Visão geral do pipeline gráfico

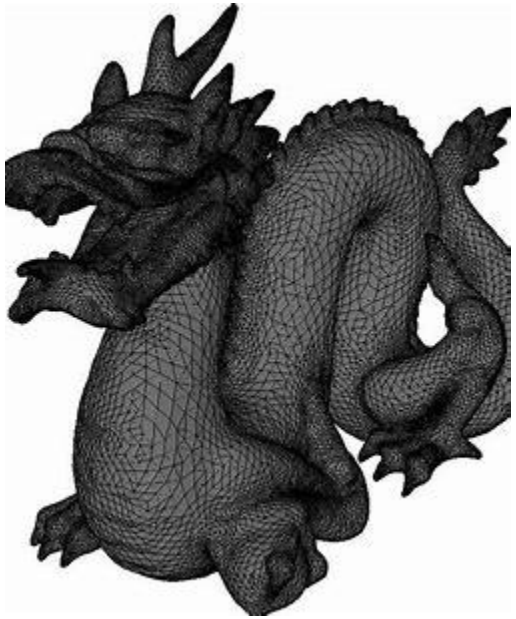
M. Cristina

SCC250 Computação Gráfica

Pipeline gráfico

Dada uma cena 3D (modelo geométrico), renderizar uma imagem 2D (matriz de pixels)

Modelo + propriedades + texturas



Algoritmo de *rendering*

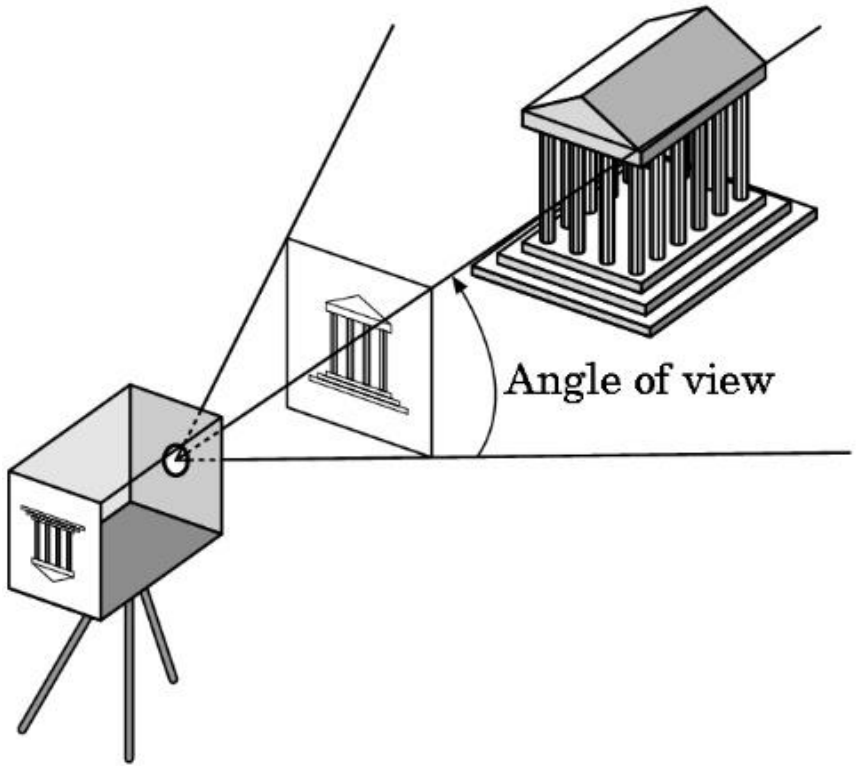
Imagem

Requisitos

- Um sistema de coordenadas usado para descrever a geometria dos objetos que compõem uma cena: sistema de coordenadas do mundo (*World Coordinate System, WCS*)

- Uma câmera que define como a cena está sendo observada: a posição da câmera em relação à cena determina o que vai ser visto: sistema de coordenadas da câmera (*Camera Coordinate System*, CCS) (ou sistema de coordenadas do observador, *View Coordinate System*, VCS)

Requisitos: Cena + Câmera



Requisitos

- O processo de modelagem de um objeto 3D é feito no Sistema de Coordenadas do Objeto (*Object Coordinate System*, OCS)

- Esse processo é, em geral, feito por um artista, que cria o modelo que pode ser inserido em diferentes cenas: descritos como malhas poligonais

- What is a Polygon Mesh? (conceptartempire.com)
- Introduction to 3D Polygon Mesh - 3D Studio

Requisitos

- O processo de modelagem de um objeto 3D é feito no Sistema de Coordenadas do Objeto (*Object Coordinate System*, OCS)

- Esse processo é, em geral, feito por um artista, que cria o modelo que pode ser inserido em diferentes cenas: descritos como malhas poligonais

- Serão necessárias transformações geométricas entre esses diferentes sistemas de coordenadas!

Malhas poligonais

- Malhas de triângulos são a representação padrão: porque?

- Outras representações são possíveis (equações de funções implícitas, superfícies e curvas suaves formadas por Bézier *patches*...)

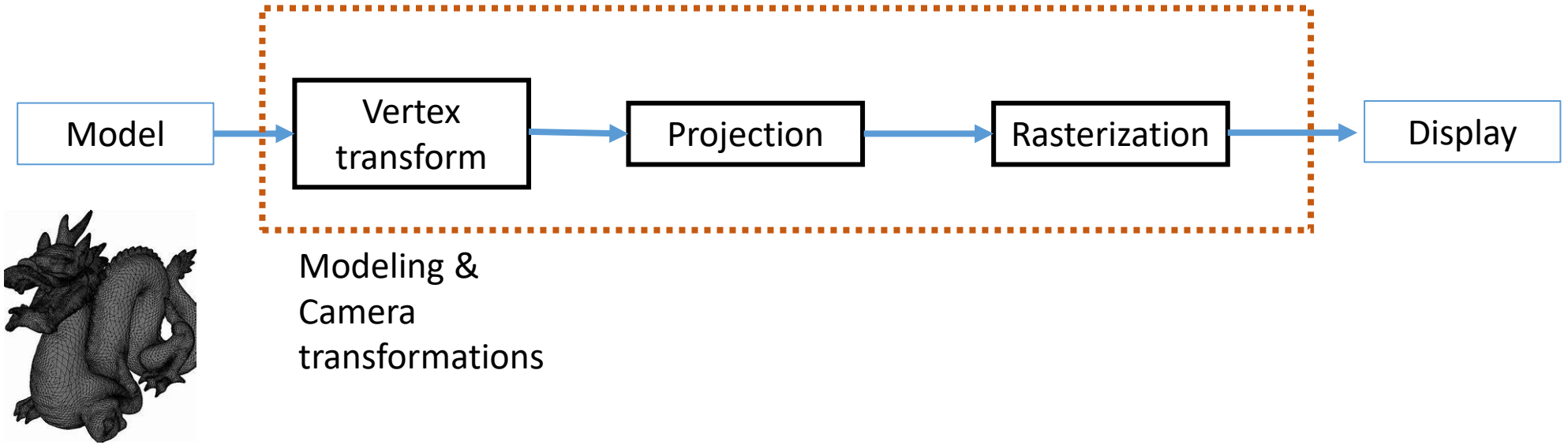
Malhas poligonais

- Malhas de triângulos são a representação padrão
  - Triângulos são primitivas planares, o que simplifica a implementação do hardware gráfico, que pode utilizar essencialmente somadores para interpolação linear
  - Tabela dos vértices + tabela dos polígonos

  - Superfícies curvas podem ser aproximadas arbitrariamente por representações poligonais

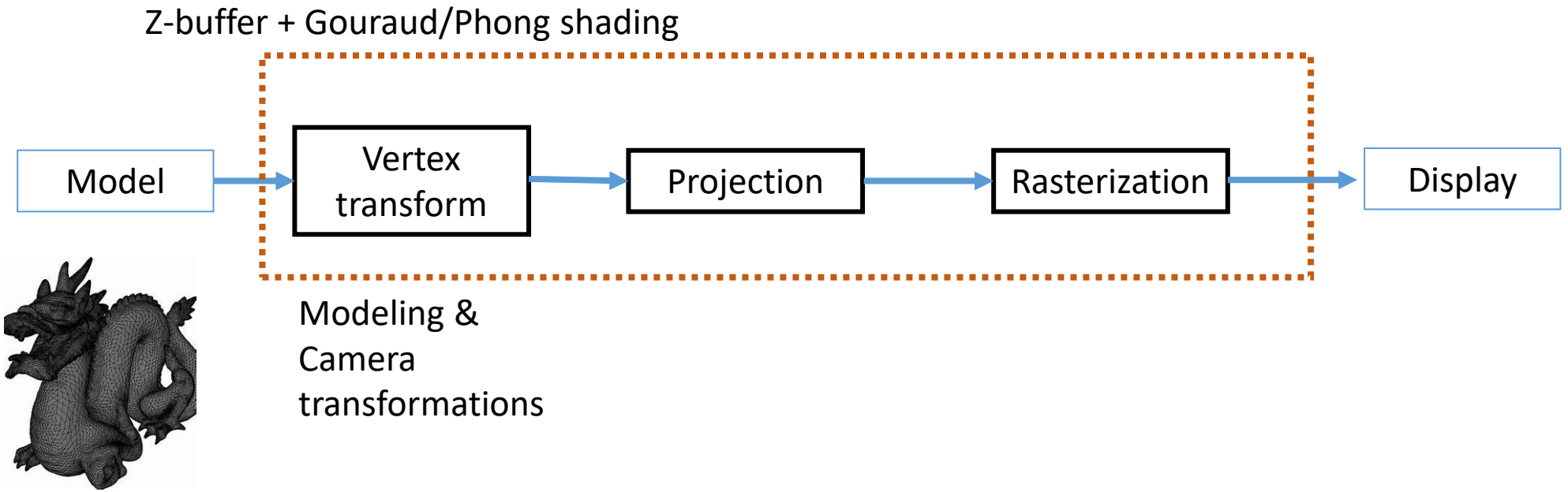
Pipeline de Rendering

- Passos para renderizar uma cena 3D em uma tela 2D



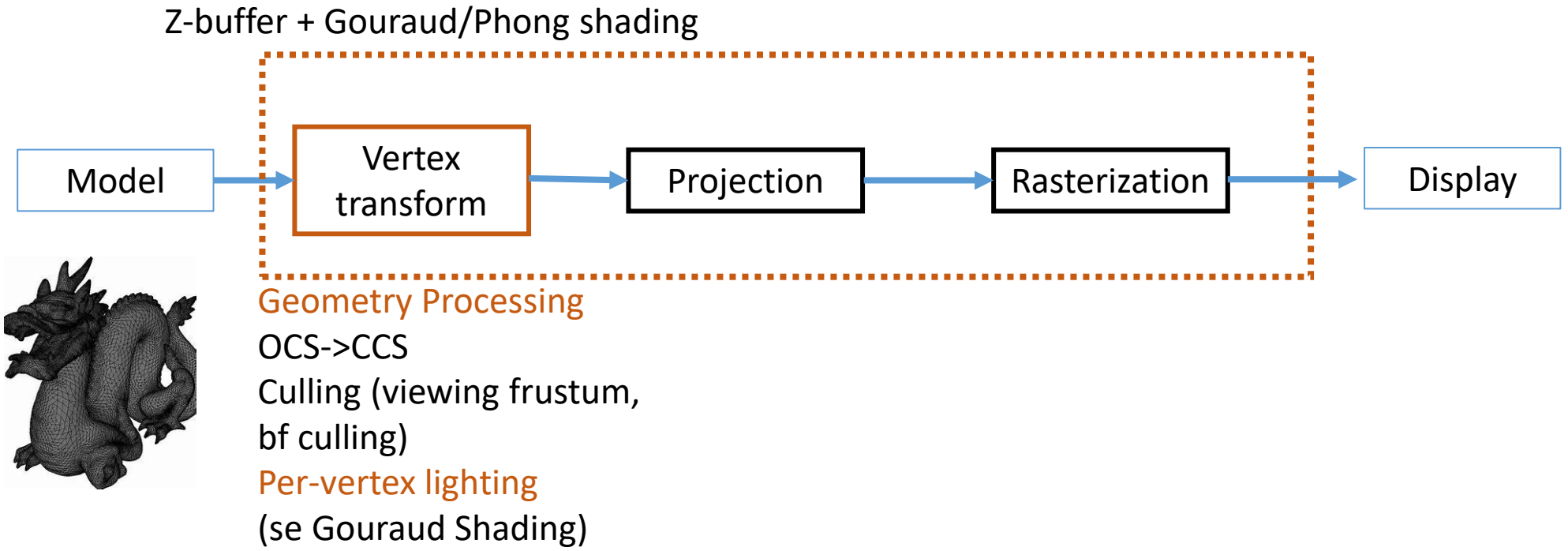
Pipeline de Rendering (rasterização)

- Passos para renderizar uma cena 3D em uma tela 2D



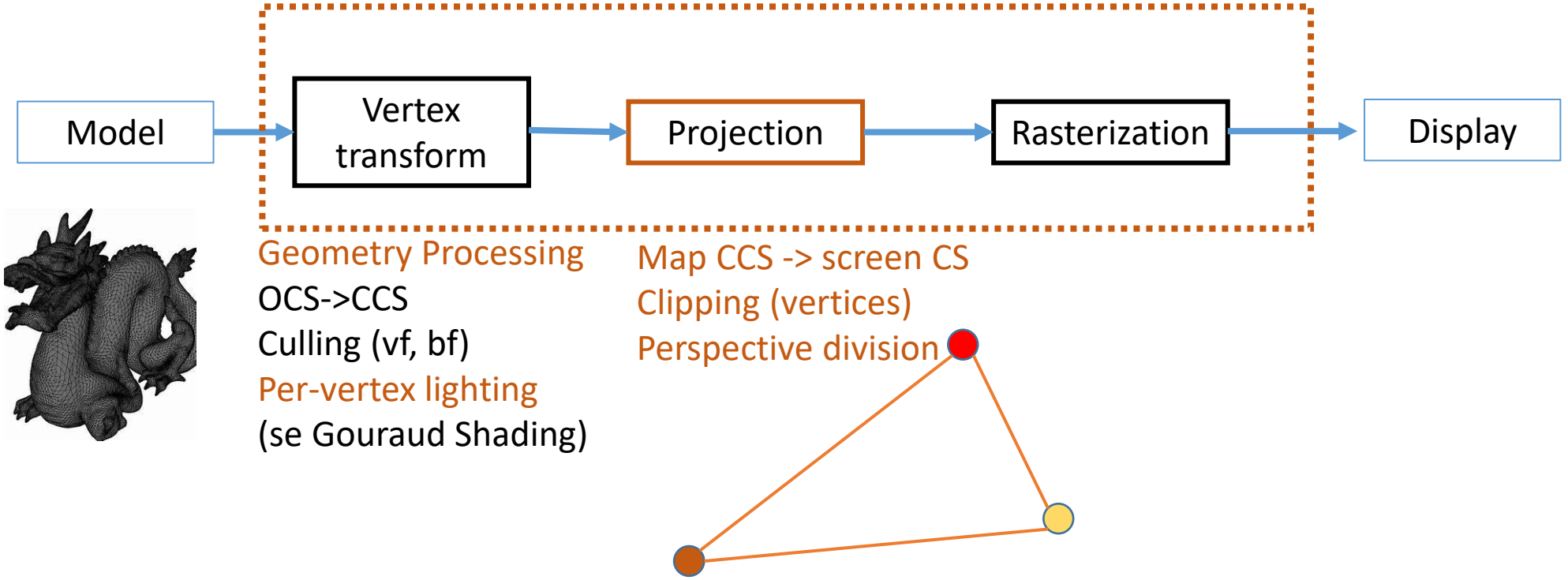
Pipeline de Rendering (rasterização)

- Passos para renderizar uma cena 3D em uma tela 2D



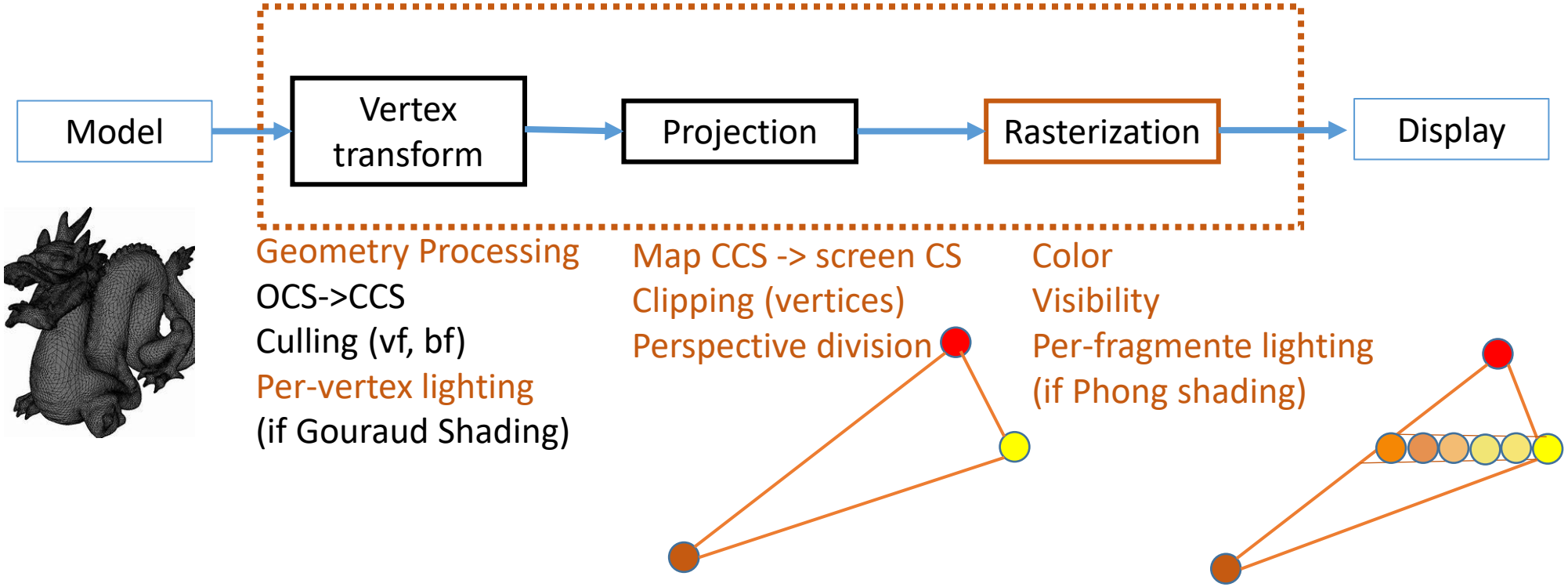
Pipeline de Rendering

- Passos para renderizar uma cena 3D em uma tela 2D



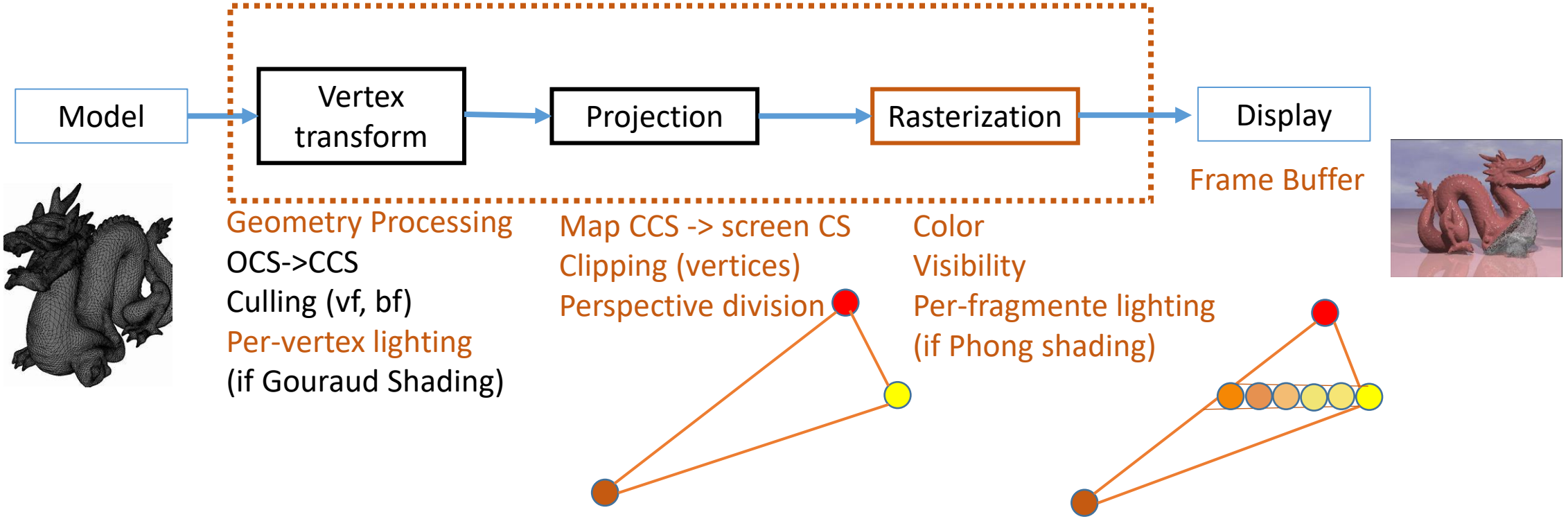
Pipeline de Rendering

- Passos para renderizar uma cena 3D em uma tela 2D



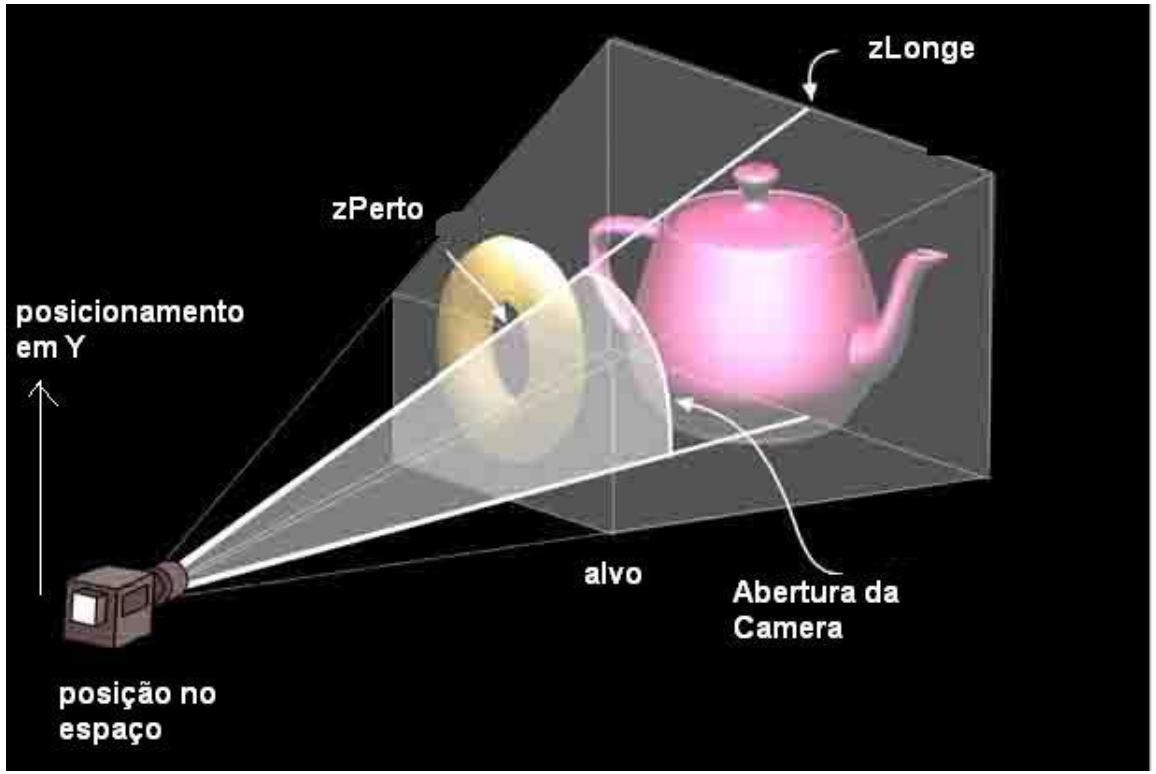
Pipeline de Rendering

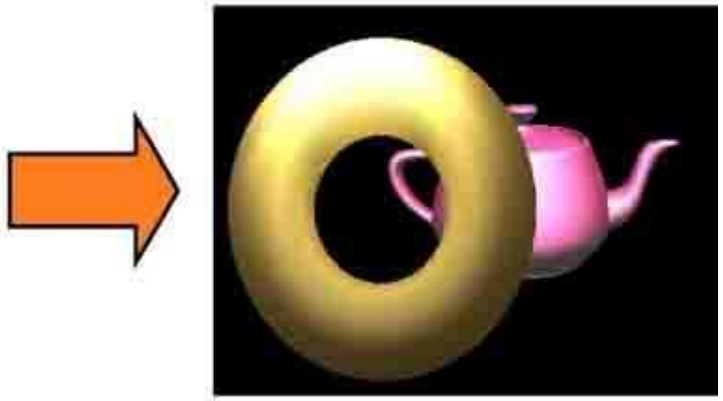
- Passos para renderizar uma cena 3D em uma tela 2D



Rendering é executado do ponto de vista da câmera

- Os vértices dados pelas coordenadas do CCS são projetados





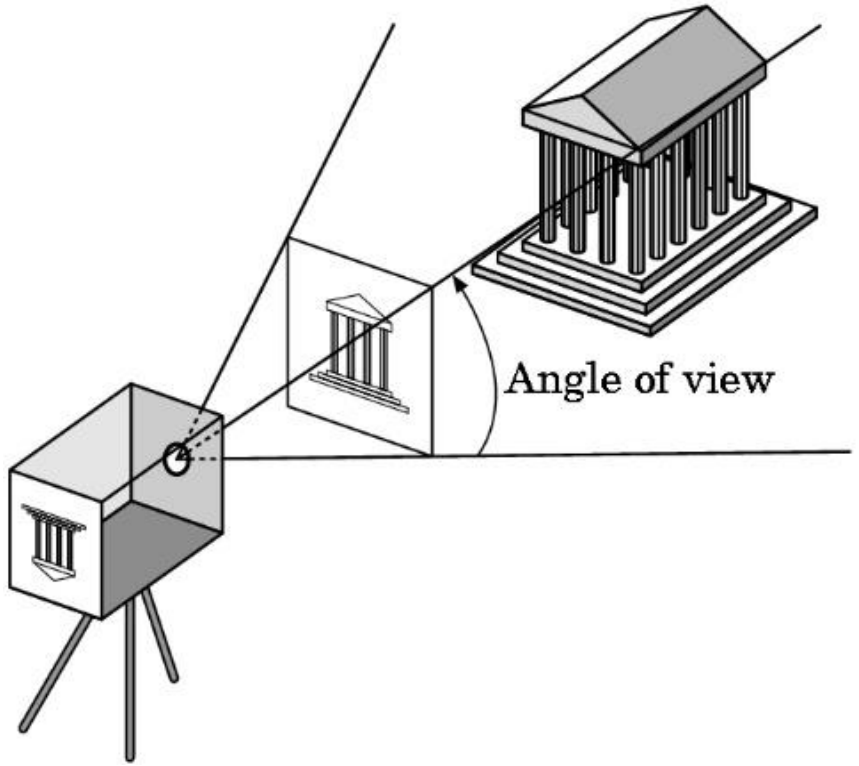
Processamento geométrico: culling

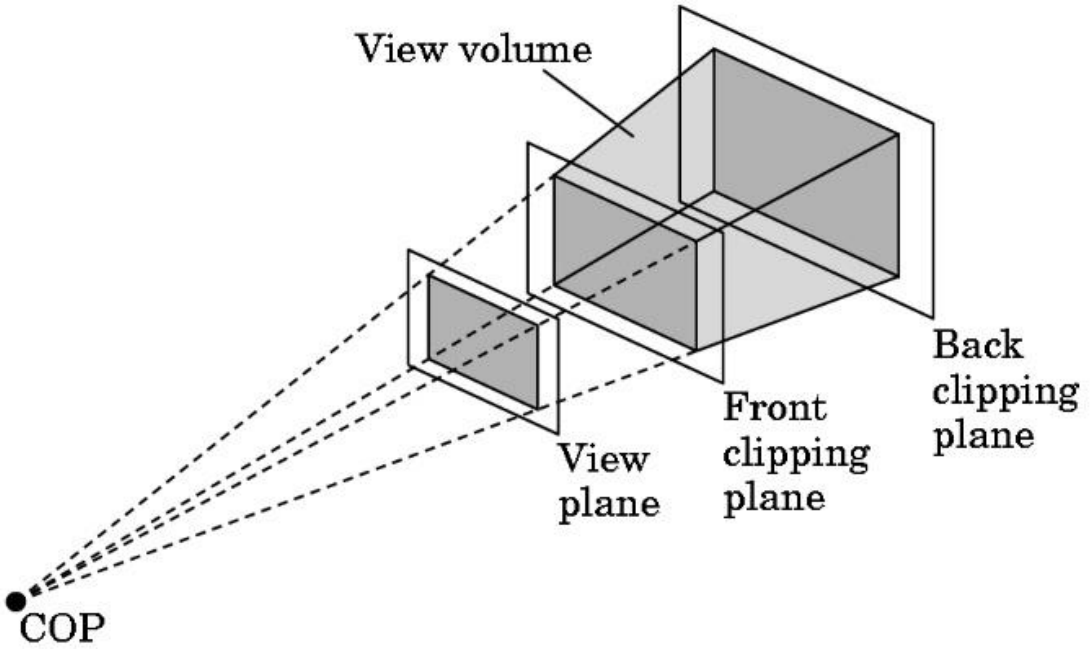
- Reduz o número de primitivas a serem renderizadas
- *View frustum culling*: remove objetos fora do campo de visão da câmera
- **Clipping**: `recorta´ as partes de objetos que estão fora do campo de visão
- *Back-face culling*: remove as primitivas não visíveis ao observador (estão `de costas´ para a câmera) (*facing away from viewer*) (pode ocorrer na etapa de processamento geométrico, ou após projeção e mapeamento para *viewport*)

- Porque?

Processamento geométrico: *culling*

- *View frustum culling*: remove objetos fora do campo de visão da câmera





Iluminação

- *Gouraud shading* – computado no espaço da câmera (cena no CCS)

- Porque?

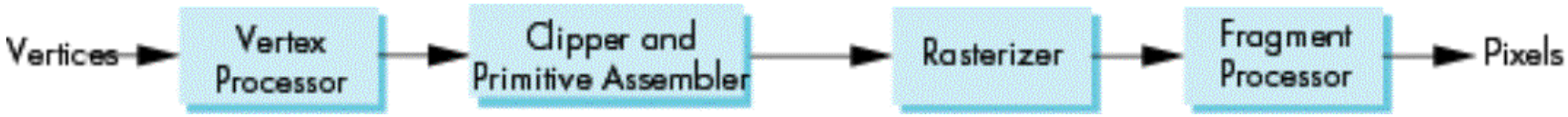
Iluminação

- *Gouraud shading* – computado no espaço da câmera (cena no CCS)

- Porque o cálculo da iluminação requer computar um ângulo entre um `raio de luz´ incidente à primitiva e a normal à ela, bem como calcular a distância da primitiva à fonte
  - As distâncias e ângulos não seriam preservados se isso fosse feito depois da transformação de projeção perspectiva

OpenGL

- API com funções para acessar os recursos da placa gráfica para renderizar malhas de triângulos
  - Programação OpenGL: acesso às funcionalidades da API, por meio de alguma linguagem de programação (C, C++, Java, ...)
- Uma implementação do pipeline conceitual visto



- Garante portabilidade da aplicação: pode ser executada em GPUs diferentes

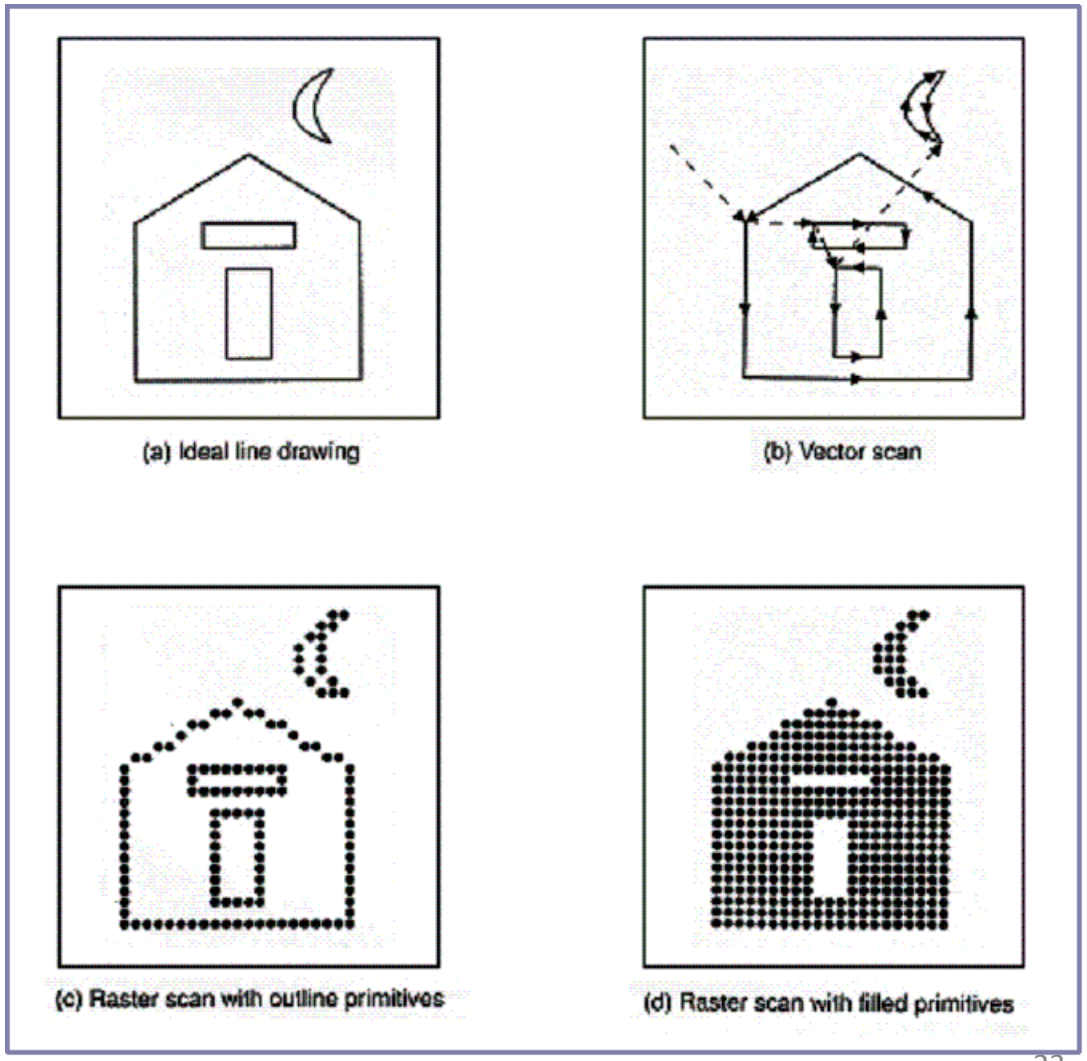
Shaders

- Programas para executar na GPU, que usam as funções da API OpenGL
- Aplicação (p.ex., um jogo) gera códigos na linguagem GLSL, que são compilados e linkados para gerar um executável

Representações gráficas

- *Vector* vs *raster*

- Vector vs Raster Graphics | Buddy Media - YouTube



Dispositivos

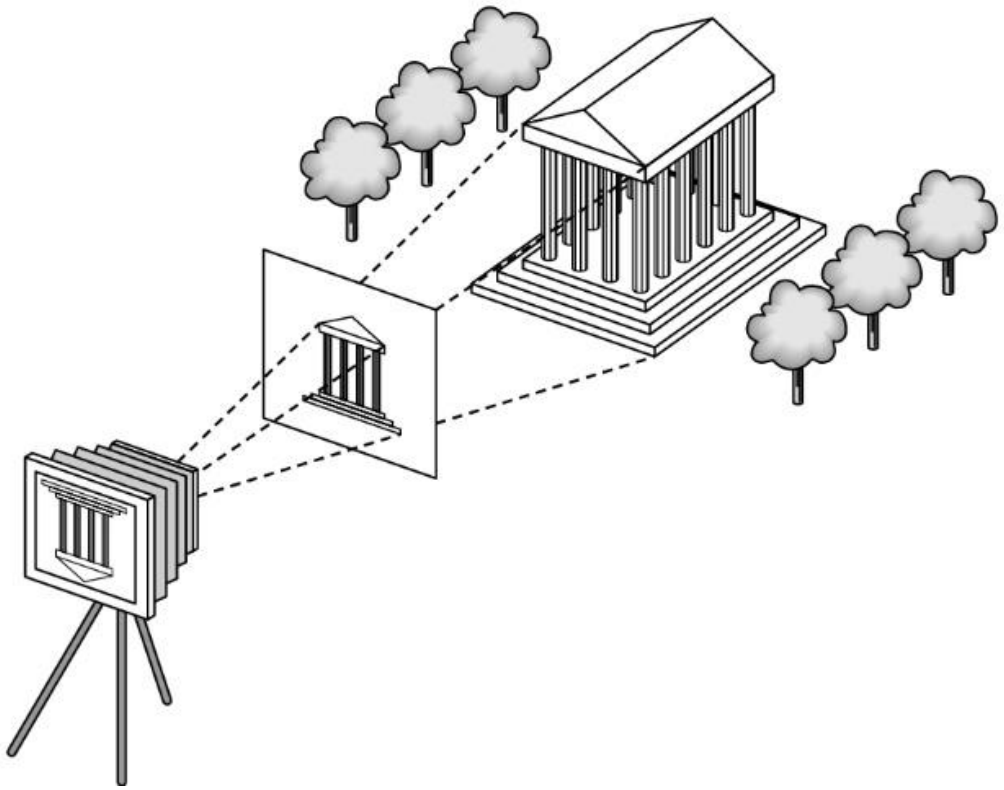
- Tecnologia *raster*: imagem mantida em um frame buffer (tem uma resolução e uma profundidade)

- Tecnologia *vector*: instruções para gerar a imagem. Ex. plotter
  - High-Speed 3D Printed Arduino Pen Plotter - YouTube

Formação de imagens

Modelo de câmera sintética

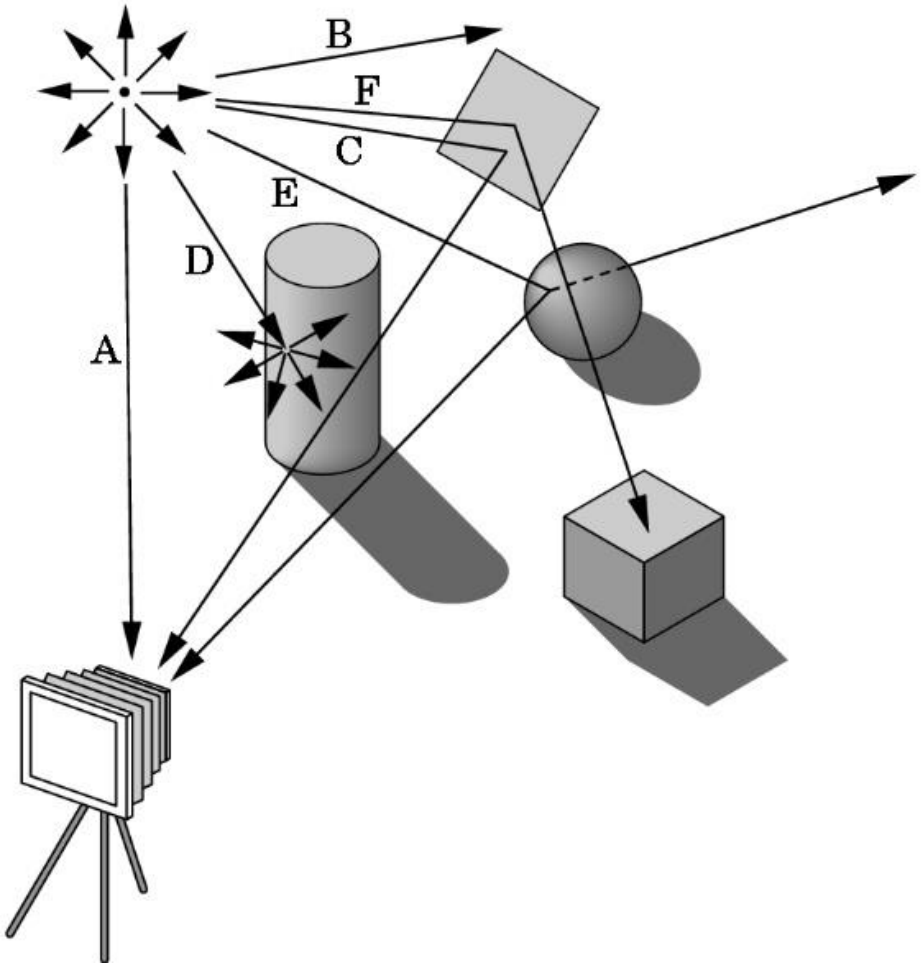
- Cena
- Câmera (observador)
- Iluminação (fonte de luz)



Formação de imagens

Modelo de câmera sintética

- *Ray tracing*
  - Algoritmo de rendering
  - Uma maneira de `simular´ o processo



Formação de imagens

Pipeline OpenGL

- Arquitetura em pipeline que processa primitivas gráficas descritas por triângulos, uma de cada vez, na ordem em que foram geradas pela aplicação
- Só considera efeitos locais de luz