

# Computação Gráfica e Interfaces

2017-2018  
Fernando Birra

# Modelos e Arquiteturas

2017-2018  
Fernando Birra

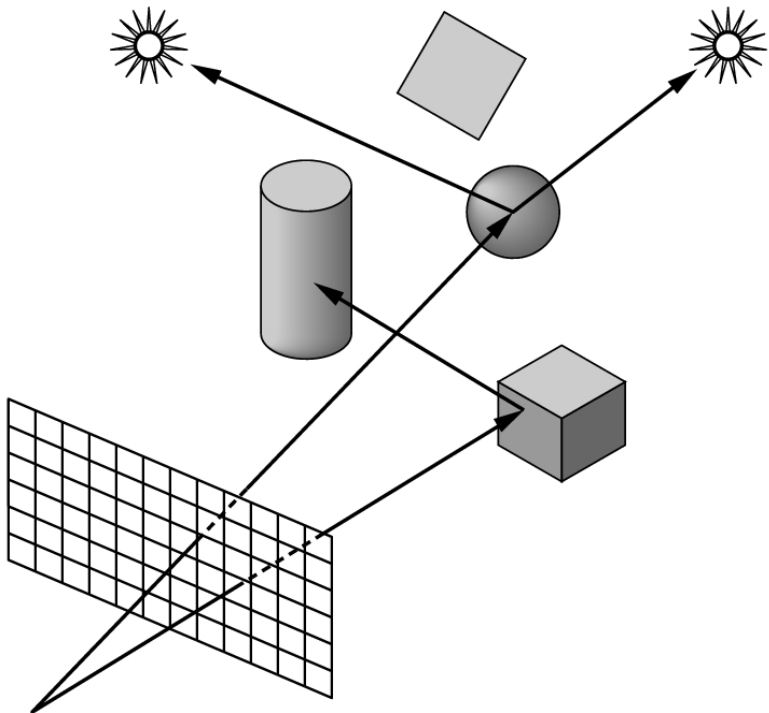
# Objetivos

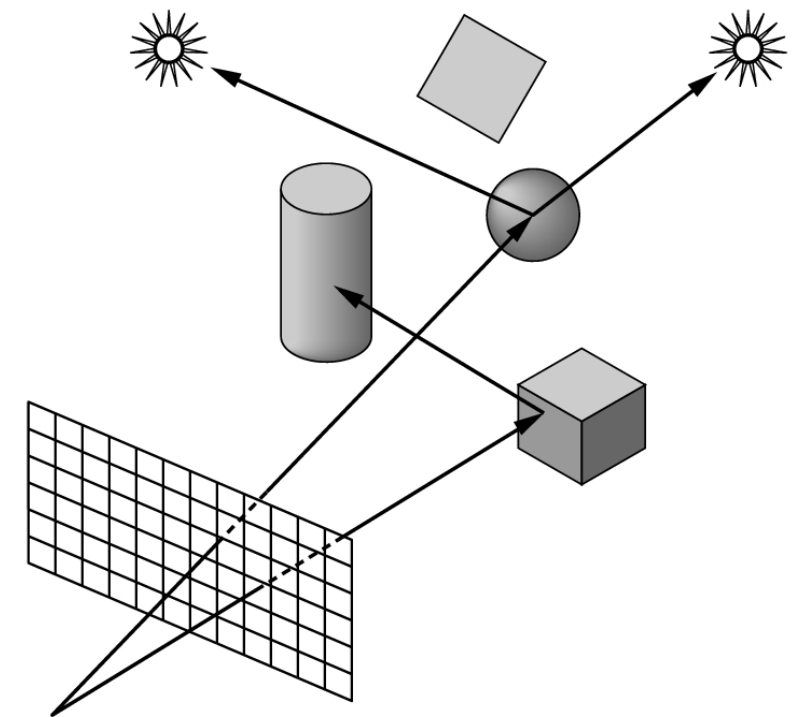
- Princípios de desenho de sistemas gráficos
- Introdução à arquitetura em pipeline
- Examinar as componentes dum sistema gráfico

# Síntese de Imagens (revisita)

- Podemos usar o modelo da câmara sintética para servir de guia ao desenho de hardware/software?
- A interface de programação de aplicações (API) apenas necessita permitir a especificação de:
  - Objetos
  - Materiais
  - Câmara (observador)
  - Luzes
- Mas como se implementa o sistema?

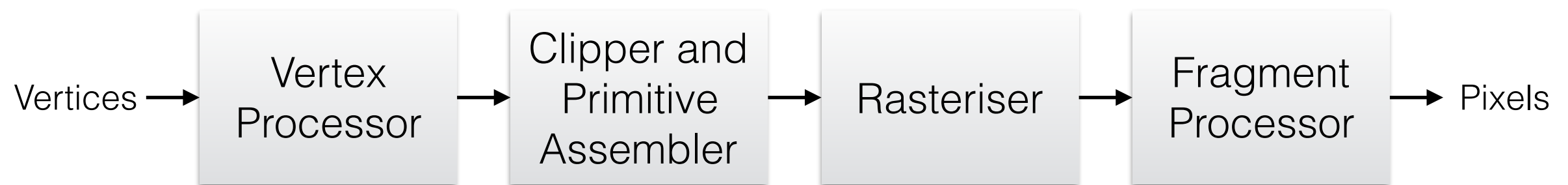
# Abordagem Física

- **Ray tracing:** Seguir o percurso dos raios de luz partindo do centro de projeção até que sejam absorvidos ou se percam no espaço
    - Permite reproduzir efeitos globais:
      - Reflexões múltiplas
      - Objetos translúcidos
      - Sombras
    - Lento
    - Requer o conhecimento de toda a cena, a cada momento
  - **Radiosidade:** método baseado nas transferências de energia entre os objetos presentes na cena
    - Muito lento
- 
- O diagrama ilustra o processo de ray tracing. Um ponto de origem (olho) no canto inferior esquerdo emite um raio que atravessa uma grade (plano da imagem) e atinge uma esfera. A partir da esfera, dois raios secundários são gerados: um reflete-se para cima e para a esquerda, atingindo um sol, e o outro reflete-se para baixo e para a direita, atingindo um cubo. O cubo, por sua vez, emite um raio que atinge um cilindro. O cilindro emite um raio que atinge a grade. A grade também emite um raio que atinge a esfera. Este processo de reflexão múltipla demonstra como o ray tracing captura efeitos globais como reflexões e sombras.



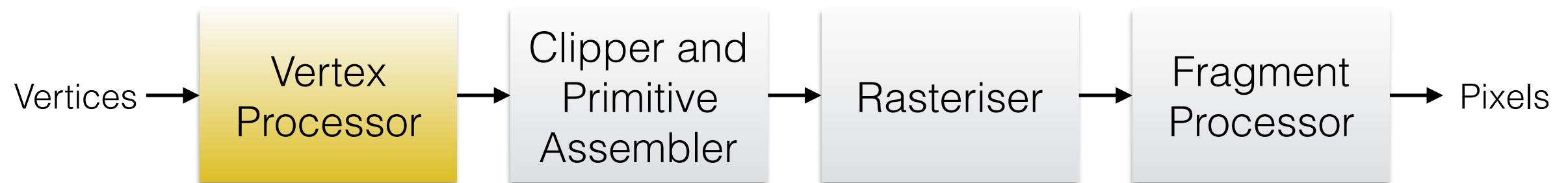
# Abordagem Prática

- Processar os objetos isoladamente, na ordem pela qual são gerados pela aplicação
  - Apenas se podem considerar fenómenos locais de iluminação, sem ter em atenção os outros objetos.
- Arquitetura em pipeline



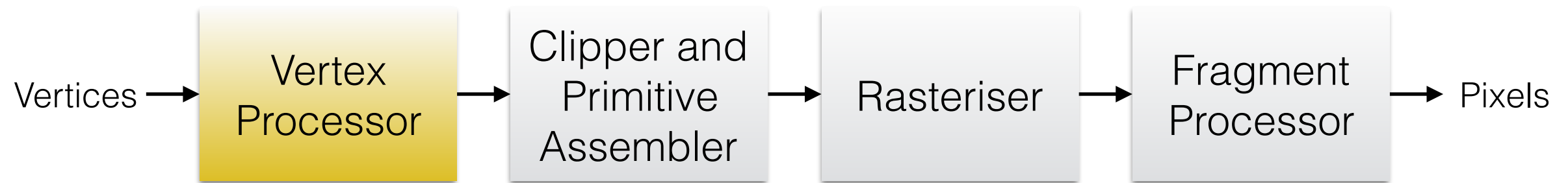
- Todas as fases podem ser implementadas no hardware gráfico (GPU)

# Processamento de Vértices



- Uma grande parte do trabalho no pipeline gráfico consiste em efetuar mudanças entre sistemas de coordenadas:
  - Coordenadas dos pontos em referenciais próprios dos objetos
  - Coordenadas no referencial da câmara
  - Coordenadas no referencial do ecrã
- Cada mudança de referencial equivale a uma operação matricial (mudança de base)
- O processamento de vértices também inclui a transformação de atributos (p.ex. cor), associados pela aplicação a cada vértice, em valores que irão variar durante o varrimento dos pixels gerados para as primitivas gráficas (triângulos na sua grande maioria)

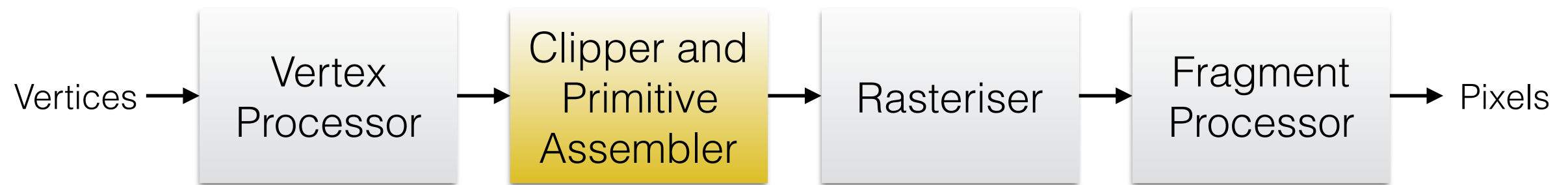
# Projeção



- A projeção permite combinar os objetos 3D e as características 3D do observador numa imagem 2D
  - Projeção perspetiva: as linhas projetantes encontram-se no centro de projeção (ponto)
  - Projeção paralela: as linhas projetantes são paralelas entre si (direção)
- A projeção é também realizada através duma operação matricial (multiplicação por uma matriz de projeção)

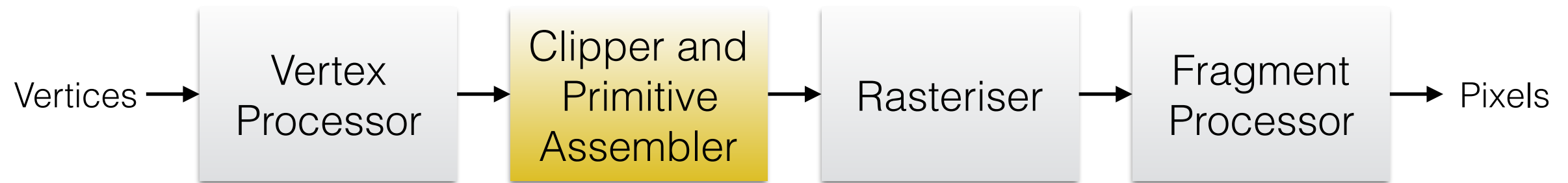


# Primitive Assembly

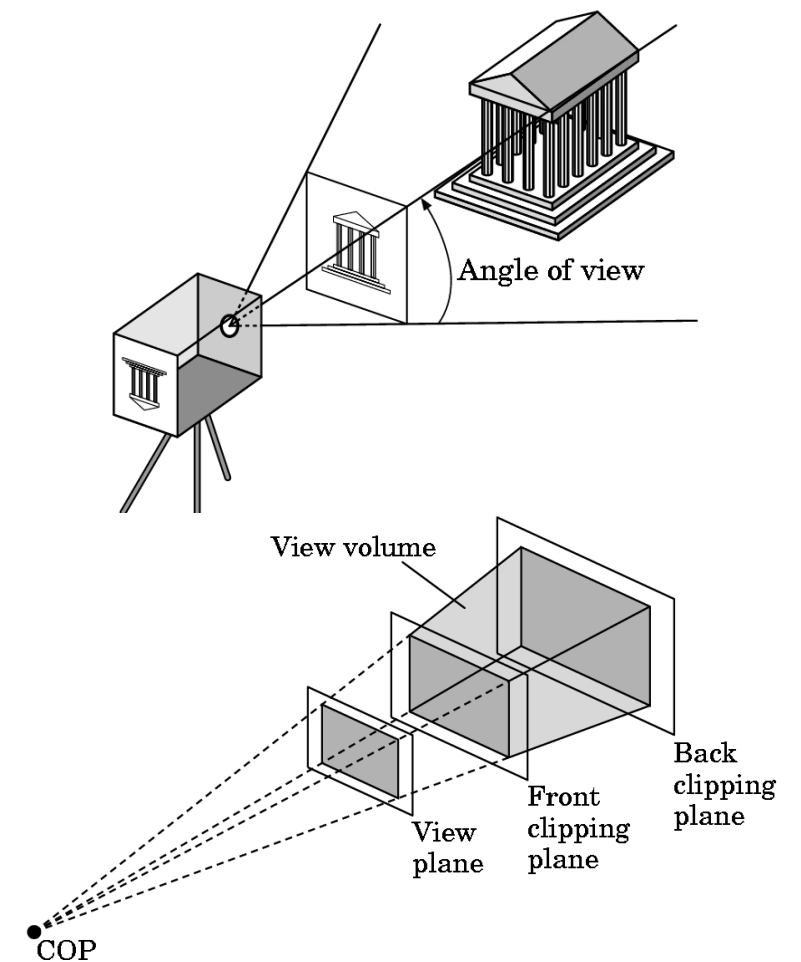


- Os vértices submetidos ao pipeline são agrupados para a formação das primitivas, antes que se possa proceder ao seu varrimento (geração dos pixels que as constituem) e recorte (clipping):
  - segmentos de reta
  - polígonos (apenas triângulos em WebGL)
  - Linhas Curvas
  - Superfícies Curvas

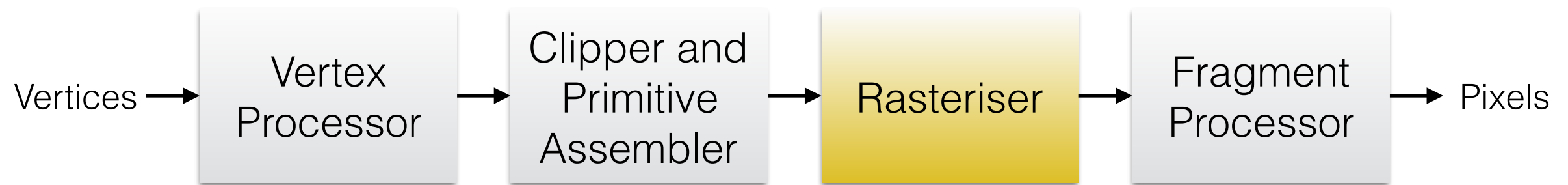
# Recorte



- Uma câmara, quer seja real ou virtual, apenas consegue “ver” parte do mundo:
- Os objetos que não estão dentro dessa mesma parte visível são recortados (total ou parcialmente)

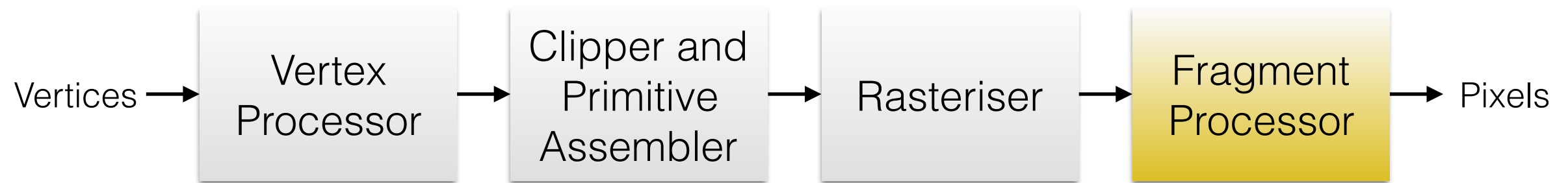


# Conversão por varrimento



- Se um objeto não for recortado na sua totalidade (descartado), os pixels correspondentes no *frame buffer* terão que ser gerados
- Um *rasteriser* produz um conjunto de fragmentos (potenciais pixels no final do pipeline) para cada objeto
- Sendo potenciais pixels, os fragmentos possuem:
  - localização no *frame buffer* (*coordenadas x e y*)
  - atributos de cor e profundidade (*coordenada z*)
- Os atributos associados aos vértices poderão ser propagados para os fragmentos e interpolados sobre os objetos pelo módulo de varrimento.

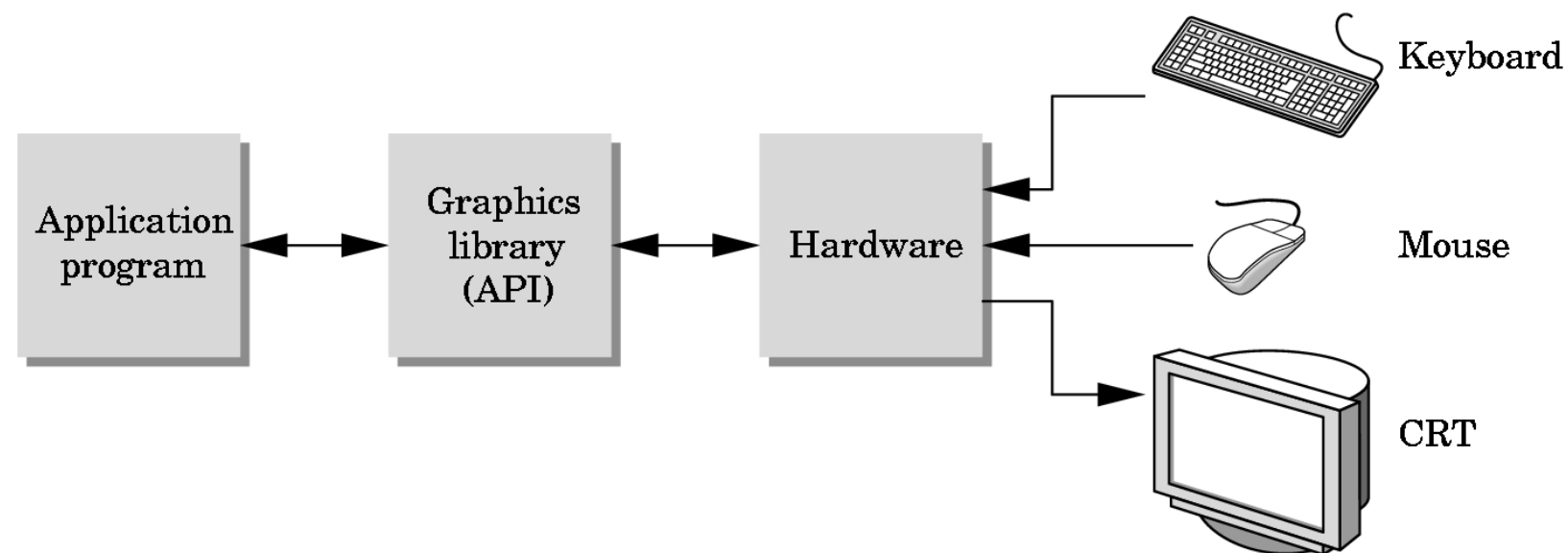
# Processamento de Fragmentos



- Os fragmentos são processados para determinar a cor final do pixel correspondente no *frame buffer*
- As cores podem ser determinadas por mapeamento de texturas ou por interpolação de valores de atributos nos vértices
- Os fragmentos poderão ser “bloqueados” por outros mais perto da câmara
  - Remoção de Superfícies Ocultas (**H**idden **S**urface **R**emoval)

# A API

- O programador “vê” o sistema gráfico através da interface exposta ao nível do software (**A**pplication **P**rogrammer **I**nterface)



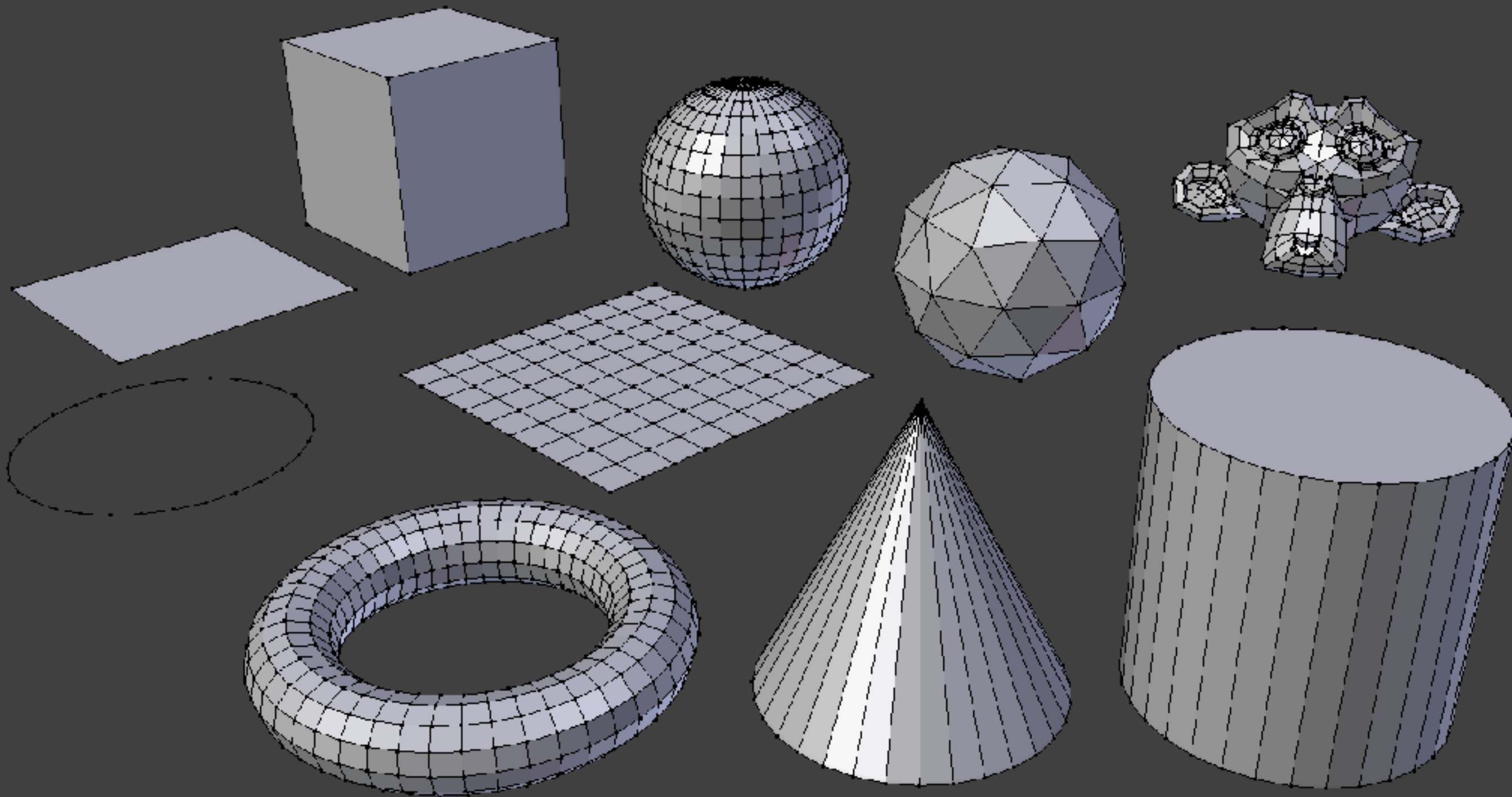
# Conteúdo da API

- Funções para especificar os componentes que permitem sintetizar a imagem:
  - objetos
  - observador
  - fontes de luz
  - materiais
- Outros:
  - Input dos dispositivos (mouse, teclado)
  - Capacidades do sistema

# Primitivas Gráficas

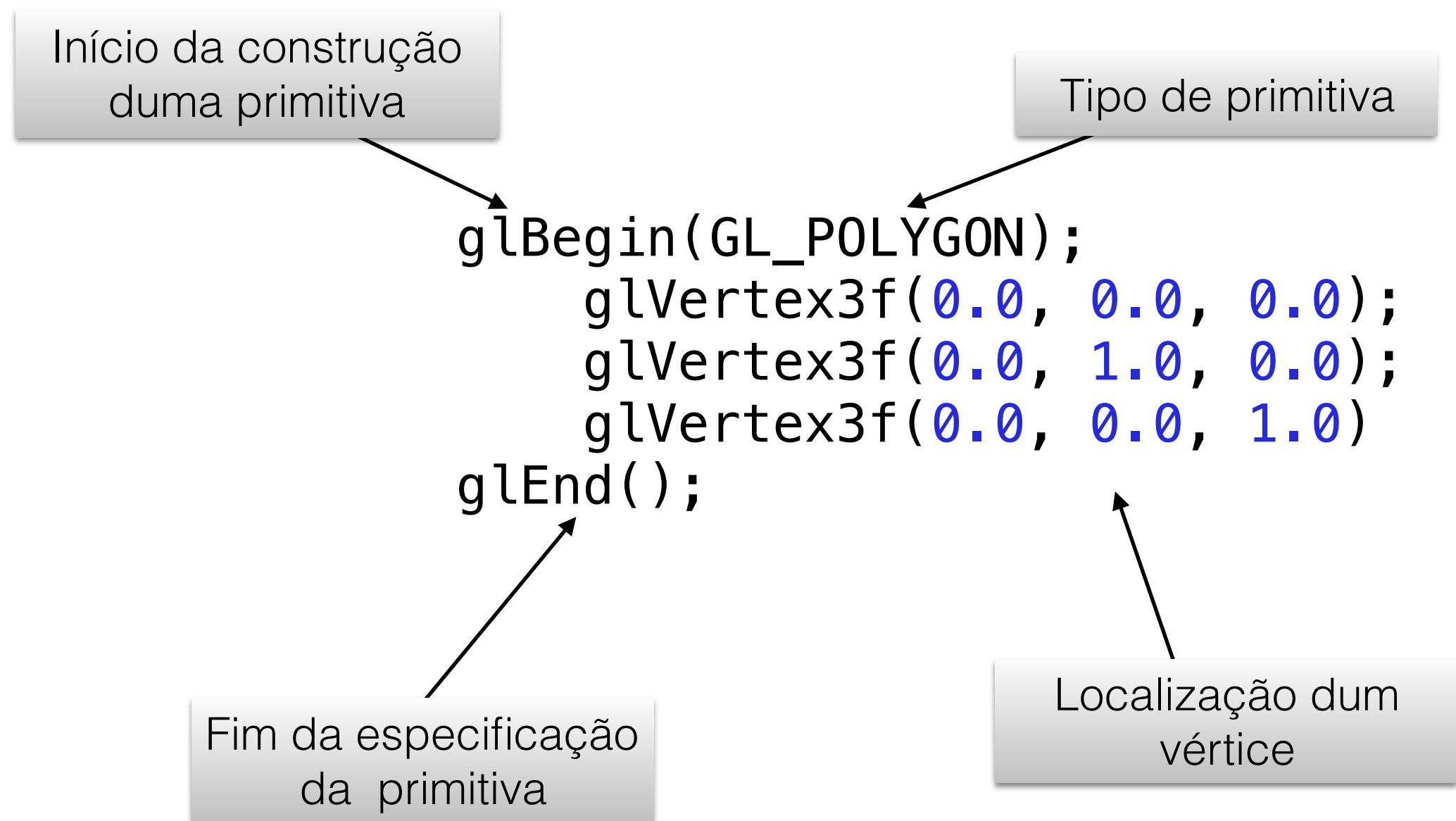
- A maior parte das APIs gráficas suportam um reduzido número de primitivas gráficas:
  - Pontos (objetos 0D)
  - Segmentos de reta (objetos 1D)
  - Polígonos (objetos 2D)
  - Linhas e superfícies curvas
- Todas as primitivas são definidas através de pontos no espaço - vértices

# Primitivas Gráficas





# Exemplo (deprecado)



# Exemplo (GPU)

- Colocar os valores associados aos vértices (normalmente incluindo as coordenadas) num vetor

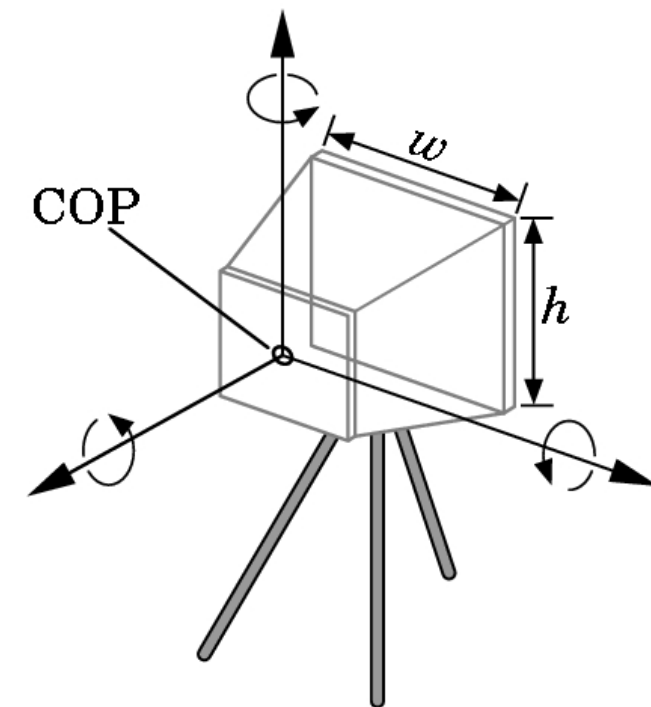
```
var points = [  
    vec3(0.0, 0.0, 0.0),  
    vec3(0.0, 1.0, 0.0),  
    vec3(0.0, 0.0, 1.0)  
];
```

Este é um exemplo redutor, com apenas um triângulo. Mas o vetor poderá conter 3N vértices para desenhar N triângulos efetuando uma única chamada.

- Enviar o vetor para o GPU
- Pedir ao GPU para desenhar usando o tipo de primitiva pretendida (pontos, linhas, triângulos, etc.)

# Especificação da câmara

- 6 graus de liberdade para a posição (3) + orientação (3)
  - posição do centro da lente (**C**enter **O**f **P**rojection)
  - Orientação (vetor 3D)
- Graus de liberdade adicionais
  - Lente (distância focal)
  - Tamanho do filme/sensor ( $w \times h$ )
  - Orientação do plano do filme/sensor (perpendicular/oblíquo)



# Especificação de Materiais e Luzes

- Características das luzes:
  - fontes de luz pontual (*point source*) vs. áreas luminosas (*distributed light sources*)
  - próximas vs distantes
  - cor
- Propriedades dos materiais
  - absorção - cor do material
  - reflexão
    - difusa
    - especular

