Introdução à Computação Gráfica

por Rossana B Queiroz

Processamento Gráfico

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

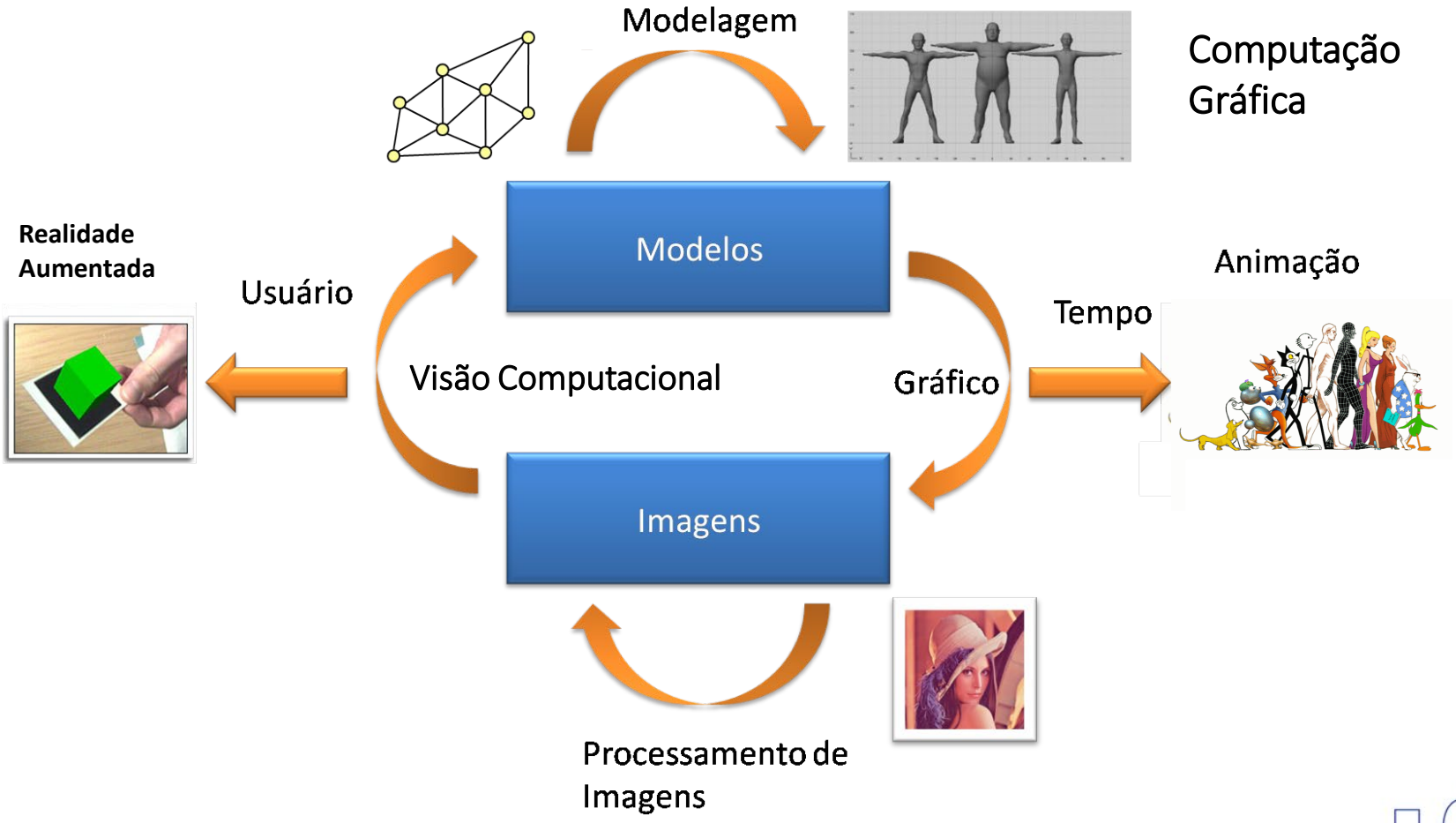
Processamento Gráfico

- Exemplos
  - Geração de filmes através de computador
  - Correção de olhos vermelhos em fotos
  - Aplicação de filtros de imagens
  - Geração de desenhos animados no computador
  - Detecção e reconhecimento de objetos em imagens
  - Geração de cenas em jogos 2D e 3D
  - Modelagem de sólidos baseado em física
  - Modelos 3D para prototipação de matrizes.
  - Simulação de ambientes nocivos
  - Visualização de gráficos sobre 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

Processamento Gráfico



Processamento de imagens

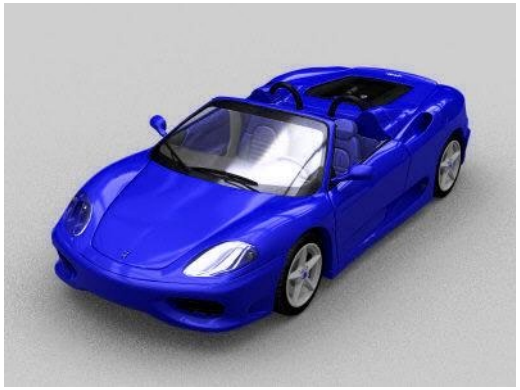
- Processamento de imagens visa a obtenção de informações da imagem para produção de dados a respeito da mesma ou modificação da imagem
- Tratamento de imagens:
  - Trata da geração de novas imagens a partir de imagens de entrada. A rigor não extrai informações da imagem.
  - Exemplos de aplicativos 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

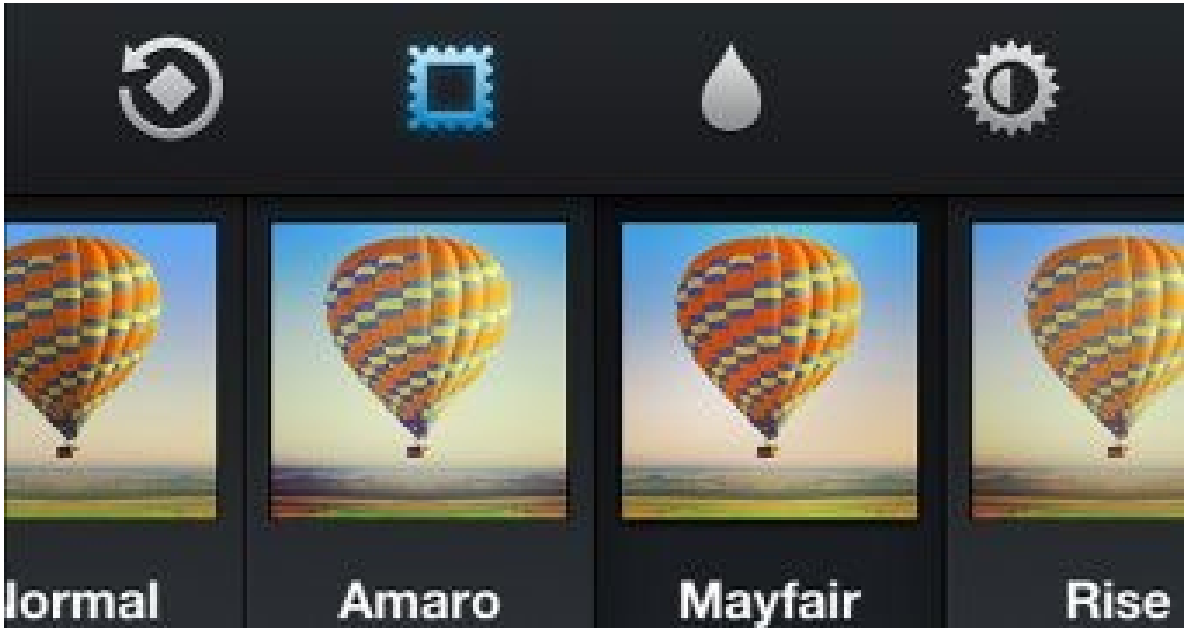


Filtro de imagem aplica efeitos de foto envelhecida

Processamento de Imagens

- Tratamento de imagens:

  Para correção, melhoria ou destaque de informações relevantes da imagem

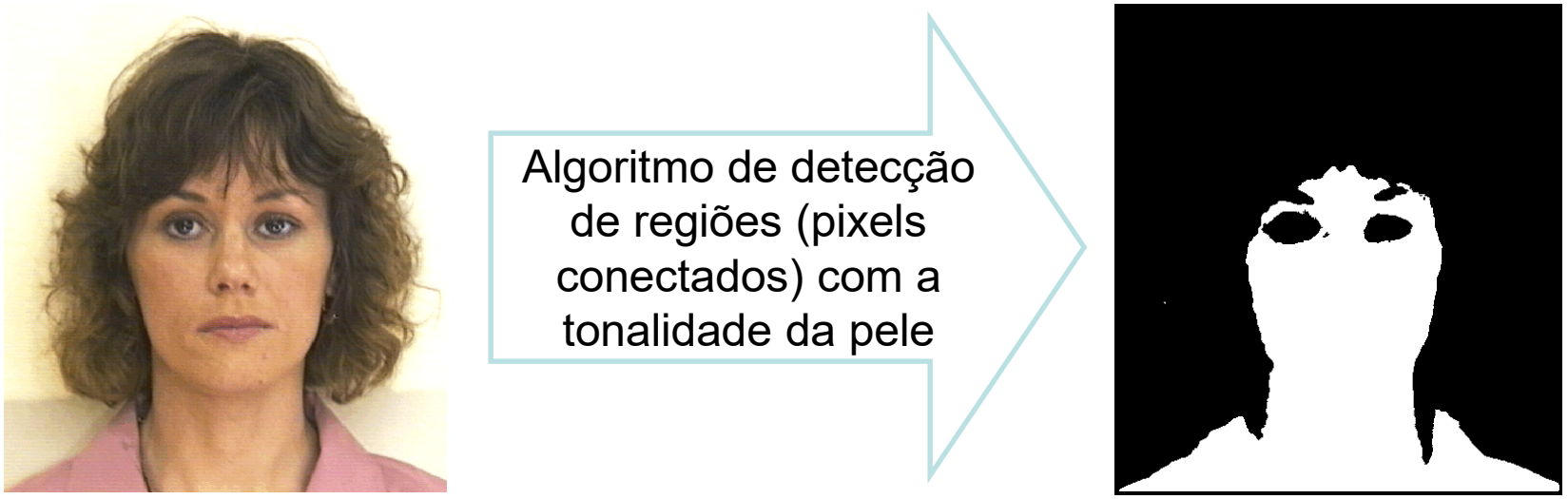


Processamento de Imagens

- 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



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

Processamento de Imagens

- Visão Computacional

Resultado: INS1457 - Fulano de Tal - AUTORIZADO

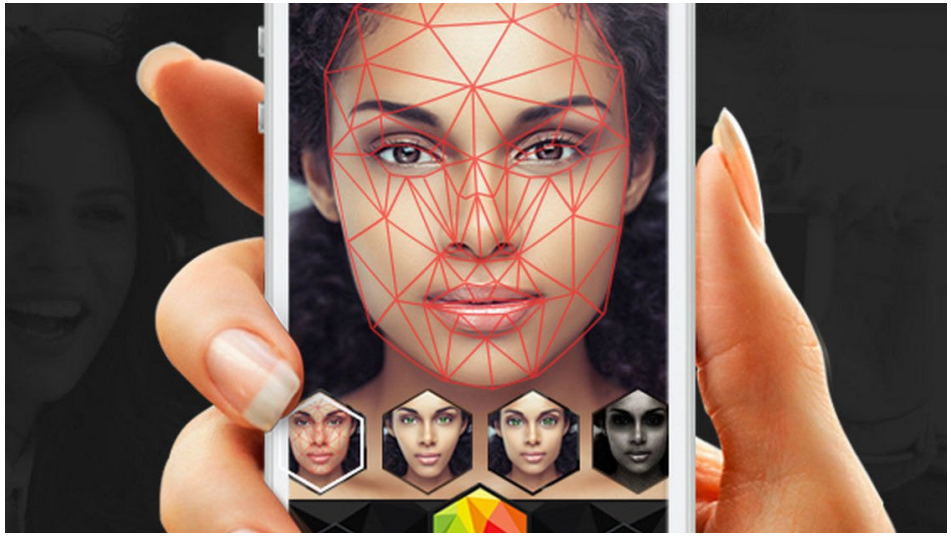
Sistema de reconhecimento de placas e autenticação de veículos para condomínios: Câmera fotografa o veículo e sistema faz a detecção e o reconhecimento dos números da placa.

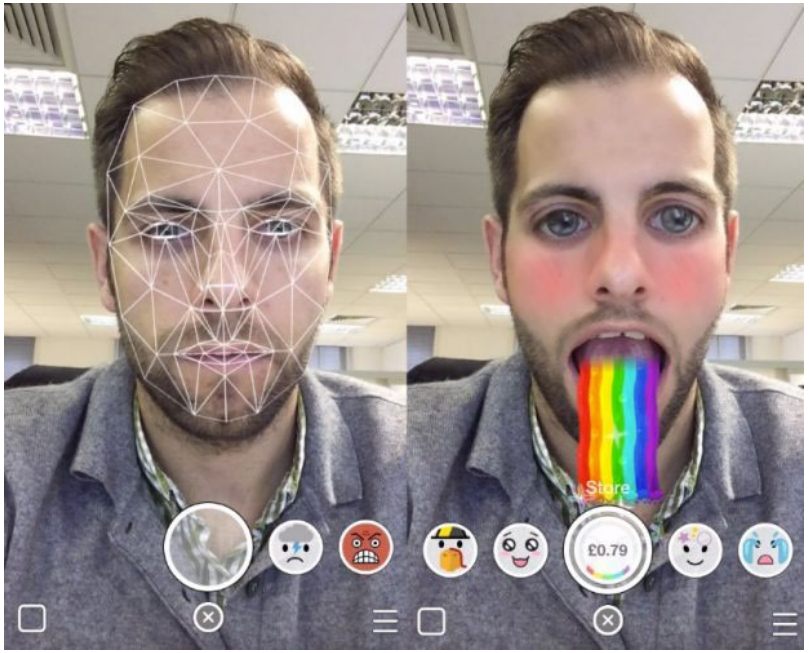
Processamento de Imagens

- Realidade Aumentada (AR)
  - Detecção de marcações ou padrões na imagem
  - Inserção de elementos de Computação Gráfica

Processamento de Imagens

- Realidade Aumentada (AR)





Computação Gráfica

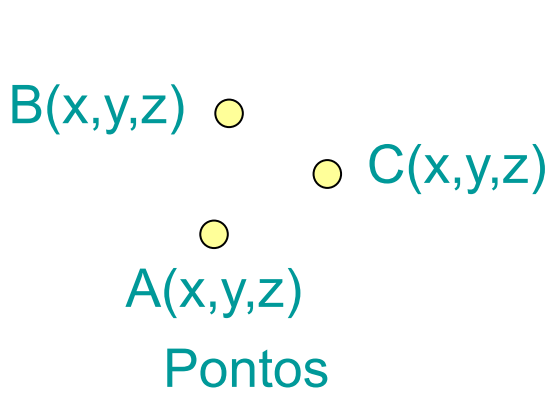
- É um processo de sintetizar imagens a partir de um conjunto de dados.
  - Transformação de dados em imagem. Dados são a parte do modelo, são a descrição da cena ou imagem a ser sintetizada.
- O processo de síntese de dados em imagem requer um alto custo computacional.

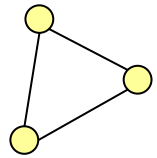
Computação Gráfica

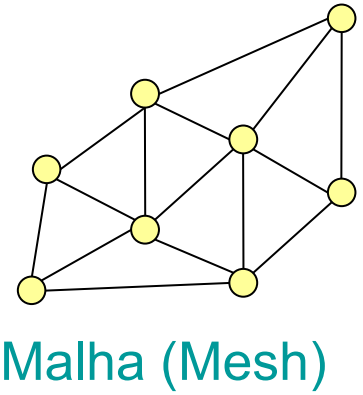
- A CG está intimamente ligada a idéia de se obter o “melhor resultado” com o menor custo computacional possível. Este paradoxo fomenta as pesquisas na área.
  - Muitos algoritmos são criados com este intuito: Algoritmos de compressão de imagens, redução de malha poligonal, representação de objetos em mapas ou cenários de jogos, mapeamento de texturas, ...
- Computação Gráfica não é só 3D, é 2D também

Computação Gráfica

- Modelagem em CG:
  - As **primitivas** básicas em CG são:







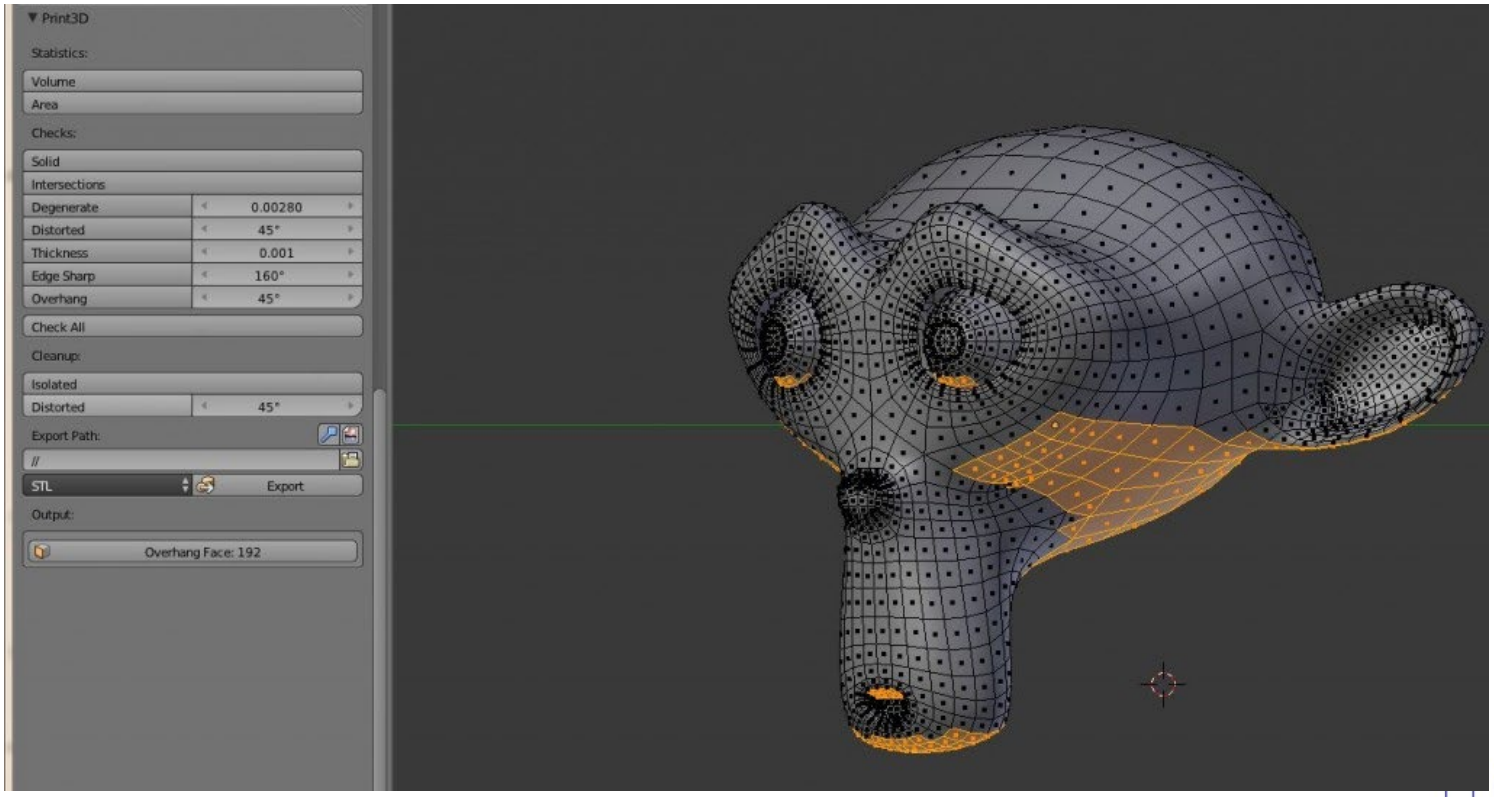
  - Pontos são conectados para formarem polígonos, quem são conectados para formar algum objeto...

Computação Gráfica

- Modelagem em CG:

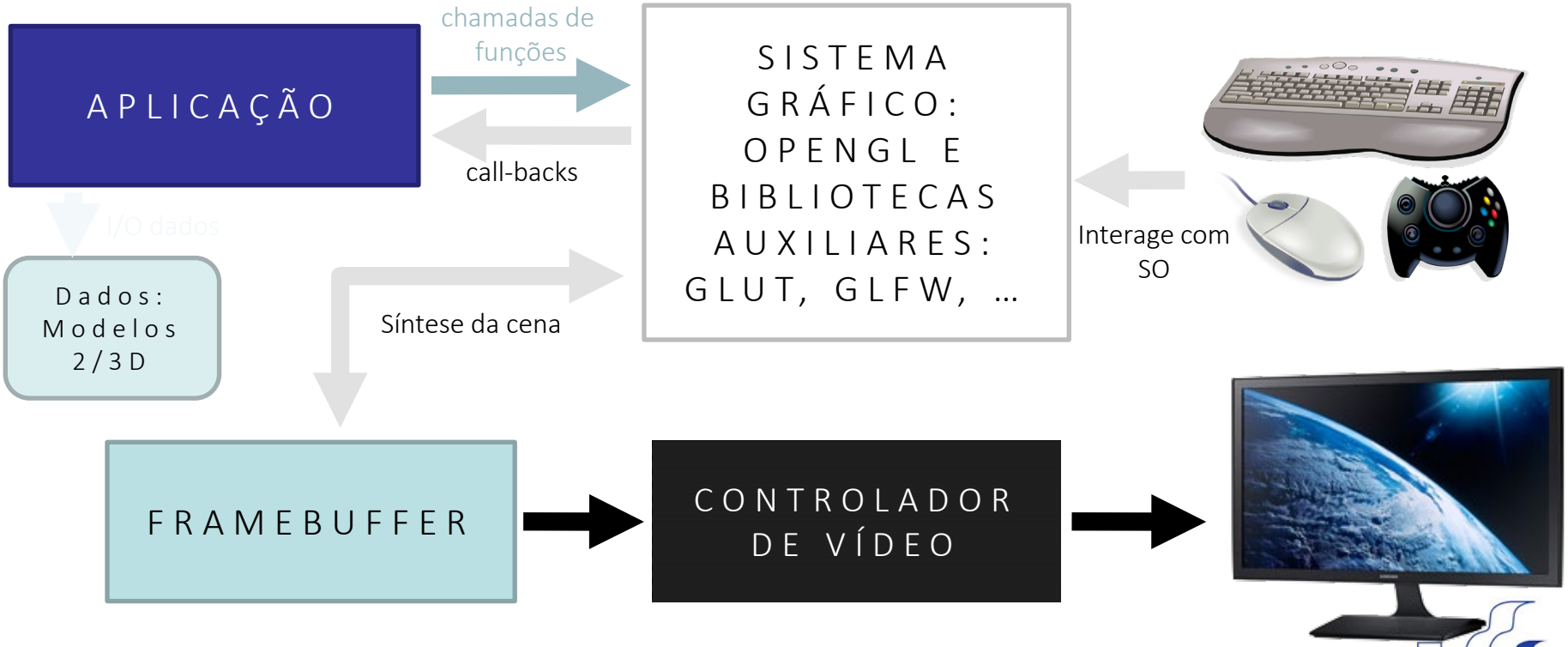
Computação Gráfica

- Modelagem em CG:



Computação Gráfica

Esquema conceitual de aplicações de CG:



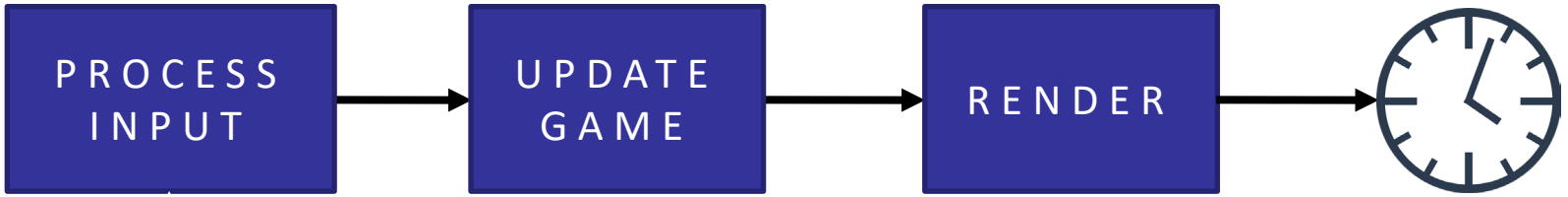
Computação Gráfica

- 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)

Processamento Gráfico

- *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

Processamento gráfico para Jogos digitais

- *Game Loop*

```
while(true) {
  processInput();
  update(); // estado + lógica
  render();
  sync();   // controle tempo!
}
```

DURANTE O SEMESTRE VOLTAREMOS AO *GAME LOOP*!

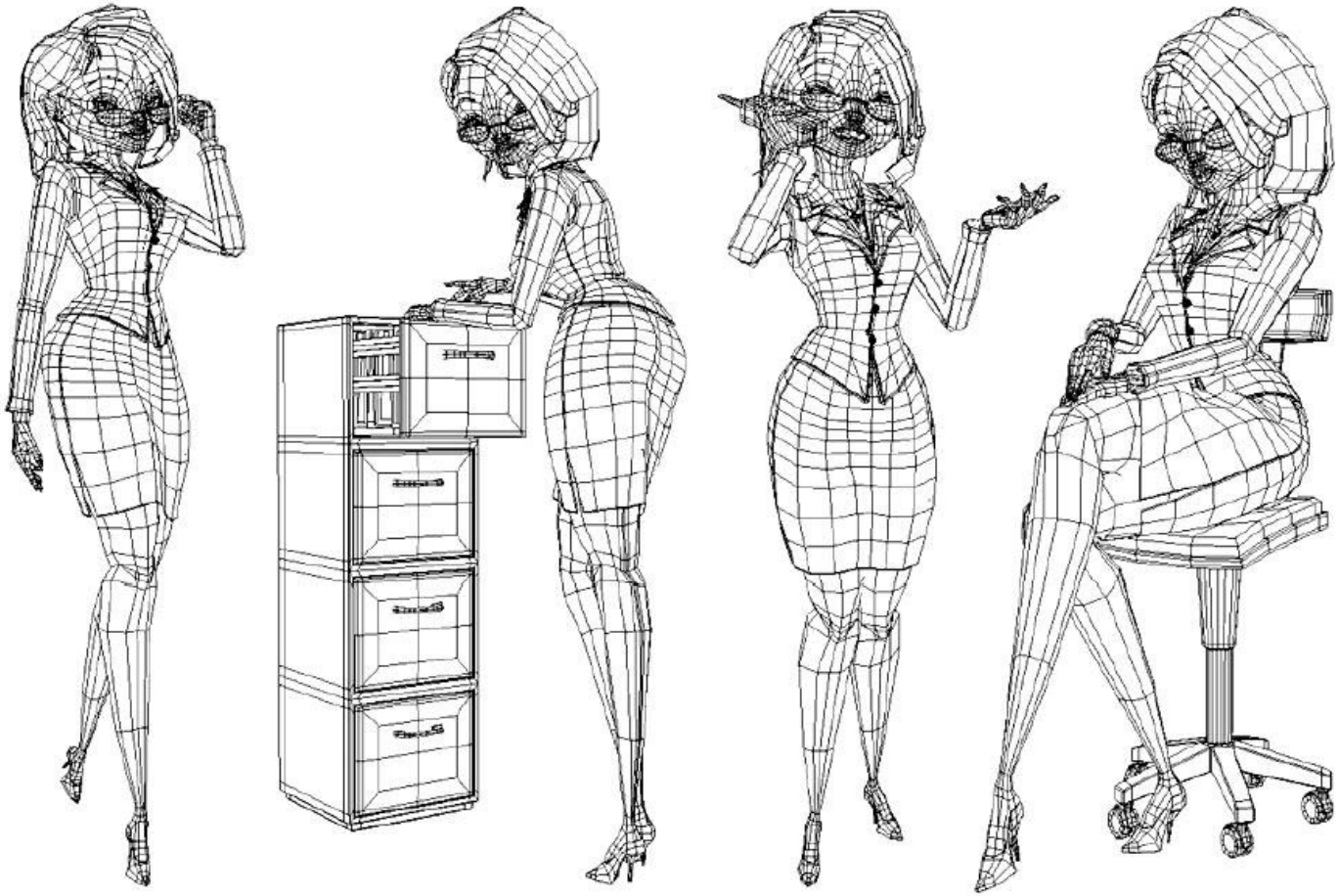
Renderização

- 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 a técnica pela qual obtemos uma imagem gerada a partir de um modelo
- Este modelo é uma descrição de uma cena e pode conter informações sobre geometria, cores, propriedades e texturas de objetos, iluminação e sombreamento
  - Tipos de modelos conhecidos?

Rendering – Exemplo

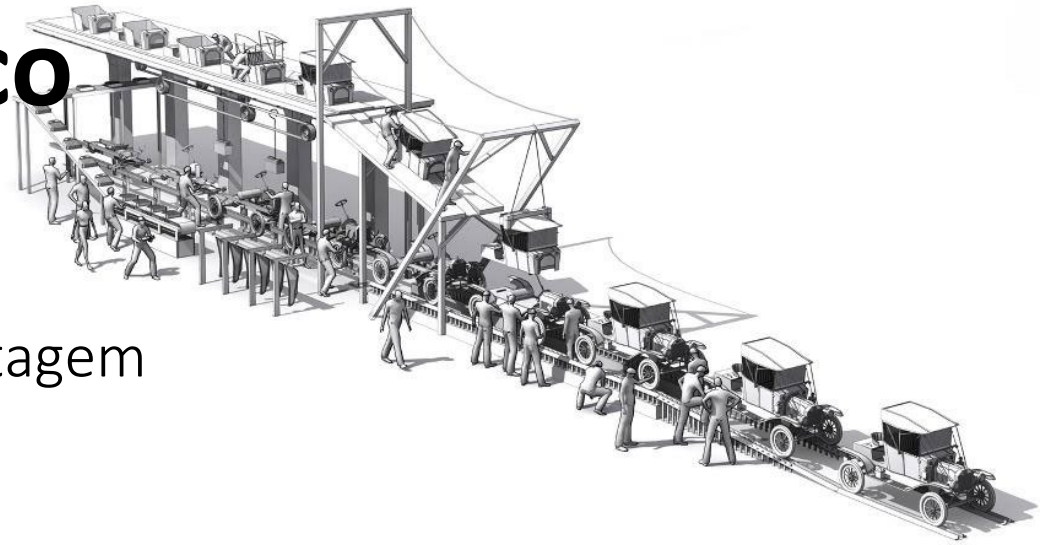


Rendering – Exemplo

Rendering – Exemplo

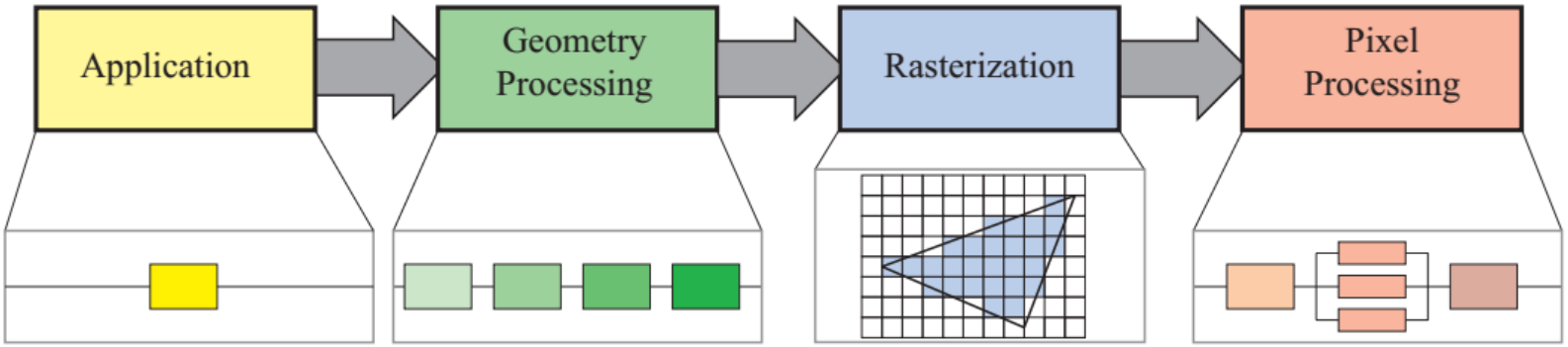
Rendering - Iluminação

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...

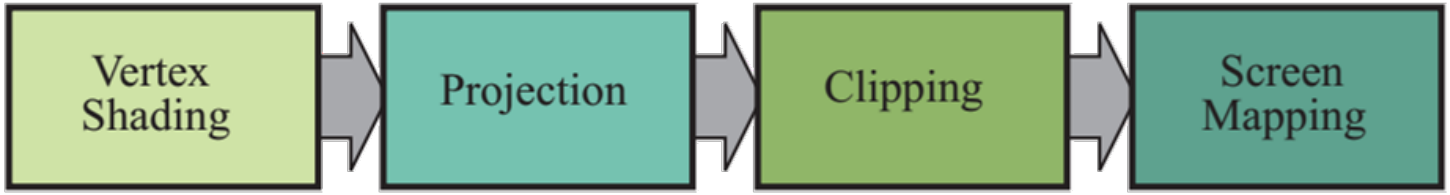
Usualmente em CPU, mas atualmente já pode usar a GPU de modo geral usando o modo compute shader

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*

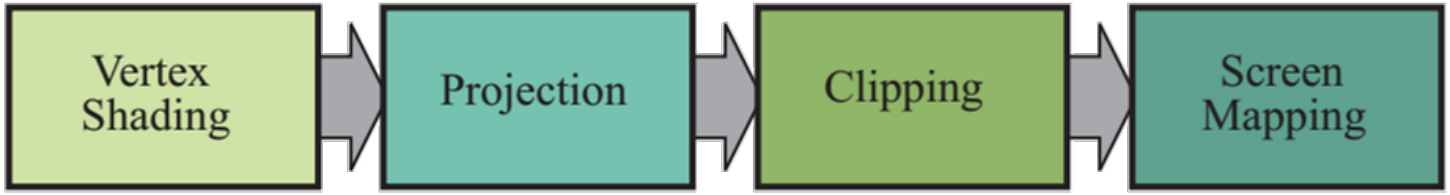
- Estágio de Geometria



  - 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*

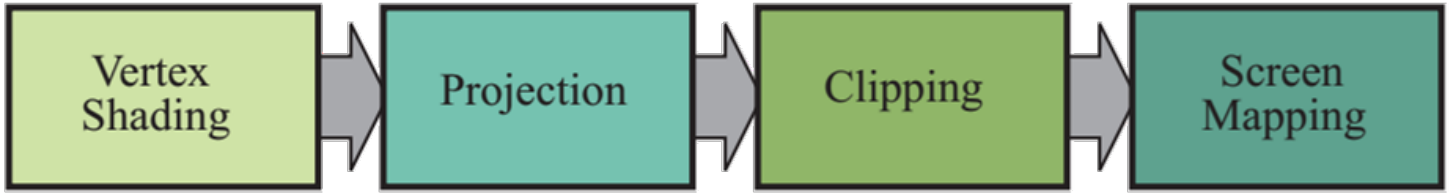
- Estágio de Geometria



  - 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)

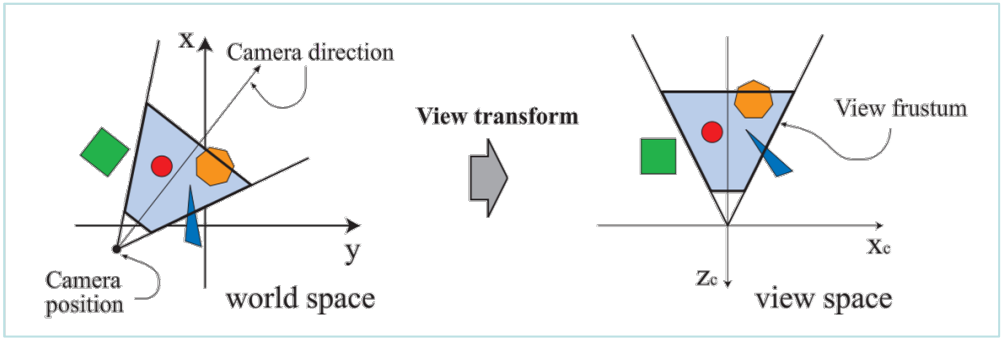
Etapas do *Pipeline*

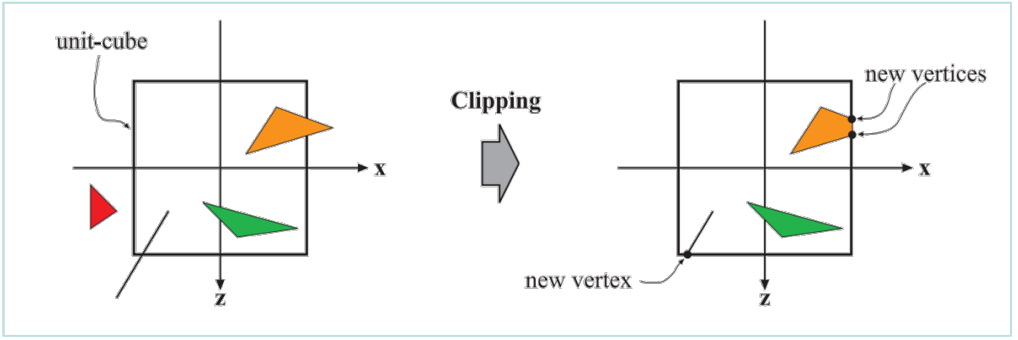
- Estágio de Geometria

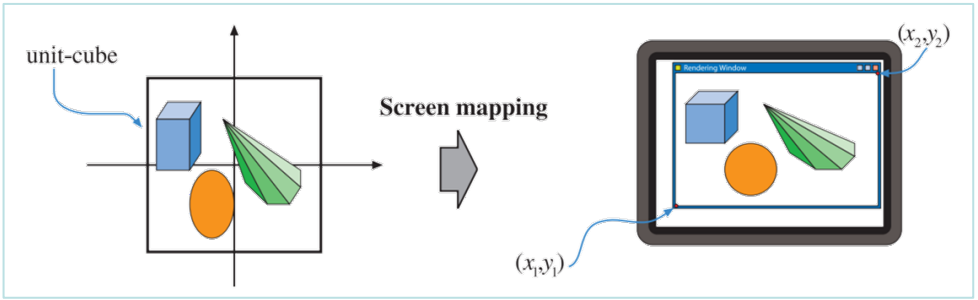


  - Processamento dos Vértices (posição, normal, cor, coordenadas de texturas)
    - Apenas os objetos dentro do frustum serão processados em um espaço normalizado (clipping)
    - Cálculo do efeito da luz sobre o material: shading

Etapas do *Pipeline*



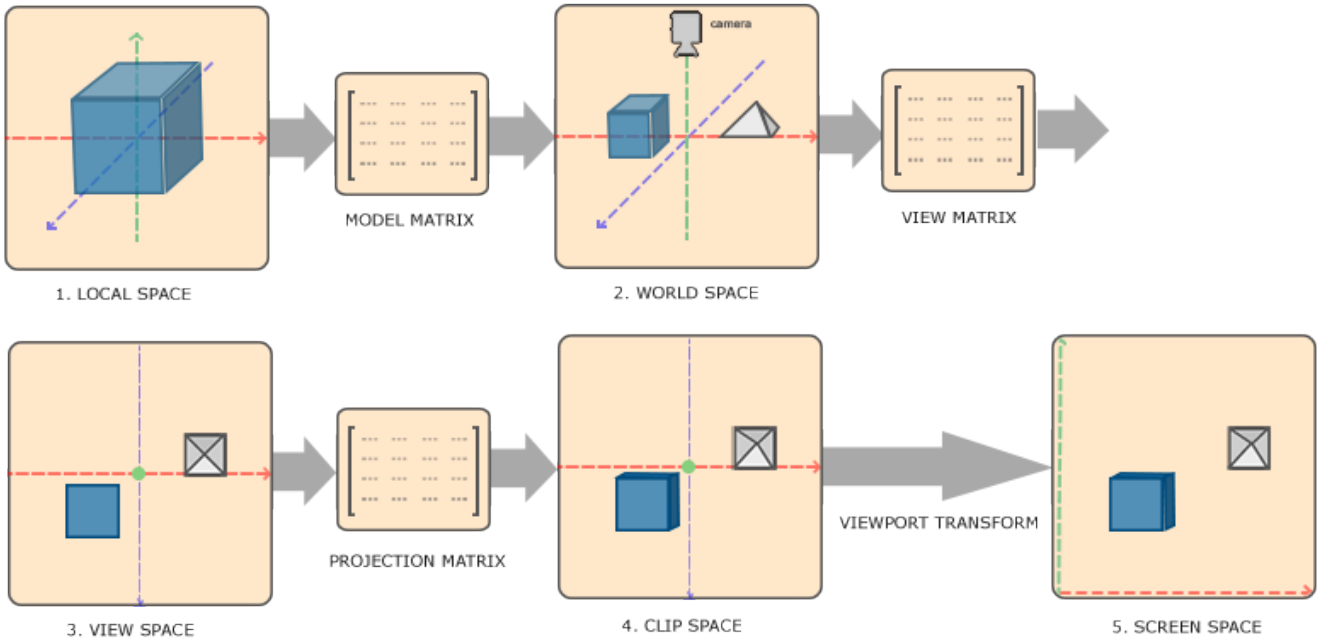




[Akenine-Möller *et al.* 2018]

Etapas do *Pipeline*

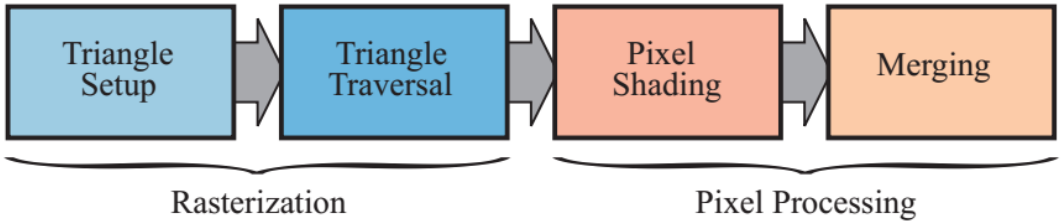
- Estágio de Geometria: sistemas de coordenadas



[LearnOpenGL 2019]

Etapas do *Pipeline*

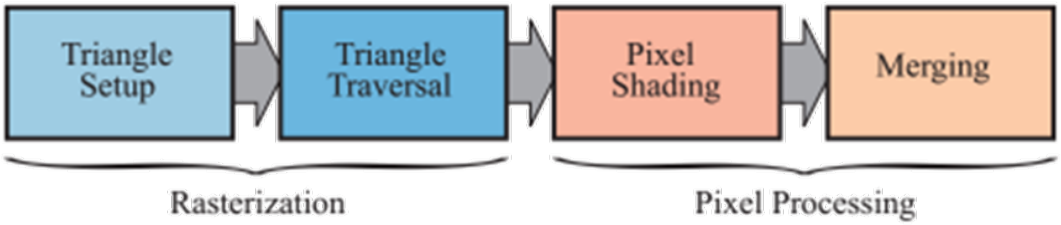
- Rasterização e Processamento de Pixel



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

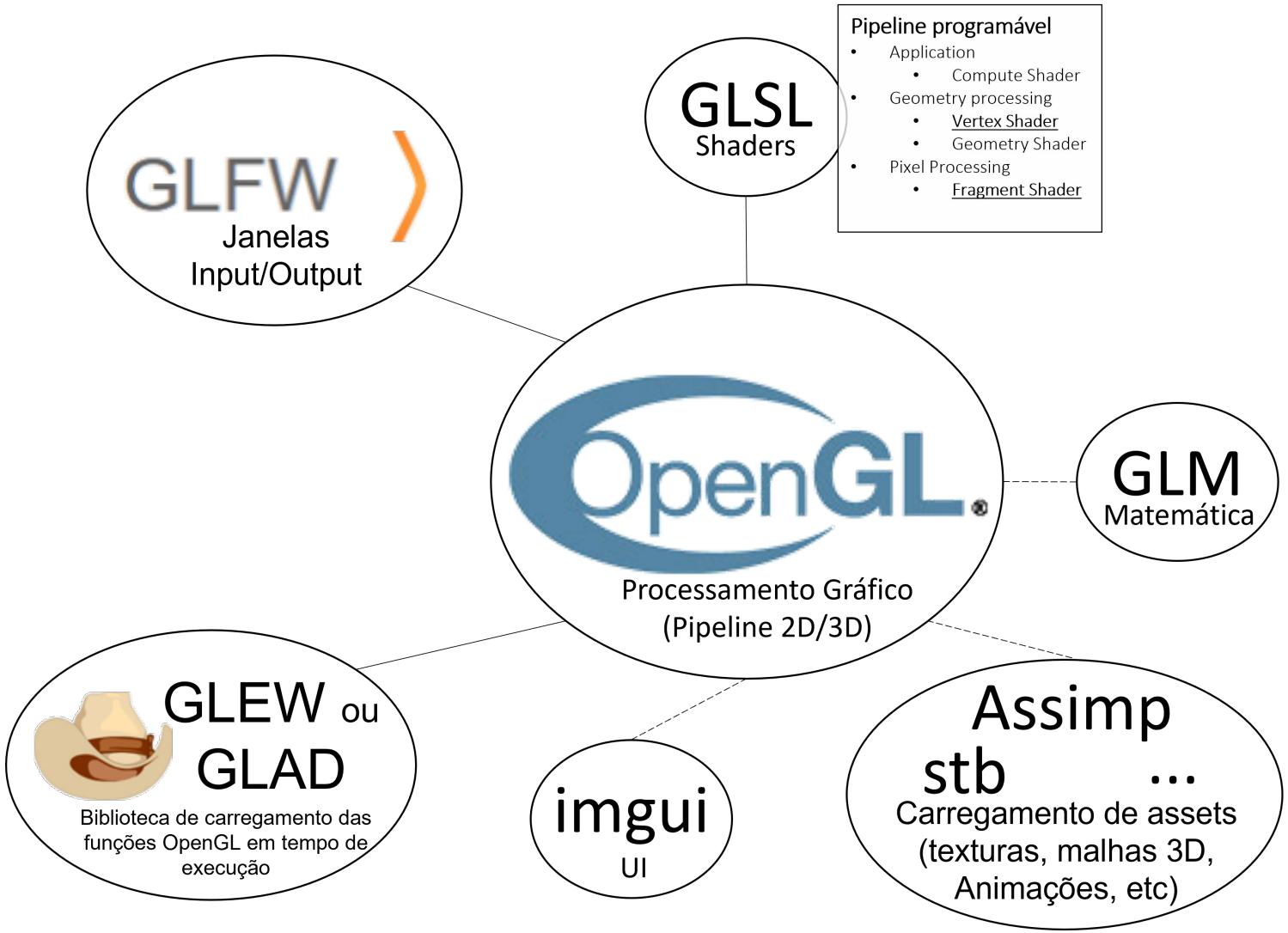
Etapas do *Pipeline*

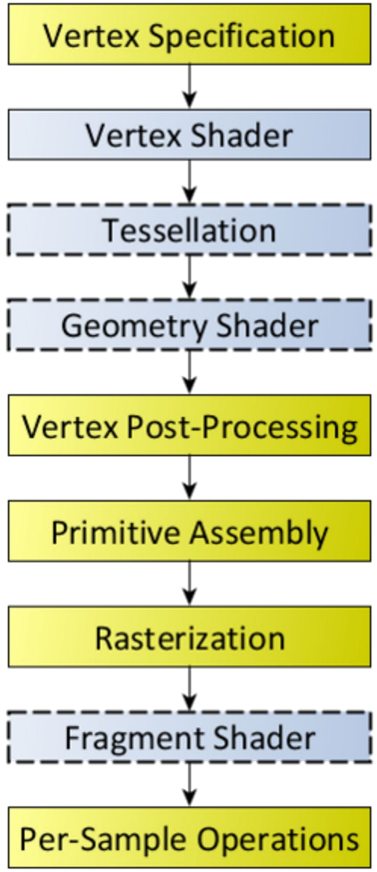
- Rasterização e Processamento de Pixel



  - 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)

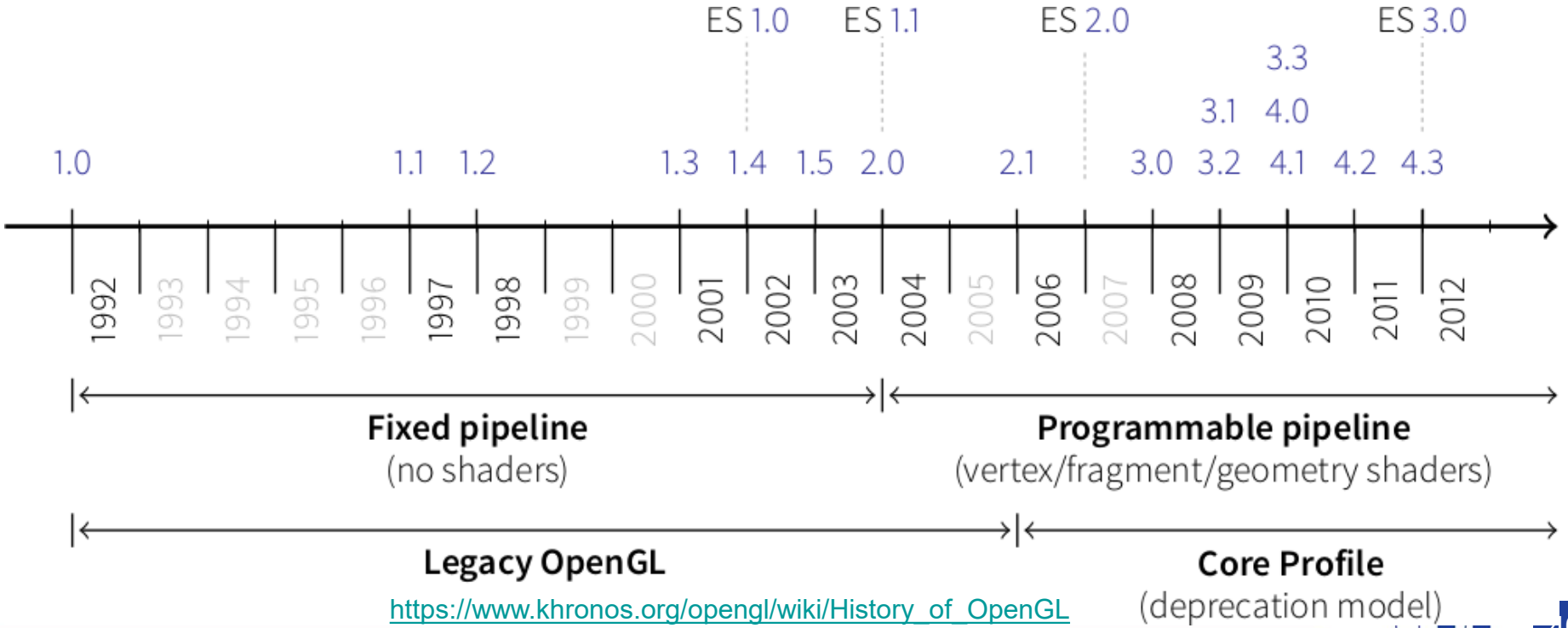




Fonte: Khronos Group

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