Introdução (2)

Computação Gráfica

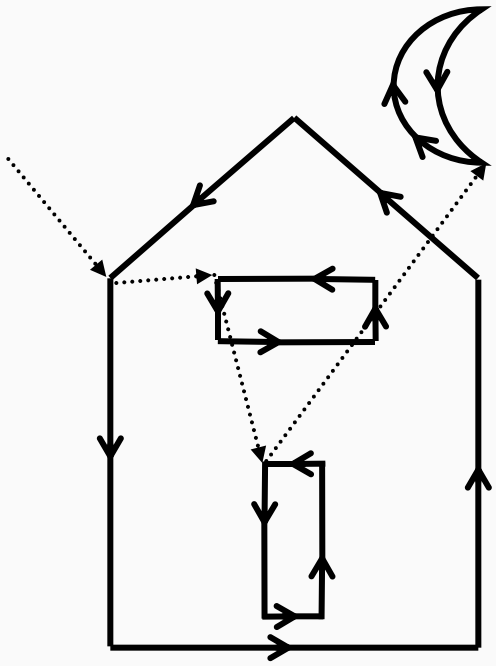
Inverno 2011/2012

Fonte: [Autobahn WIP] http://hof.povray.org/

Sumário

- Dispositivos de saída
  - Vectoriais
  - *Raster*
- Arquitectura de um sistema gráfico interactivo
  - Paradigma da camara fotográfica
  - Transformação de visualização
  - Luz e sombreamento
  - Projecção
  - Rasterização

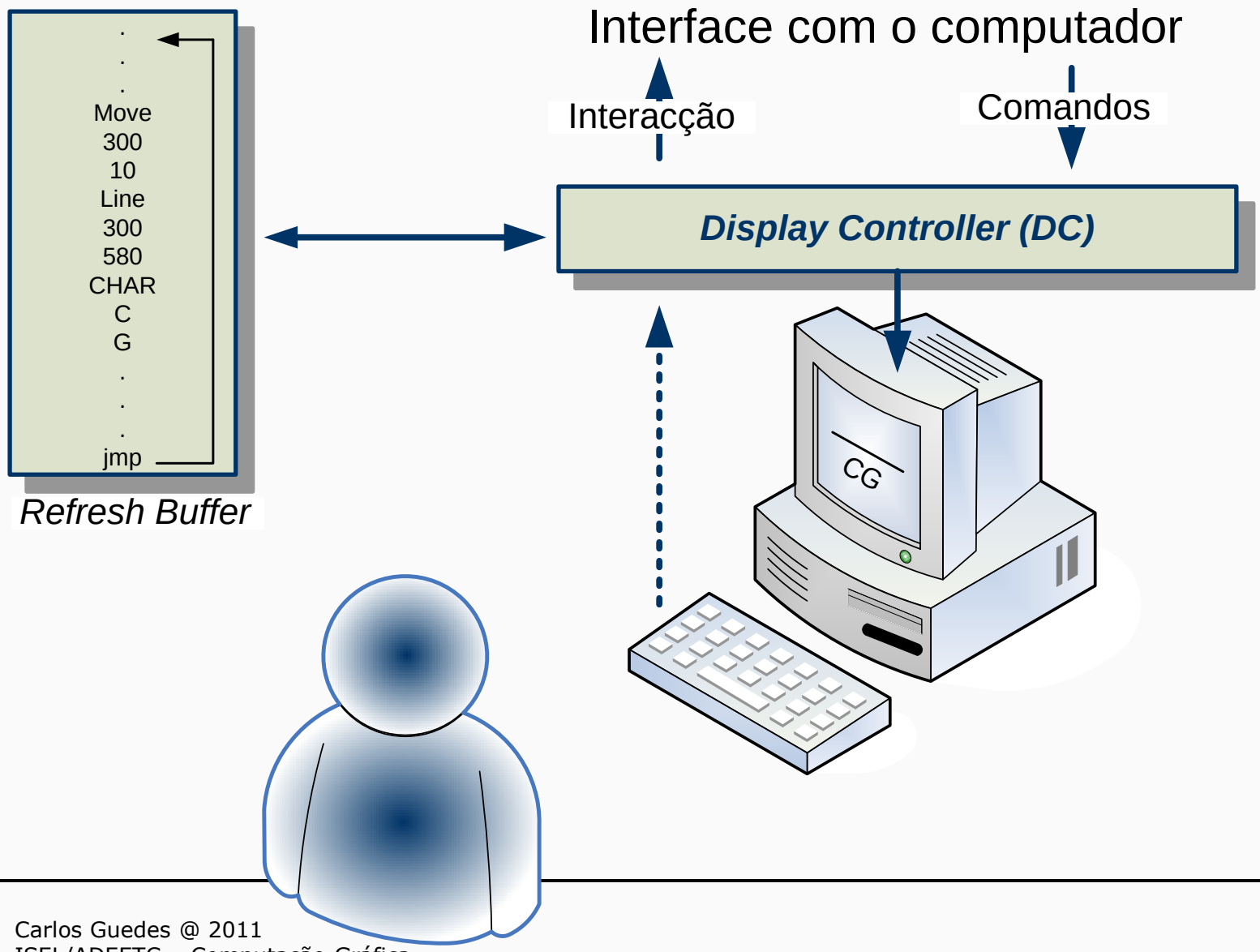
Dispositivos vectoriais



- Os dispositivos vectoriais foram introduzidos em meados dos anos sessenta
- O termo vector é, neste contexto, sinónimo de **linha**
- Estes dispositivos utilizam muitas das técnicas ainda em uso nos sistemas actuais

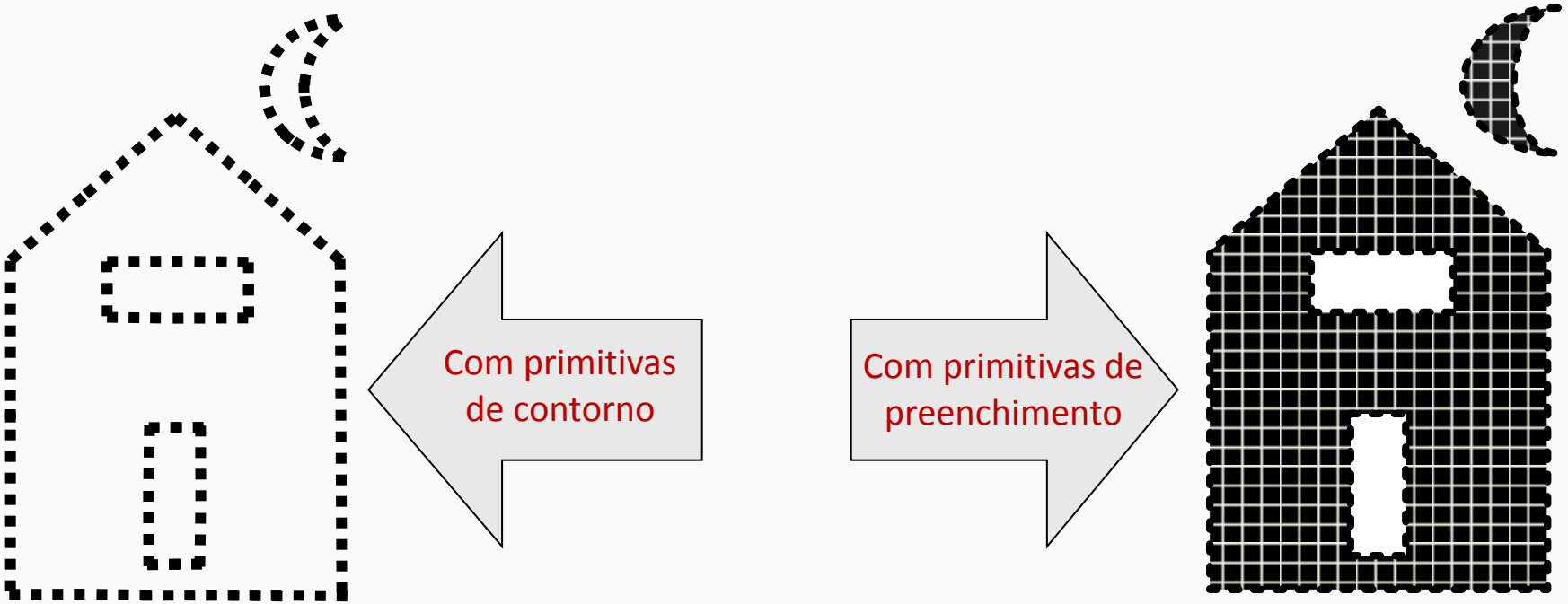
- Estes dispositivos são capazes de desenhar pontos, linhas e caracteres, convertendo as coordenadas em voltagens, de forma ao feixe de electrões excitar a camada de fósforo presente no monitor CRT
- Note-se que o feixe anda arbitrariamente pelo ecrã, de acordo com as directivas ditadas pelo comandos – *Random scan*

Dispositivos vectoriais

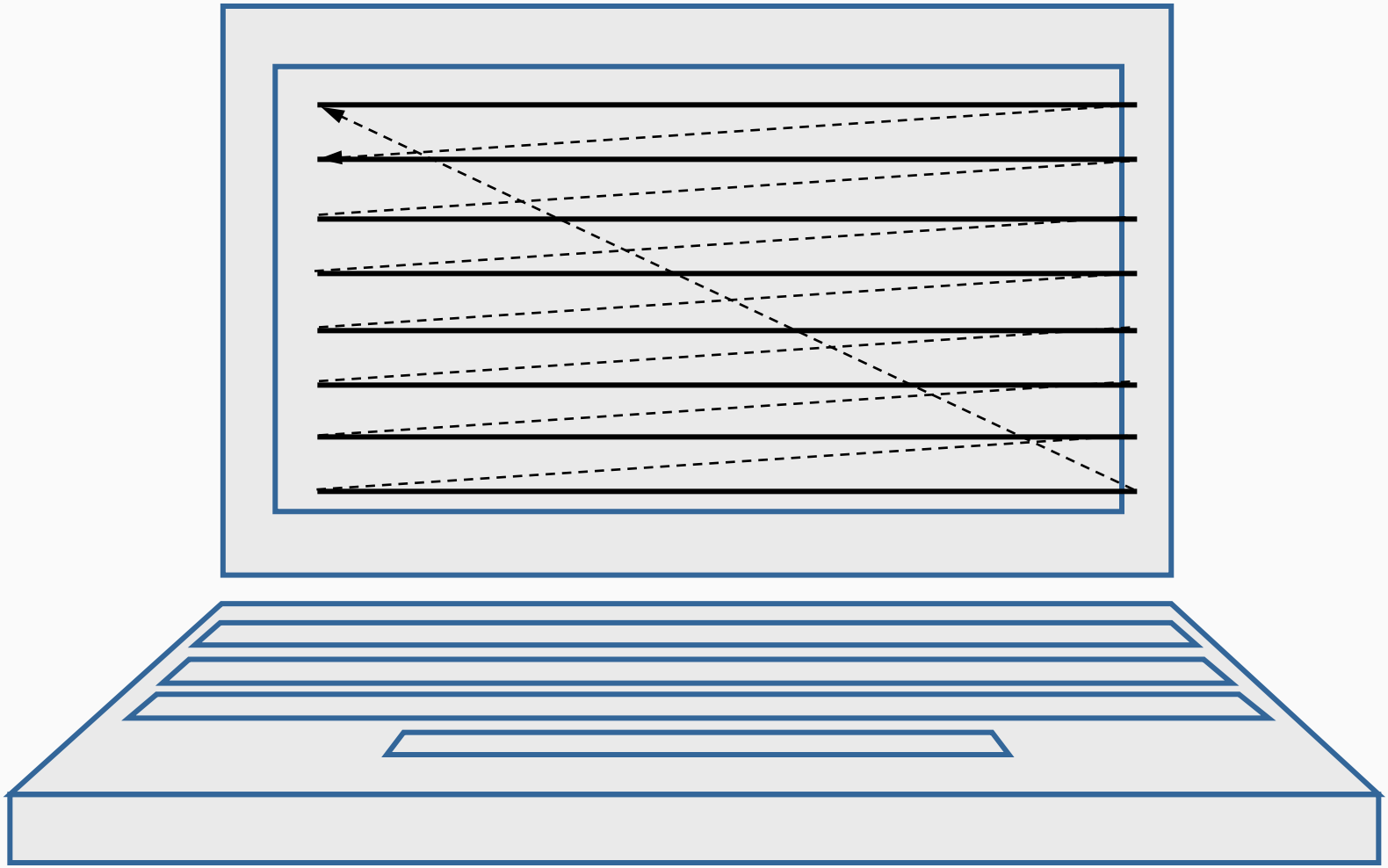


Dispositivos *raster*

- A tecnologia utilizada na televisão serviu de base aos dispositivos *raster*, que **são os sistemas utilizados hoje em dia**
- A sua massificação começou no inicio dos anos setenta, impulsionada pelas preços “mais baixos” das memórias RAM
- A imagem a ser mostrada é guardada numa matriz pixéis (*raster*)

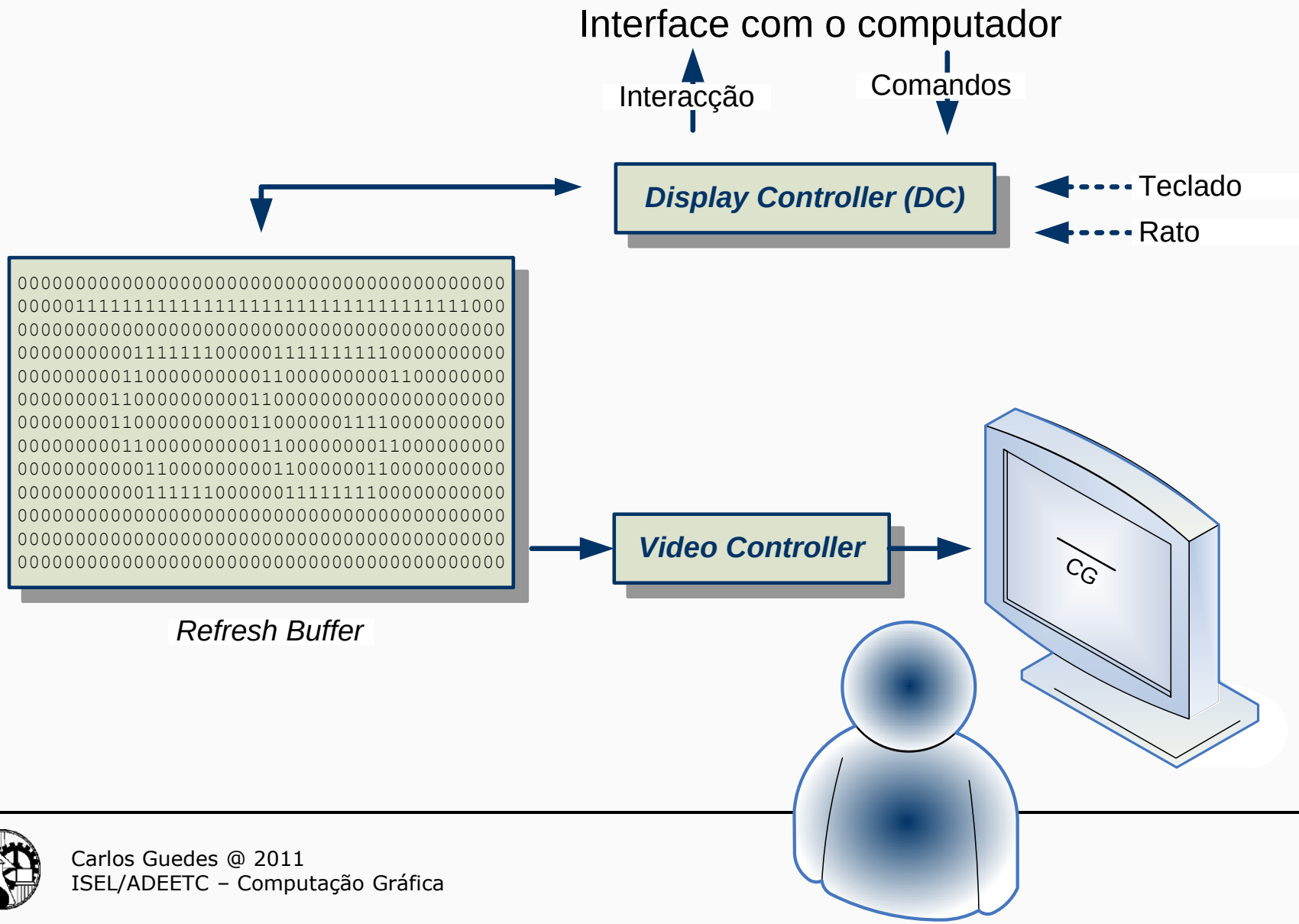


Dispositivos *raster*



Raster Scan

Dispositivos Raster



Raster *vs* vectoriais

+

Vantagens dos sistemas *raster*

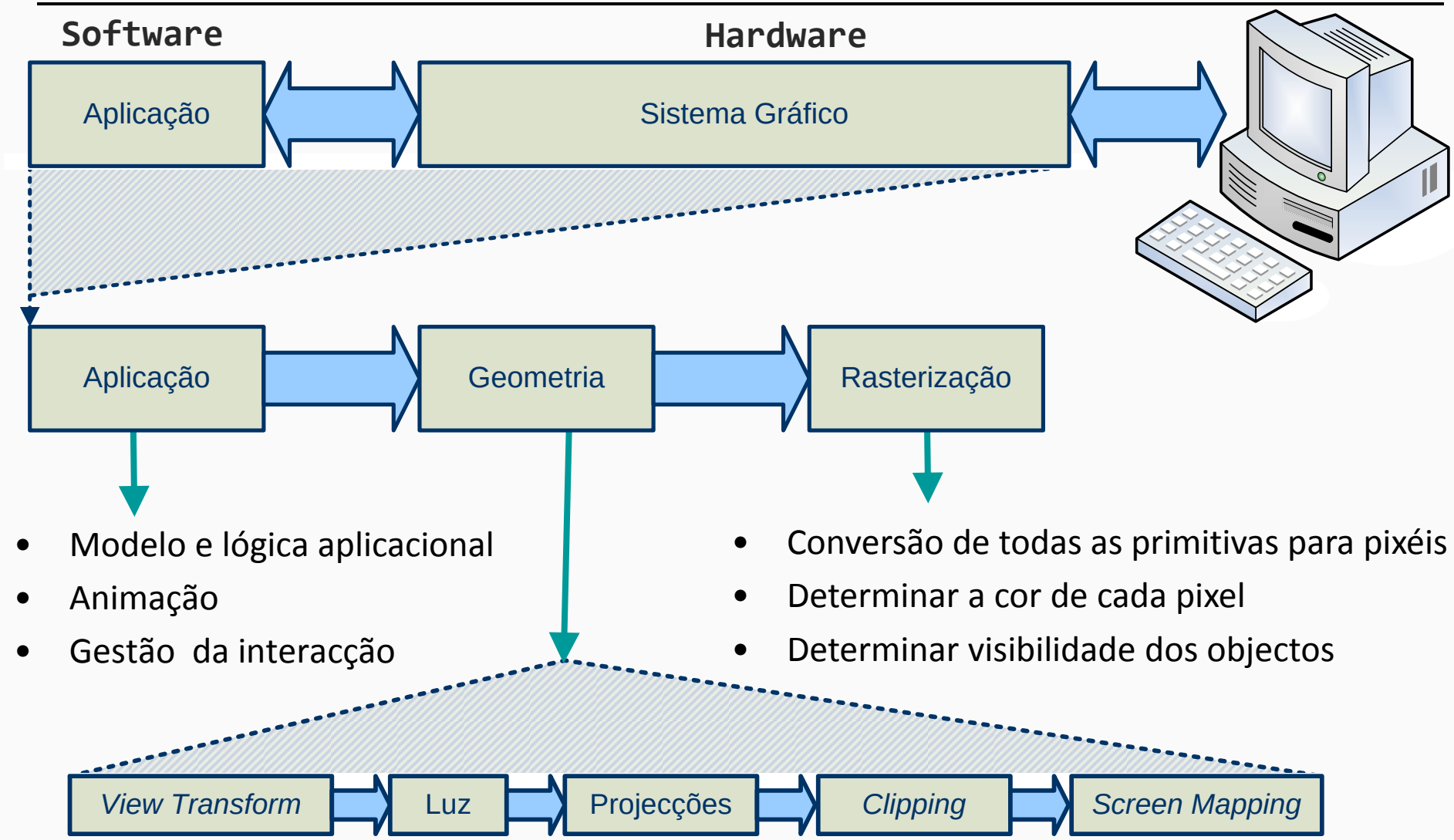
- Menos dispendiosos
- Capacidade de mostrar áreas a cheio
- O refrescamento é independente da complexidade do conteúdo a apresentar

Desvantagens dos sistemas *raster*

- Necessidade de todas as linhas serem convertidas num conjunto discreto de pixéis (discretização) – computacionalmente mais exigente
- As linhas e curvas são aproximações efectuadas à custa de pixéis

-

Arquitectura de um sistema gráfico interactivo



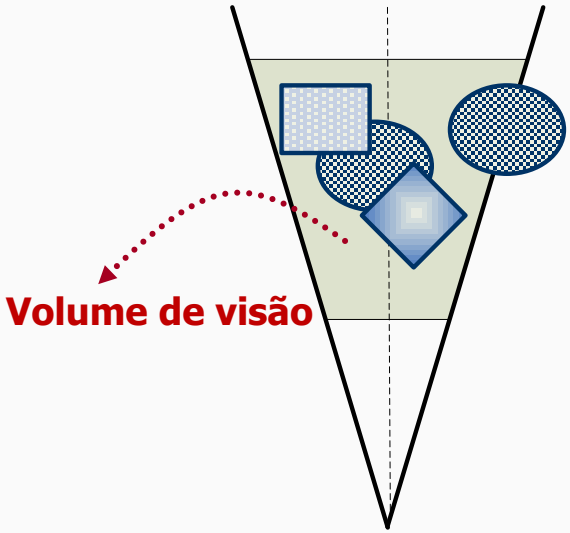
- Modelo e lógica aplicacional
- Animação
- Gestão da interacção

- Conversão de todas as primitivas para pixéis
- Determinar a cor de cada pixel
- Determinar visibilidade dos objectos

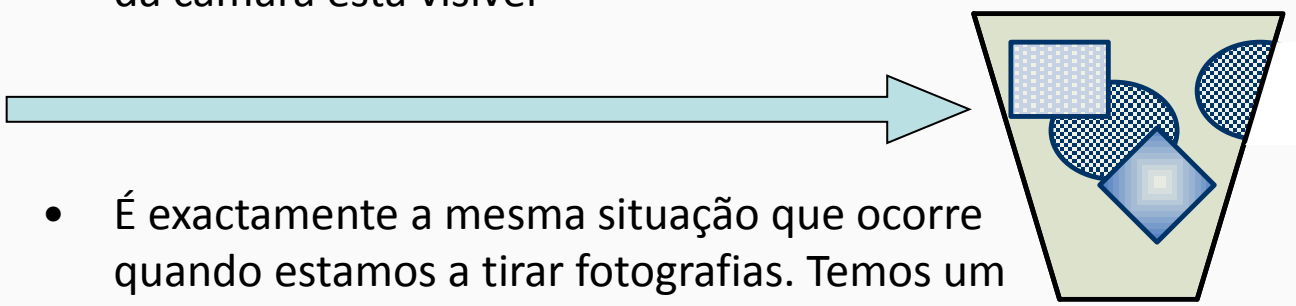
Paradigma da câmara fotográfica

- Antes de avançar para as transformações de vistas (`View Transform`), iremos falar um pouco sobre o **paradigma da câmara fotográfica**

- Sempre que se olha para o mundo, criado virtualmente, a visualização está a ser feita por uma câmara, normalmente designada de vista



- Apenas o que está dentro do volume de visão da câmara está visível

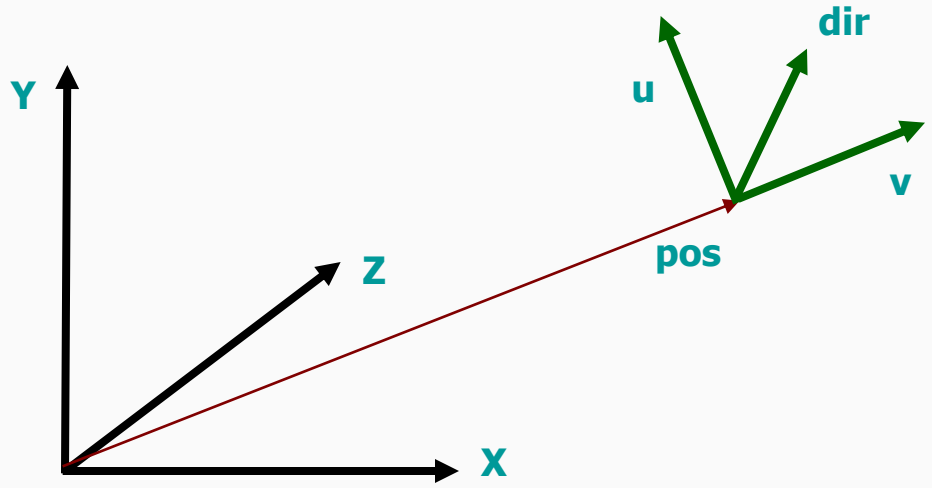


- É exactamente a mesma situação que ocorre quando estamos a tirar fotografias. Temos um **volume de visão** finito e apenas ficará para a posterioridade o que ficar enquadrado com esse volume

Paradigma da câmara fotográfica

- A câmara representa os olhos de quem interage com o mundo virtual

- Para representar uma câmara são necessários 4 vectores (para 3D)
  - A posição dentro do mundo - **pos**
  - Uma base para o seu espaço **(u, v, dir)**

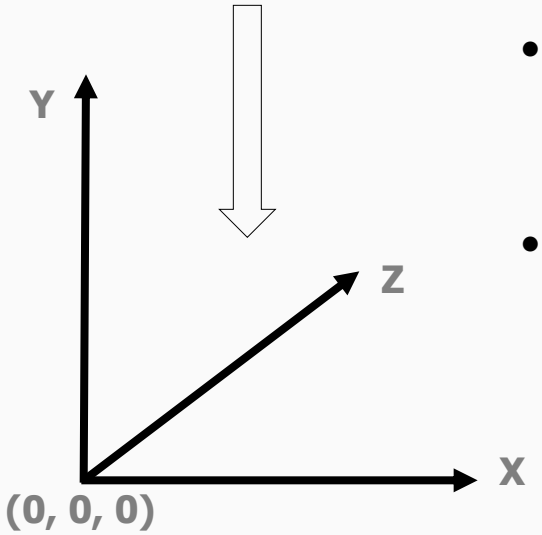


View Transforms

- É importante ter a noção que existem uma série de diferentes espaços em computação gráfica, cada um com o seu sistema de coordenadas

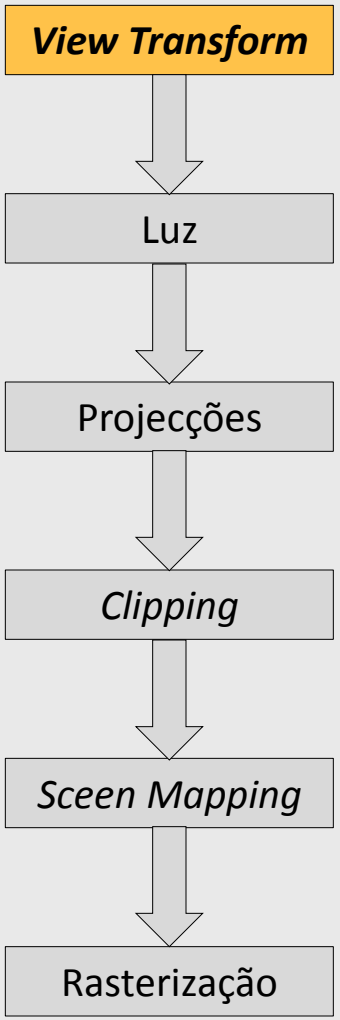
- Os dois principais são o do **mundo** (*WCS – **W**orld **C**oordinate **S**ystem*) e o da **câmara** (*VCS – **V**iew **C**oordinate **S**ystem*)

O espaço do mundo é onde se desenrola a acção



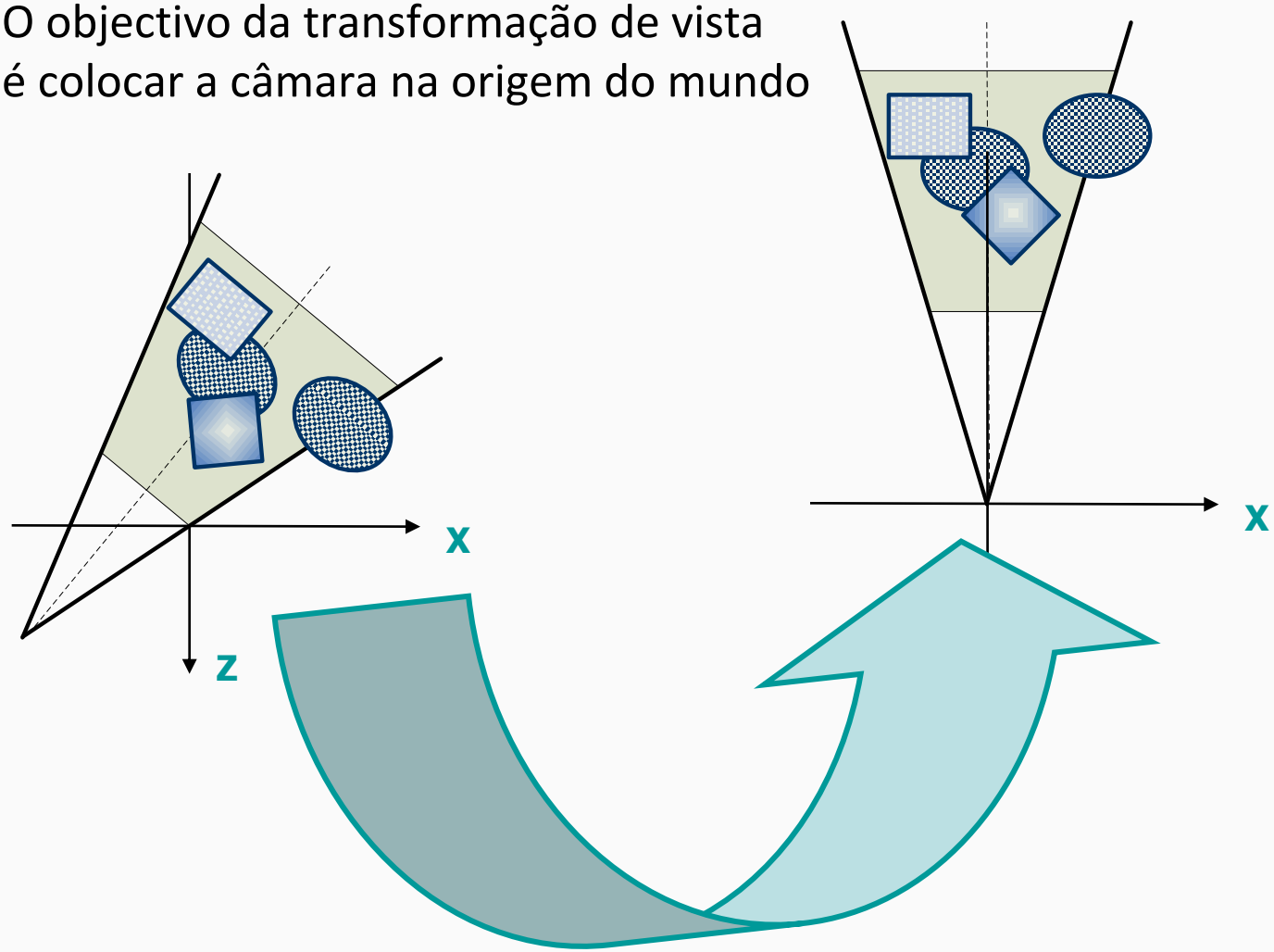
- O espaço da câmara terá outro sistema de coordenadas: **X' Y' Z'** (***u, v*** *e* ***dir***)

- Para que todos os objectos tenham apenas um espaço, é necessário efectuar uma **transformações de coordenadas** de todos os vértices dos objectos



View Transforms

- O objectivo da transformação de vista é colocar a câmara na origem do mundo



View Transform

Luz

Projecções

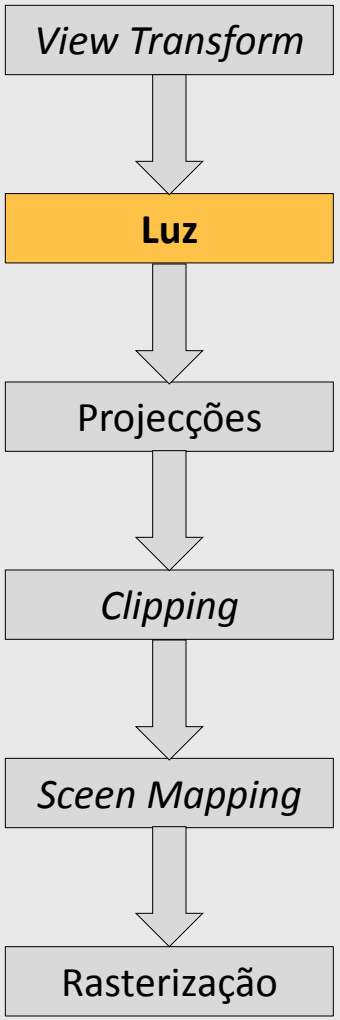
Clipping

Sceen Mapping

Rasterização

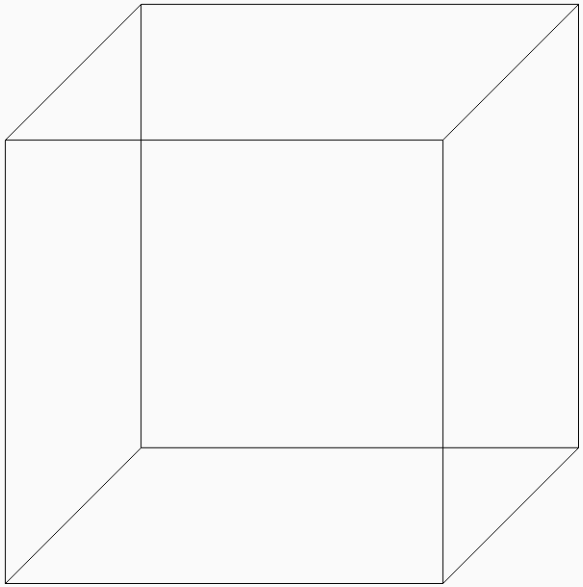
Exemplo: Luz e sombreamento

- Para dar um **maior realismo** aos modelos criados computacionalmente é possível colocar em cena um ou mais pontos de luz
- Nesta fase do processamento o resultado da iluminação é calculado; iluminando algumas partes, escurecendo outras e projectando sombras
- Para isso são usadas equações que modelam o **efeito da luz nas superfícies**. É necessário saber as características da luz incidente nos objectos, assim como as **propriedades destes**.
- Igualmente para aumentar o realismo, os modelos podem ter cor e/ou **texturas**, embora esse processamento seja feitos mais adiante no *pipeline* de processamento.

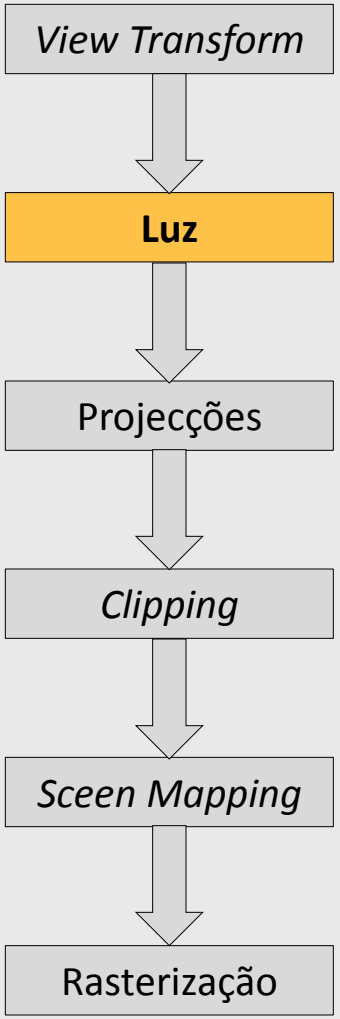


Exemplo: Luz e sombreamento

Está realista?

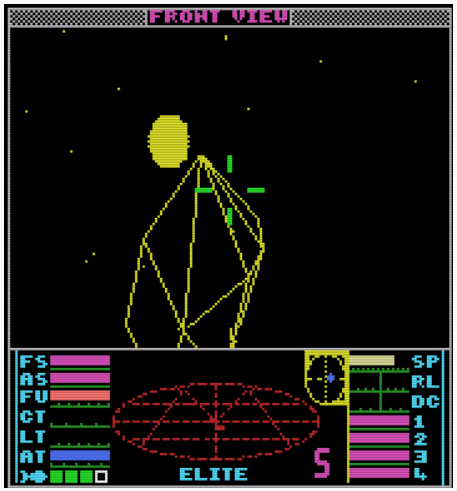


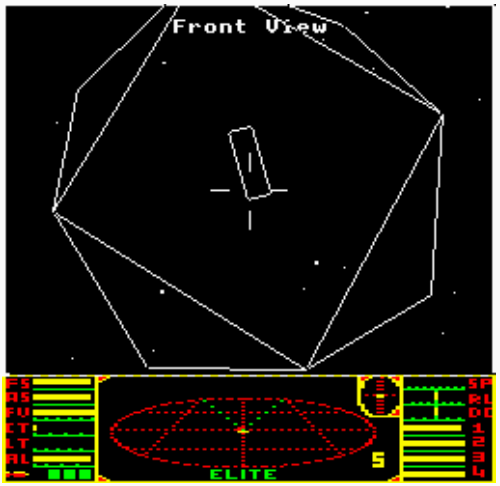
Modo: ***Wireframe***



Exemplo: Luz e sombreamento

Exemplo deste tipo de gráficos: **Jogo ELITE**, 1984



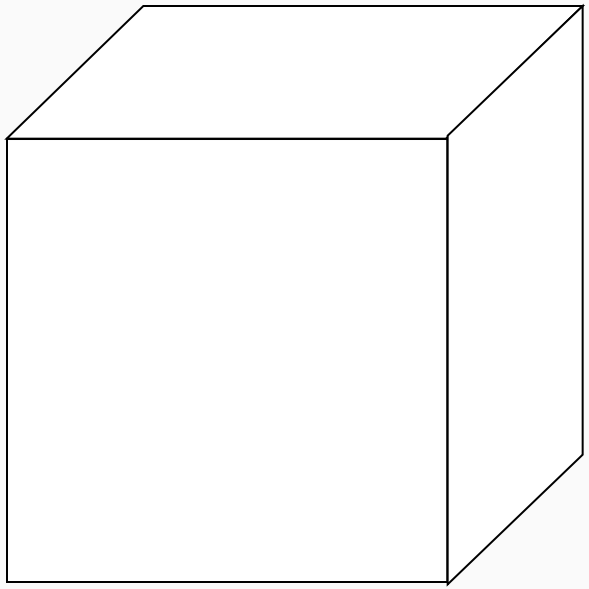


Modo: ***Wireframe***

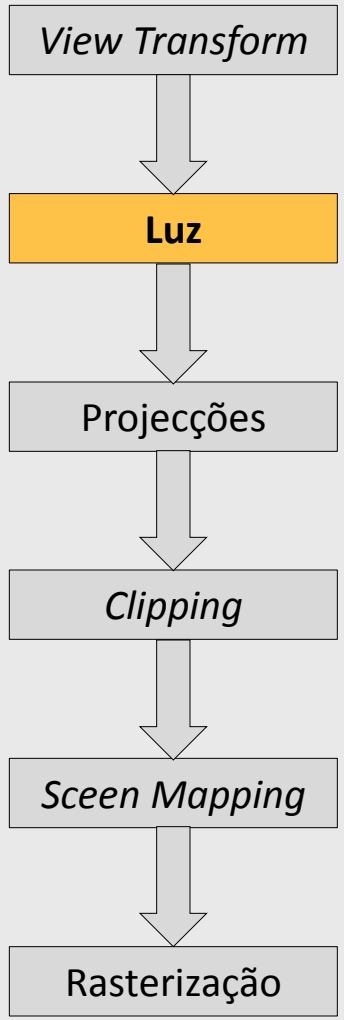
- *View Transform*
- **Luz**
- Projecções
- *Clipping*
- *Sceen Mapping*
- Rasterização

Exemplo: Luz e sombreamento

Melhor?



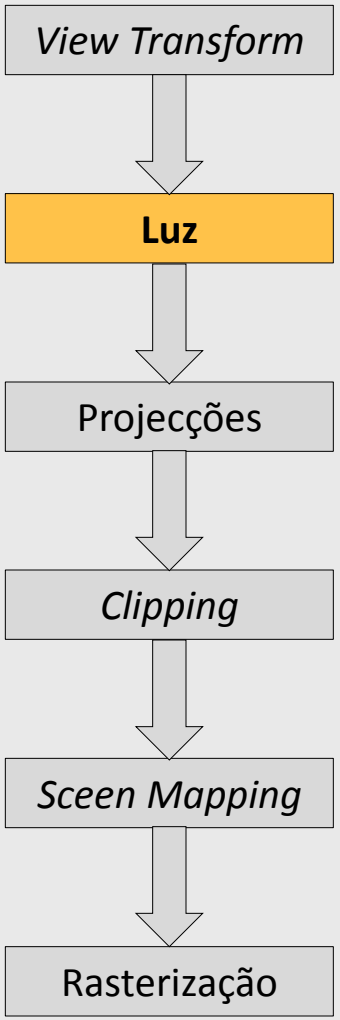
Modo: ***Remoção de linhas ocultas***



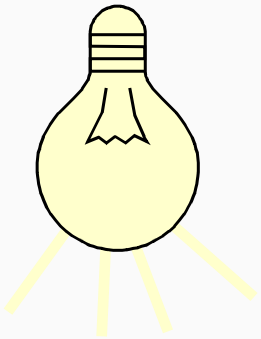
Exemplo: Luz e sombreamento

E agora?

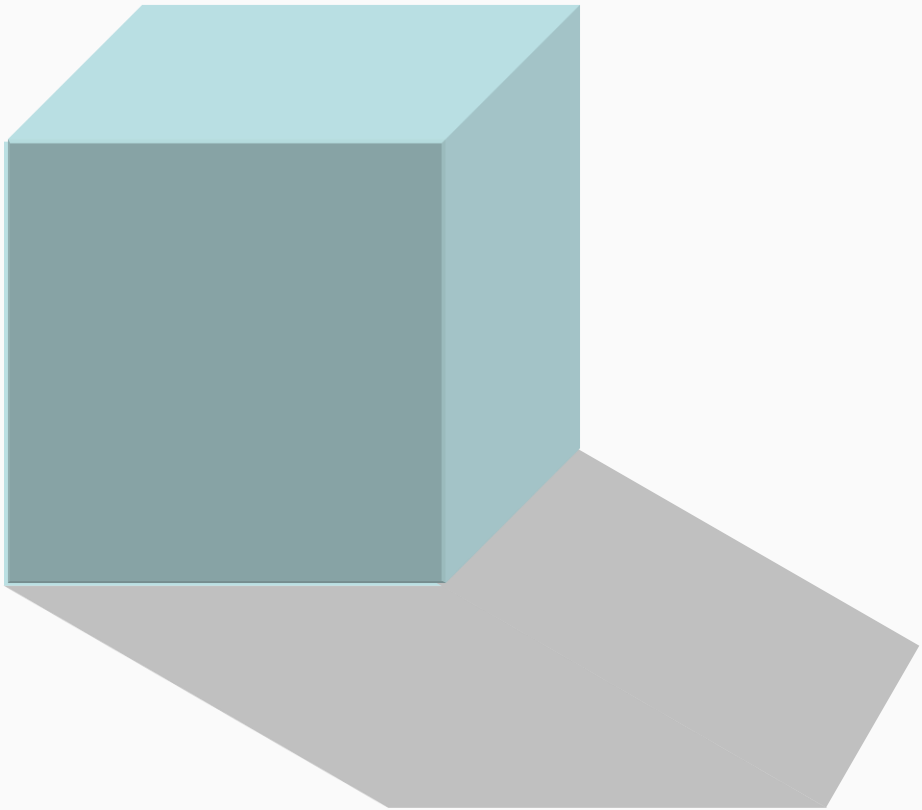
Modo: ***Sombreamento constante***



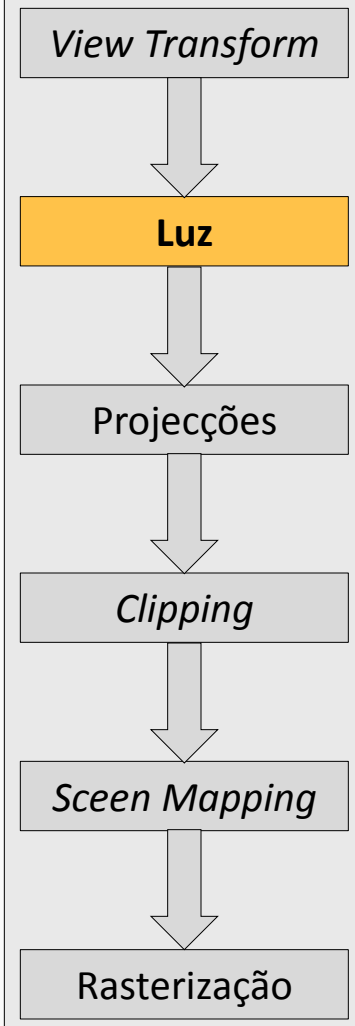
Exemplo: Luz e sombreamento



Melhor!



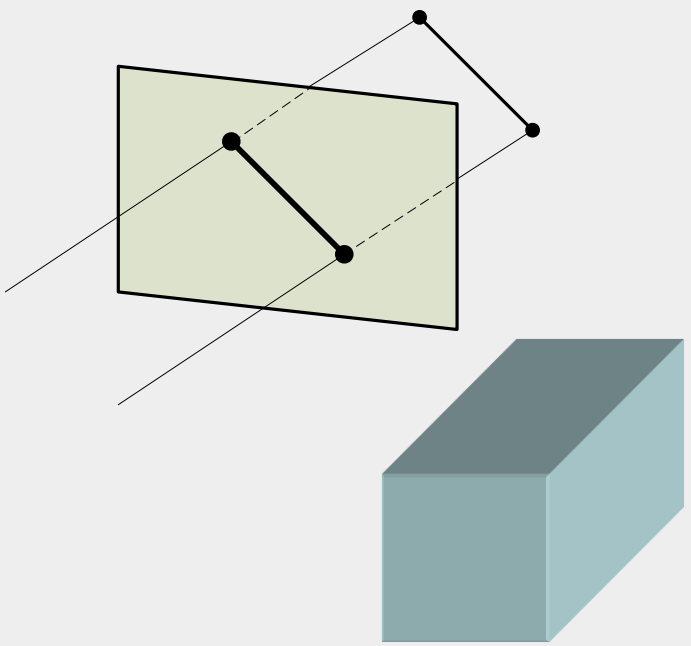
Modo: ***Iluminação – componente difusa***



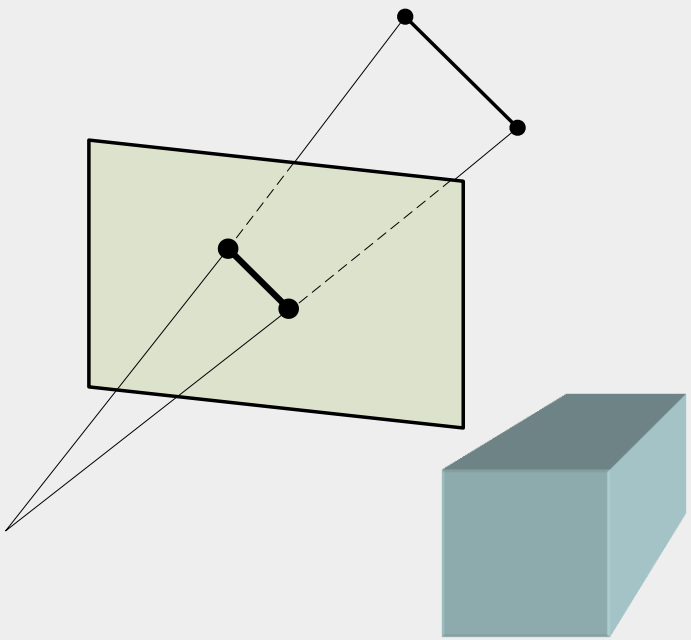
Projecção

- Nesta fase volume de visão é transformado num volume cúbico unitário – **volume canónico de visualização**.
- A transformação é efectuada utilizando um de **dois tipos de projecção**: Paralela ou Perspectiva.

Paralela



Perspectiva



View Transform

Luz

Projecções

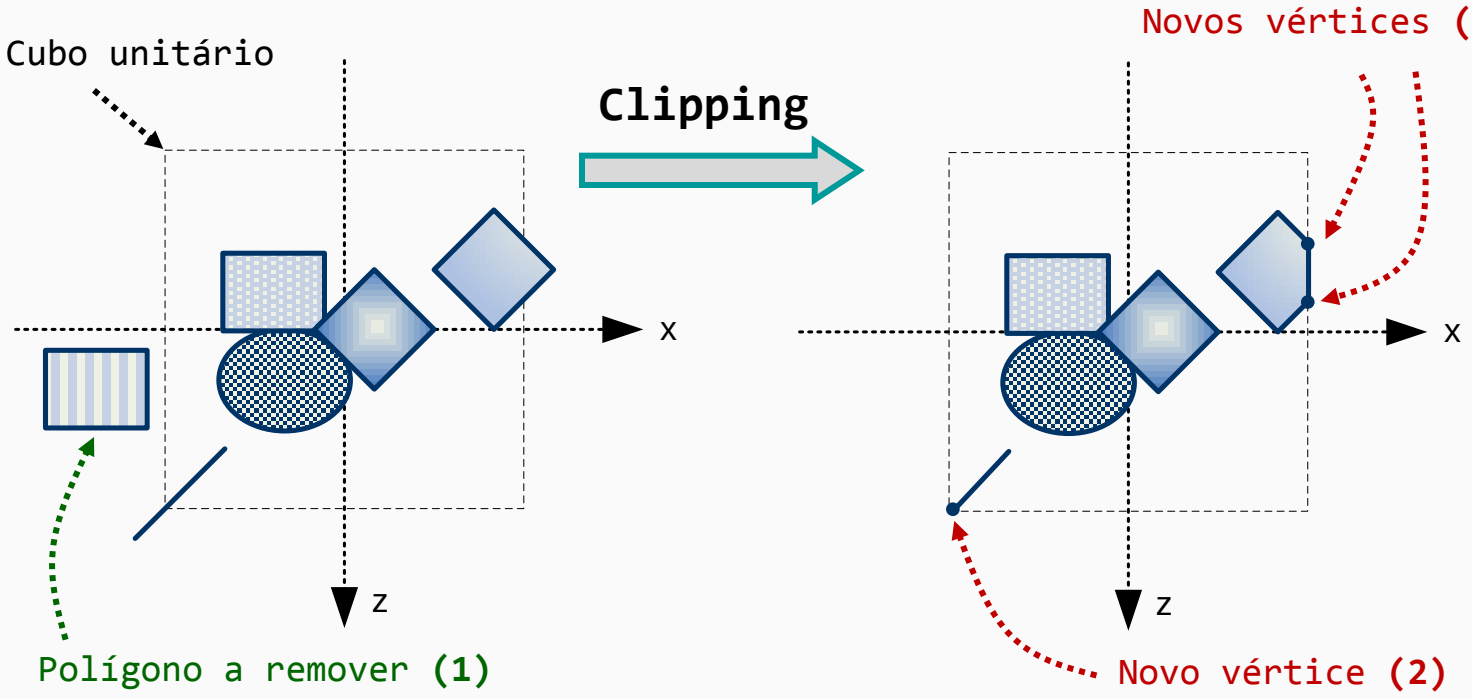
Clipping

Sceen Mapping

Rasterização

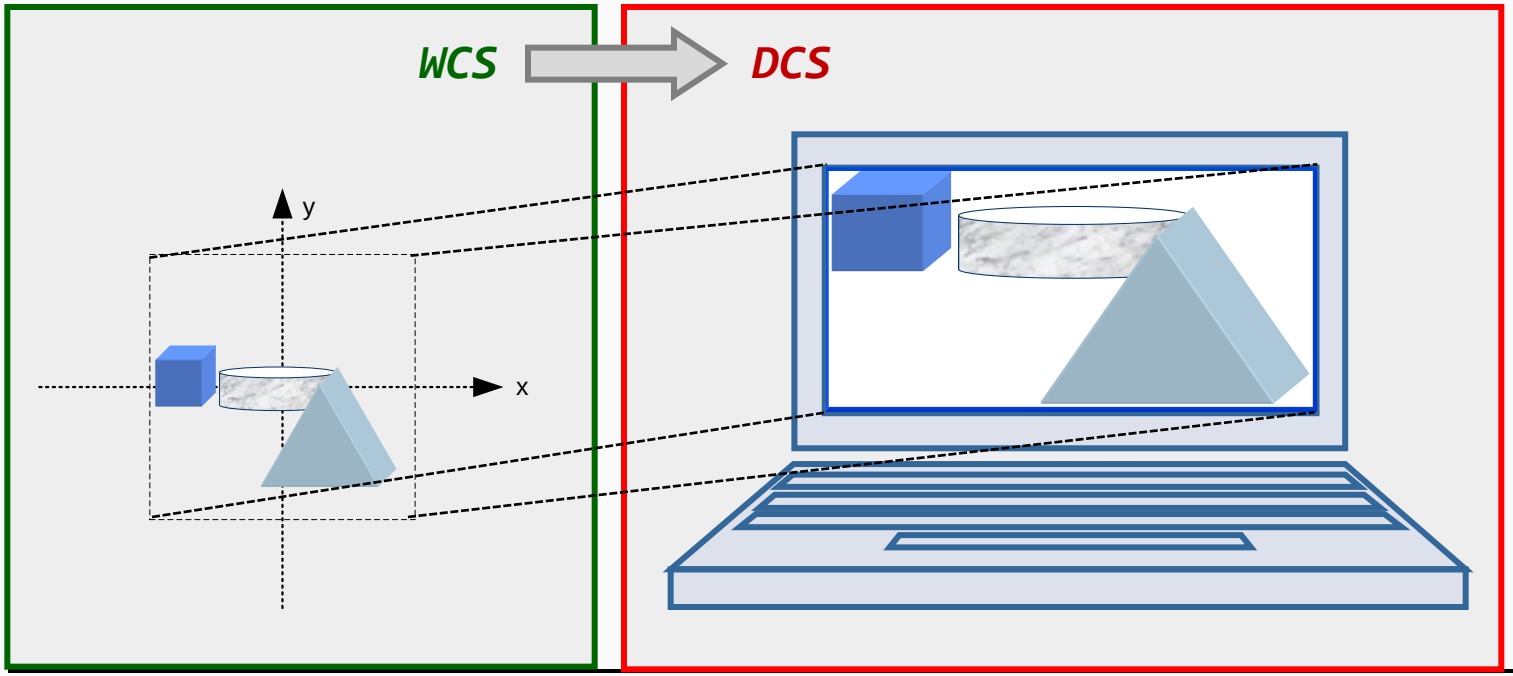
Clipping

- A seguir às projecções, após encontrar o volume canónico de visualização, não é necessário considerar todos os polígonos:
  - **(1) Removem-se** os que se encontram **totalmente fora**
  - **(2) Recortam-se** os que se encontram **parcialmente dentro**, criando **novos vértices** nos polígonos existentes



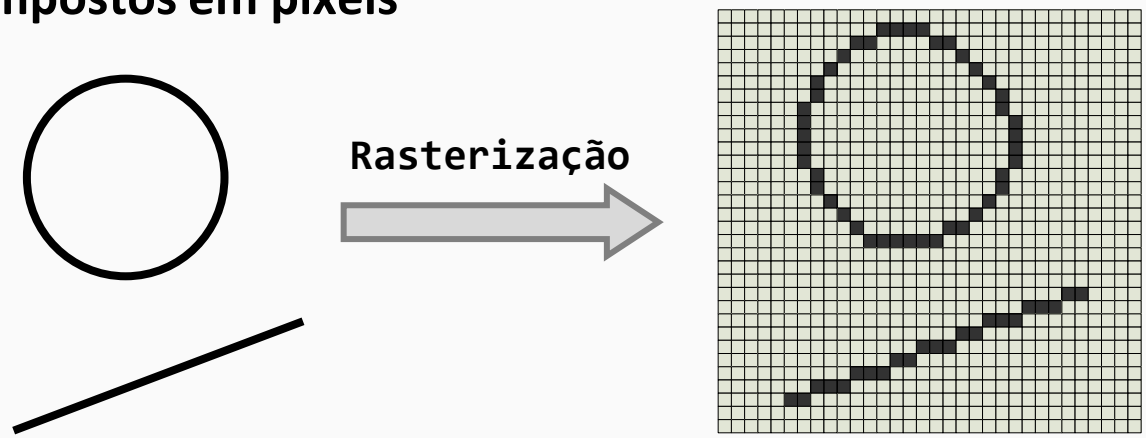
Screen Mapping

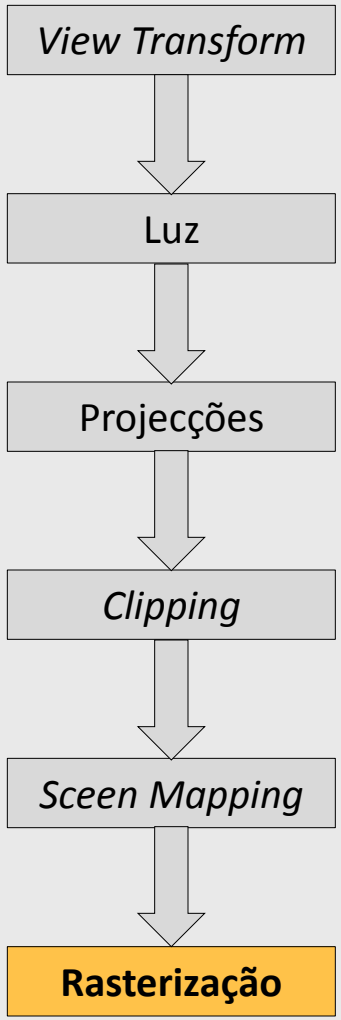
- Até esta fase, toda a cena estava em coordenadas do mundo (*World Coordinate System* - ***WCS***)
- **É necessário efectuar um nova mudança de coordenadas**, desta vez para as coordenadas do dispositivo (*Device Coordinate System* – ***DCS***)



Rasterização

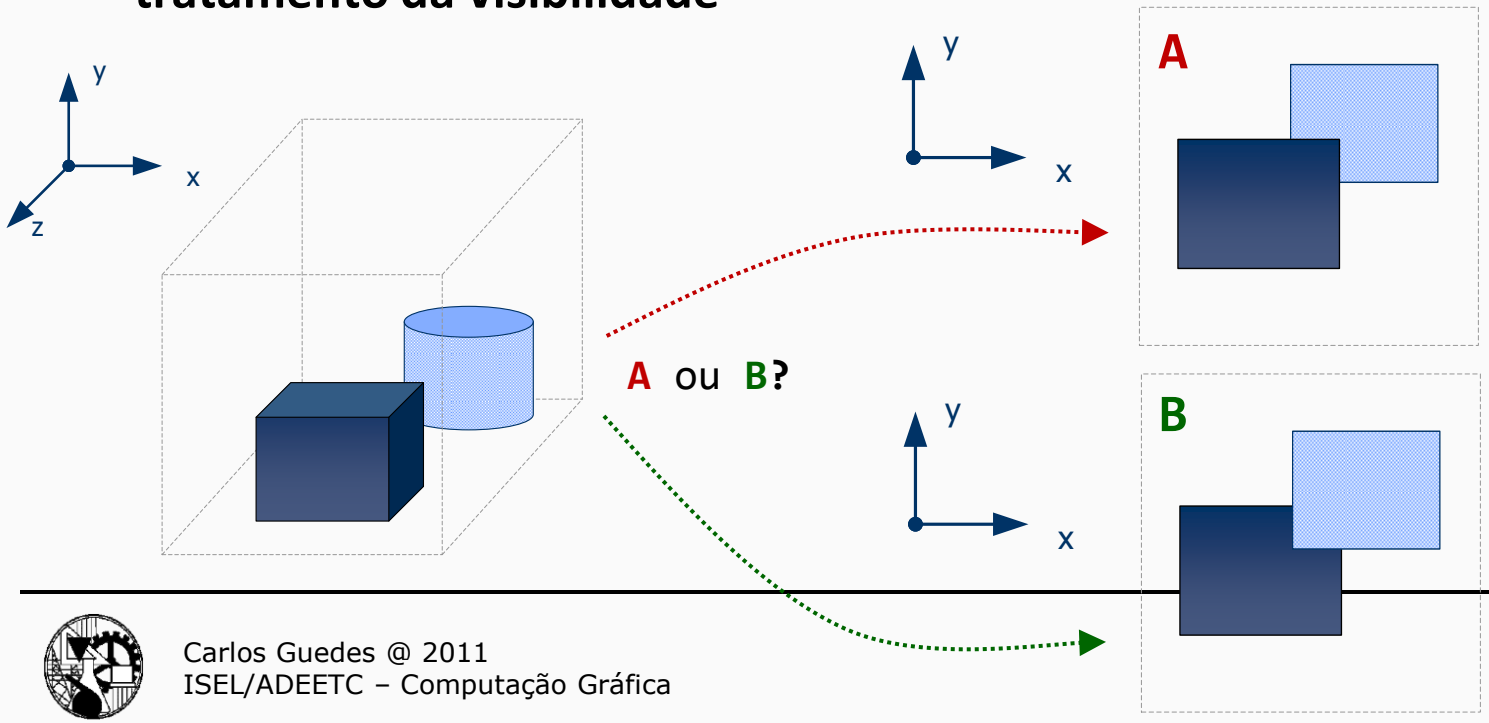
- Esta fase representa **uma alteração nos objectos alvo** das manipulações
- Deixa-se de manipular polígonos, **passa-se a manipular pixéis**
- O principal objectivo desta fase é determinar a cor correcta de cada *pixel* a colocar na matriz *raster*
- Para isso, todos os polígonos, rectas, etc…, **têm de ser decompostos em pixéis**

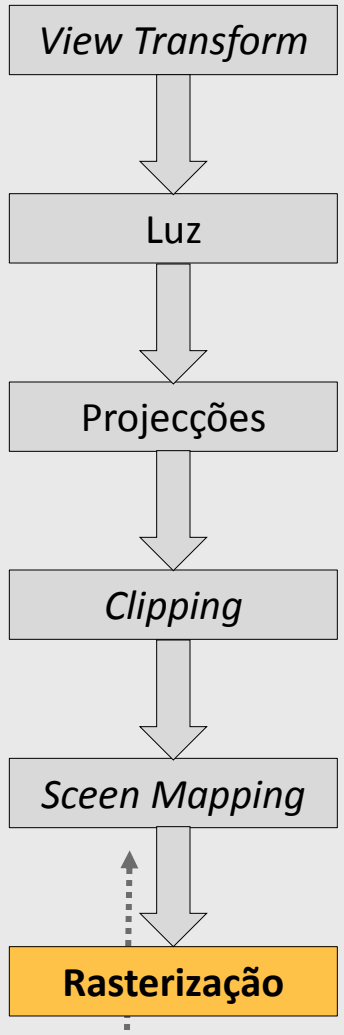




Rasterização

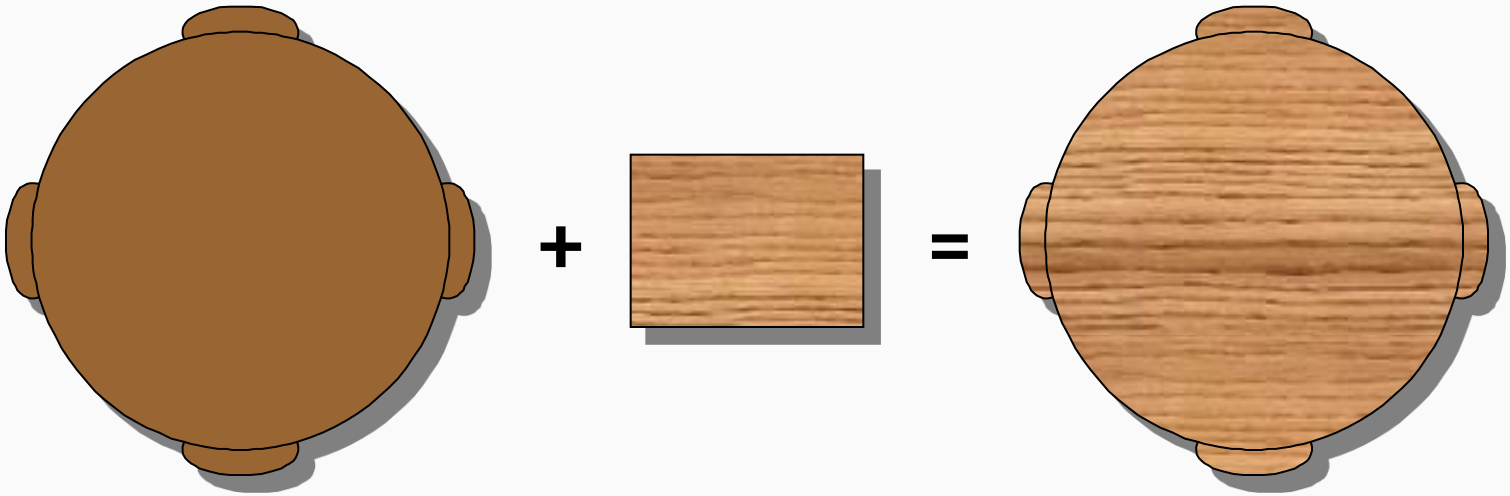
- Note-se que nesta fase, **as cenas são já a duas dimensões**, resultado das projecções efectuadas, num plano 2D
- No entanto a coordenada em **z** não foi descartada
- Serve para indicar a distância ao observador, e por isso, é essencial para um dos processamentos efectuados nesta fase: **tratamento da visibilidade**





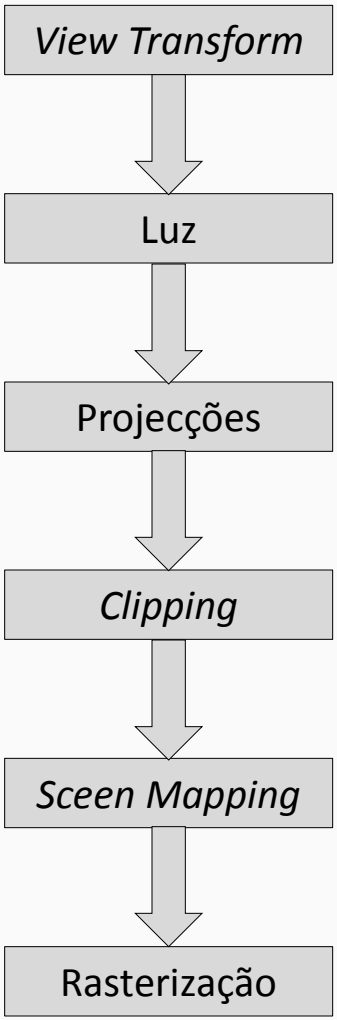
Rasterização

- **Será utilizando o algoritmo *z-buffer***, que de acordo com a coordenada em **z**, determina quem está à frente
- É igualmente nesta fase que **são aplicadas texturas**, com vista ao aumento do realismo



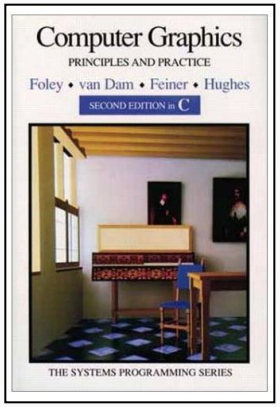
View Transform
Luz
Projecções
Clipping
Sceen Mapping
Rasterização

Resumo do processamento



- **Transformação de vista** para colocar a câmara na origem (*WCS*)
- Dar **realismo** às cenas através de efeitos de luzes e sombreamento de polígonos.
- Determinar o **volume canónico de visão**;
  Projectar os objectos num plano 2D.
- **Remover** os objectos fora do volume de visão;
  Recortar os que se encontram parcialmente contidos nele
- Converter a cena de coordenadas do mundo para as **coordenadas do dispositivo**
- **Conversão** de todas as primitivas para pixéis; Determinar a cor de cada pixel e **visibilidade** dos objectos; Aplicar **texturas**.

Referências



Computer Graphics: Principles and Practice in C,
James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Addison-Wesley Professional; 2nd edition (1995)

Real-Time Rendering,
Tomas Moller, Eric Haines, AK Peters, Ltd.; 2nd edition (July 2002)

Brown Exploratories:
`http://www.cs.brown.edu/exploratories/freeSoftware/home.html`