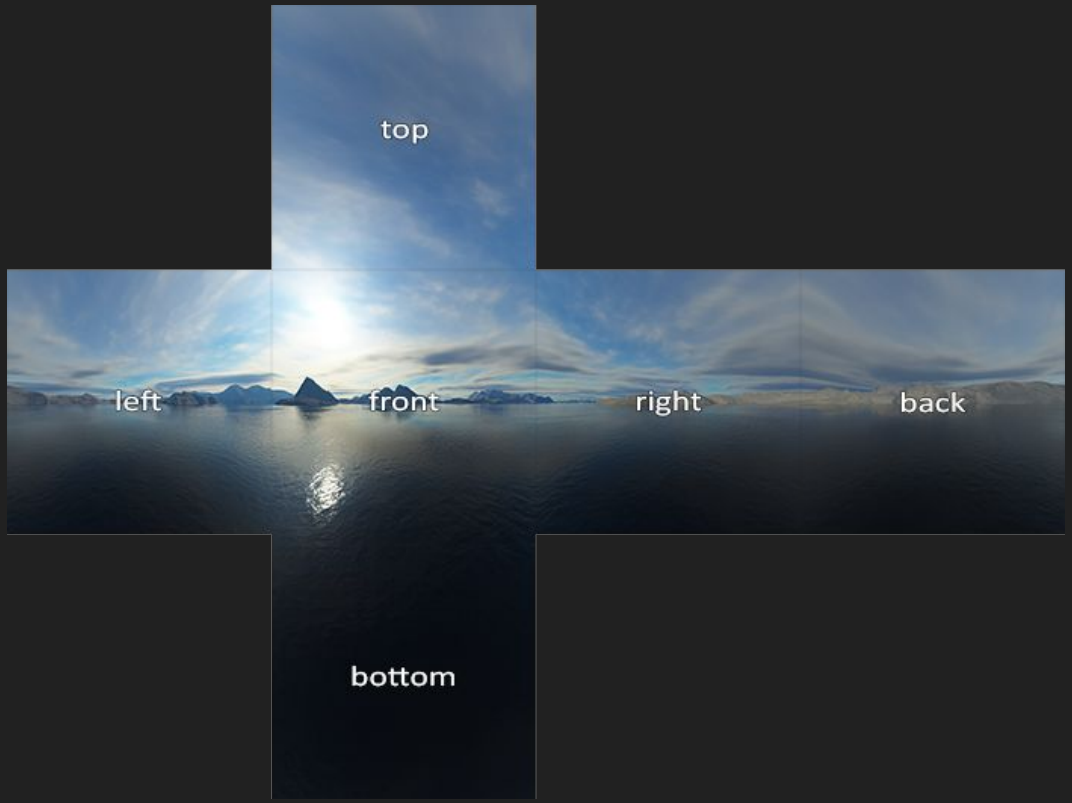
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Bump e Displacement Maps



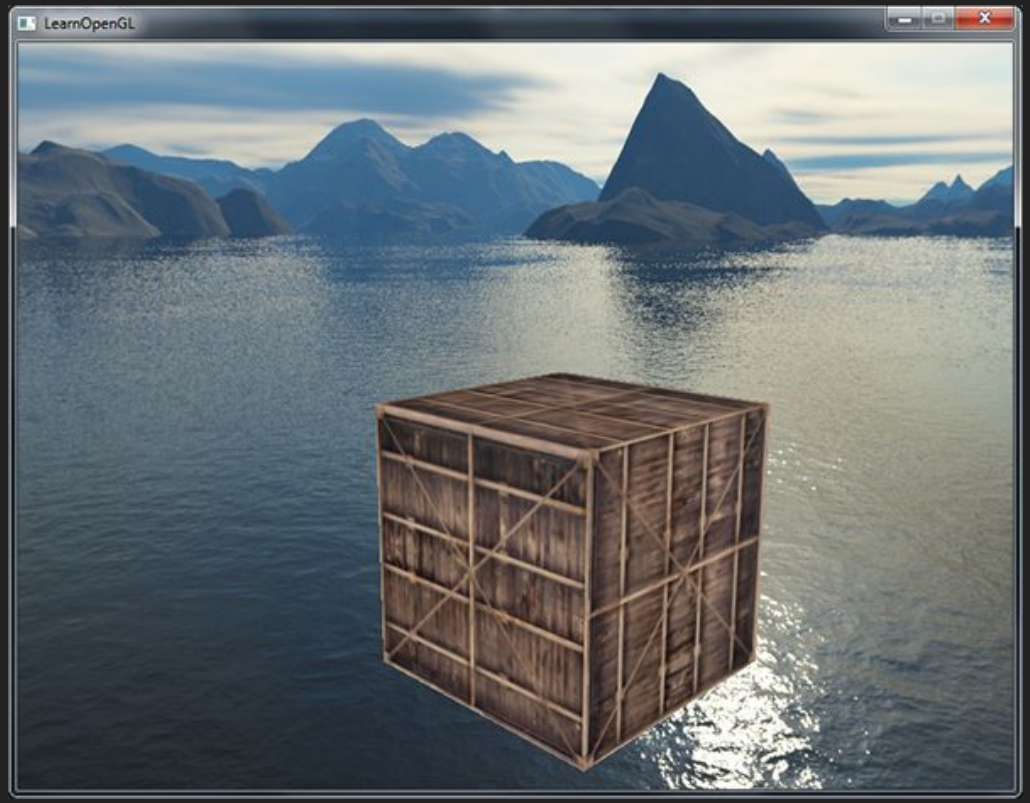
http://blog.digitaltutors.com/bump-normal-and-displacement-maps/

Cube Map



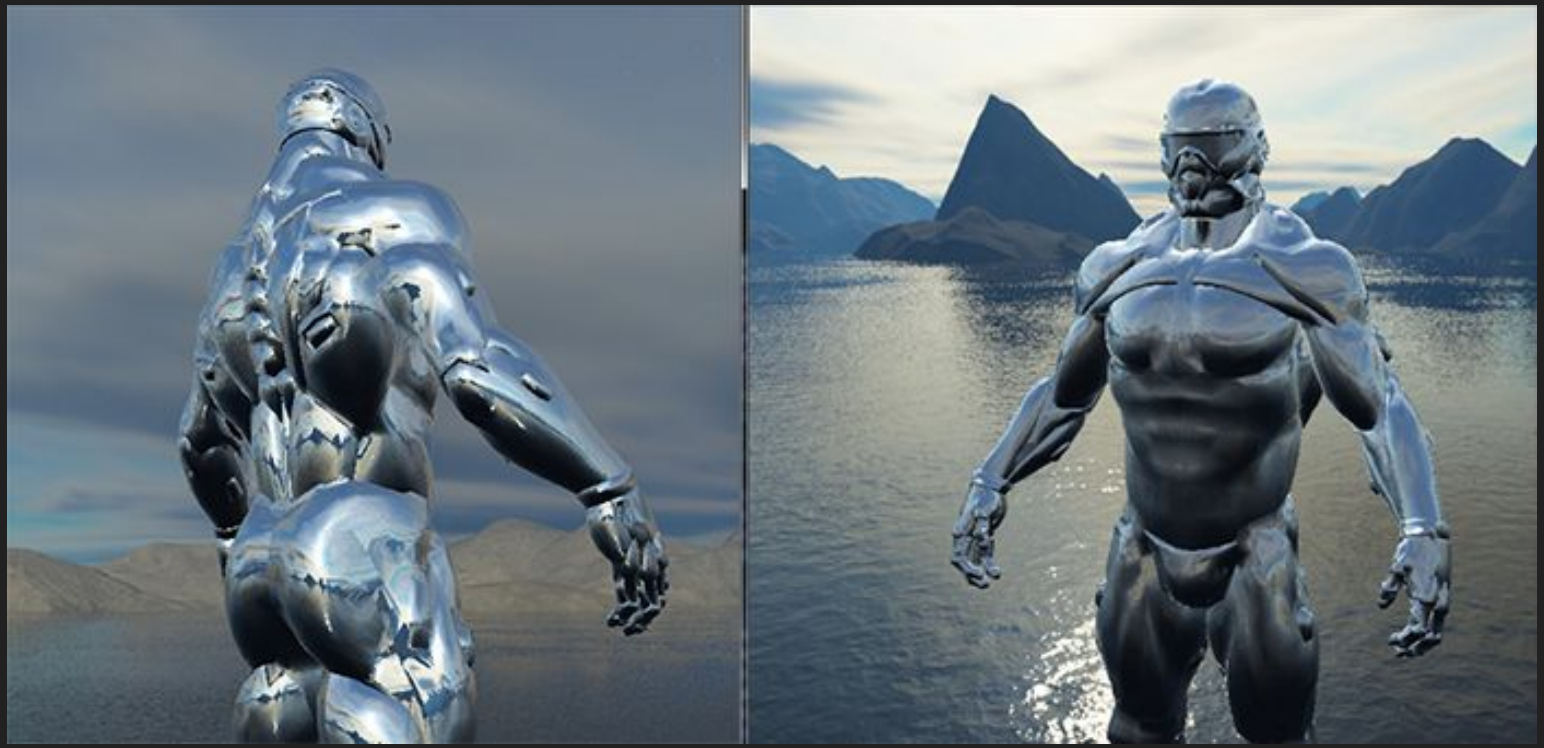
http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Skybox



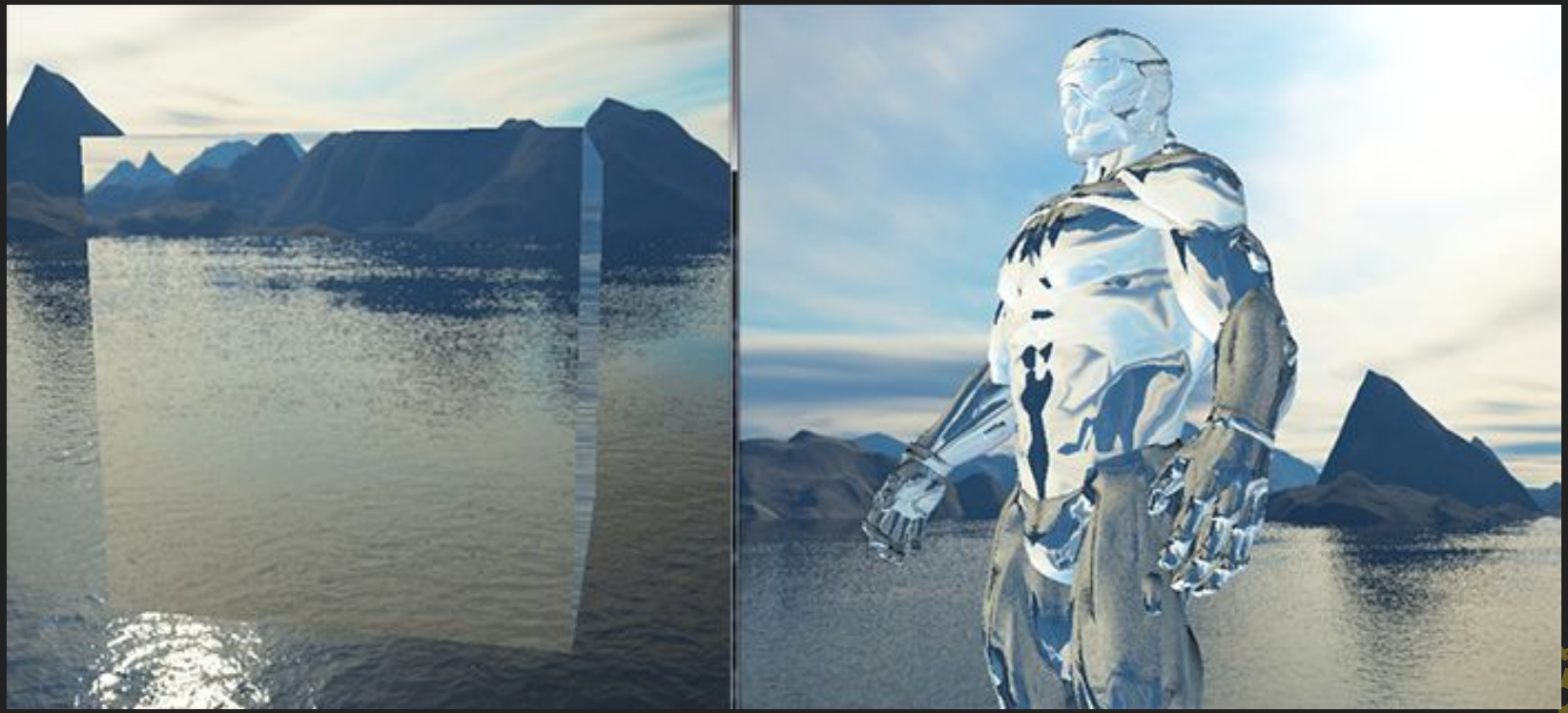
http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Reflexão



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

Refração

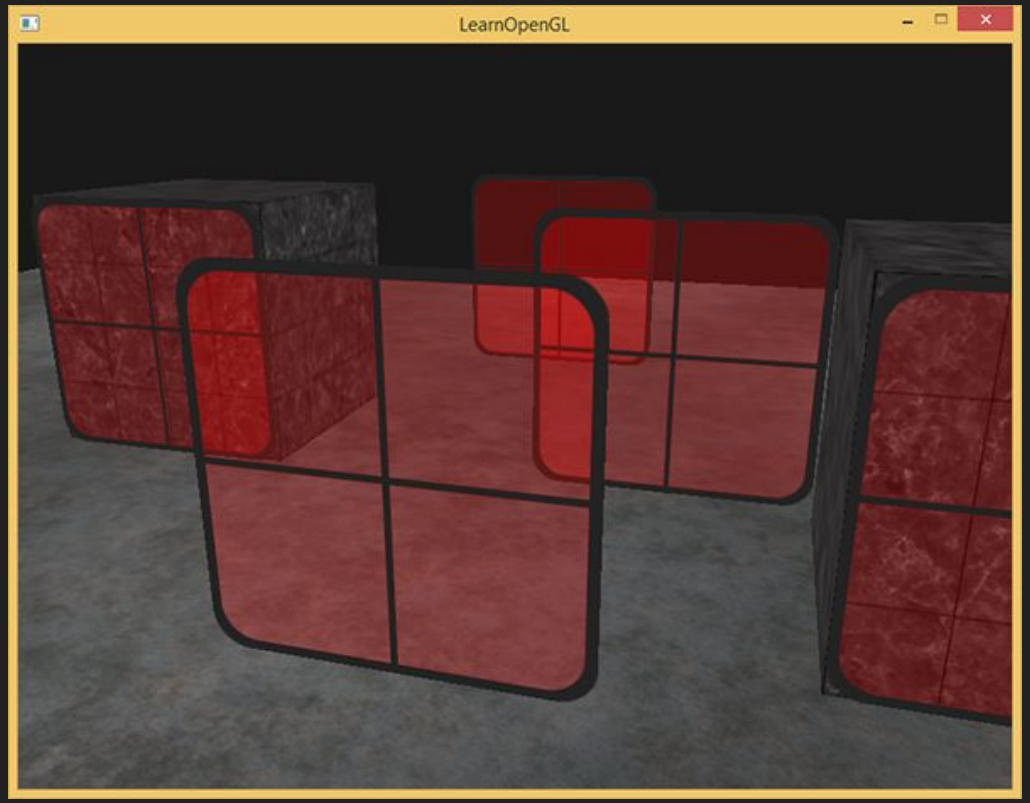


http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

6. Renderização

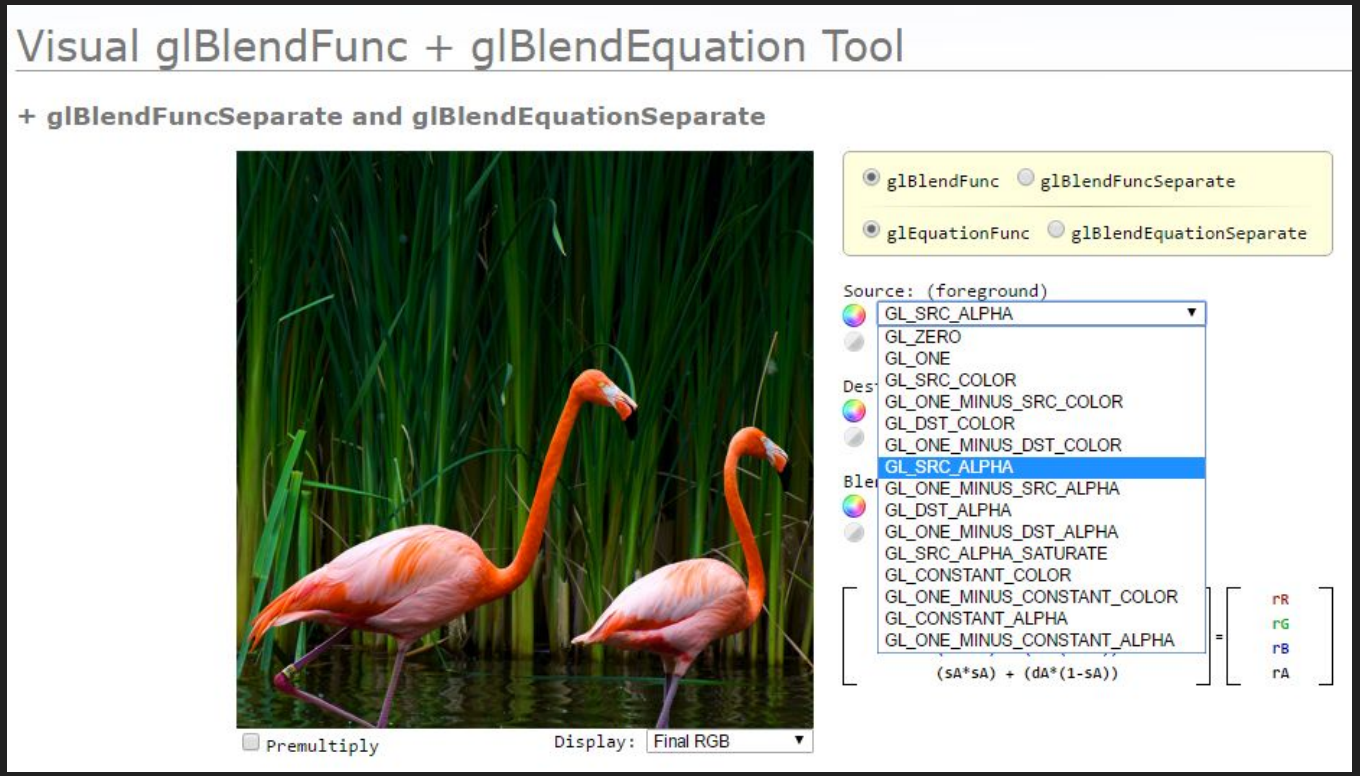
- Blending
  - Aritmética entre Alpha
  - Transparência
  - Translucência

Blending



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

Blending

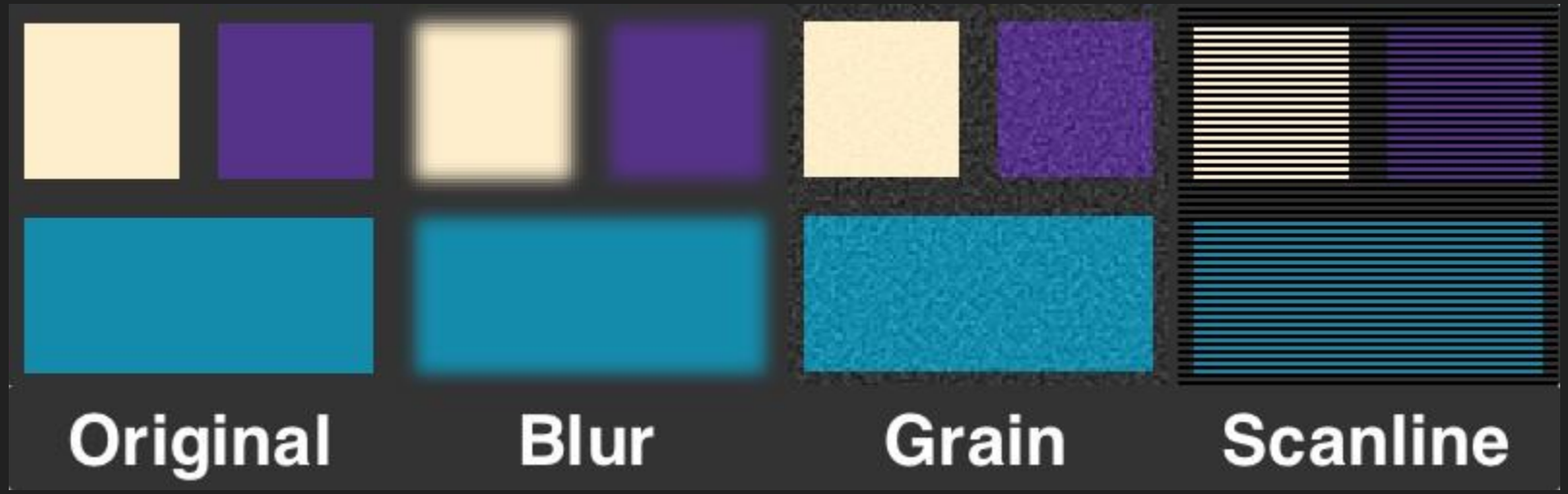


http://andersriggelsen.dk/glblendfunc.php

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/

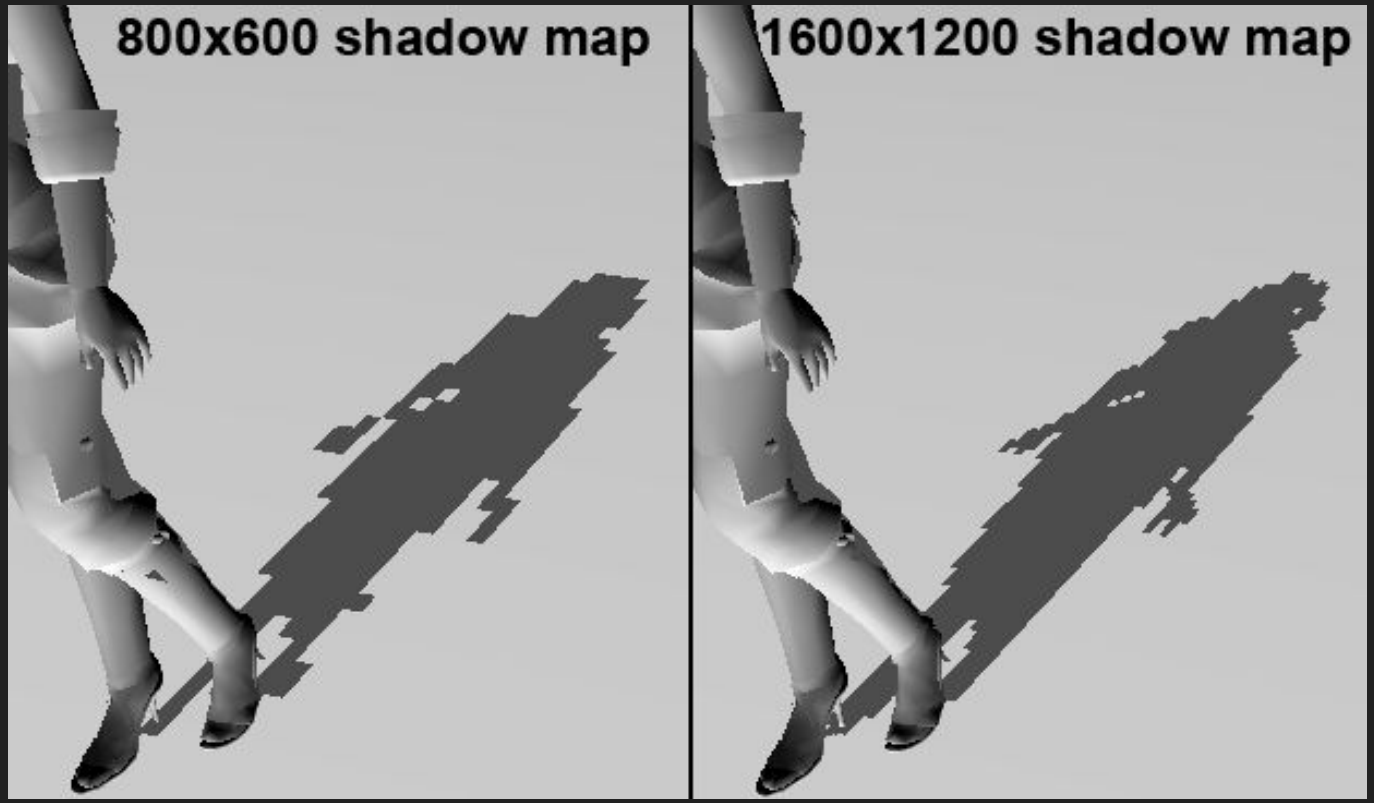
Depth of Field

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

6. Renderização

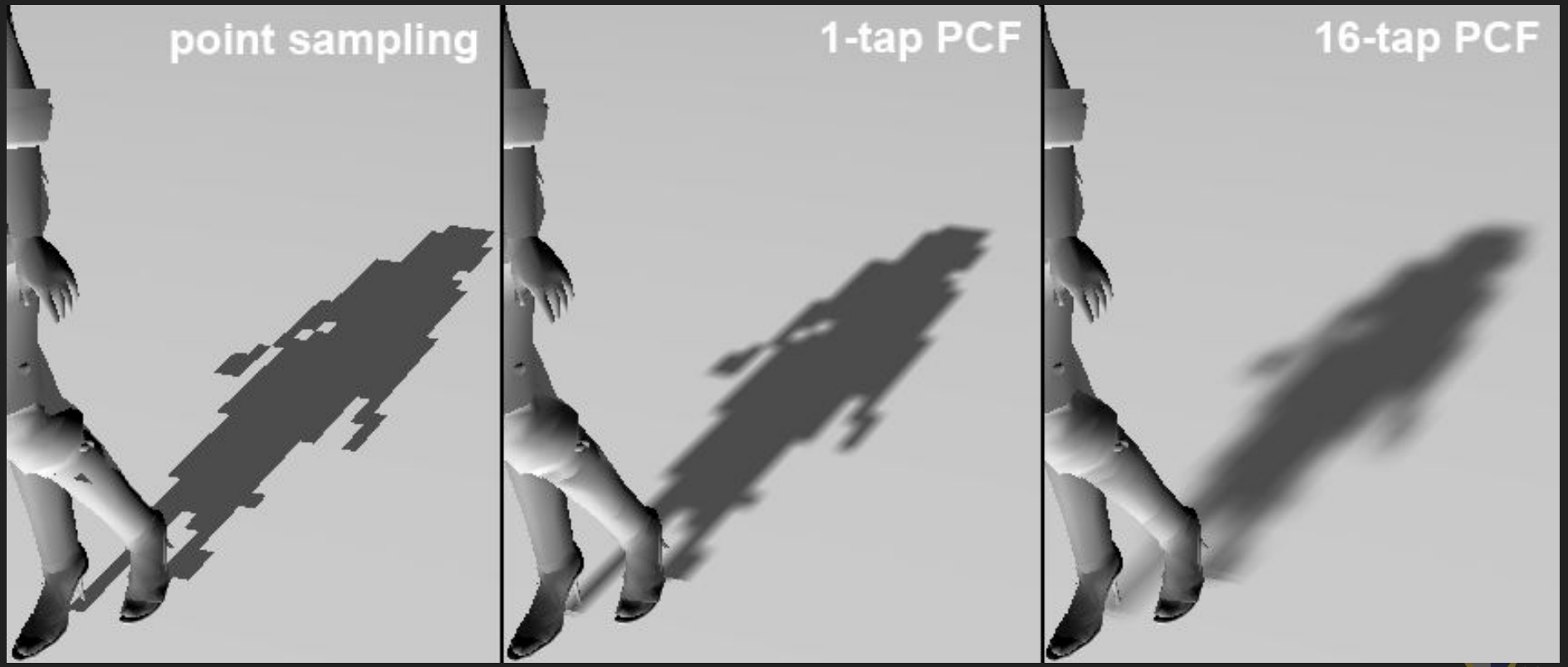
- Sombra
  - Alto custo
  - Mapeamento
  - Aproximação
  - Anti-aliasing

Shadow Map



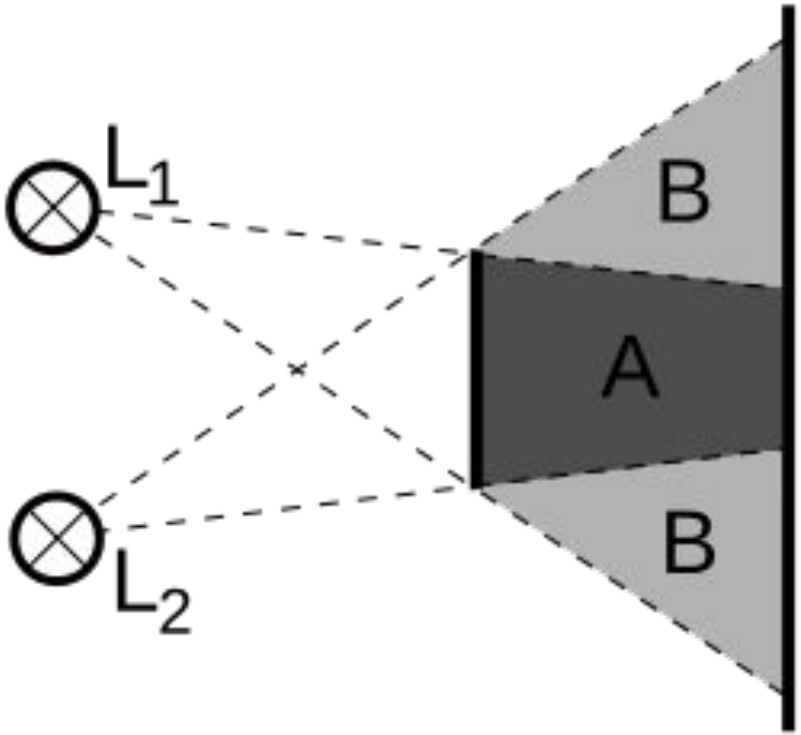
https://takinginitiative.wordpress.com/2011/05/25/directx10-tutorial-10-shadow-mapping-part-2/

Shadow Map



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

Soft Shadow e Penumbra

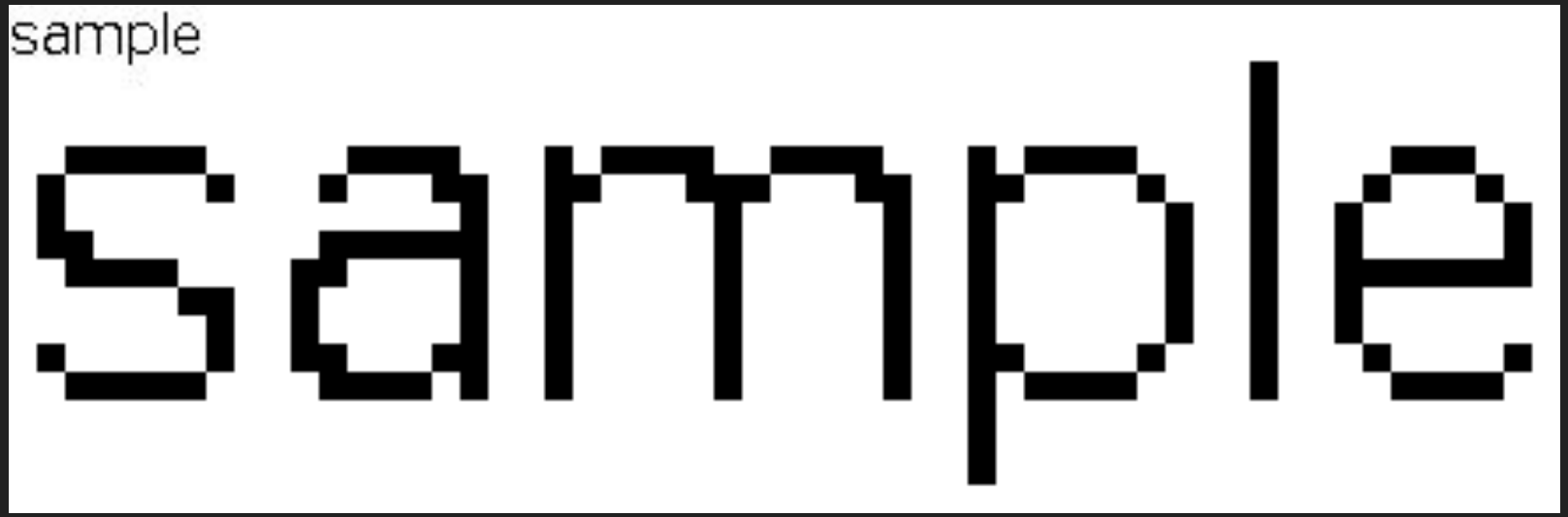


https://en.wikipedia.org/wiki/Umbra,_penumbra_and_antumbra

6. Renderização

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

Rasterização simples



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

Rasterização simples

sample

sample

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

Ray tracing

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

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/

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