Transformações Modelagem

Iluminação (*Shading*)

Transformação Câmera*

Recorte

Projeção*

Rasterização

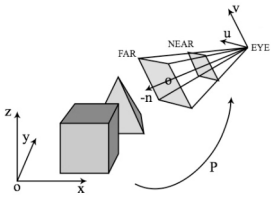
Visibilidade

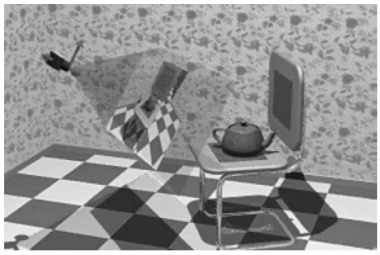
✓Mapeamento de coordenadas de Universo para câmera

✓Escolha da projeção: perspectiva ou ortográfica

*Veremos nesta ordem por motivos didáticos

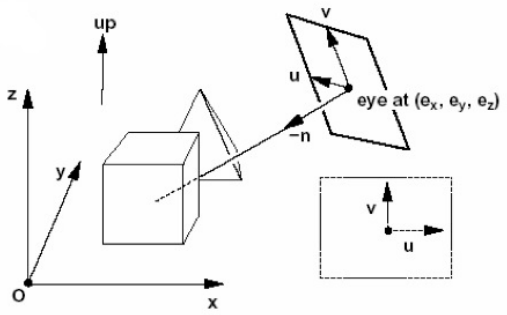
Resumo





Sist. Coordenadas Câmera (SCC)

- Envolve uma translação à origem do sistema e uma mudança de base ortogonal



Marcelo Walter - UFPE

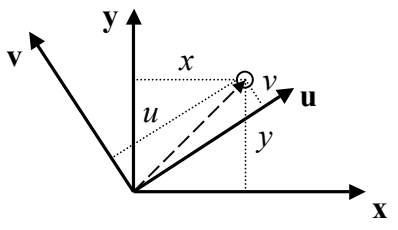
Definindo o Sistema de Coordenadas de Câmera - SCC

- Precisamos especificar:
  - Origem do SCC: ponto qualquer no espaço
  - Direção para onde a câmera está apontando
  - Direção “para cima”: *up vector*
- A partir destas três informações um sistema de coordenadas ortogonal pode ser estabelecido
- Ver por exemplo: *Foley at al. Introduction to Computer Graphics. p. 201*

Definindo SCC

- Transformar as coordenadas, ou seja, dado os sistemas de coordenadas xyz e uvn e um ponto p=(x,y,z) encontrar p=(u,v,n)

Em 2D esquematicamente:



Definindo SCC

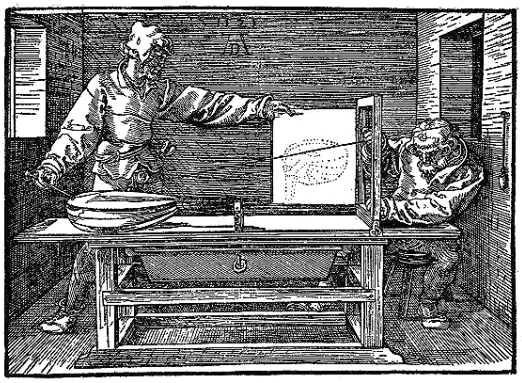
$$\begin{pmatrix} u \\ v \\ n \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Onde u_x = **x . u**, u_y = **y . u,** u_z = **z . u** etc

Produto escalar

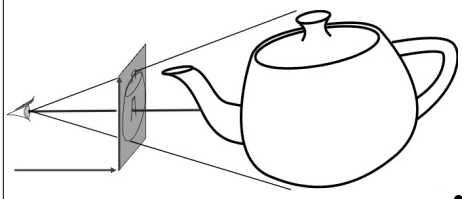
Volume de Visualização

- Finalmente, precisamos definir uma área do espaço que será considerada
- Somente os objetos dentro do volume serão considerados
- Implica definirmos um tipo de **PROJEÇÃO**



Albrecht Durer doing perspective projections in 1525.
http://www.viewmuseum.com/

Projeções



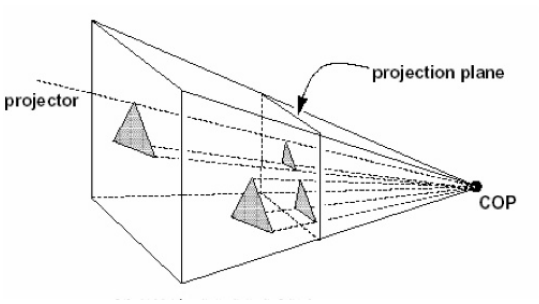
- Genericamente, projeções transformam pontos em um sistema de coordenadas com N dimensões em pontos num sistema com dimensão menor do que N
- Para nós interessa apenas o caso de 3D -> 2D

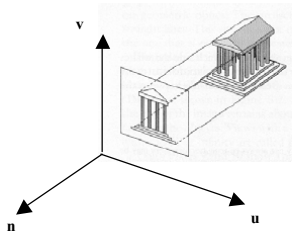
Projeções

- Uso de linhas de projeção, denominadas *raios de projeção*
- Origem em um *centro de projeção* (COP) e passam por cada parte do objeto (no nosso caso vértices) interseccionando um *plano de projeção* onde finalmente tem-se as coordenadas 2D

Tipos de Projeções

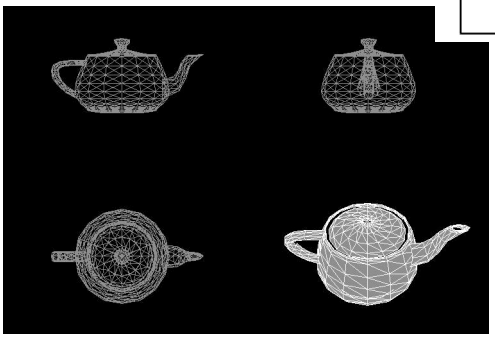
- Em CG 2 tipos: *perspectiva* e *ortográfica*
- Diferença: na projeção ortográfica o COP está no infinito, logo os raios de projeção são paralelos uns aos outros

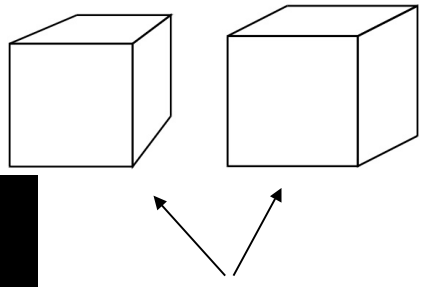




Tipos de Projeções

- Diferença Visual





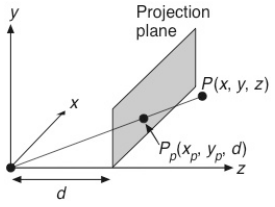
Qual é qual?

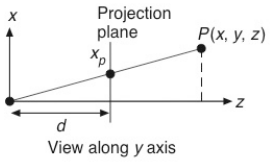
Obtendo Coordenadas de Projeção

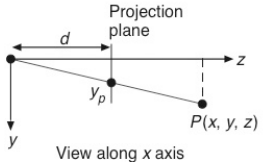
- Precisamos “livrar-nos” de uma dimensão
- Para projeções *ortográficas* é simples:
  - Basta descartarmos a coordenada equivalente a ***n*** no SCC (assumindo o plano de projeção em n = 0)

$$\begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

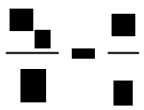
Obtendo Coordenadas de Projeção

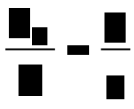




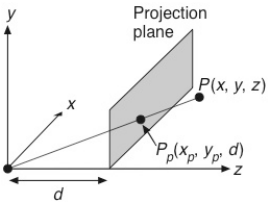


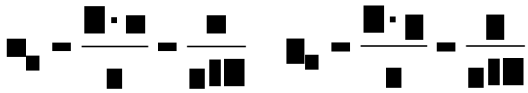
- Para projeções perspectivas?
- Assumindo que o plano de projeção encontra-se perpendicular ao eixo ***n*** do SCC e a uma distância ***d***
- Por semelhança de triângulos temos:



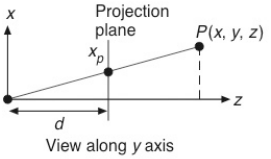


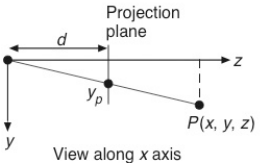
Obtendo Coordenadas de Projeção

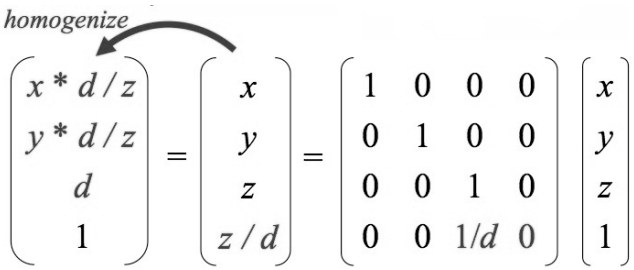




matricialmente:





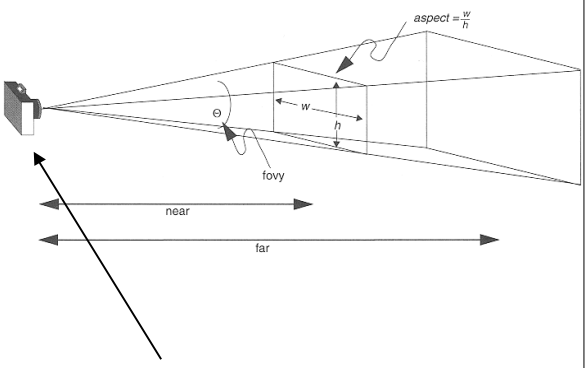


E em OpenGL?

Em OpenGL
Perspectiva

Qual o *up vector* neste caso?

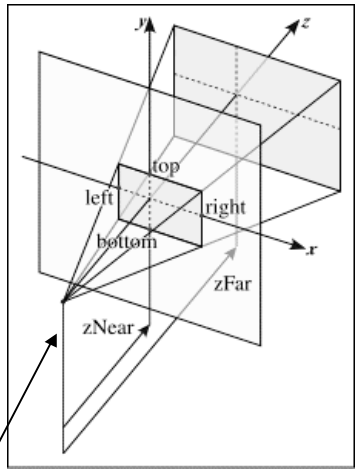
```
void gluPerspective(
  GLdouble fovy,
  GLdouble aspect,
  GLdouble zNear,
  GLdouble zFar
);
```



A posição é definida anteriormente com `glTranslate`

Em OpenGL
Perspectiva

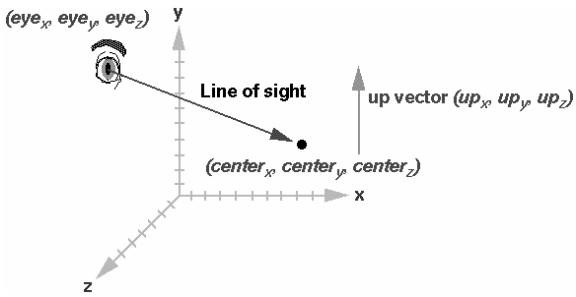
```
void glFrustum(
  GLdouble left,
  GLdouble right,
  GLdouble bottom,
  GLdouble top,
  GLdouble near,
  GLdouble far
);
```



A posição é definida anteriormente com `glTranslate`

Em OpenGL
Perspectiva

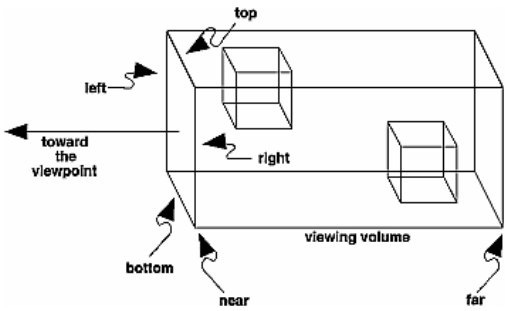
```
void gluLookAt(
  GLdouble eyex,
  GLdouble eyey,
  GLdouble eyez,
  GLdouble centerx,
  GLdouble centery,
  GLdouble centerz,
  GLdouble upx,
  GLdouble upy,
  GLdouble upz,
);
```



A posição é diretamente definida
IMPORTANTE: Esta chamada assume um *znear* e *zfar* (*front and back clipping planes*) já estabelecidos.

Em OpenGL
Ortográfica

```
void glOrtho(
  GLdouble left,
  GLdouble right,
  GLdouble bottom,
  GLdouble top,
  GLdouble near,
  GLdouble far
);
```



Perguntas?