

# Computação Gráfica

## Unidade 4

prof. Dalton S. dos Reis  
dalton.reis@gmail.com

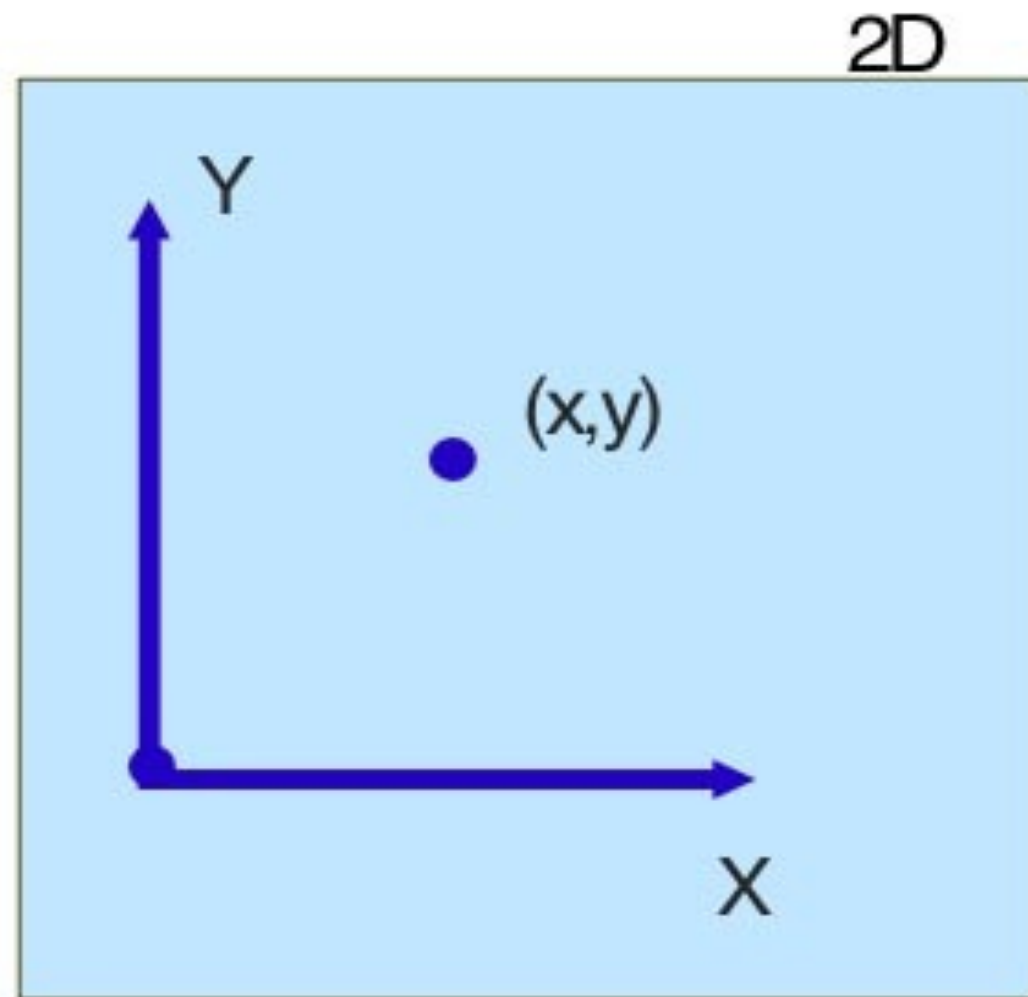
FURB - Universidade Regional de Blumenau  
DSC - Departamento de Sistemas e Computação  
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital  
[www.inf.furb.br/gcg](http://www.inf.furb.br/gcg)



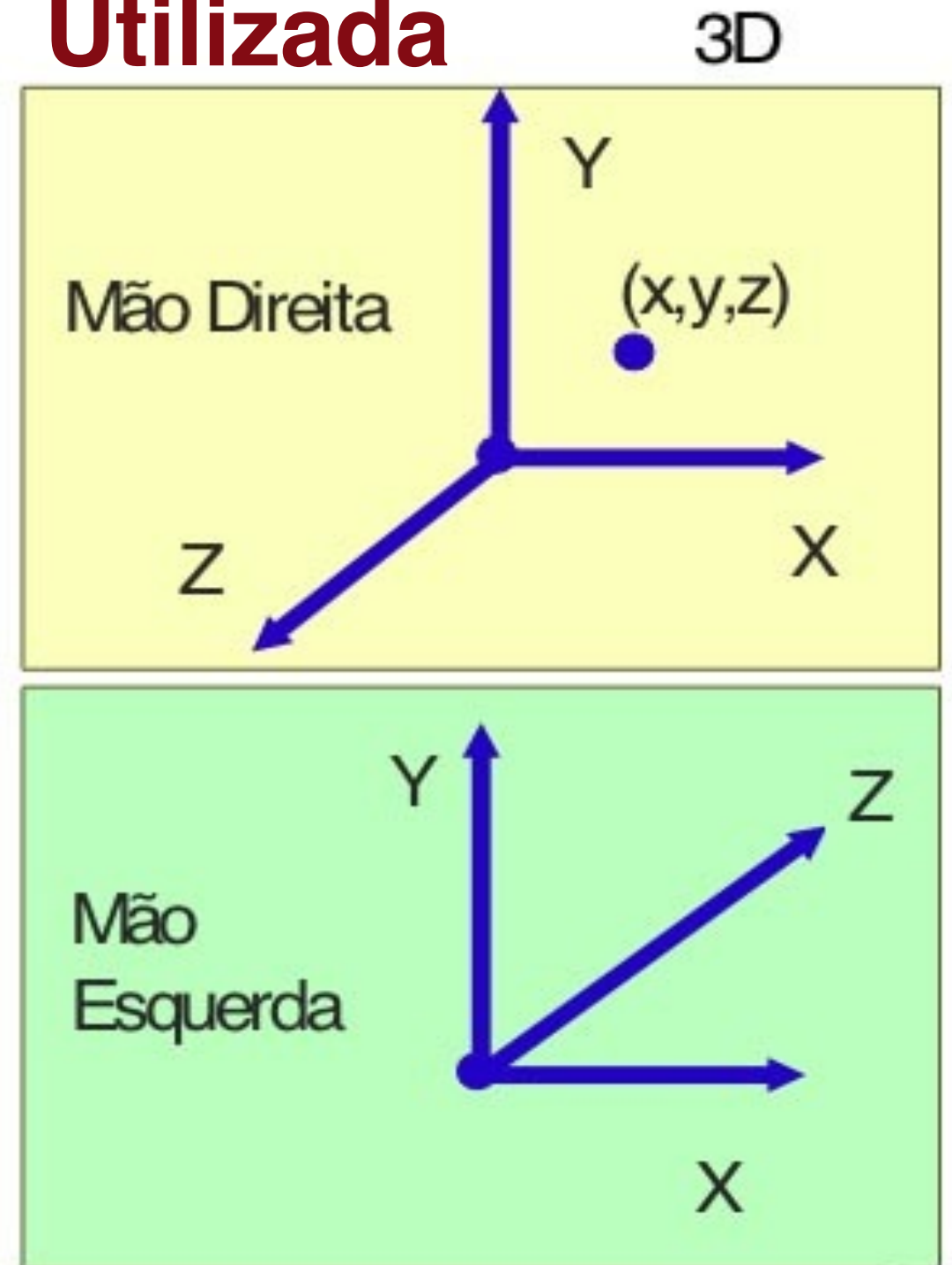
- **Conceitos básicos de 3D**

- Pipeline de visualização: loop, display e render
- Desenho: polígonos, círculos e curvas cúbicas (splines)
- Sistemas de referência, Câmera sintética
- Projeções: ortogonal e perspectiva, Coordenadas homogêneas,
- Transformações geométricas 3D, Transformações inversas,
- Composição de transformações geométricas
- **Objetivos Específicos**
  - Demonstrar conhecimentos teóricos e práticos nos algoritmos básicos de geometria computacional e transformações geométricas 3D.
- **Procedimentos Metodológicos**
  - Aula expositiva dialogada Material programado
  - Atividades em grupo (laboratório)
- **Instrumentos e Critérios de Avaliação**
  - Trabalhos práticos (avaliação 4)

# Sistemas de coordenadas



## Utilizada

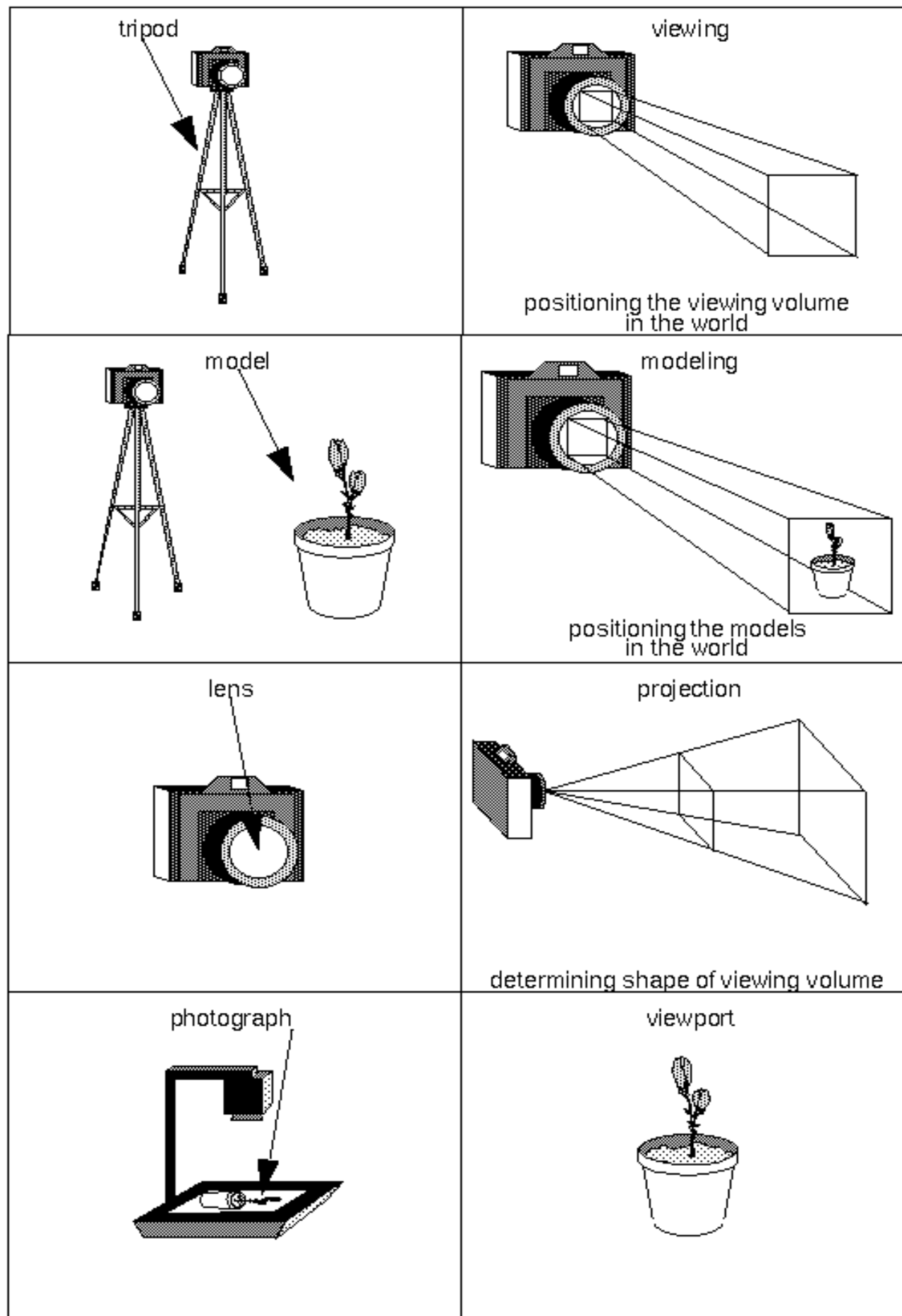


With a Camera

With a Computer

# Câmera OpenGL

- O modelo de visualização em OpenGL, é similar a uma câmera fotográfica!
  - Tripé: *viewing*
  - Modelo: modelo
  - Lente: projeção
  - Papel: *viewport*



# Câmera OpenGL

## Transformações de Projeção:

A matriz de projeção (*Projection*) é outra estrutura importante na renderização de gráficos 3D. Ela basicamente tem o mesmo comportamento de uma matriz de **modelação-visualização** (*Model/View*), porém, é utilizada especificamente para simular o aspecto da visão humana, onde objetos distantes possuem menor tamanho do que objetos mais próximos, quando referente à visão perspectiva.

A ideia é que ela represente coordenadas de visão de um objeto visualizador conceitual, como por exemplo, uma câmera.

# Câmera OpenGL

- As transformações devem ser feitas na seguinte ordem, no seu código:
  - Transformações de projeção
  - Transformações de modelagem
- Transformações de projeção (*Projection*) e *viewport* podem acontecer em qualquer ponto do código antes do modelo ser desenhado (*ModelView*)

# Matrizes de transformação

**glMatrixMode(GL\_PROJECTION);**

- Define tipo e parâmetros da projeção

**glMatrixMode(GL\_MODELVIEW);**

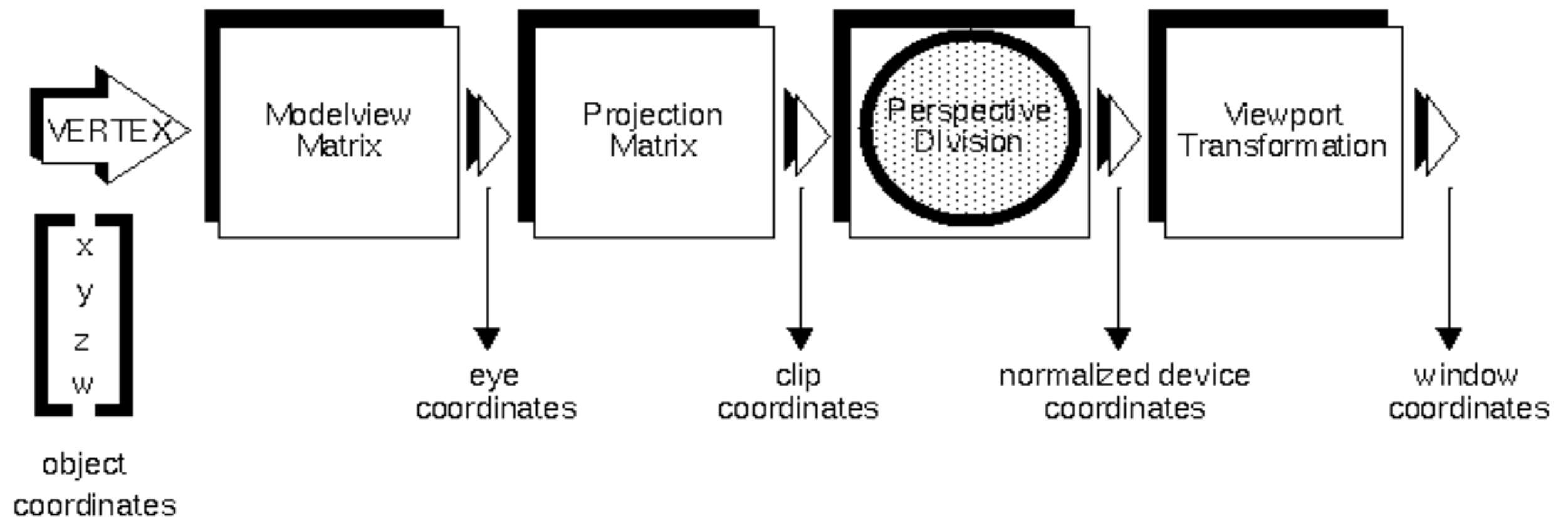
- Define câmera
- Define transformações geométricas

# Etapas *Pipeline* de Visualização Clássico

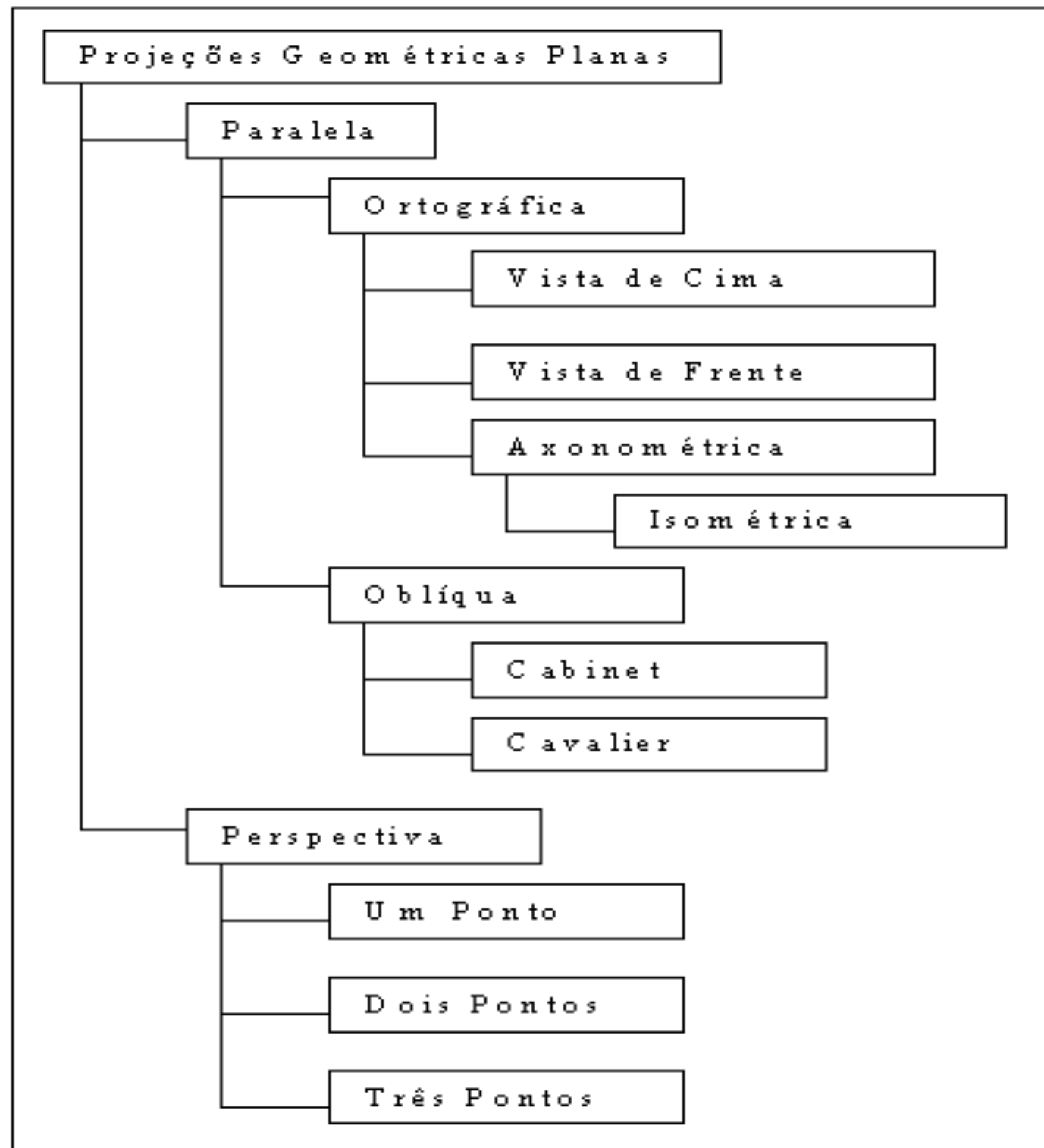
Objetos descritos no SRU	
	<b>TRANSFORMAÇÕES GEOMÉTRICAS</b>
Objetos descritos no SRU (Transformados)	
	<b>MUDANÇA DE SRU PARA SRC</b>
Objetos descritos no SRC	
	<b>RECORTE DE PROFUNDIDADE</b>
Objetos descritos no SRC (Recortados)	
	<b>TRANSFORMAÇÕES DE PROJEÇÃO</b>
Objetos descritos no SRC (Projetados)	
	<b>RECORTE CONTRA A JANELA DE SELEÇÃO</b>
Objetos descritos no SRC (Recortados)	
	<b>MUDANÇA DE SRC PARA SRD</b>
Objetos descritos no SRD	
SRU: Sistema de Referência do Universo SRC: Sistema de Referência da Câmera SRD: Sistema de Referência do Dispositivo	



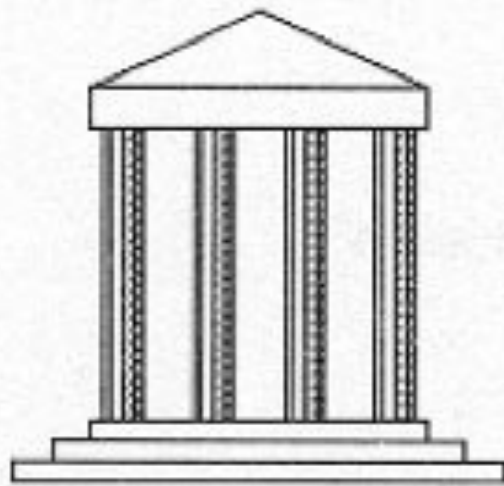
# *Pipeline* de transformações



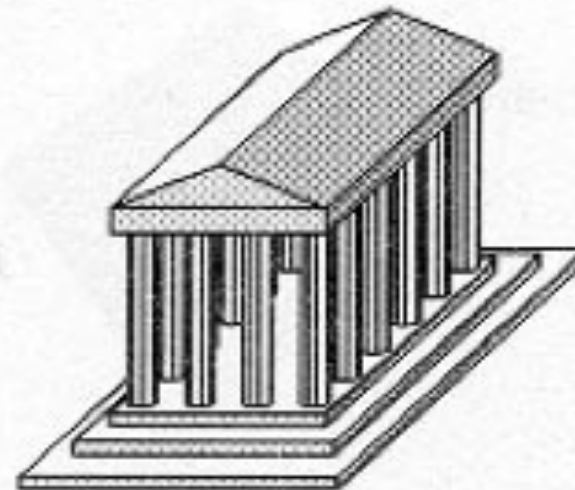
# Hierarquia das projeções



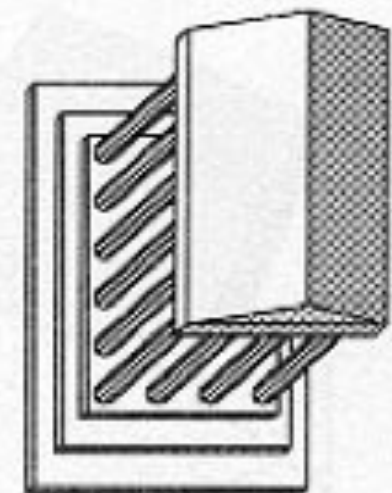
# Projeções



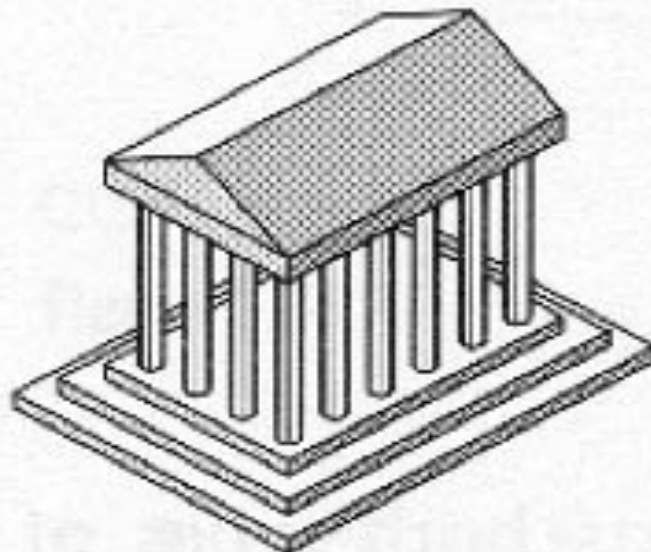
Front elevation



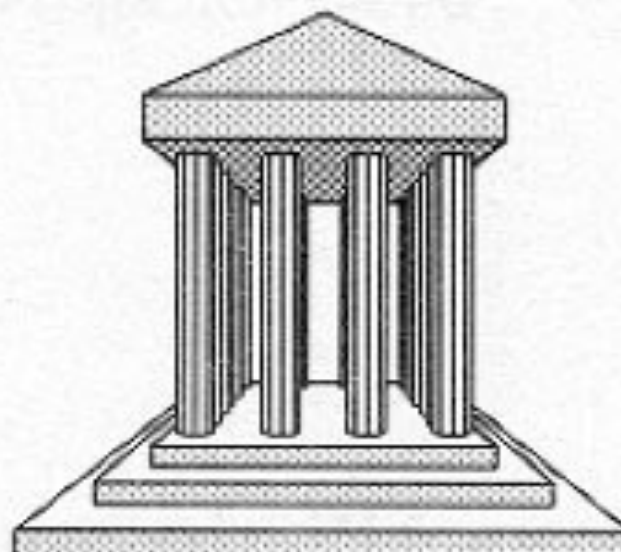
Elevation oblique



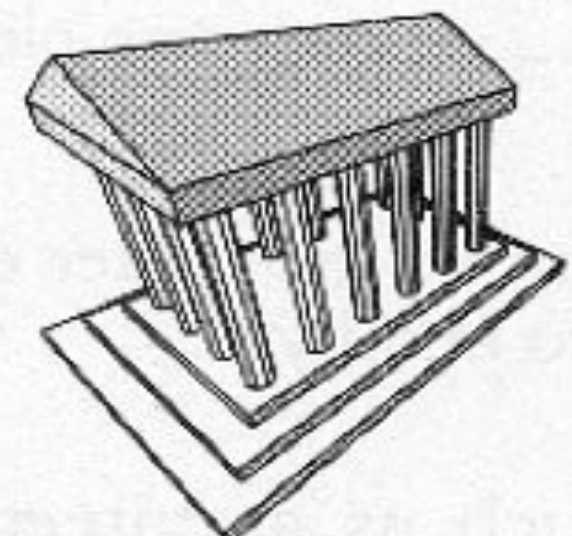
Plan oblique



Isometric

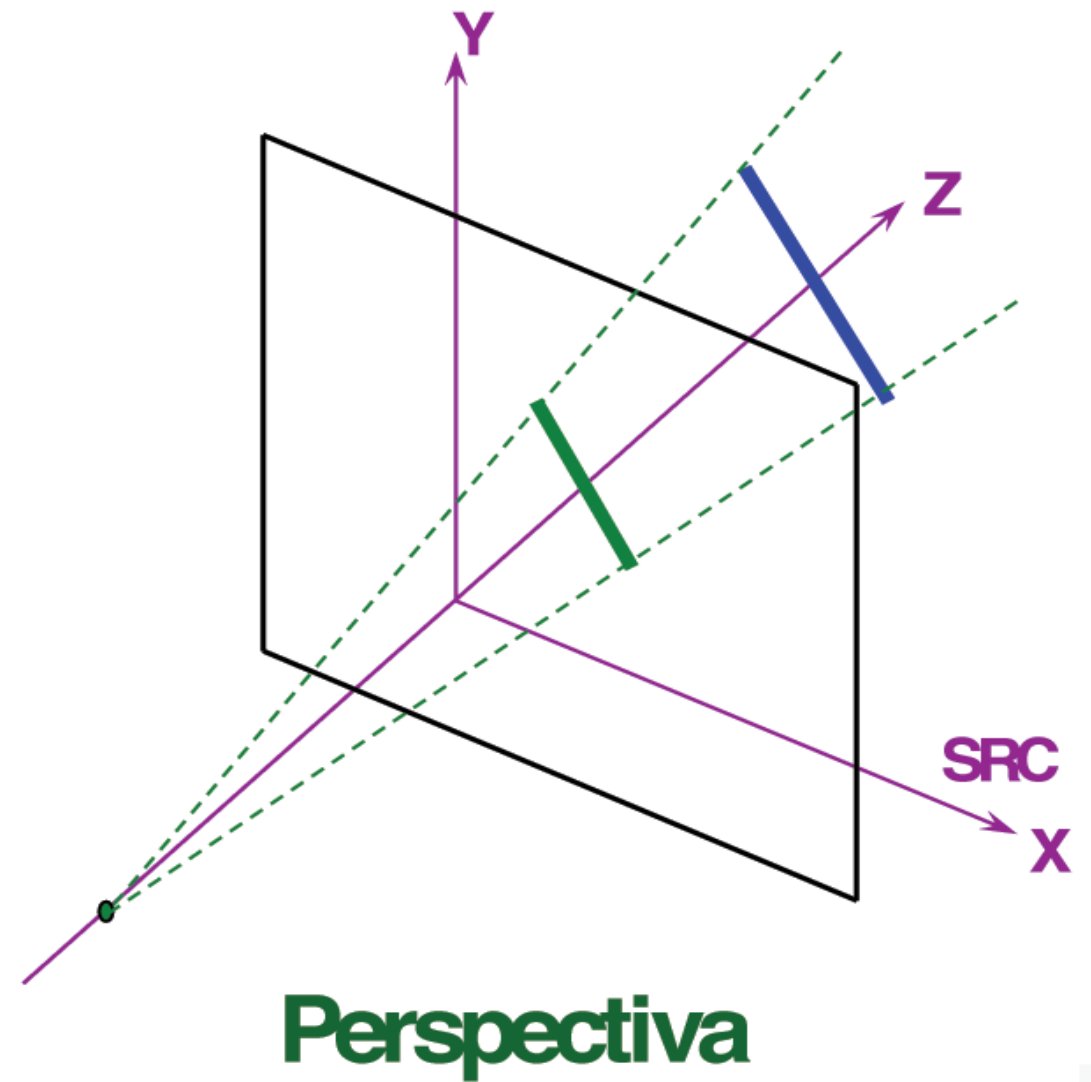
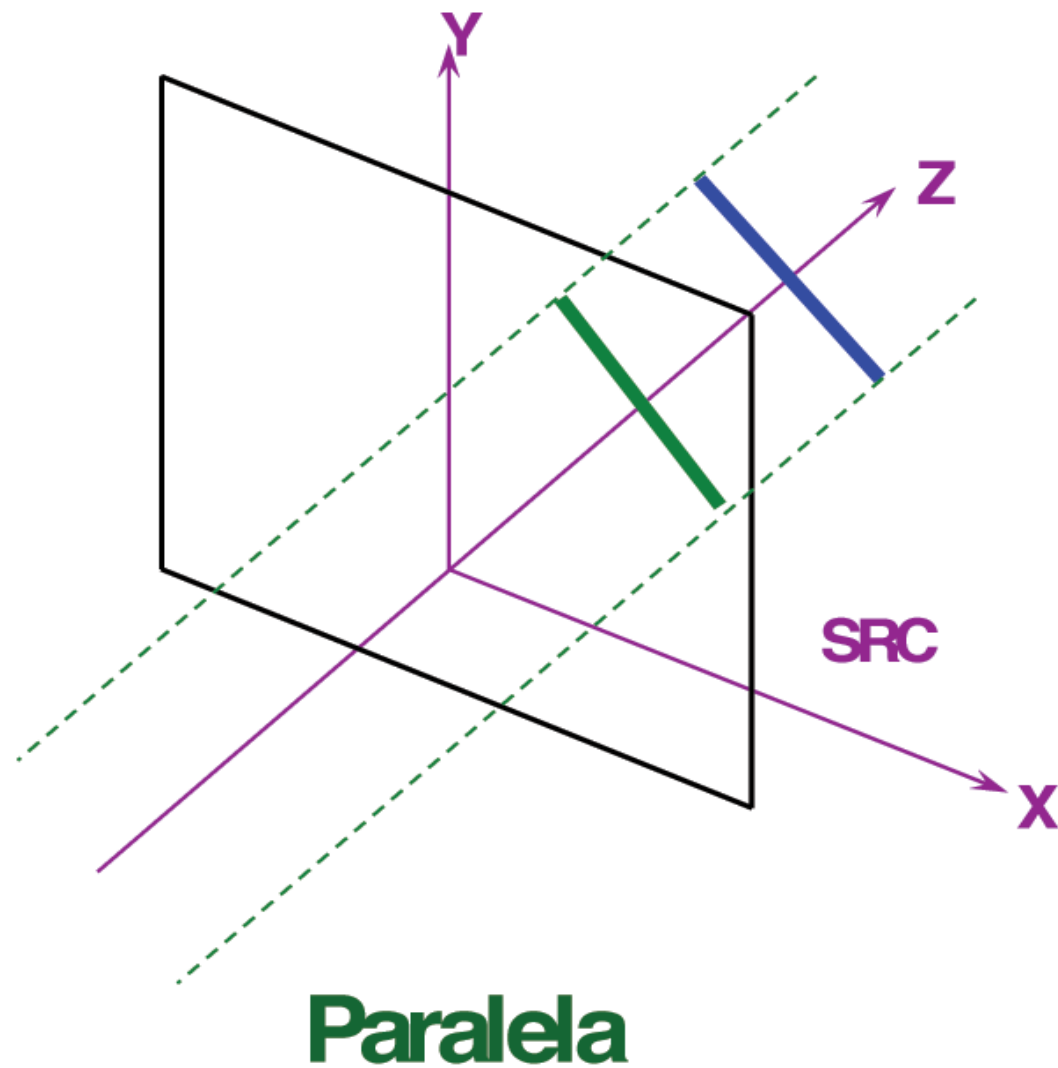


One-point perspective



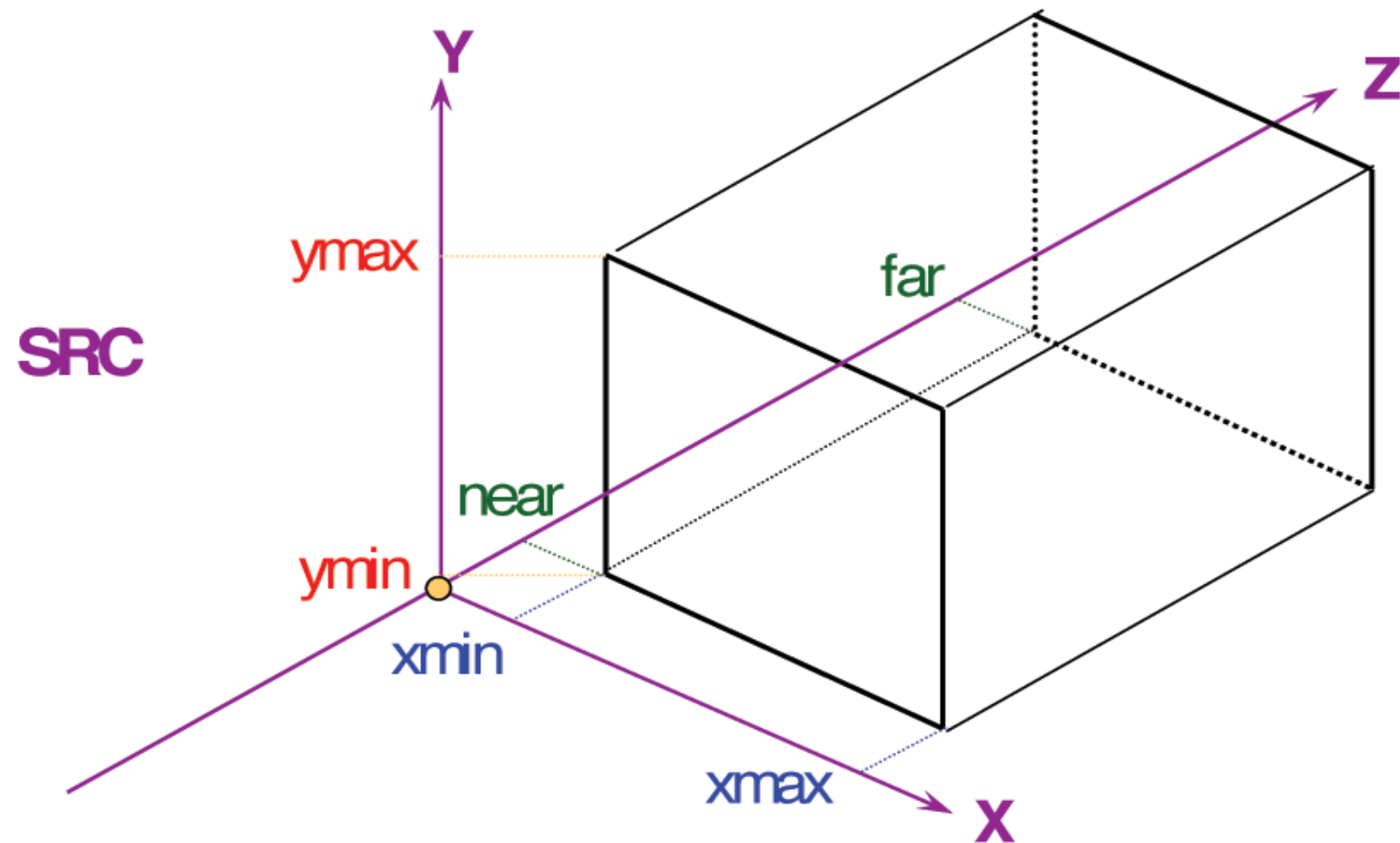
Three-point perspective

# Projeções em OpenGL



# Projeção paralela ortográfica

- Determina um paralelepípedo:
- ***glOrtho*** ( xmin, xmax, ymin, ymax, near, far)



# Projeção em perspectiva

- Centro de projeção fixo: *eye* (posição da câmera)
- Duas possibilidades:
  - Determina um tronco de pirâmide:  
`glFrustum`
  - Determina o ângulo de visão  
`gluPerspective`

# Definição do volume de visualização

**glFrustum(left, right, bottom, top, near, far);**

- não precisa ser simétrico

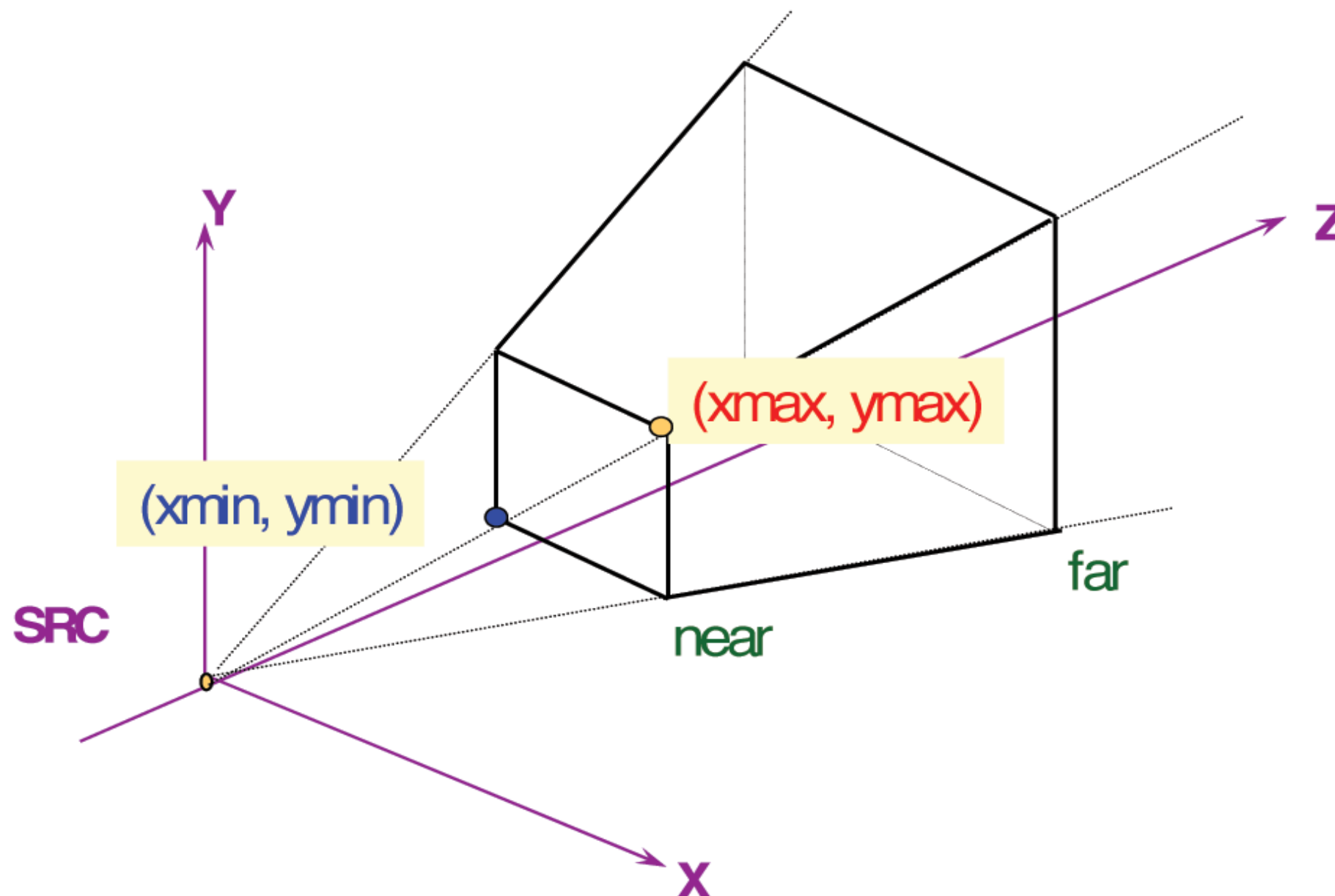
**gluPerspective(fovy, aspect ratio, near, far);**

- simétrico



# glFrustum

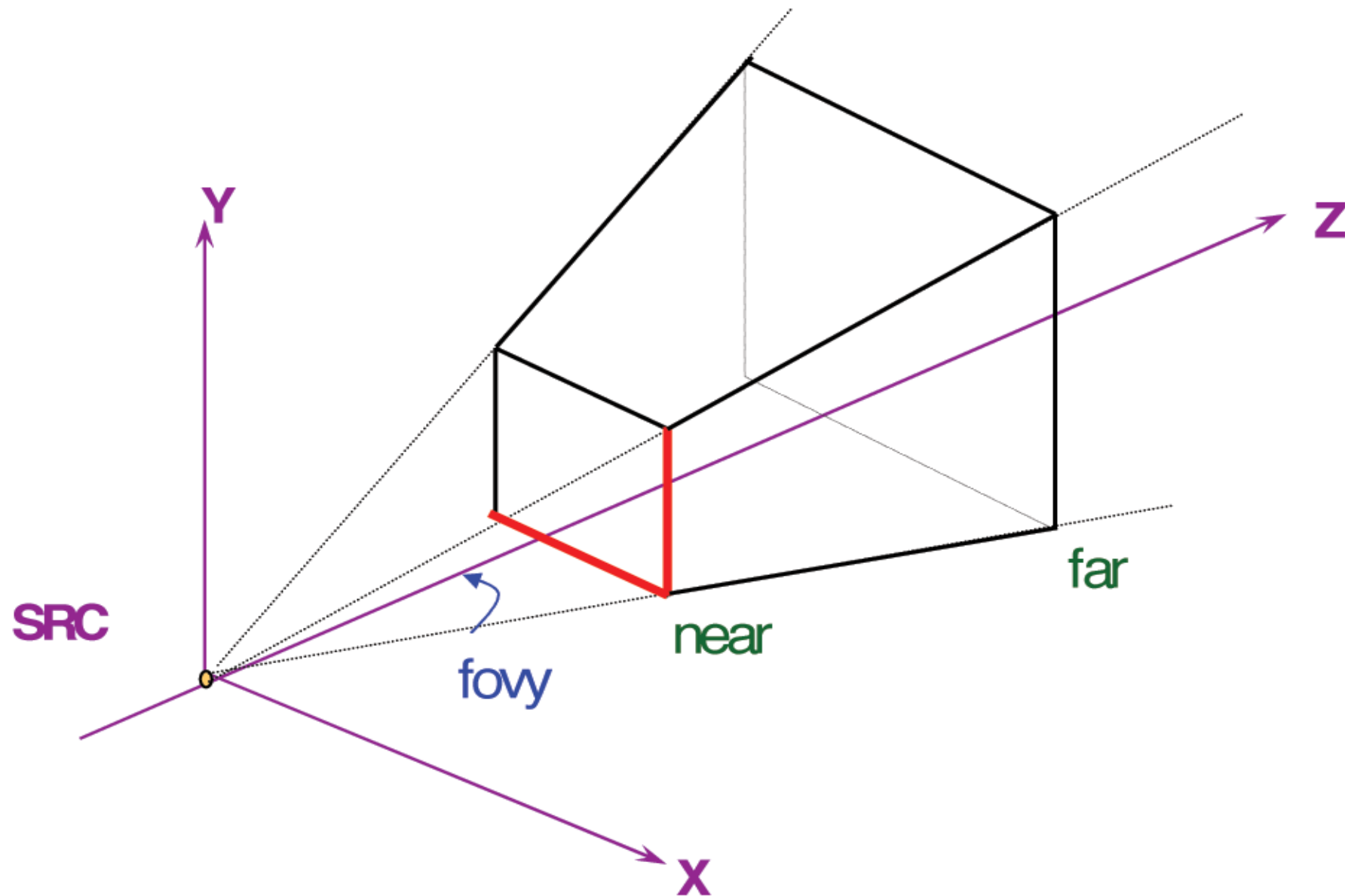
***glFrustum***( $x_{\min}$ ,  $x_{\max}$ ,  $y_{\min}$ ,  $y_{\max}$ ,  $near$ ,  $far$ )





# gluPerspective

***gluPerspective*** ( fovy, aspect, near, far )



# gluPerspective

***gluPerspective*** ( fovy, aspect, near, far )

```
protected override void OnResize(EventArgs e)
{
    base.OnResize(e);

    GL.Viewport(ClientRectangle.X, ClientRectangle.Y, ClientRectangle.Width, ClientRectangle.Height);

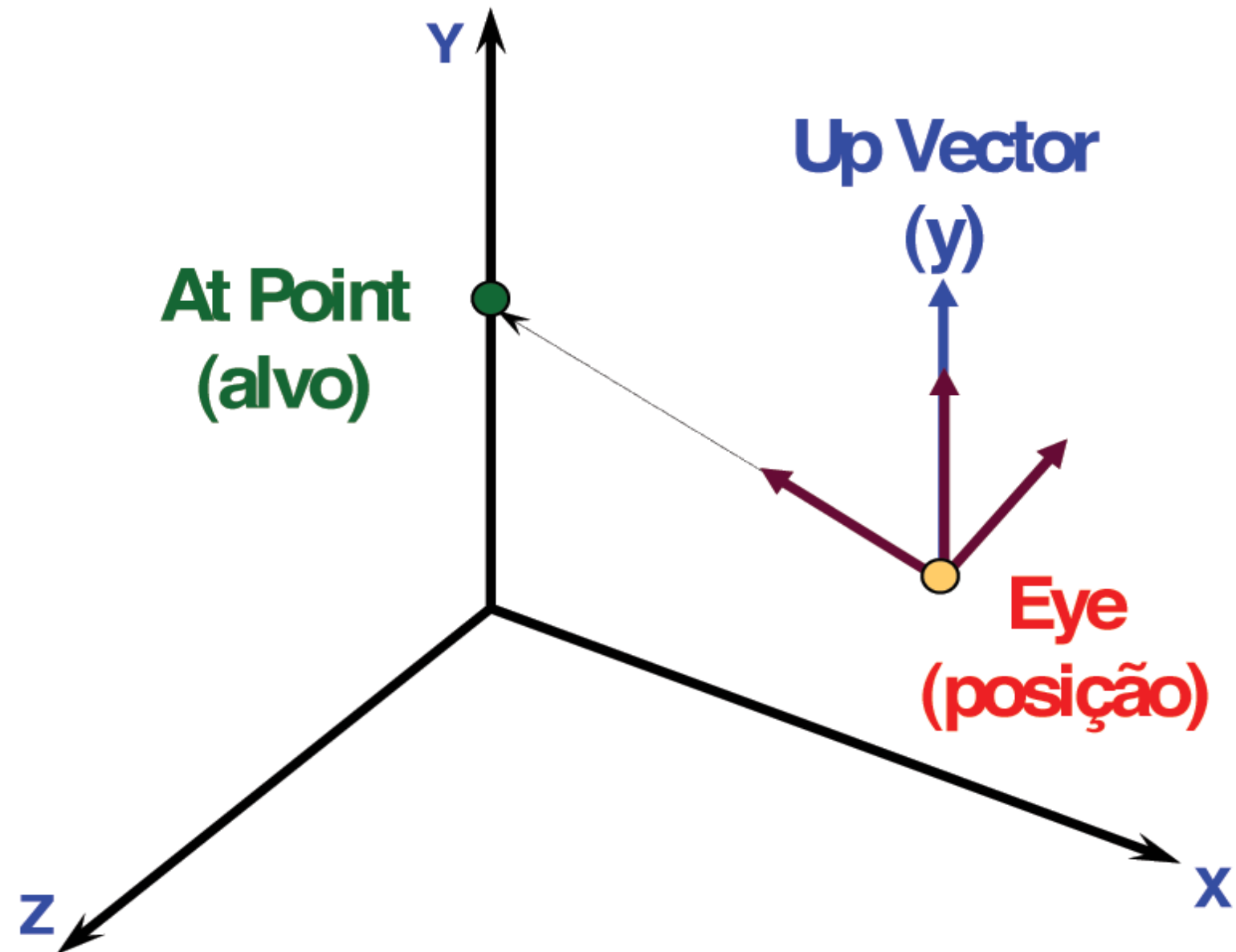
    Matrix4 projection = Matrix4.CreatePerspectiveFieldOfView( (float)Math.PI / 4, Width / (float)Height, 1.0f, 50.0f);
    GL.MatrixMode(MatrixMode.Projection);
    GL.LoadMatrix(ref projection);
}

protected override void OnUpdateFrame(FrameEventArgs e)
{
    base.OnUpdateFrame(e);
}
```

X

# Projeção em perspectiva

***gluLookAt*** ( *eyex, eyey, eyez, atx, aty, atz, upx, upy, upz* )



# Projeção em perspectiva

***gluLookAt*** ( *eyex*, *eyey*, *eyez*, *atx*, *aty*, *atz*, *upx*, *upy*, *upz* )

```
protected override void OnRenderFrame(FrameEventArgs e)
{
    base.OnRenderFrame(e);

    GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

    Matrix4 modelview = Matrix4.LookAt(eye, target, up);
    GL.MatrixMode(MatrixMode.Modelview);
    GL.LoadMatrix(ref modelview);

    mundo.Desenha();

    this.SwapBuffers();
}
```

# O modelo de câmera OpenGL

- A cena é construída na origem e definimos uma posição arbitrária para a câmera
- `void gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz);`
  - Eye: localização da câmera
  - Center: para onde a câmera aponta
  - Up: vetor de direção de topo da câmera (0, 1, 0)

# Computação Gráfica

## Unidade 04

prof. Dalton S. dos Reis  
dalton.reis@gmail.com

FURB - Universidade Regional de Blumenau  
DSC - Departamento de Sistemas e Computação  
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital  
<http://www.inf.furb.br/gcg/>

