Introdução à Computação Gráfica

por Rossana B Queiroz

INTRODUÇÃO

- Cenas realistas/**convincentes**
- Aplicação é interdisciplinar

Jogo: God of War™ Ragnarök
(Santa Monica Studio, 2022)

Processamento Gráfico

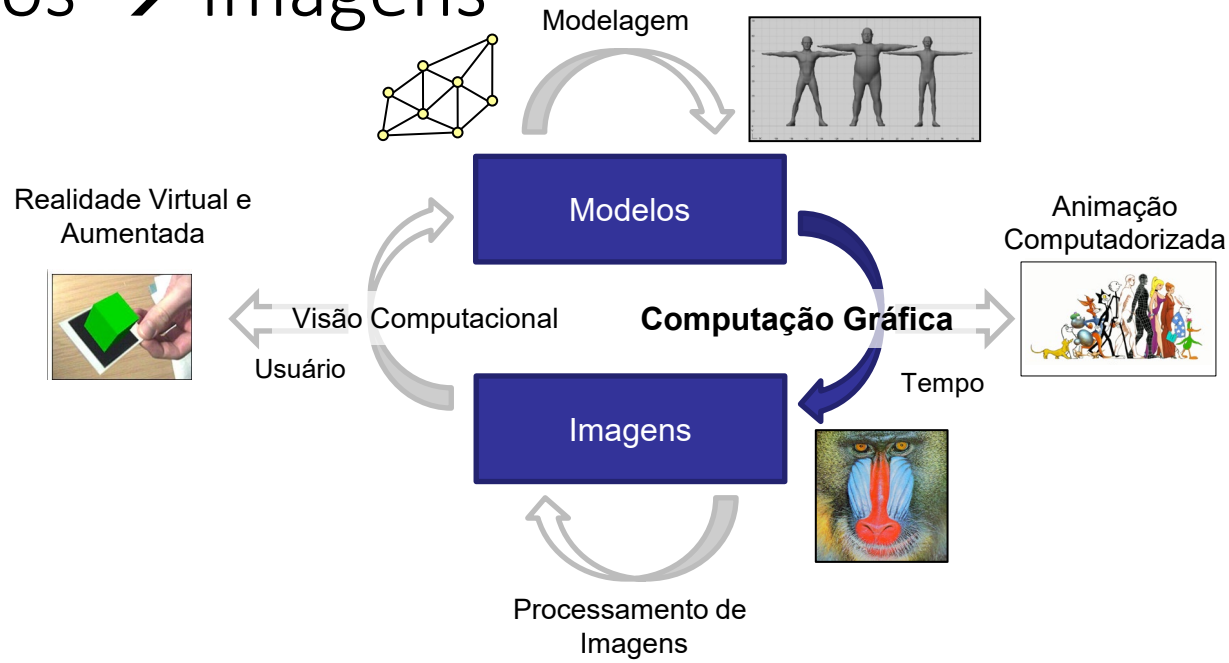
É o processamento de informações visuais, tanto para geração de imagens, quanto obtenção de dados.

Processamento Gráfico

- Divide-se em duas grandes áreas (ou linhas de pesquisa):
  - Processamento de imagens
    - Tratamento de imagens
    - Visão computacional
  - Computação Gráfica
    - Síntese de imagens
    - Pipeline - processo ou cadeia de renderização:
      - *Pipeline* 2D
      - *Pipeline* 3D

Computação Gráfica

- Subárea do Processamento Gráfico
- Modelos → Imagens



Processamento de imagens

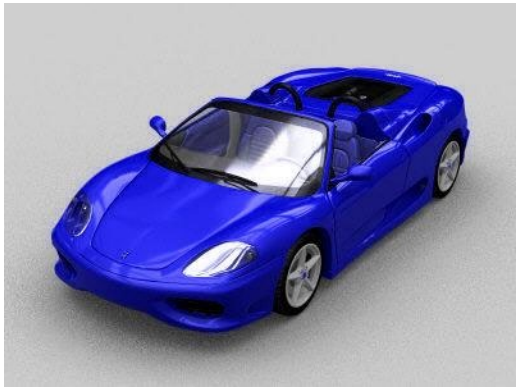
- Processamento de imagens visa a obtenção de informações da imagem para produção de dados a respeito da mesma ou modificação da imagem
- Tratamento de imagens:
  - Trata da geração de novas imagens a partir de imagens de entrada. A rigor não extrai informações da imagem.
  - Exemplos de aplicativos comerciais:
    - GIMP (software livre), Adobe Photoshop, Paint Shop Pro, ...

Processamento de imagens

- Tratamento de imagens:

  Para efeitos artísticos ou correção/destaque de cores da imagem

Filtro de imagem que troca uma cor por outra



Filtro de imagem aplica efeitos de foto envelhecida

Visão Computacional

- É um processo de analisar a imagem e obter dados que possuem algum significado.
- Exige um alto processamento computacional para extrair dados de uma imagem.
  - **Normalmente, implica em percorrer todos os pontos da imagem e, para cada ponto, analisar a sua vizinhança.**
- Exemplo: detecção de rostos, robótica, reconhecimento de placas de veículos, autenticação baseada em imagens, etc.

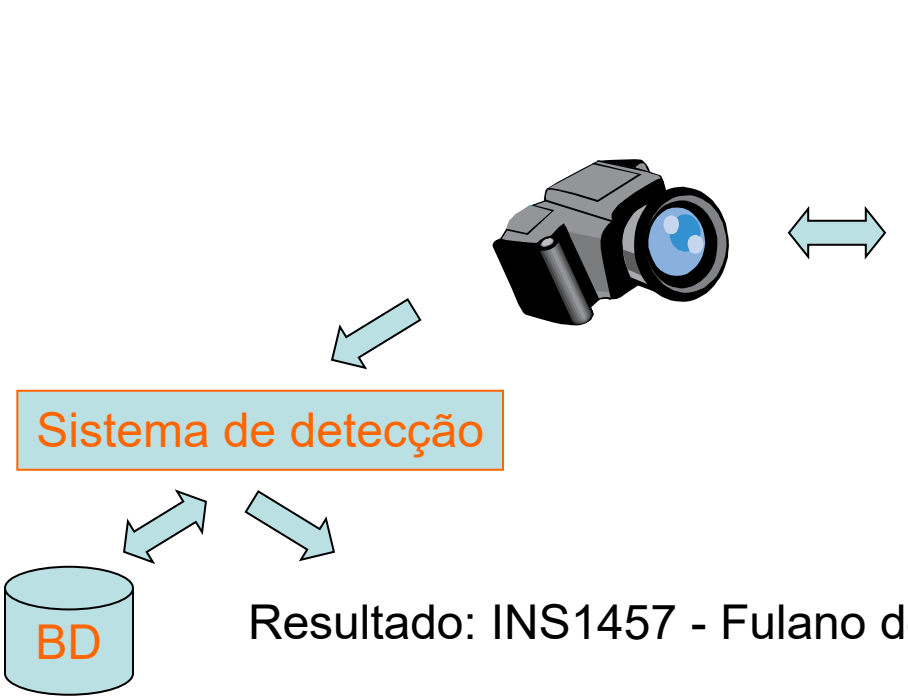
Processamento de Imagens

- Visão Computacional

Algoritmo de detecção de regiões (pixels conectados) com a tonalidade da pele

Detecção de pele usando Redes Neurais Artificiais (Bittencourt & Osório, 2002)

Visão Computacional



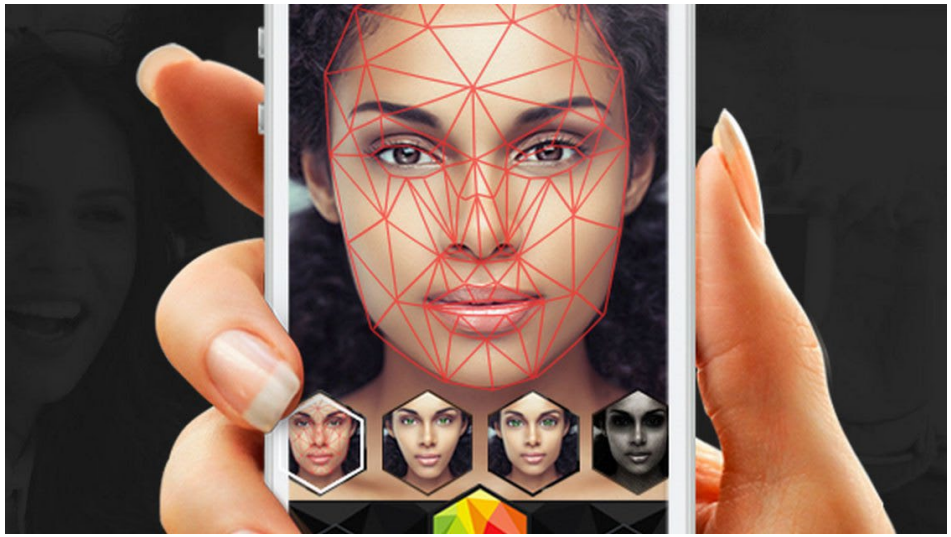
Resultado: INS1457 - Fulano de Tal - AUTORIZADO

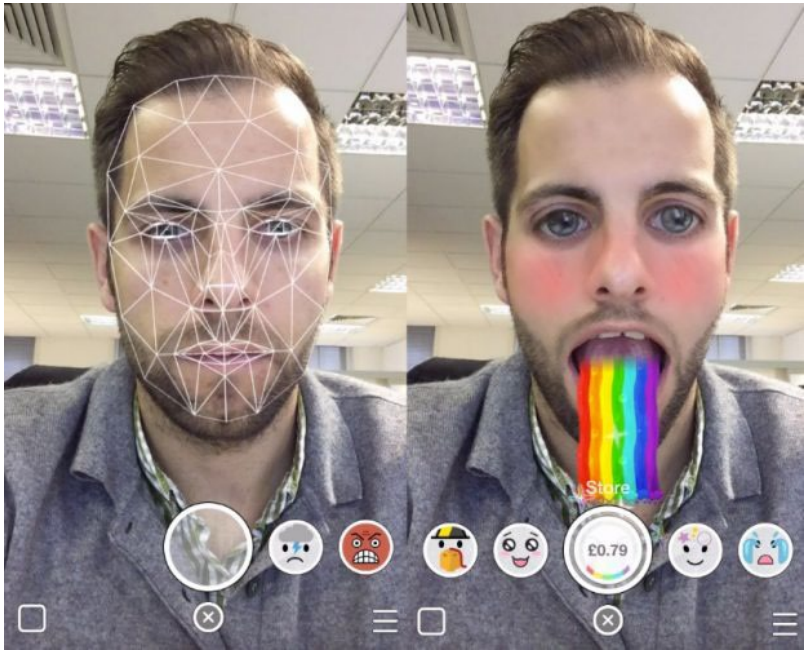
Sistema de reconhecimento de placas e autenticação de veículos para condomínios: Câmera fotografa o veículo e sistema faz a detecção e o reconhecimento dos números da placa.

Realidade Aumentada

- Detecção de marcações ou padrões na imagem
  - Visão Computacional/Processamento de Imagens
- Inserção de elementos de Computação Gráfica
  - Computação Gráfica

Realidade Aumentada

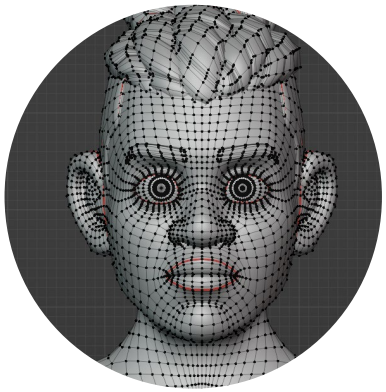




Computação Gráfica

- É a área que processa **modelos** e gera uma **imagem**
  - Transformação de dados em imagem. Dados são a parte do modelo, descrevendo a informação da cena que será **renderizada**
    - Geometria
    - Aparência
    - Ação

A Tríade da CG



Forma

Modelagem Geométrica

Aparência

Rendering

Ação

Animação

https://studio.blender.org/characters/snow/v2/

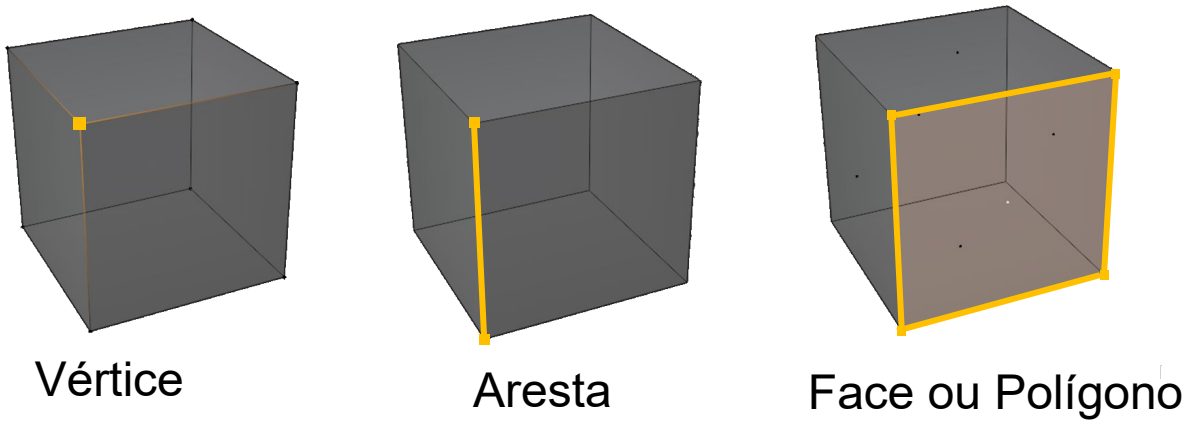
Animação Computadorizada

- Arte de criar imagens em movimento utilizando o computador.
  - Sub-área da Computação Gráfica responsável pelo elemento Ação da tríade
    - Transformações ao longo do tempo
    - Seqüência de imagens com coerência temporal
- Quando utilizada em filmes, é também chamada de *CGI - Computer Generated Imagery*

Luxo Jr., 1986
Curta-metragem da Pixar

Primitivas Gráficas

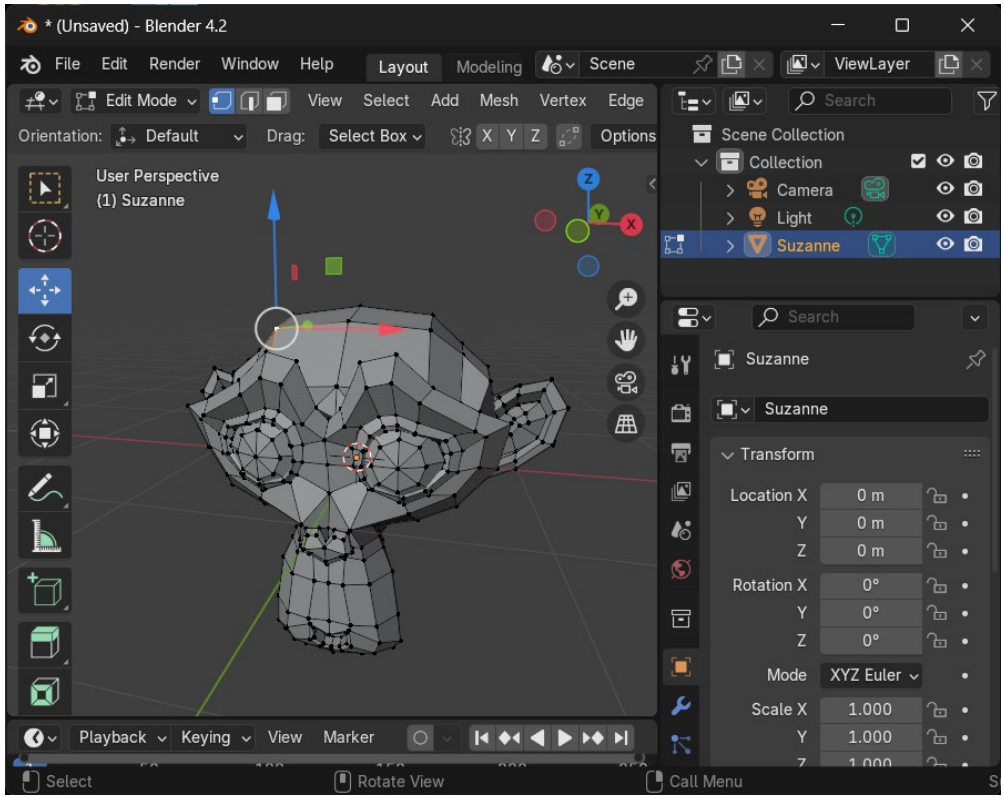
- Modelagem Poligonal – Descrição da FORMA dos objetos que compõem a cena a partir de primitivas geométricas
  - As **primitivas** básicas em CG são:



  - **Vértices** são conectados para formarem **polígonos**, que são conectados para formar a **malha poligonal** do objeto modelado

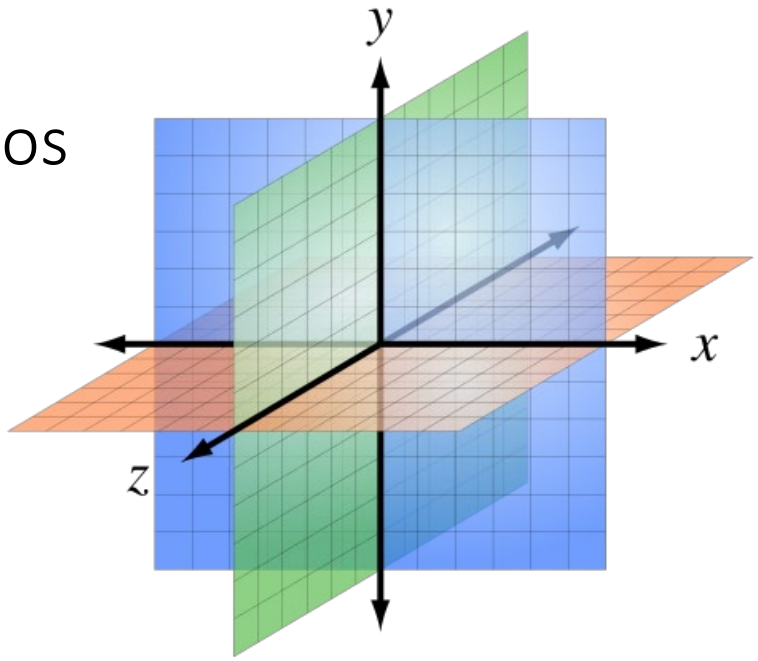
Modelagem Poligonal

- Processo de criar modelos (geralmente em 3 dimensões) a partir da adição e operações (transformações matemáticas) de/nas primitivas gráficas
- Geralmente feita com auxílio de uma interface gráfica avançada que facilita a navegação e manipulação dos objetos e suas primitivas



Representação do Mundo

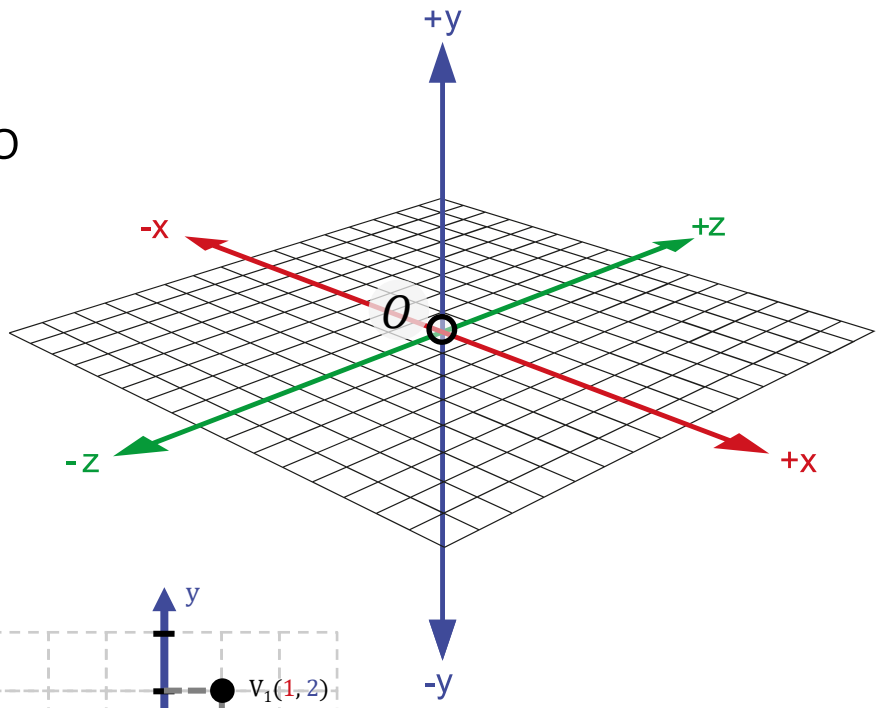
- Sistema cartesiano com 3 eixos perpendiculares entre si
  - x - largura
  - y - altura
  - z - profundidade

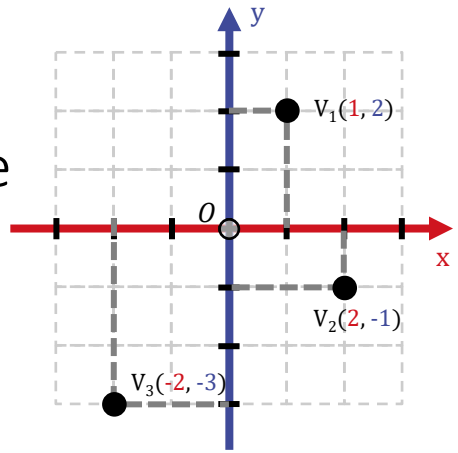


⚠ OBSERVAÇÃO! É bem comum livros de matemática e mesmo alguns softwares considerarem o eixo z como o de altura e o y como de profundidade. Isso não é um problema, desde que mantenhamos a coerência em nossa representação

Representação do mundo

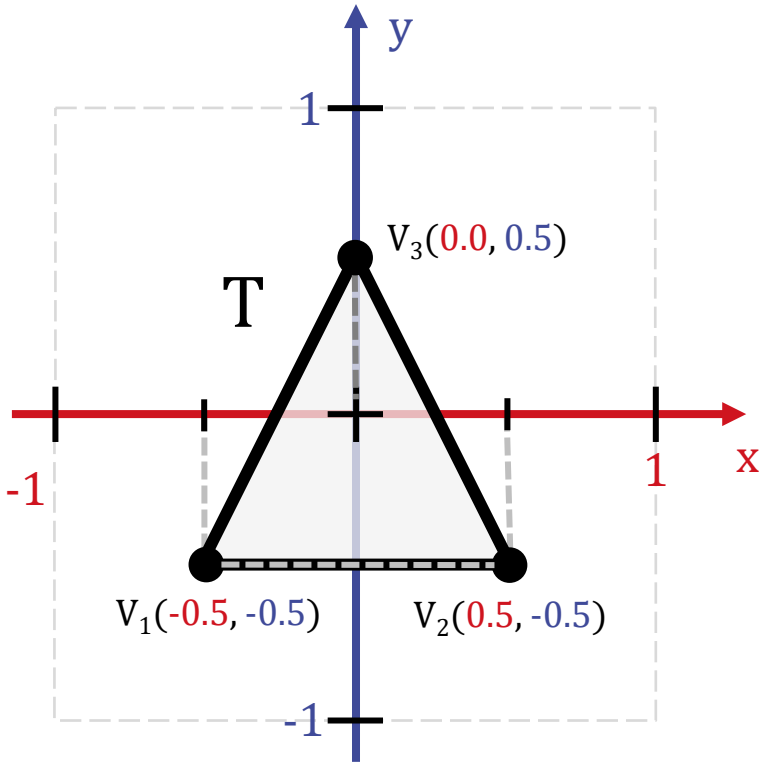
- Origem (O) do sistema cartesiano
  - Ponto onde os 3 eixos interceptam entre si
  - Nesse ponto, as coordenadas nos 3 eixos possuem o valor 0
    - $O(0.0, 0.0, 0.0)$
- Os vértices são pontos que possuem suas coordenadas definidas nas dimensões desse sistema
  - 2D: duas dimensões – $V(x, y)$
  - 3D: três dimensões – $V(x, y, z)$





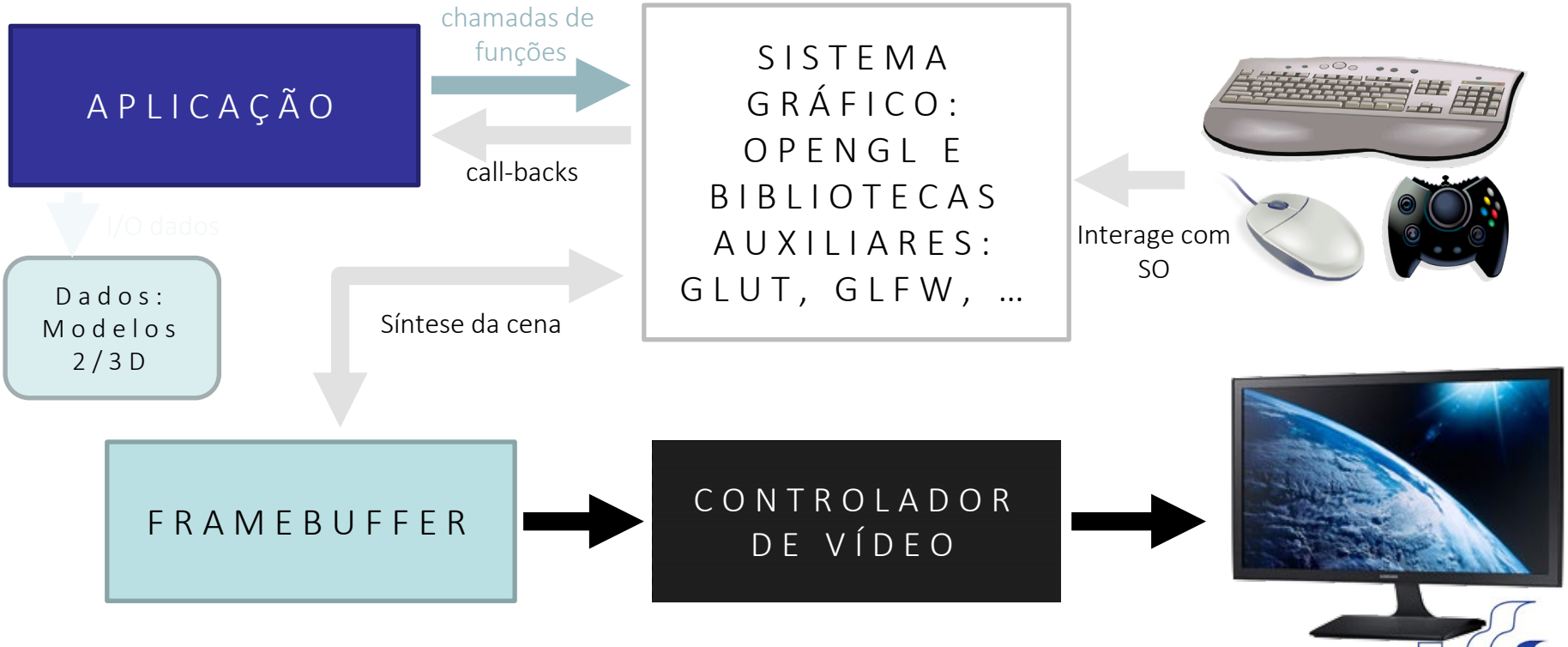
Primitivas Gráficas

- Os polígonos (ou faces) são formados pela conexão entre vértices
  - O polígono com menor número de vértices é o triângulo
    - 3 vértices não colineares entre si
    - $T(V_1, V_2, V_3)$



Computação Gráfica

Esquema conceitual de aplicações de CG:



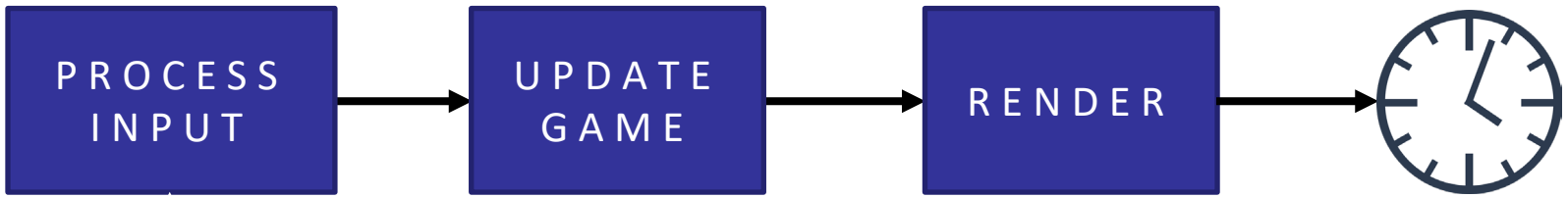
Computação Gráfica

- **Frame buffer**: É uma porção de memória usada para criar o pixel map que será enviado para o monitor.

- **Double buffering**: técnica que utiliza um buffer auxiliar para criar imagem enquanto um buffer é desenhado (alternância). Usado para evitar o flicker (tremer a imagem)

Aplicações Gráficas

- *Game Loop* [1]
  - Objetivo: separação do código entre tratamento de entrada, processamento do jogo (estado) e questões relacionadas a progressão de tempo do jogo:



[1] Game Programming Patterns

Aplicação Gráfica

- *Game Loop*

```
while(true)
{
    processInput(); //detecção de eventos de entrada
    update(); //estado + lógica
    render(); //chamada de desenho (drawcall)
}
```

Renderização

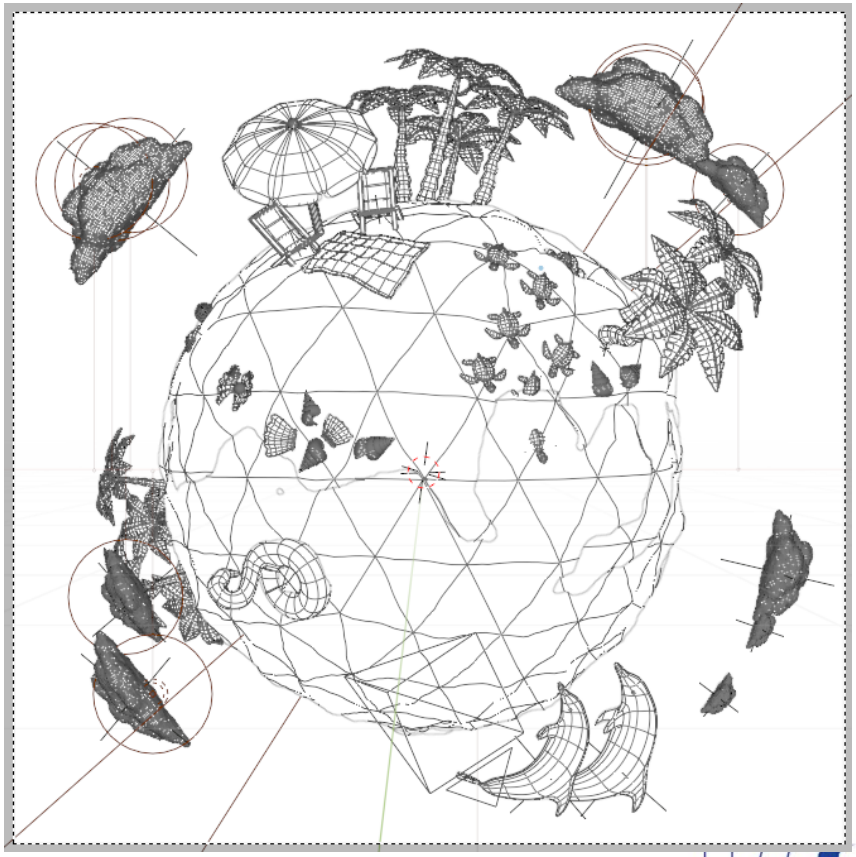
- Uma imagem é uma distribuição de energia luminosa num meio bidimensional (o plano do filme fotográfico, por exemplo)
- Dados uma descrição do ambiente 3D e uma câmera virtual, calcular esta energia em pontos discretos (tirar a fotografia)
- Resolver equações de transporte de energia luminosa através do ambiente!!

Renderização

- É o processo pela qual **obtemos uma imagem** gerada a partir do **processamento das informações visuais** que a descrevem (**modelo**)
- Este modelo é uma descrição de uma cena e pode conter informações sobre geometria, cores, propriedades e texturas de objetos, iluminação/sombreamento, etc.. projetados a partir de um observador virtual (câmera sintética)

Rendering – Exemplo

- Wireframe view
  - Apenas primitivas, sem sombreamento (*shading*)



Rendering – Exemplo

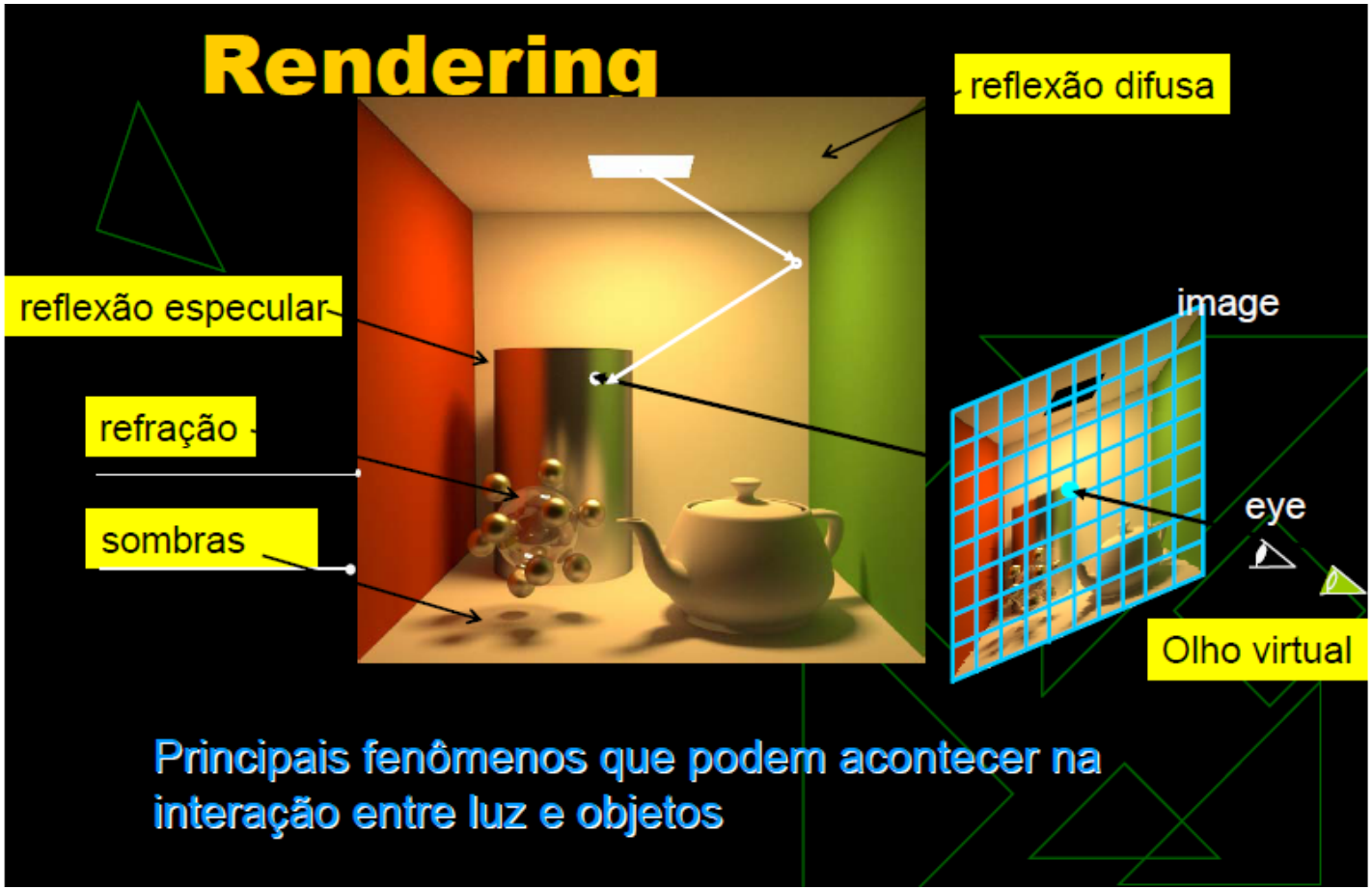
- Solid view
  - Apenas geometria, aplicando-se sombreamento (*shading*)

Rendering – Exemplo

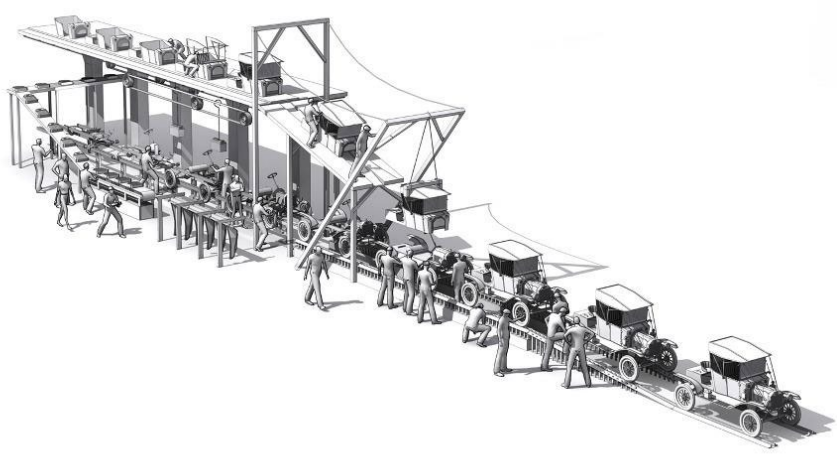
- Render view
  - Geometria e materiais
  - Iluminação
  - Pós processamento (se houver)

⚠ Neste exemplo, apesar de já estarmos mostrando o resultado com iluminação, estamos visualizando dentro do editor, enxergamos ainda alguns elementos da interface

Rendering - Iluminação

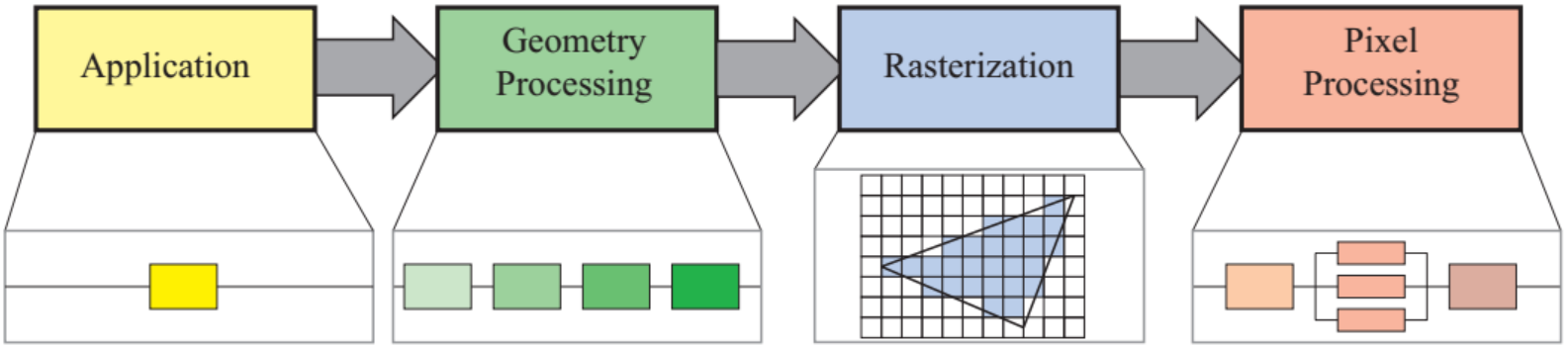


Pipeline Gráfico



- Pipeline
  - Idéia de linha de montagem
- Idealmente:
  - Uma tarefa sequencial que pode ser dividida em *n* estágios para realizer o trabalho *n* vezes mais rápidos
  - Os estágios trabalham em paralelo entre si
    - O estágio em si também pode ser paralelizado (múltiplas instâncias do mesmo estágio), desde que a tarefa permita
  - Todos os estágios precisam ser atrasados de acordo com o estágio mais lento

Etapas do *Pipeline*



[Akenine-Möller *et al.* 2018]

*cada um desses estágios pode ser um *pipeline* e/ou ter etapas paralelizadas

Etapas do *Pipeline*

1. Estágio de Aplicação
   - Execução na CPU, implementado em software
   - Controlado pela aplicação
     - Detecção de colisão, simulação física, animação, IA...

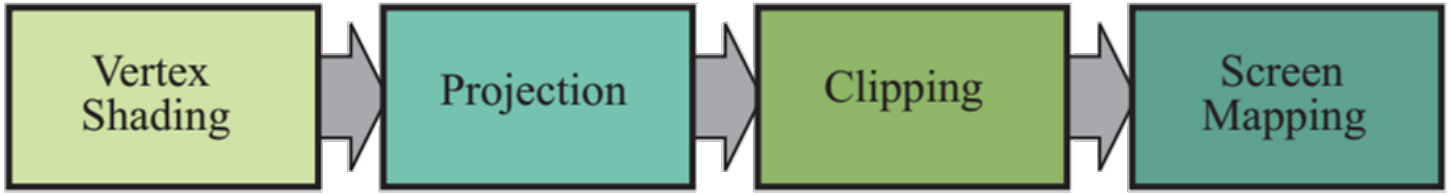
Usualmente em CPU, mas atualmente já pode usar a GPU de modo geral usando o modo compute shader

Podem ser realizados em GPU

2. Estágio de Geometria
   - Transformações e projeções
   - Controla o que, como e onde vai ser desenhado
3. Estágio de Rasterização
   - Recebe a informação de 3 vértices (triângulo) e encontra todos os pixels que podem ser encontrados dentro desse triângulo
4. Processamento de Pixel
   - Executa um programa por pixel para determinar a sua cor e pode testar se ele é vivível ou não, assim como executar o *blending* de cores

Etapas do *Pipeline*

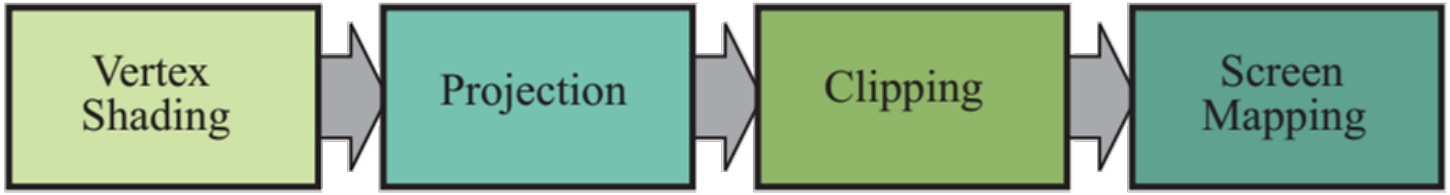
- Estágio de Geometria



  - Processamento dos Vértices (posição, normal, cor, coordenadas de texturas)
    - Executados pelo Vertex Shader
    - Mapeamento de coordenadas: matriz de modelo, matriz(es) de transformação

Etapas do *Pipeline*

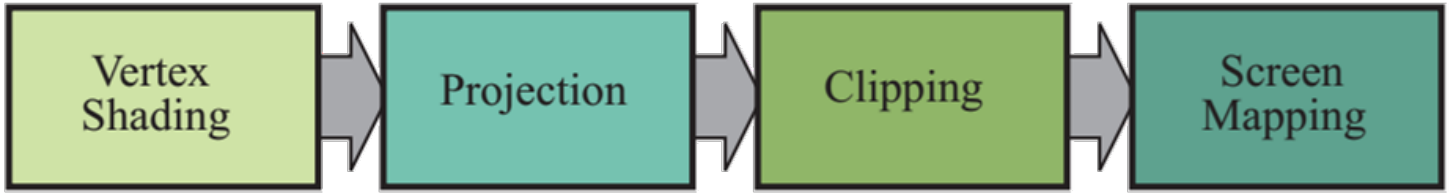
- Estágio de Geometria



  - Processamento dos Vértices (posição, normal, cor, coordenadas de texturas)
    - Posição e orientação da camera sintética: espaço de *view*
    - Projeção da camera (ortogonal vs. perspectiva) forma o volume de visualização (view frustum)

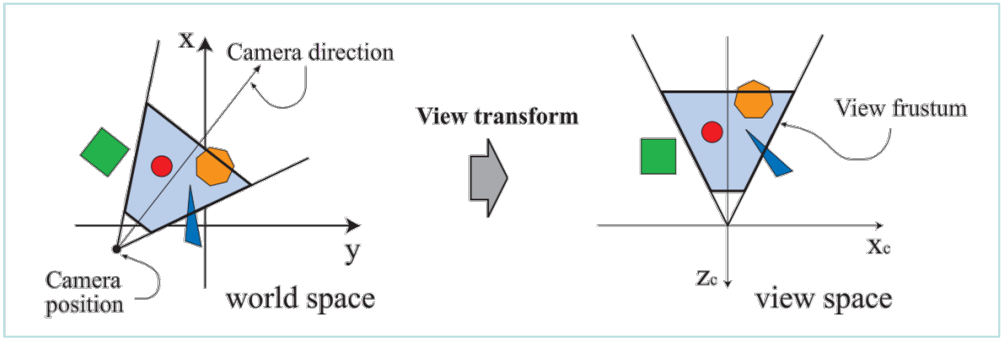
Etapas do *Pipeline*

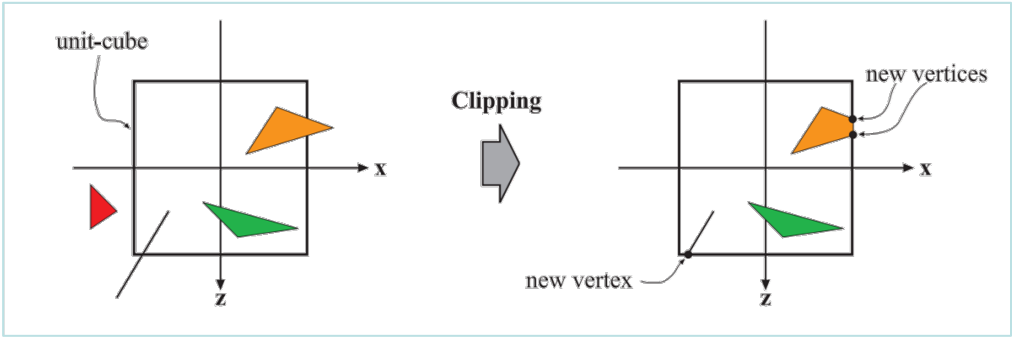
- Estágio de Geometria

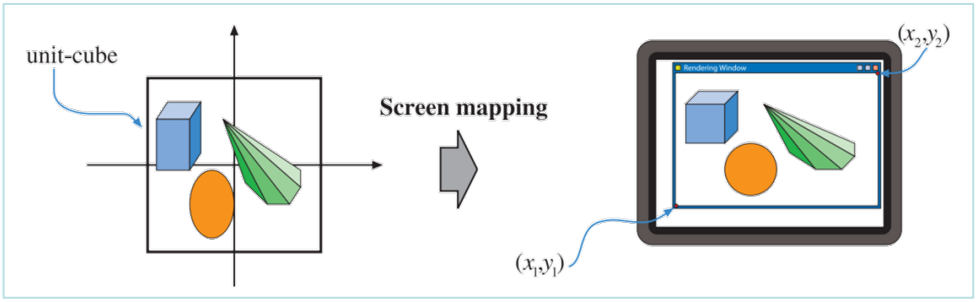


  - Processamento dos Vértices (posição, normal, cor, coordenadas de texturas)
    - Apenas os objetos dentro do frustum serão processados em um espaço normalizado (clipping)
    - Cálculo do efeito da luz sobre o material: shading

Etapas do *Pipeline*



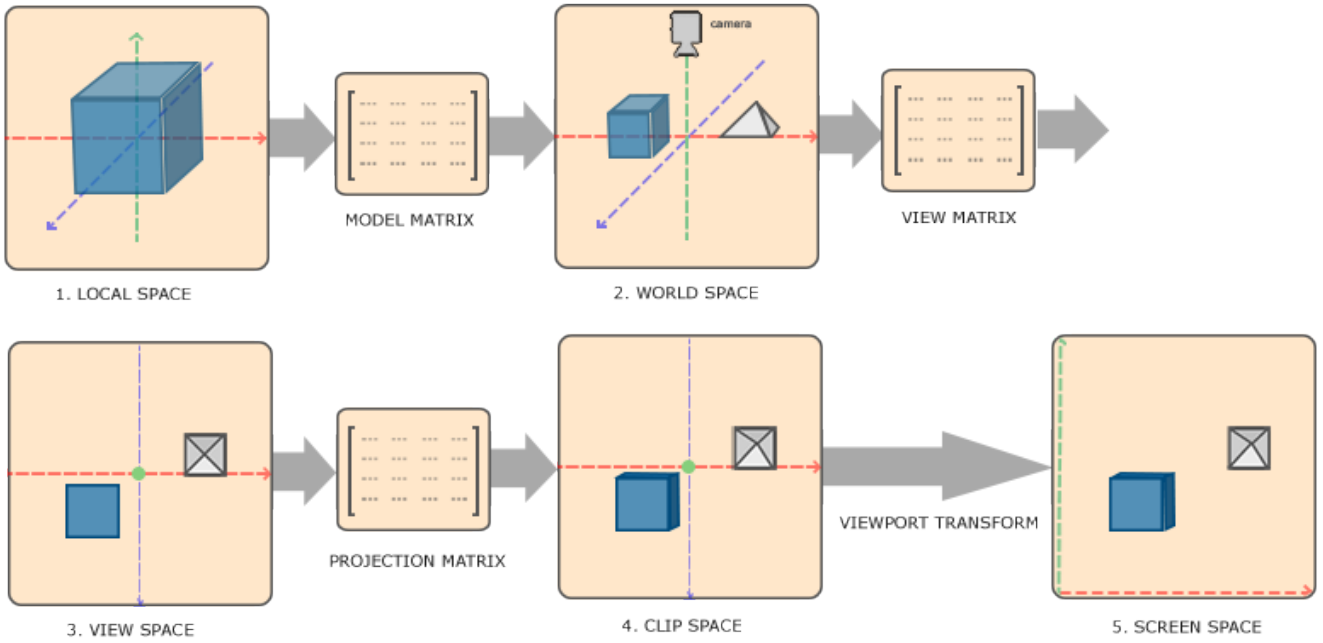




[Akenine-Möller *et al.* 2018]

Etapas do *Pipeline*

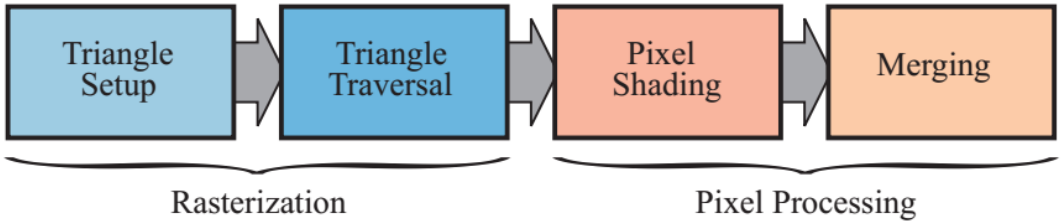
- Estágio de Geometria: sistemas de coordenadas



[LearnOpenGL 2019]

Etapas do *Pipeline*

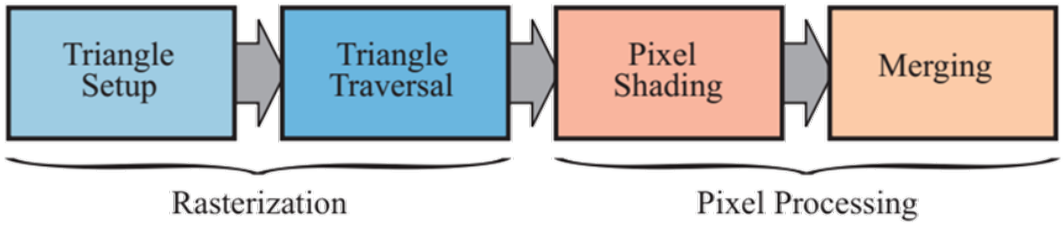
- Rasterização e Processamento de Pixel



  - Triangle Setup: cálculos dos dados do triângulo
  - Triangle Transversal: cada pixel que tem seu centro “coberto” pelo triângulo gera um *fragmento*

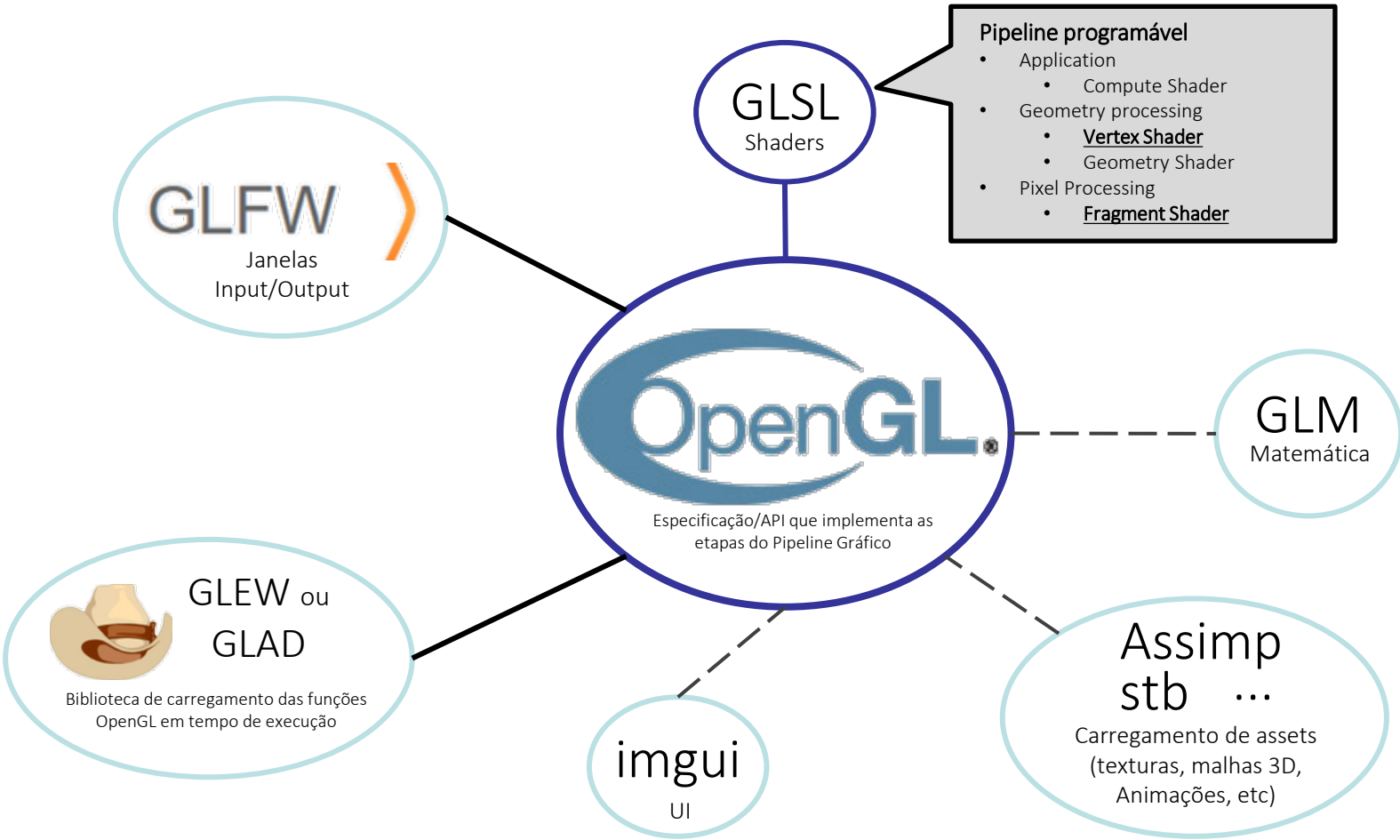
Etapas do *Pipeline*

- Rasterização e Processamento de Pixel



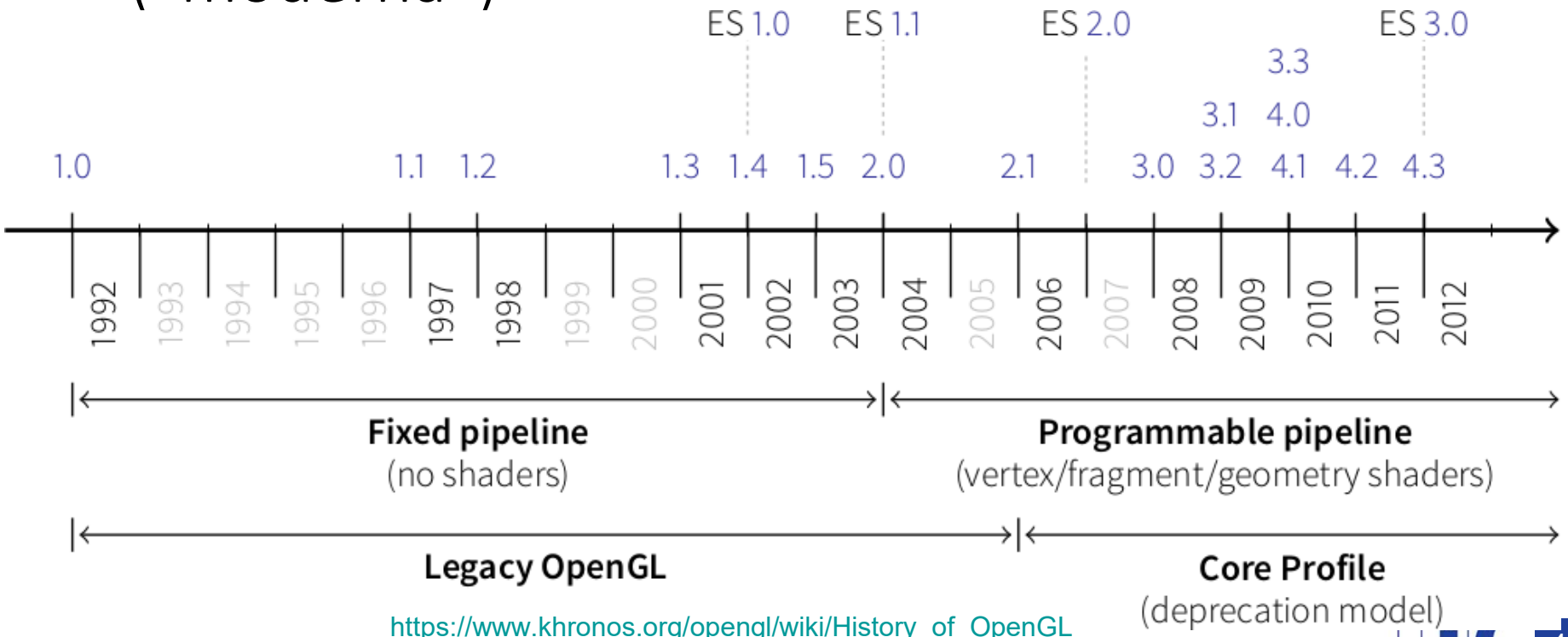
  - Pixel Shading: cálculos realizados no *fragment shader* para encontrar a cor do pixel
  - Merging: combina a cor do fragmento produzido com a cor que está no *color buffer* atual e pode também resolver problemas de visibilidade (*z-buffer*) – processa e combina as informações calculadas e as contidas no *framebuffer*

Nossa(s) ferramenta(s)



Versões

- OpenGL 2.0 (“antiga”) vs OpenGL 3.3+ (“moderna”)



https://www.khronos.org/opengl/wiki/History_of_OpenGL

Um pouco sobre a arquitetura

- Pipeline fixo vs. pipeline programável