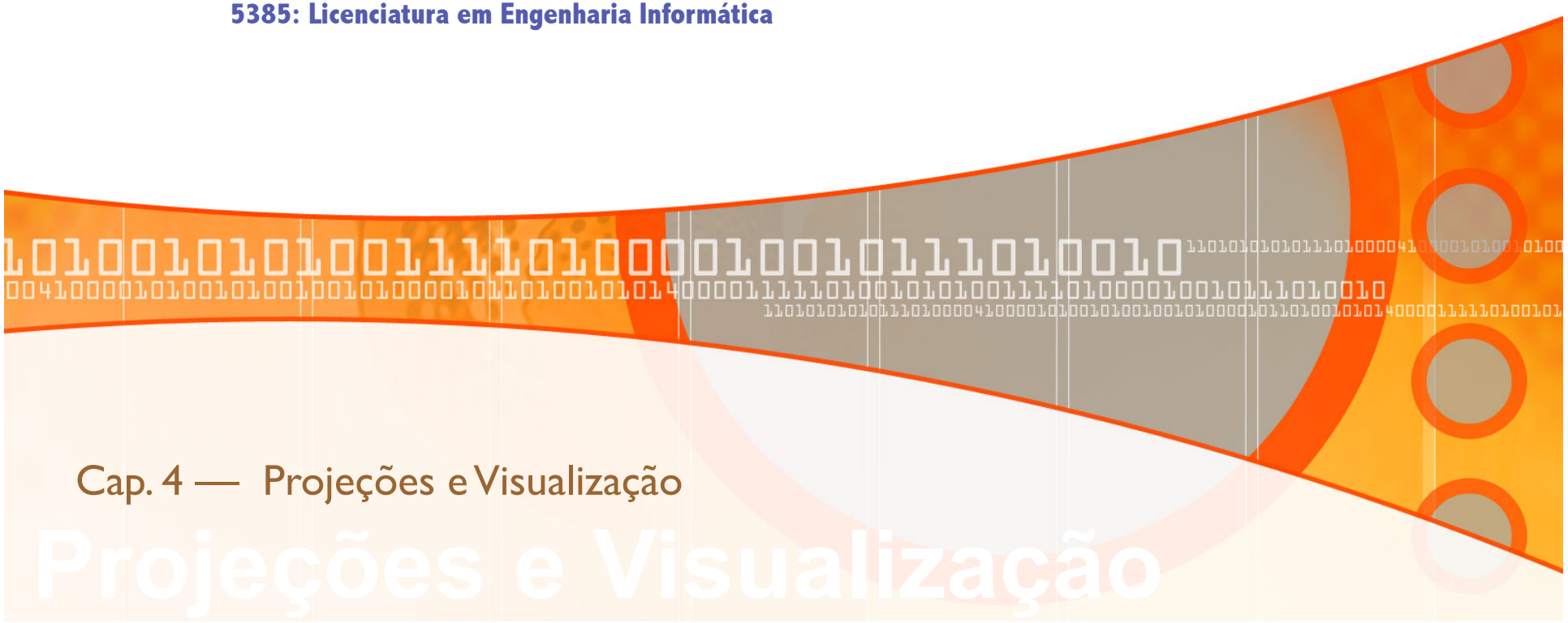
Computação Gráfica

5385: Licenciatura em Engenharia Informática



Cap. 4 — Projeções e Visualização

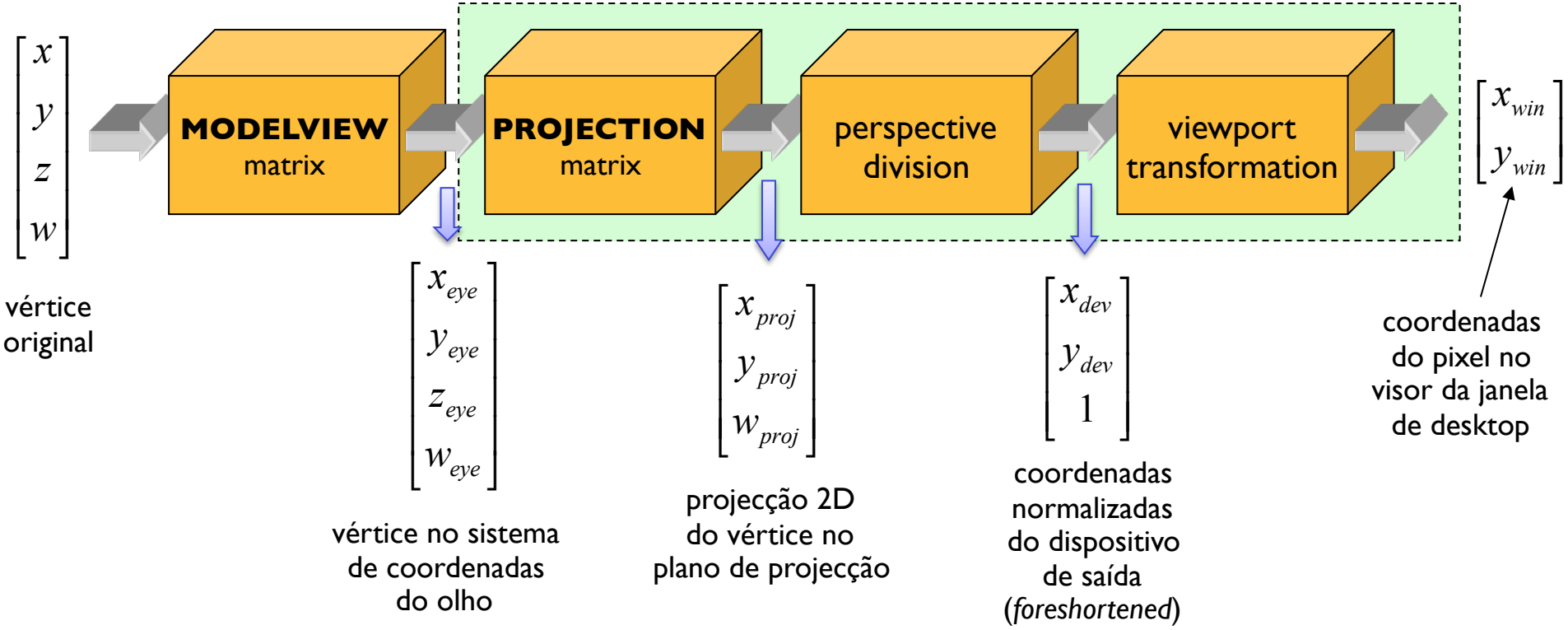
Projeções e Visualização

Sumário

…:

- Pipeline de renderização em OpenGL.
- Modelo de câmara+plano+cena.
- Tipos de câmara: câmara clássica, câmara de lente dupla de Gauss, câmara de renderização foto-realística.
- O processo de renderização de cenas 3D em OpenGL.
- Tipos de projeção: projeção paralela e projeção em perspectiva.
- Projeções em OpenGL.
- Câmara móvel. Câmara móvel em OpenGL.
- Janela de projeção. Transformação janela-visor: revisão. Razão de aspeto: revisão.
- Exemplo em OpenGL. Discussão de ideias e de questões em aberto.

Pipeline de Visualização em OpenGL®



Como se processa cenas 3D no pipeline gráfico?

A criação de uma vista de uma cena requer:

- uma descrição da geometria da cena
- uma câmara ou definição do ponto de vista (ou observador)
- um plano de projeção

Posição/direção da câmara OpenGL por omissão:

- encontra-se na origem e direcionada no sentido do eixo z negativo
- a definição da câmara permite a projeção da cena 3D (geometria) num plano para efeitos de saída gráfica.

Esta projeção pode ser feita de várias maneiras:

- ortogonal (paralelismo das linhas é preservado)
- perspectiva: 1-ponto, 2-pontos ou 3-pontos
- ortogonal oblíqua

Tipos de câmara

Antes de gerar uma imagem temos que escolher o observador (ou câmara).

Câmara clássica (ou *pinhole camera*)

- É o modelo de câmara mais usado; também em OpenGL
- profundidade de campo infinita (infinite depth of field): tudo é focado

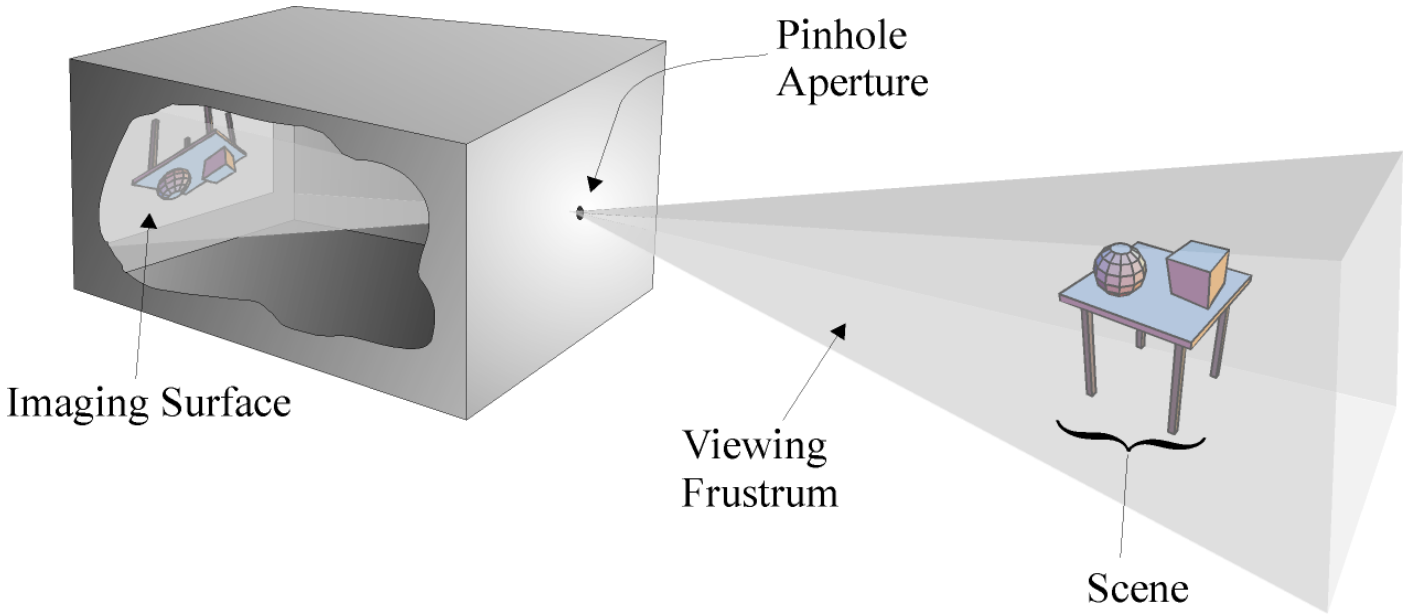
Câmara de lente dupla de Gauss

- modelo de câmara implementado na Princeton University (1995)
- lente dupla de Gauss éusada por muitas câmaras profissionais
- modela a profundidade de campo e óptica não-linear (incluindo lens flare)

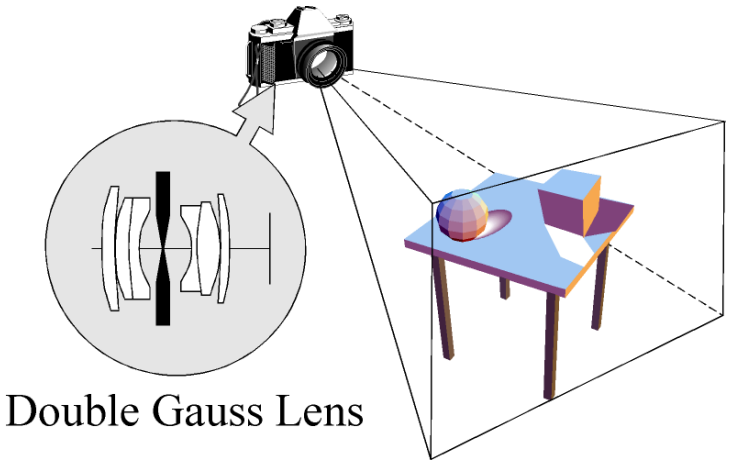
Câmara de renderização foto-realística

- emprega muitas vezes o modelo físico do olho humano para renderizar imagens
- modela a resposta dos olhos face aos níveis de brilho e cor
- modela a óptica interna do próprio olho (difracção pelas fibras da lente etc.)

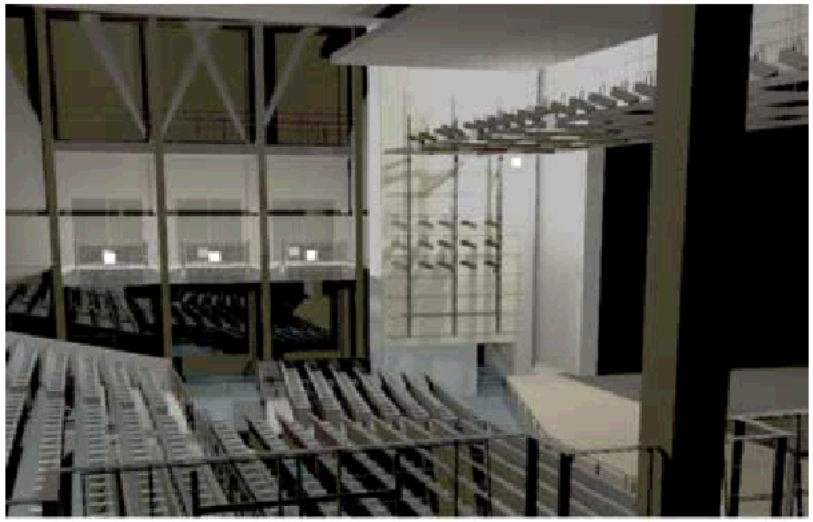
Câmara clássica

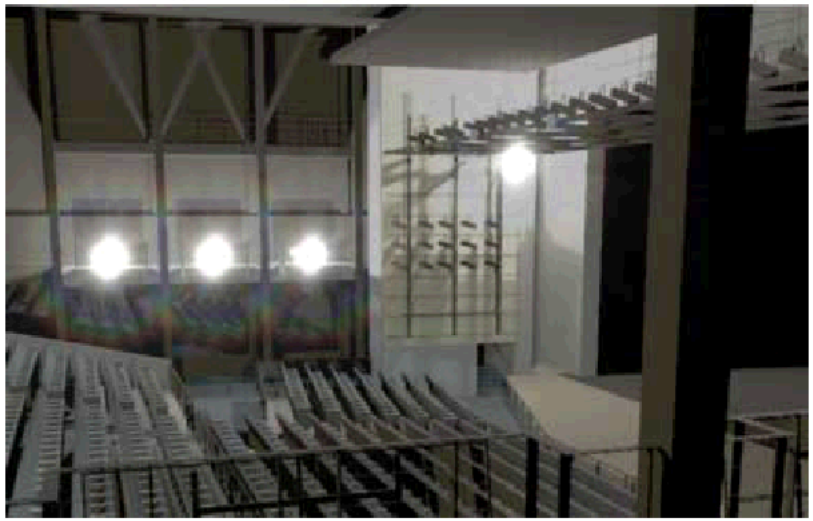


Câmara de lente dupla de Gauss



Câmara foto-realística





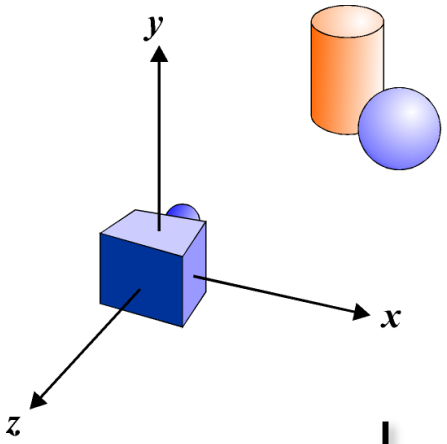
Glare & Difracção

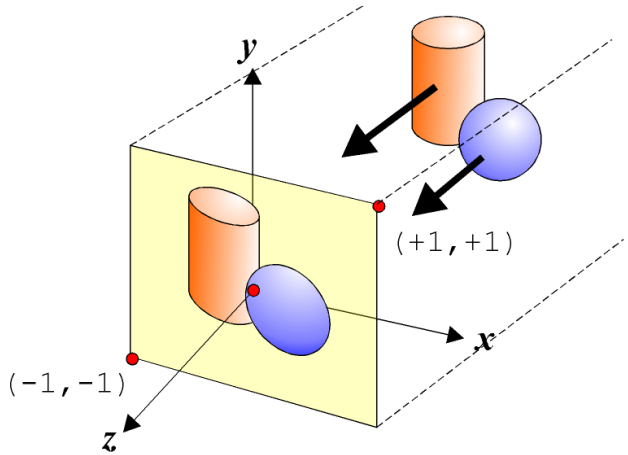
Adaptação

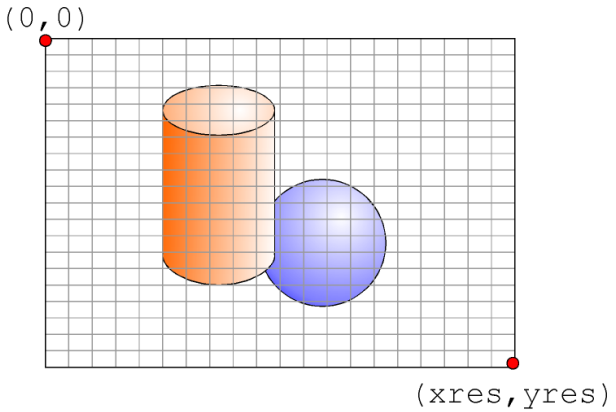
Renderização de cenas 3D em OpenGL®

Veja-se slide 3 para comparação!

sistema de coordenadas da câmara → sistema de coordenadas da janela 2D → sistema de coordenadas do dispositivo/ecrã







glMatrixMode(GL_MODELVIEW)
...

glMatrixMode(GL_PROJECTION)
...

glViewport(0,0,xres,yres)

Renderização de cenas 3D em OpenGL®: da geometria à imagem

http://www.songho.ca/opengl/gl_transform.html

GL_MODELVIEW

- É o produto da matriz de modelação (referente ao sistema de coordenadas do domínio da cena) com a matriz de visualização (referente ao sistema de coordenadas do observador/ câmara).
- Serve para mudar do sistema de coordenadas da cena 3D para o sistema de coordenadas do observador/câmara.
- É empregue para determinar a localização de cada vértice da cena 3D no sistema de coordenadas da câmara.

GL_PROJECTION

- Depois, aplicamos a matriz de projecção definida por GL_PROJECTION para mapear coordenadas da câmara para as coordenadas do plano de projecção ou, mais especificamente, da janela 2D ali definida.

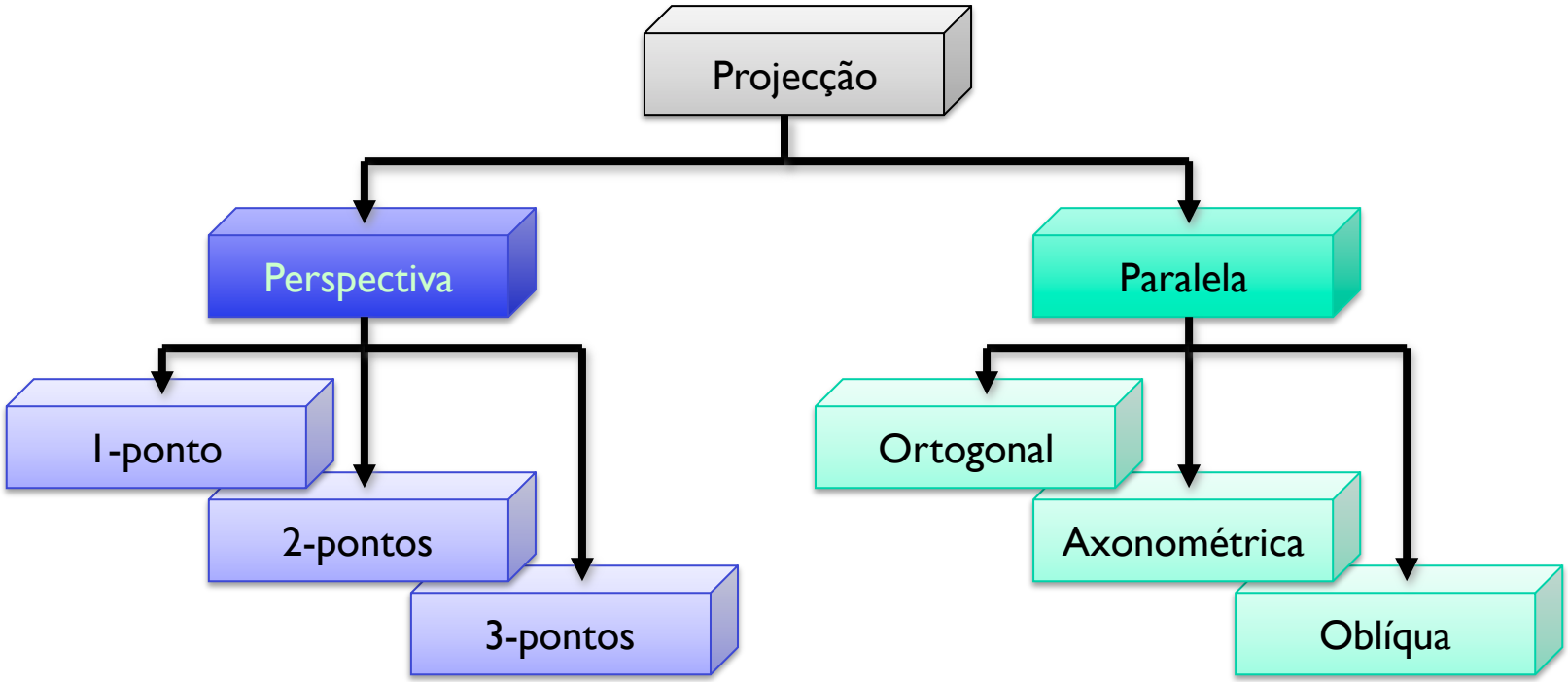
Rasterização no Visor (glViewport)

- Finalmente, estas coordenadas 2D são mapeadas para as coordenadas do dispositivo de saída através da utilização da definição dum visor (viewport) na janela de interface (dado por glViewport()).

Tipos de projeção 3D→2D

Tipo de projeção depende de 2 factores:

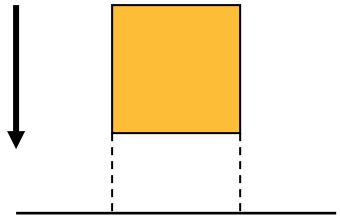
- Posição do observador (que determina a direção das projectantes or raios visuais)
- Localização e orientação do plano de projeção (janela de visualização)



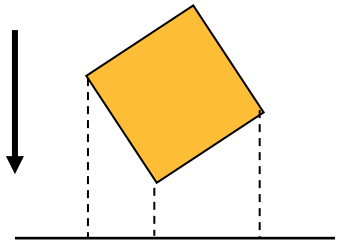
Projeções paralelas

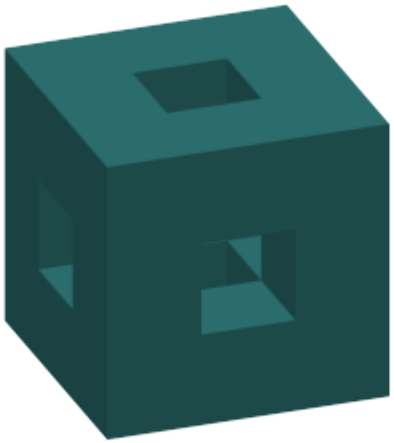
- *O observador no infinito.*
- *As projectantes são paralelas.*

ortogonal

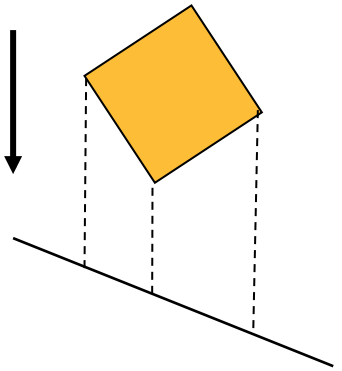


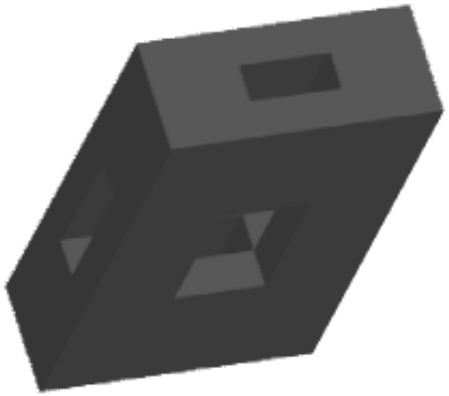
axonométrica





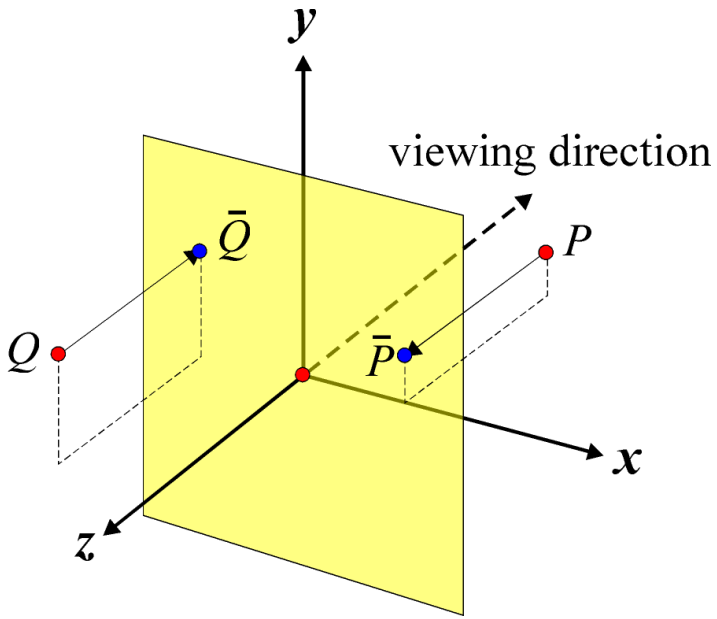
oblíqua





Matriz de projeção paralela ortogonal

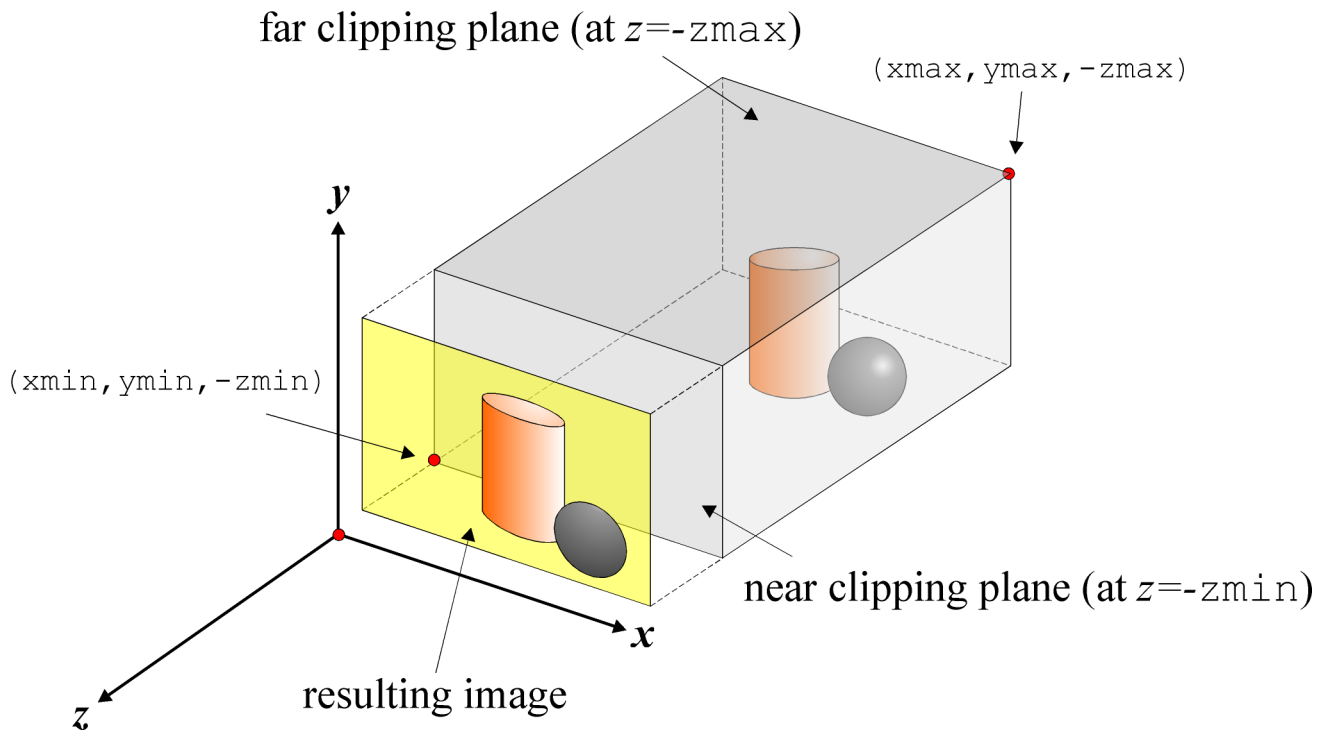
- *A mais simples de todas as projeções: as projetantes são perpendiculares ao plano de projeção.*
- *Normalmente, o plano de projeção está alinhado com os eixos (muitas das vezes em z=0).*
- *As projeções paralelas ortogonais também são designadas por vistas (em desenho técnico).*



$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \Rightarrow \overline{P} = \mathbf{M}P \text{ where } \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

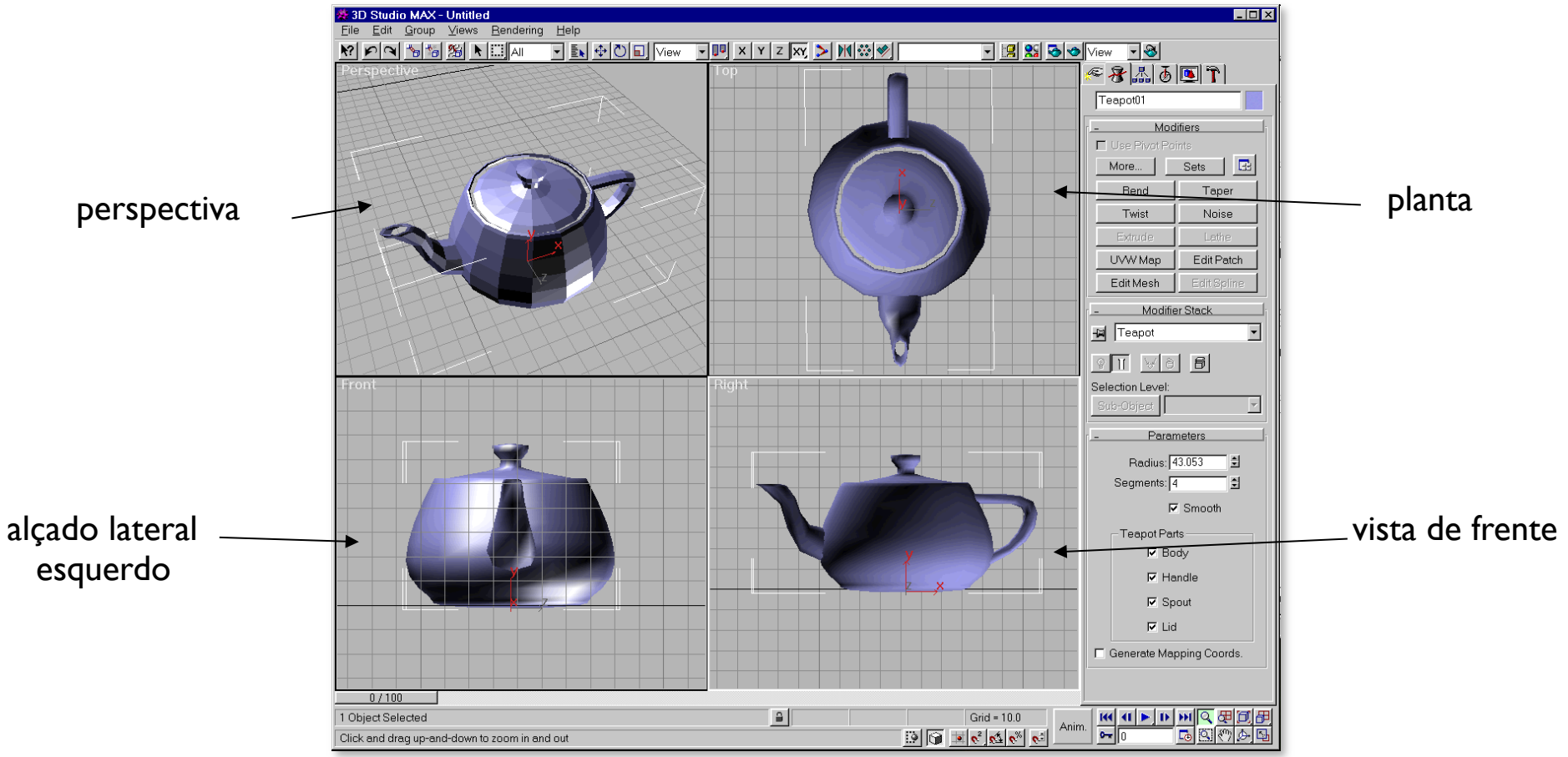
Projeções paralelas ortogonais em OpenGL®

glOrtho(xmin, xmax, ymin, ymax, zmin, zmax**);**



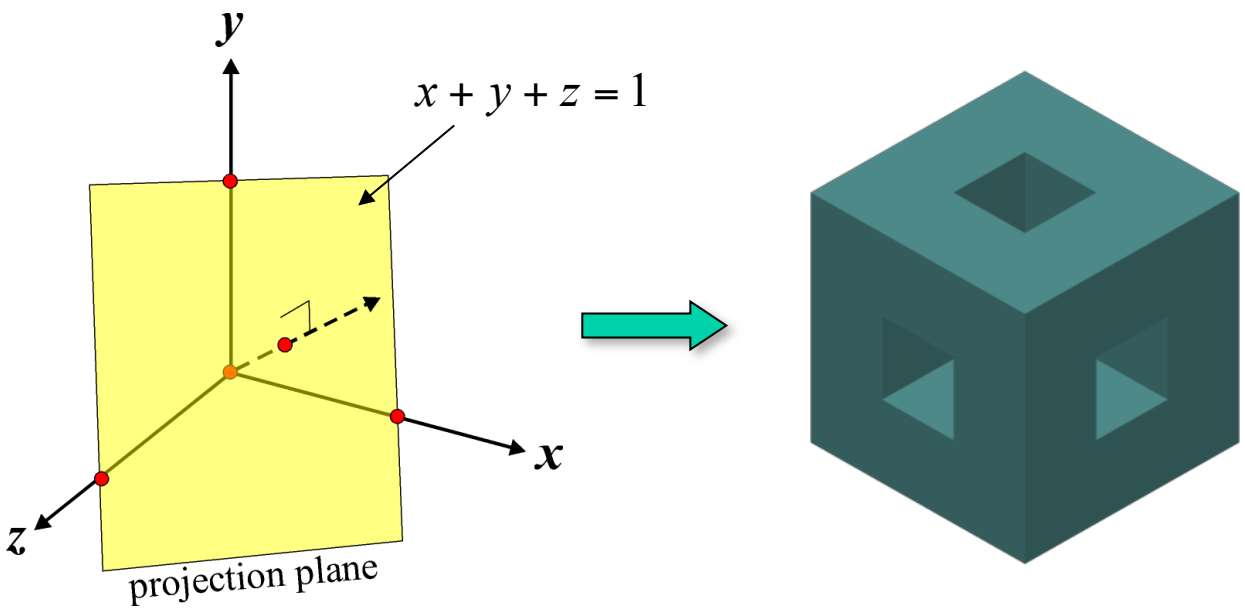
Várias projeções em vários visores: *exemplo*

- *Obtêm-se através do reposicionamento da câmara.*
- *Em alternativa, obtêm-se através do reposicionamento do/a objecto/cena.*



Projeções paralelas axonométricas: isométrica, dimétrica e cavaleira

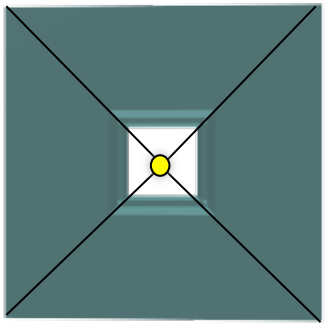
- *Se o objecto está alinhado com os eixos, o resultado é uma projecção **ortogonal**;*
- *Caso contrário, é uma projecção **axonométrica**.*
- *Se o plano de projecção intersecta os eixos XYZ à mesma distância relativamente à origem, o resultado é uma projecção **isométrica**.*



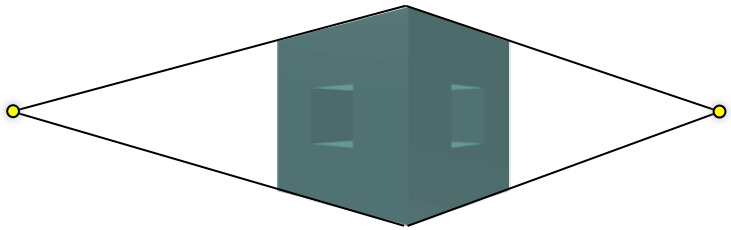
Projeções em perspectiva

- *O observador está a uma distância finita do/a objeto/cena.*
- *As projetantes não são paralelas e convergem para um ou mais pontos (observadores).*

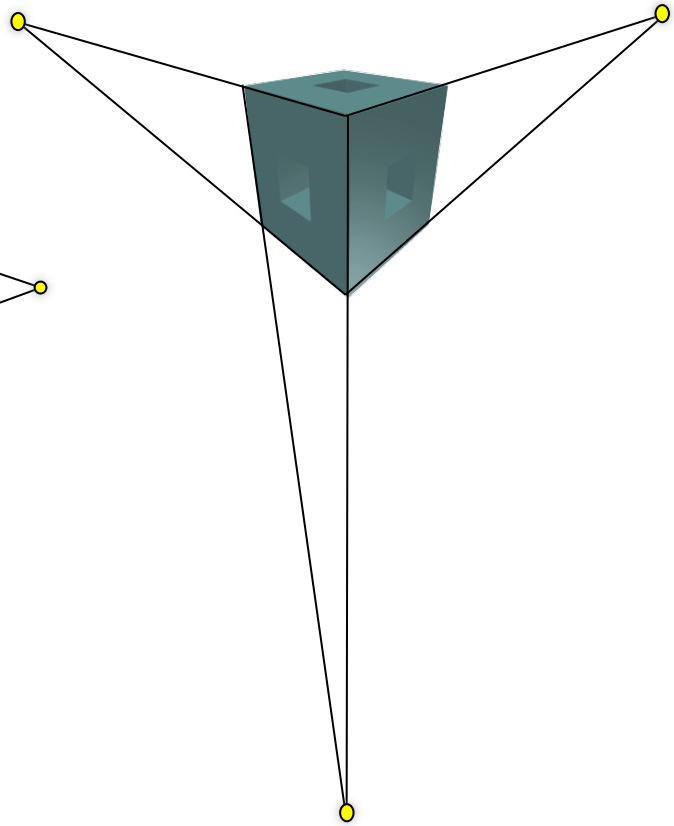
perspectiva com 1-ponto



perspectiva com 2-pontos



perspectiva com 3-pontos

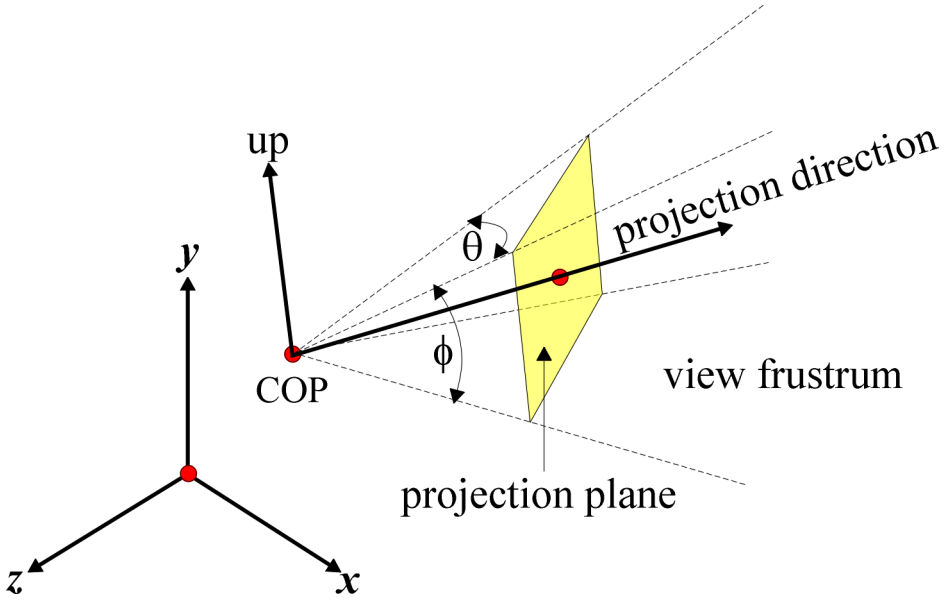


Projeções em perspectiva com um observador

- *O observador está a uma distância finita do/a objeto/cena.*
- *As projetantes or raios visuais convergem para um ponto (observador), designado por COP.*
- *Em OpenGL, só se usa um observador.*

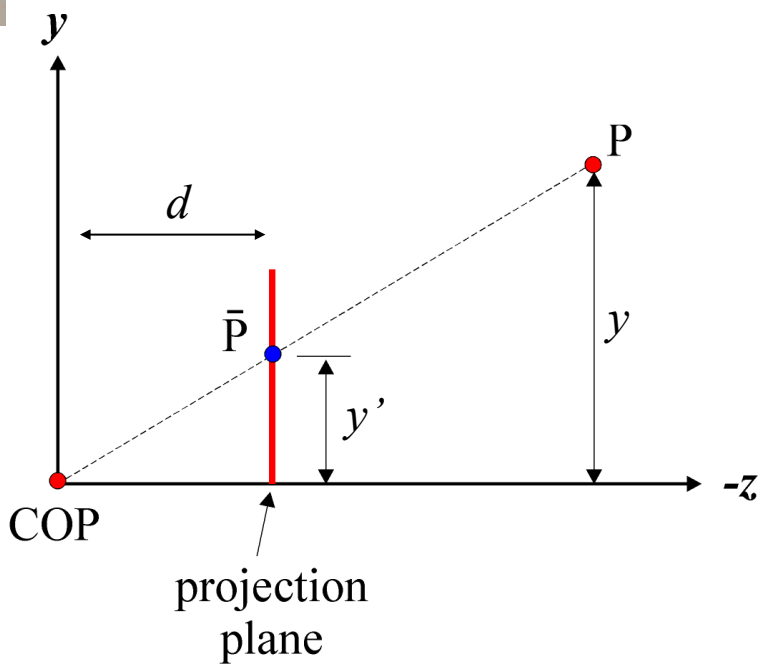
Parâmetros de projeção:

- centro de projeção (COP)
- campo de vista (θ,ϕ)
- direção de projeção
- direção *up* do eixo da câmara ou do observador



Matriz de projeção em perspetiva com um observador

– *Considere uma projeção em perspetiva com o ponto de vista na origem e a direção de observação orientada ao longo do eixo -**z**, com o plano de projeção localizado em z = -d.*

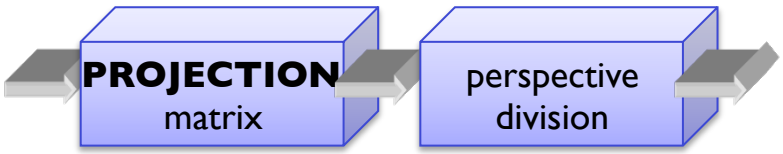


$$\begin{cases} x' = \dfrac{-xd}{z} = \dfrac{x}{-z/d} \\ y' = \dfrac{y}{-z/d} \\ z' = -d \end{cases}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{x}{-z/d} \\ \dfrac{y}{-z/d} \\ -d \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} x \\ y \\ z \\ -z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\frac{x}{-z} = \frac{x'}{d} \Rightarrow x' = \frac{x}{-z/d}$$

$$\frac{y}{-z} = \frac{y'}{d} \Rightarrow y' = \frac{y}{-z/d}$$

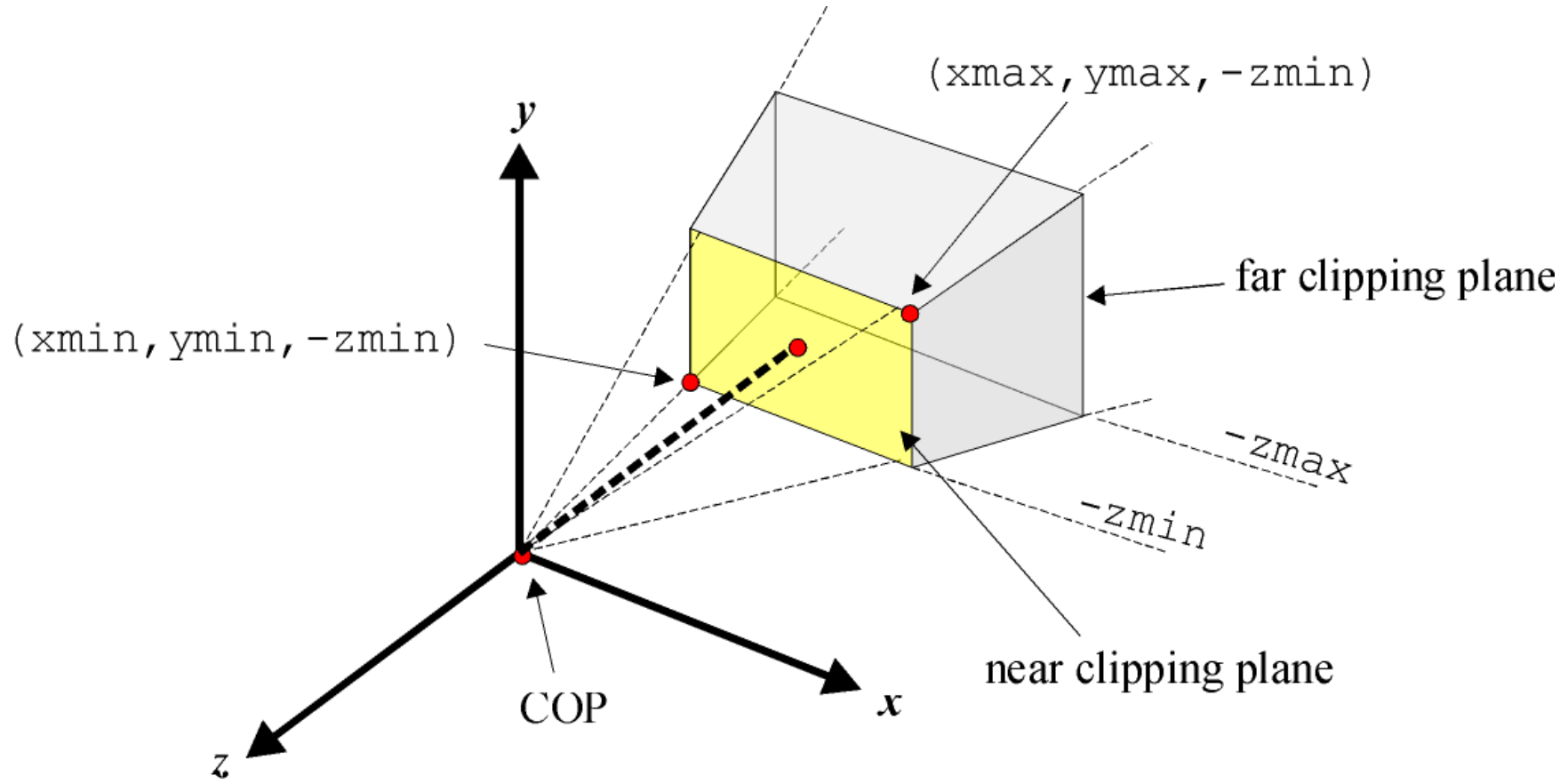


Projeção em perspetiva com um observador em OpenGL®

frustum = pirâmide truncada do campo de vista

– *Por via de **glFrustrum**:*

glFrustum(xmin, xmax, ymin, ymax, zmin, zmax**);**



Projeção em perspetiva com um observador em OpenGL® (cont.)

– *Por via de* ***glFrustrum****:*

Especificação de *glFrustrum*:

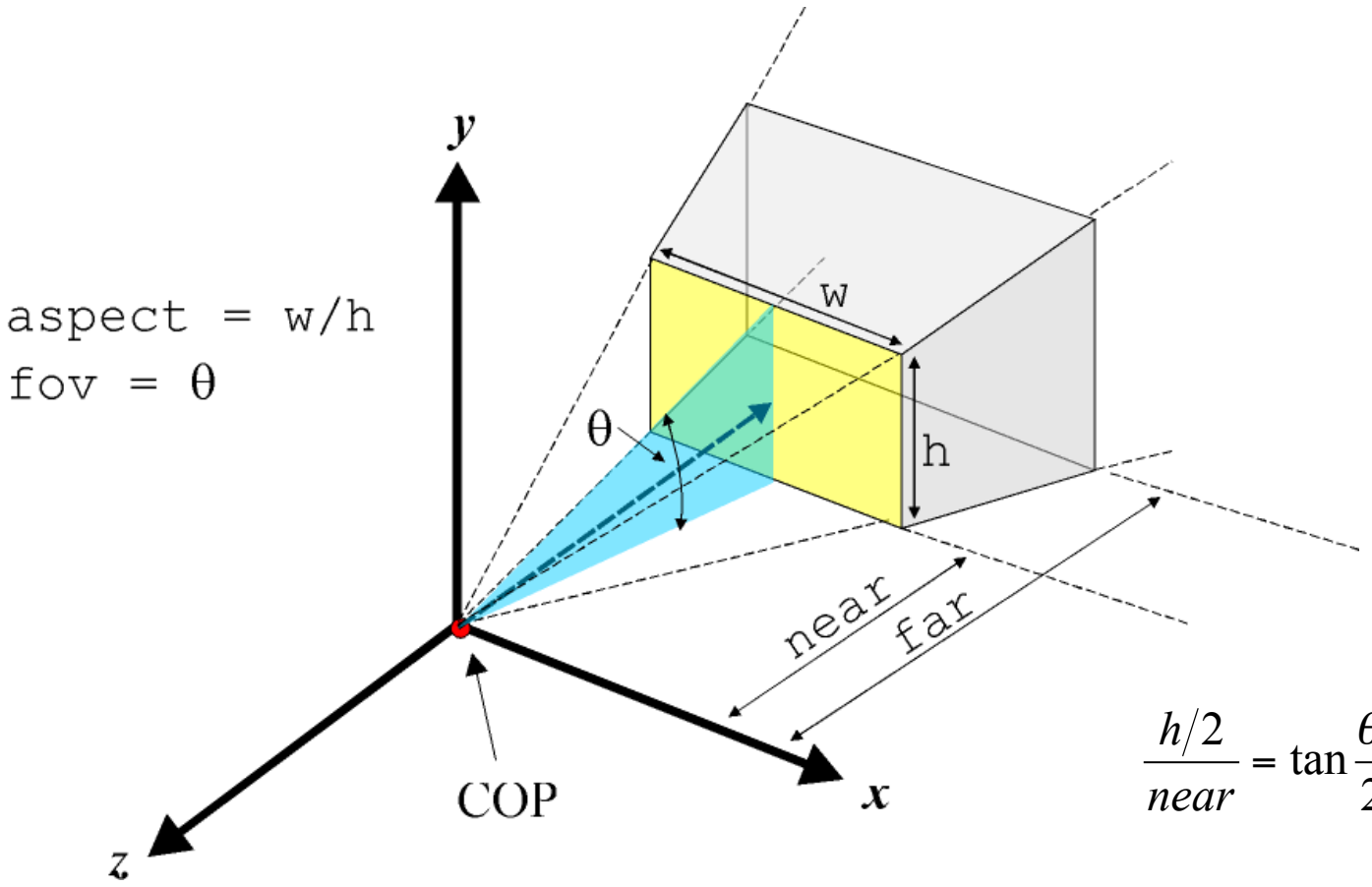
- Todos os pontos na linha definida pelo COP e (xmin,ymin,-zmin) são mapeados para o canto *inferior esquerdo* da janela.
- Todos os pontos na linha definida pelo COP e (xmax,ymax,-zmin) são mapeados para o canto *superior direito* da janela.
- zmin e zmax são especificados como distâncias positivas ao longo de -z
- A direção de observação é sempre paralela a –z
- Não é necessário ter um *frustrum simétrico*, mas um *frustrum* não-simétrico introduz *obliquidade* na projeção.
  - Por exemplo, a seguinte instrução em OpenGL especifica um frustrum não-simétrico:

```
glFrustrum(-1.0, 1.0, -1.0, 2.0, 5.0, 50.0);
```

Projeção em perspetiva com um observador em OpenGL® (cont.)

– *Por via de* ***gluPerspective***:

gluPerspective(fov, aspect, near, far**);**



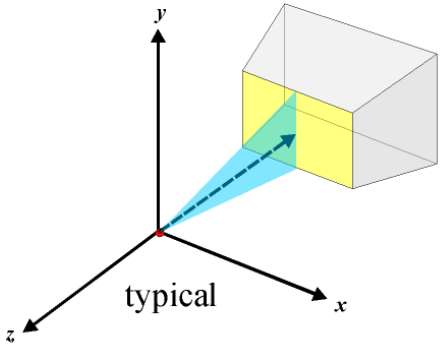
$$\frac{h/2}{near} = \tan\frac{\theta}{2} \Rightarrow h = 2\,near\tan\frac{\theta}{2}$$

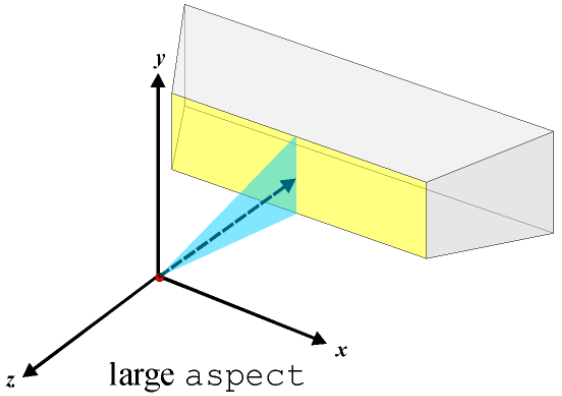
Projeção em perspetiva com um observador em OpenGL® (cont.)

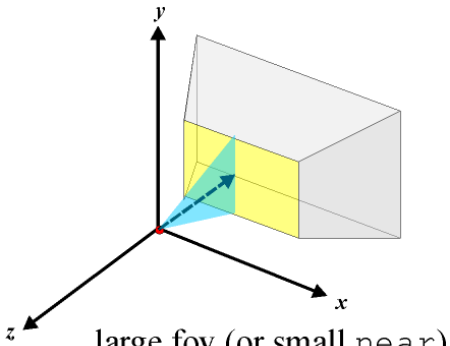
– *Por via de* ***gluPerspective***:

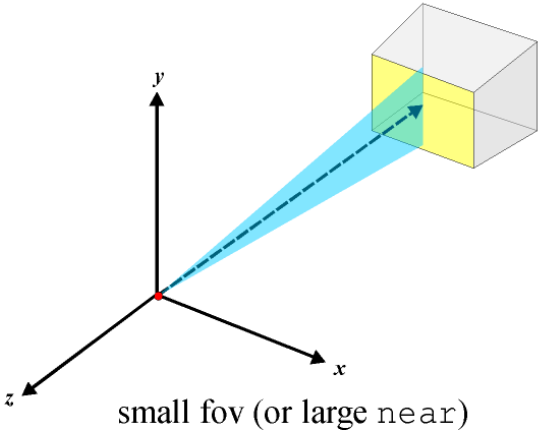
Especificação de *gluPerspective*:

- Só permite a criação de *frustra simétricos*.
- O ponto de vista (do observador) está na origem e a direcção de observação é o eixo –z.
- O ângulo *fov* do *campo de vista* tem de pertencer ao intervalo [0,180].
- *aspect* permite a criação dum *frustrum* com a mesma *razão de aspecto* do visor (*viewport*) por forma a eliminar distorção.









Exemplo:

```
gluPerspective(60, 1.0, 1.0, 50.0);
```

Câmara móvel em 3D

Limitações das projeções com *glFrustum* e *gluPerspective*:

- COP fixo e direcção de projecção (ou observação) fixa

Posicionamento e orientação arbitrários da câmara:

- Para tal, há que manipular a matriz MODELVIEW antes da criação dos modelos. Desta forma, posiciona-se a câmara relativamente aos objectos da cena.

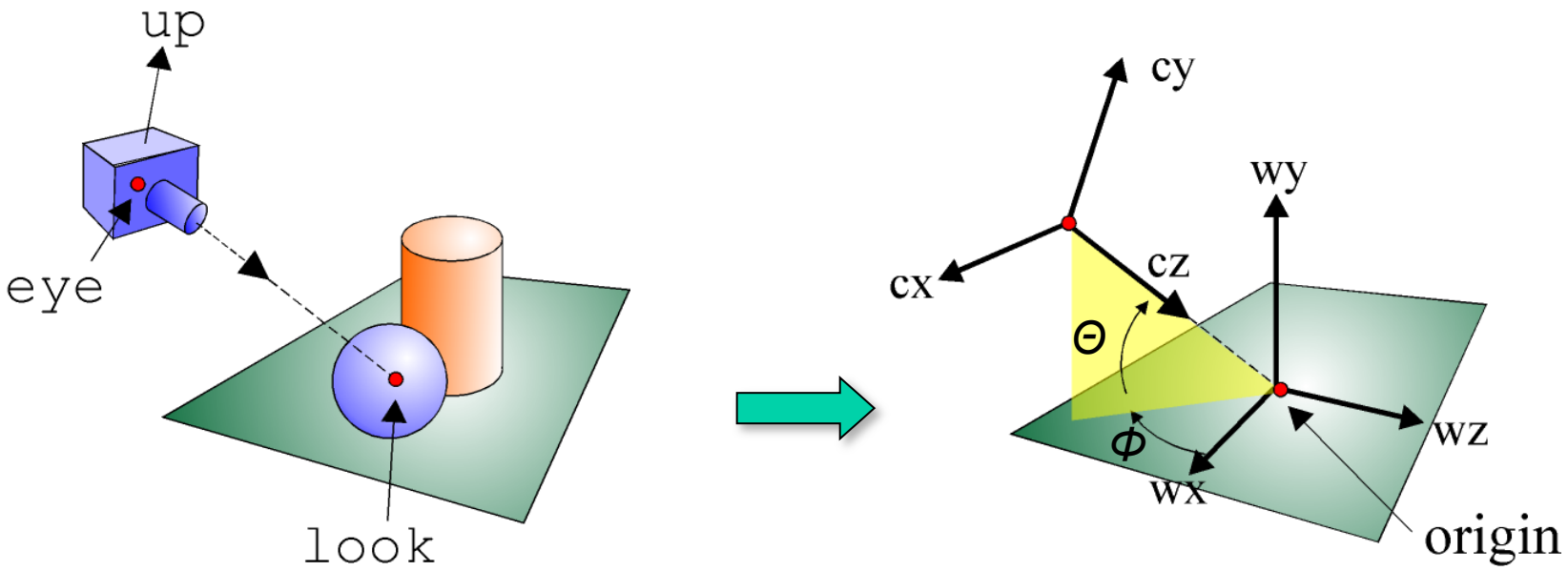
Exemplo:

- Há 2 possibilidades para posicionar a câmara em (10, 2, 10) em relação ao referencial do domínio da cena:
  - mudar o referencial do domínio da cena antes de criar os objectos usando translatef e rotatef: **glTranslatef(-10,-2,-10);**
  - usar gluLookAt para posicionar a câmara relativamente ao referencial do domínio da cena: **gluLookAt(10, 2, 10, ...);**
- Estas duas possibilidades são *equivalentes*.

Câmara móvel em 3D com a utilização de OpenGL®

– *Por via de* ***gluLookAt****:*

gluLookAt(eyex, eyey, eyez, lookx, looky, lookz, upx, upy, upz**);**



equivalente a:

```
glTranslatef(-eyex, -eyey, -eyez);
glRotatef(theta, 1.0, 0.0, 0.0);
glRotatef(phi, 0.0, 1.0, 0.0);
```

Janela de projeção

–A partir do momento que se efectua a projeção da cena 3D na janela do plano de projeção, a renderização processa-se de forma idêntica ao 2D.

Definição:

- A matriz de projecção define a transformação de coordenadas 3D do domínio da cena numa janela 2D que pertence ao plano de projecção.
- As dimensões da janela de projecção são definidas como parâmetros da projecção:
  - **glFrustrum(l,r,b,t,n,f)⇒**

  - **gluPerspective(f,a,n,f)⇒**

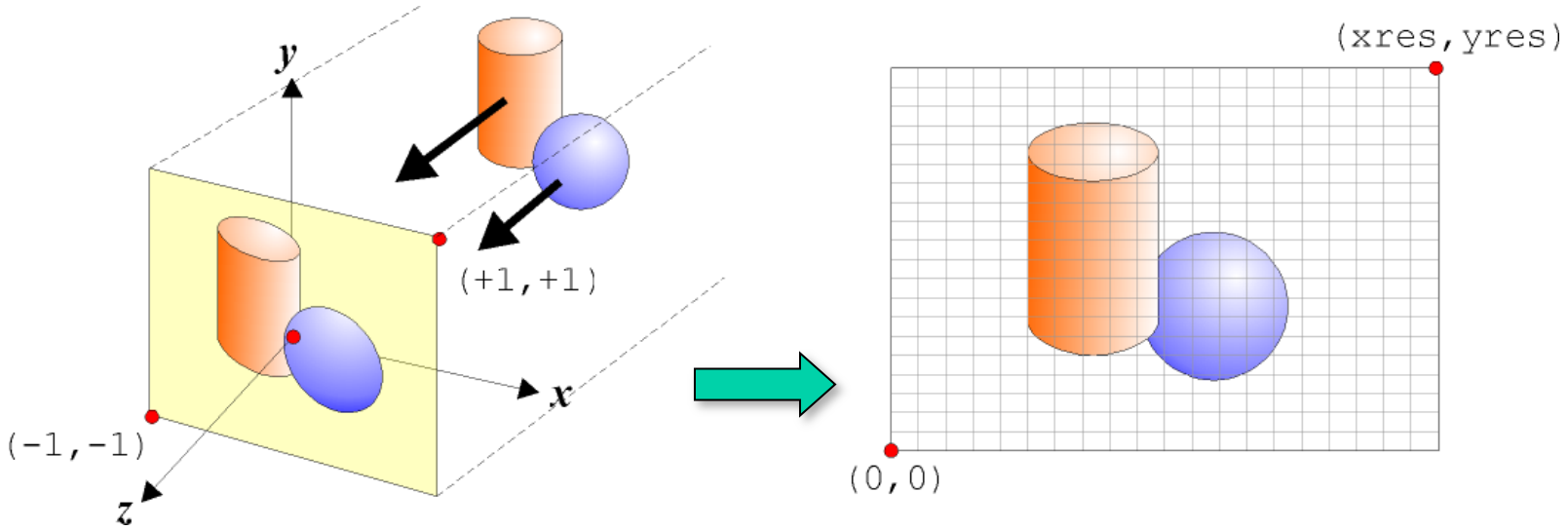
$$h = n.\tan\left(\frac{f}{2}\right)$$

$$w = h.a$$

Transformação janela-visor: *revisão*

– A partir do momento que se efetua a projeção da cena 3D na janela do plano de projeção, a renderização processa-se de forma idêntica ao 2D.

– De facto, é preciso mapear os pontos da janela de projeção para os pixéis do visor da janela da interface, por forma a determinar o pixel associado a cada vértice dos objectos da cena 3D.



coordenadas do dispositivo normalizado

coordenadas do visor da janela da interface

Transformação janela-visor: *revisão (cont.)*

Transformação de normalização:

- Após projecção no plano, todos os pontos (x_p,y_p) da janela de projeção são transformados em coordenadas (x_n,y_n) do *dispositivo normalizado*: [-1,-1]× [+1,+1].

$$x_n = 2\left(\frac{x_p - x_{\min}}{x_{\max} - x_{\min}}\right) - 1$$

$$y_n = 2\left(\frac{y_p - y_{\min}}{y_{\max} - y_{\min}}\right) - 1$$

Transformação de rasterização:

- Depois, mapeia-se a informação 2D do dispositivo normalizado para um ou mais visores:

$$x_v = (x_n + 1)\left(\frac{width}{2}\right) + left$$

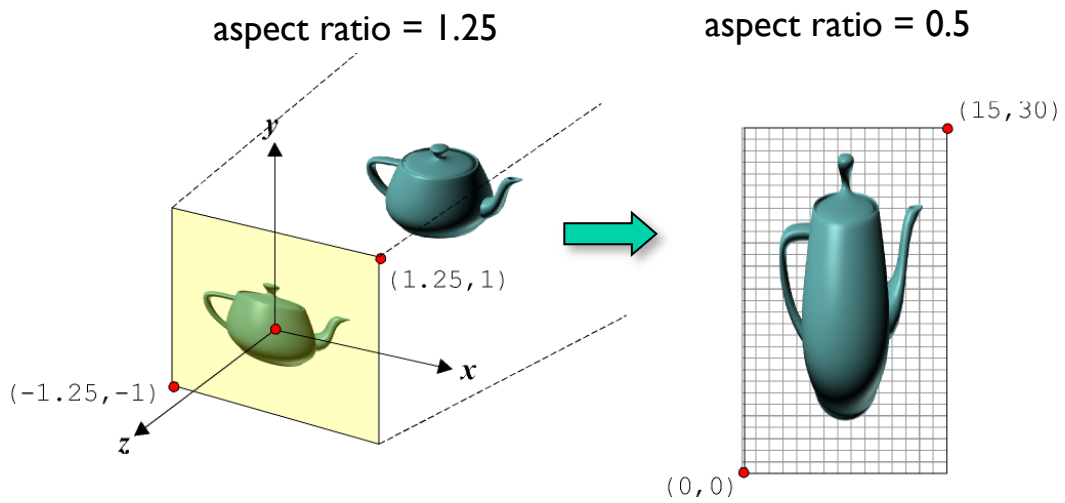
$$y_v = (y_n + 1)\left(\frac{height}{2}\right) + bottom$$

Evento *resize*:

- Normalmente, recria-se a janela após o evento *resize* da janela da interface assegurar o mapeamento correto entre as dimensões do visor e da janela:

```
static void reshape(int width, int height)
{
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(85.0, 1.0, 5, 50);
}
```

Razão de aspeto: *revisão*



Definição:

- A razão de aspect (*aspect ratio*) define a relação entre a largura (width) e a altura (height) da imagem.

Especificação em OpenGL:

- A razão de aspecto da janela de projecção é explicitamente fornecida através da função *gluPerspective*.

Como evitar distorção?:

- A razão de aspecto do visor deve ser a mesma para evitar distorção de imagem:

EXEMPLO EM OPENGL

Exemplo:
bule de chá com quatro vistas

– *Veja-se o programa completo na página web da disciplina* (teapot.zip).

```
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glPushMatrix();

  // top left: top view
  glViewport(0, Height/2, Width/2, Height/2);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 50.0);
  gluLookAt(0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  teapot();
. . .
  // bottom right: rotating perspective view
  glViewport(Width/2, 0, Width/2,  Height/2);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(70.0, 1.0, 1, 50);
  gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glRotatef(45.0, 1.0, 0.0, 0.0);
  teapot();

  glPopMatrix();

  glutSwapBuffers();
}
```

Sumário:

…:

- Pipeline de renderização em OpenGL.
- Modelo de câmara+plano+cena.
- Tipos de câmara: câmara clássica, câmara de lente dupla de Gauss, câmara de renderização foto-realística.
- O processo de renderização de cenas 3D em OpenGL.
- Tipos de projeção: projeção paralela e projeção em perspectiva.
- Projeções em OpenGL.
- Câmara móvel. Câmara móvel em OpenGL.
- Janela de projeção. Transformação janela-visor: revisão. Razão de aspeto: revisão.
- Exemplo em OpenGL. Discussão de ideias e de questões em aberto.