Computação Gráfica
Unidade 4

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

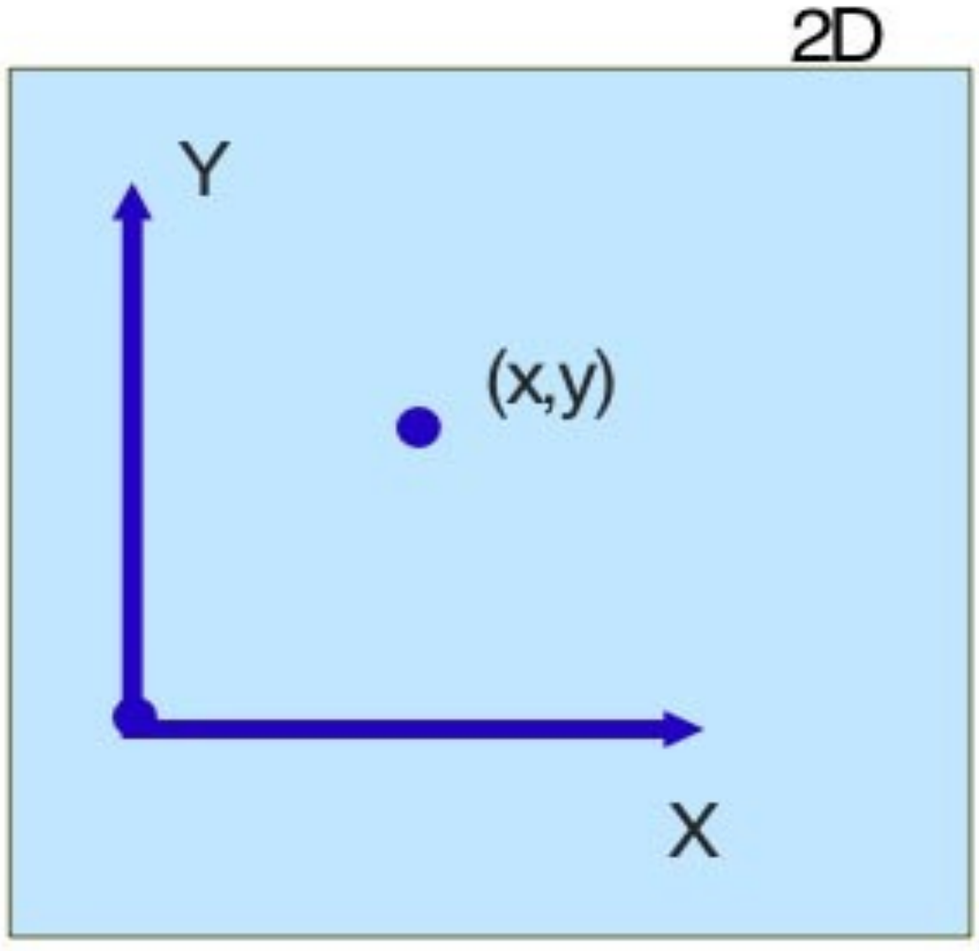
FURB - Universidade Regional de Blumenau
DSC - Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
www.inf.furb.br/gcg

Unidade 04

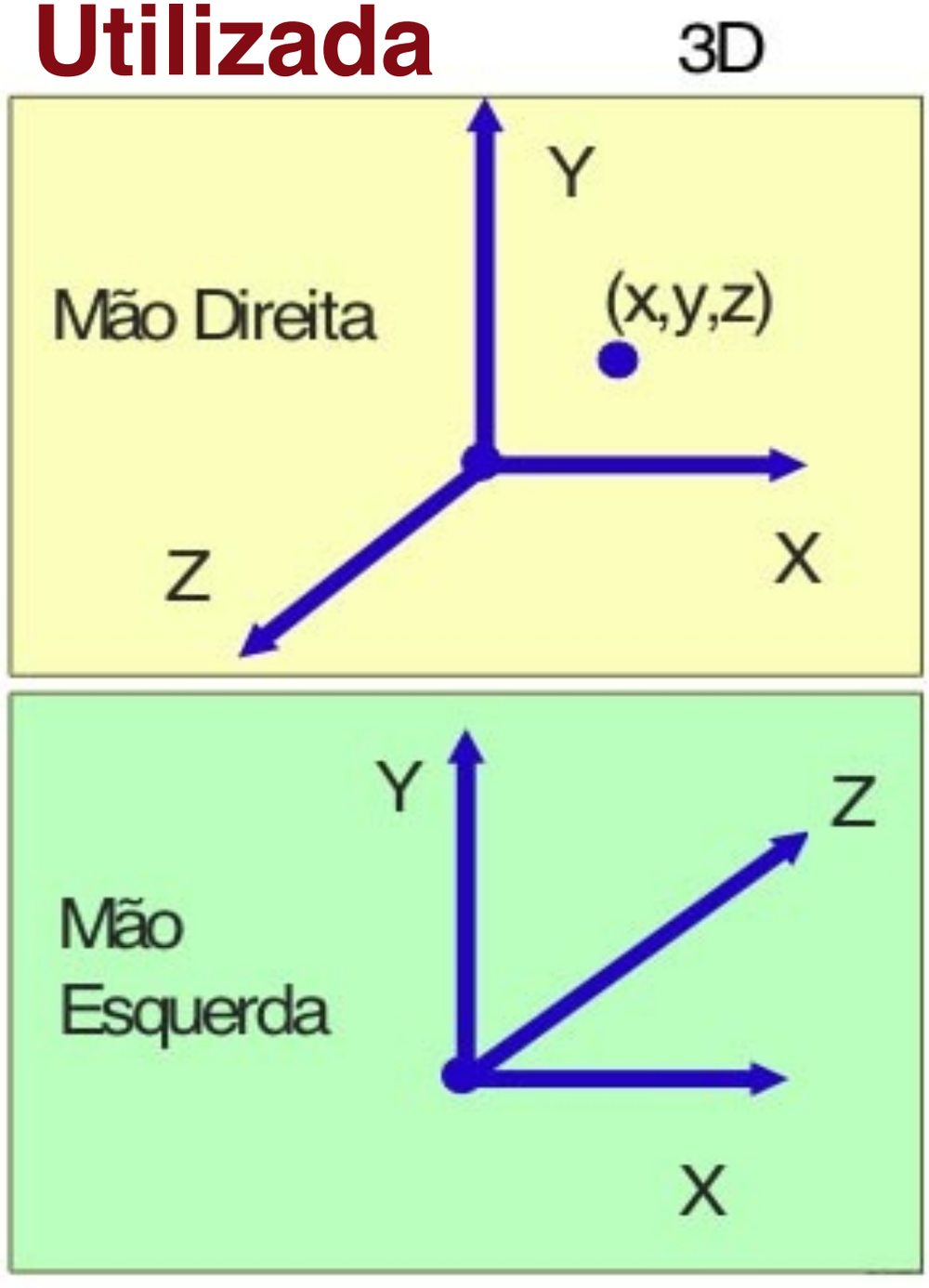
- Conceitos básicos de 3D
  - Pipeline de visualização: loop, display e render
  - Desenho: polígonos, círculos e curvas cúbicas (splines)
  - Sistemas de referência, Câmera sintética
  - Projeções: ortogonal e perspectiva, Coordenadas homogêneas,
  - Transformações geométricas 3D, Transformações inversas,
  - Composição de transformações geométricas

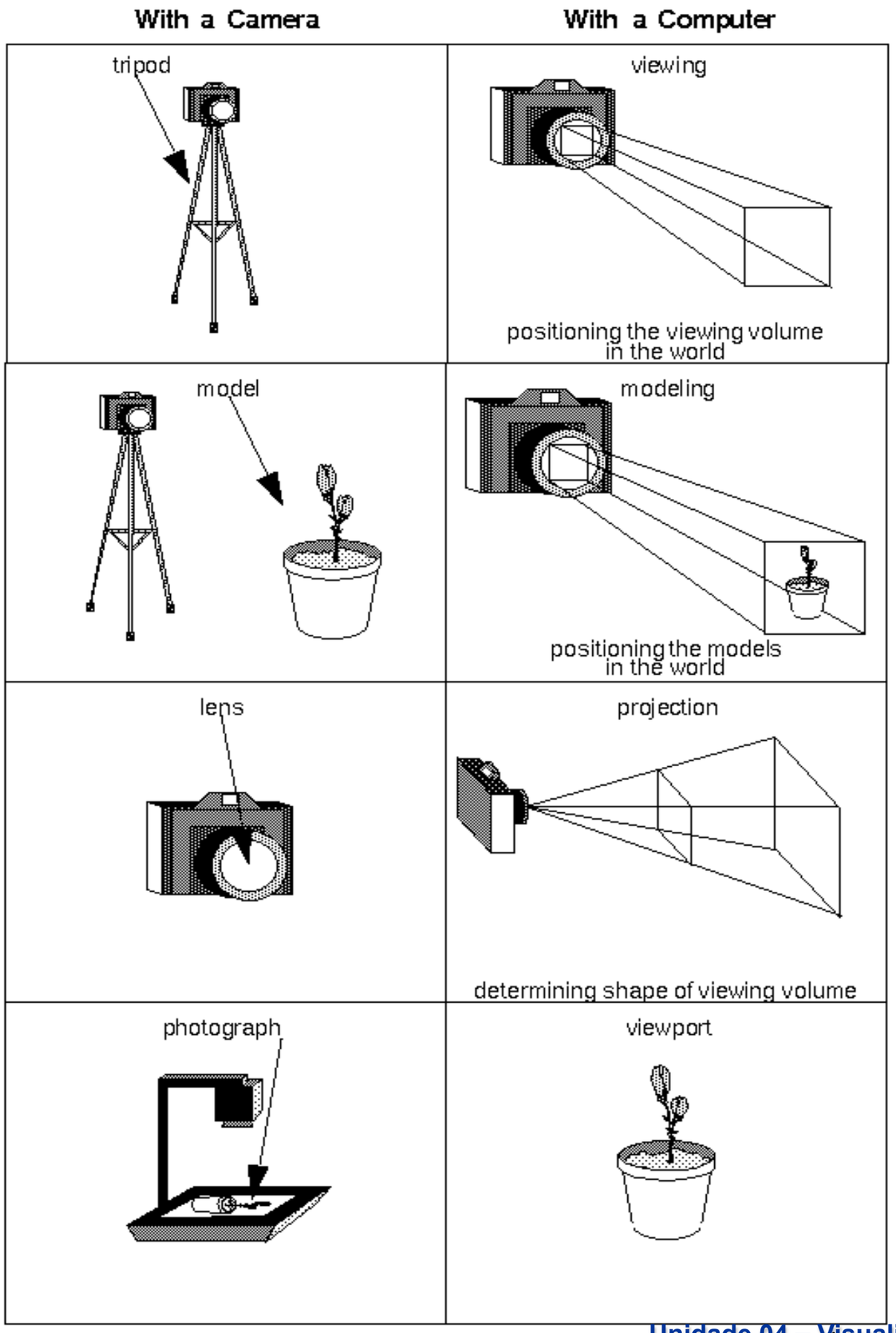
- Objetivos Específicos
  - Demonstrar conhecimentos teóricos e práticos nos algoritmos básicos de geometria computacional e transformações geométricas 3D.
- Procedimentos Metodológicos
  - Aula expositiva dialogadaMaterial programado
  - Atividades em grupo (laboratório)
- Instrumentos e Critérios de Avaliação
  - Trabalhos práticos (avaliação 4)

Sistemas de coordenadas



Utilizada





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** (*ModelView*), 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

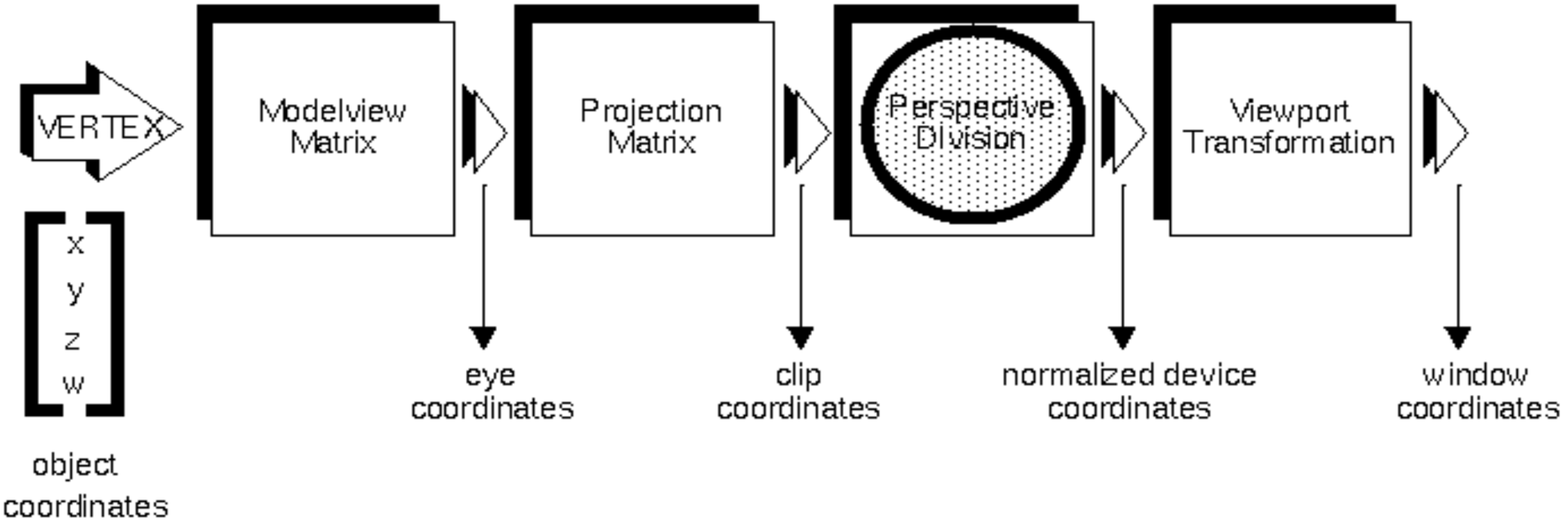
glMatrixMode(GL_MODELVIEW);

- Define câmera
- Define transformações geométricas

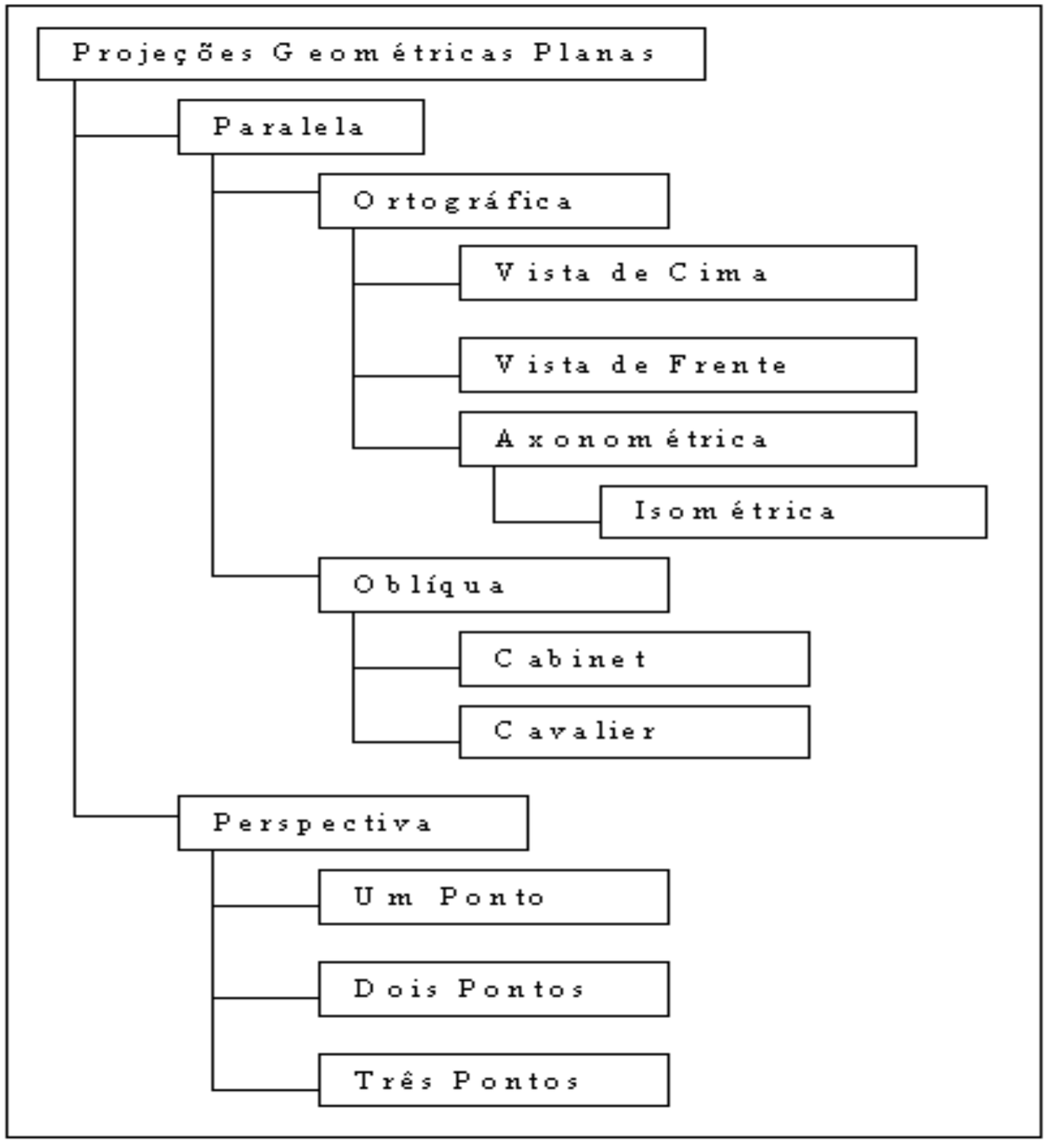
Etapas *Pipeline* de Visualização Clássico

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

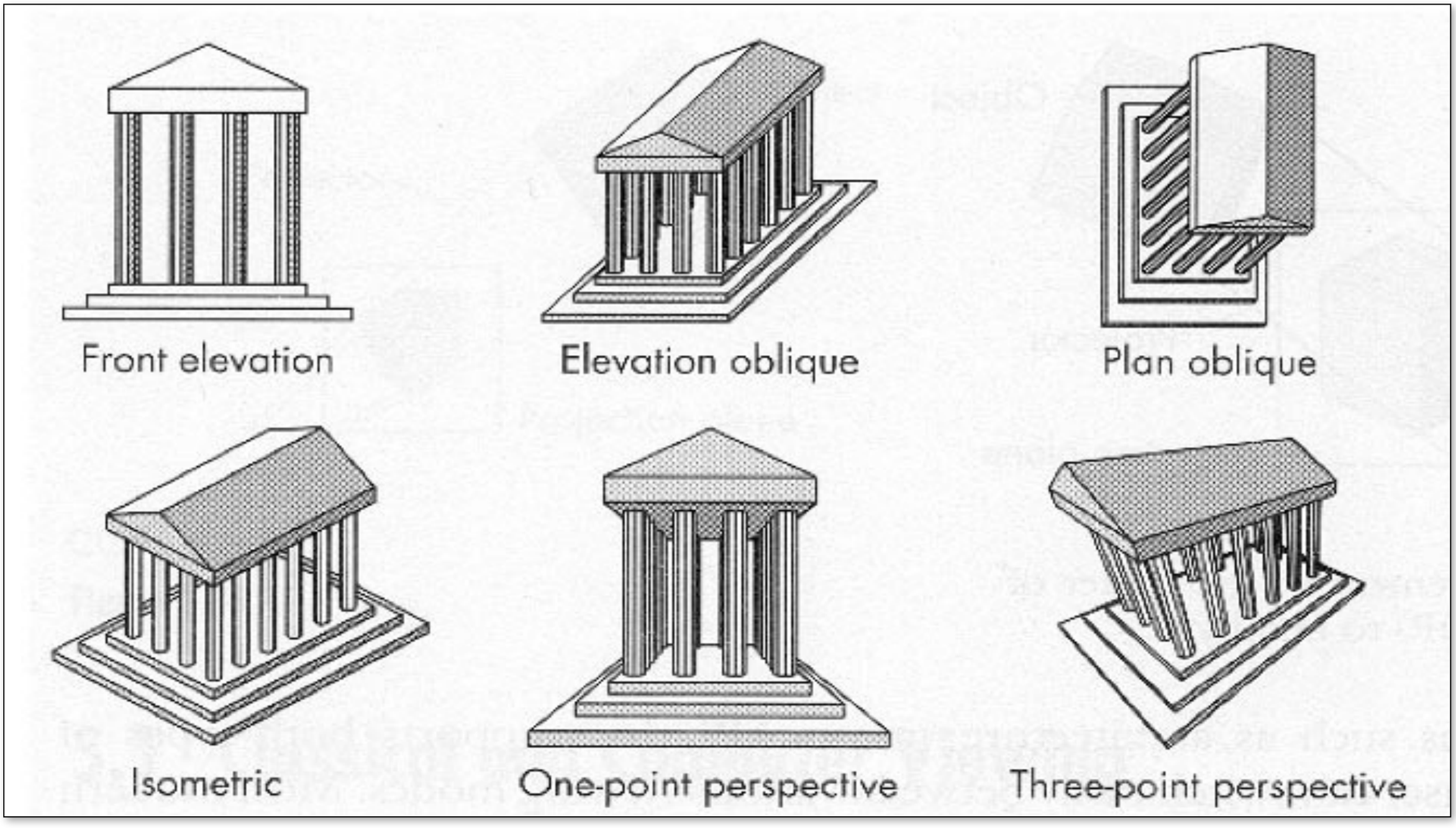
Pipeline de transformações



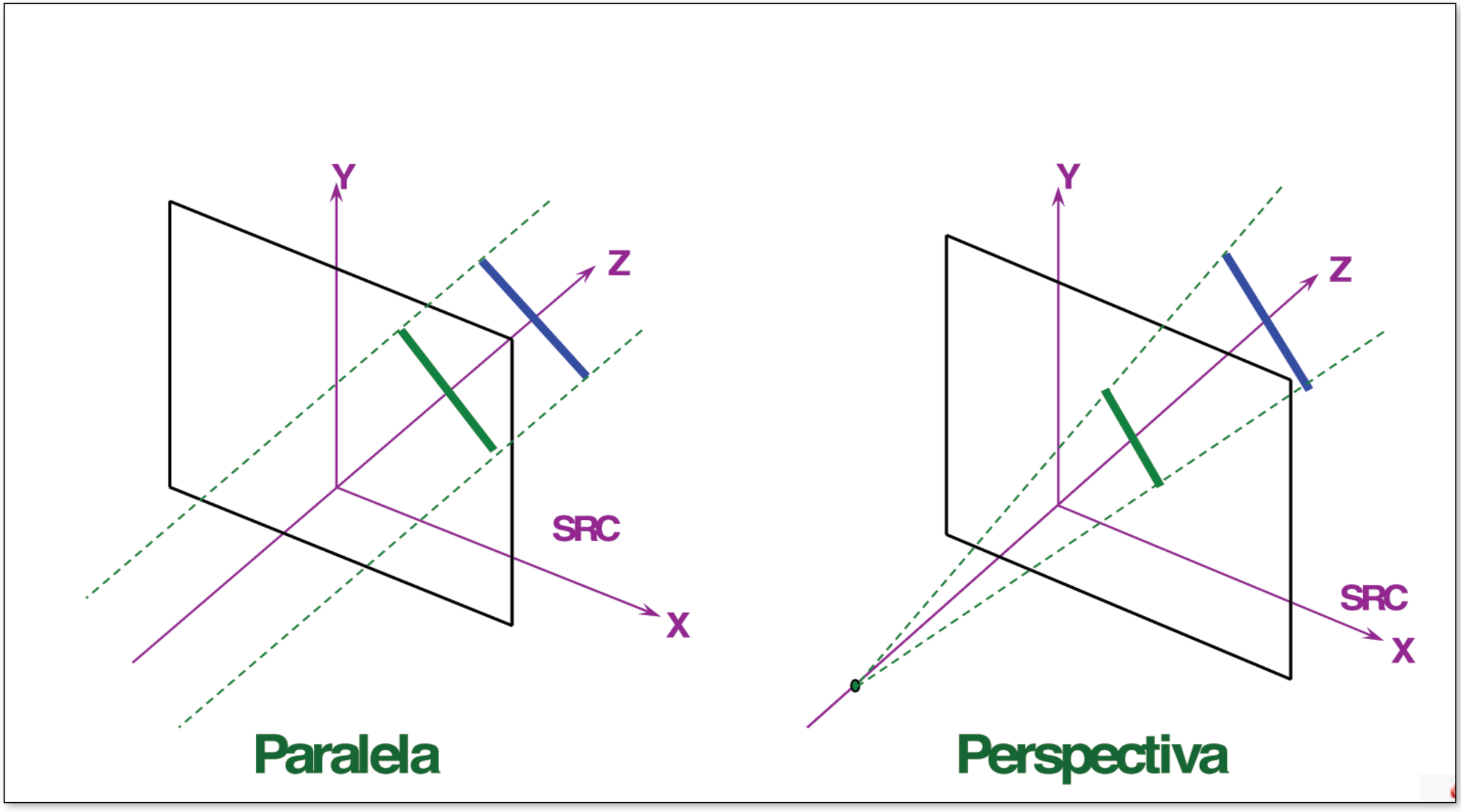
Hierarquia das projeções



Projeções

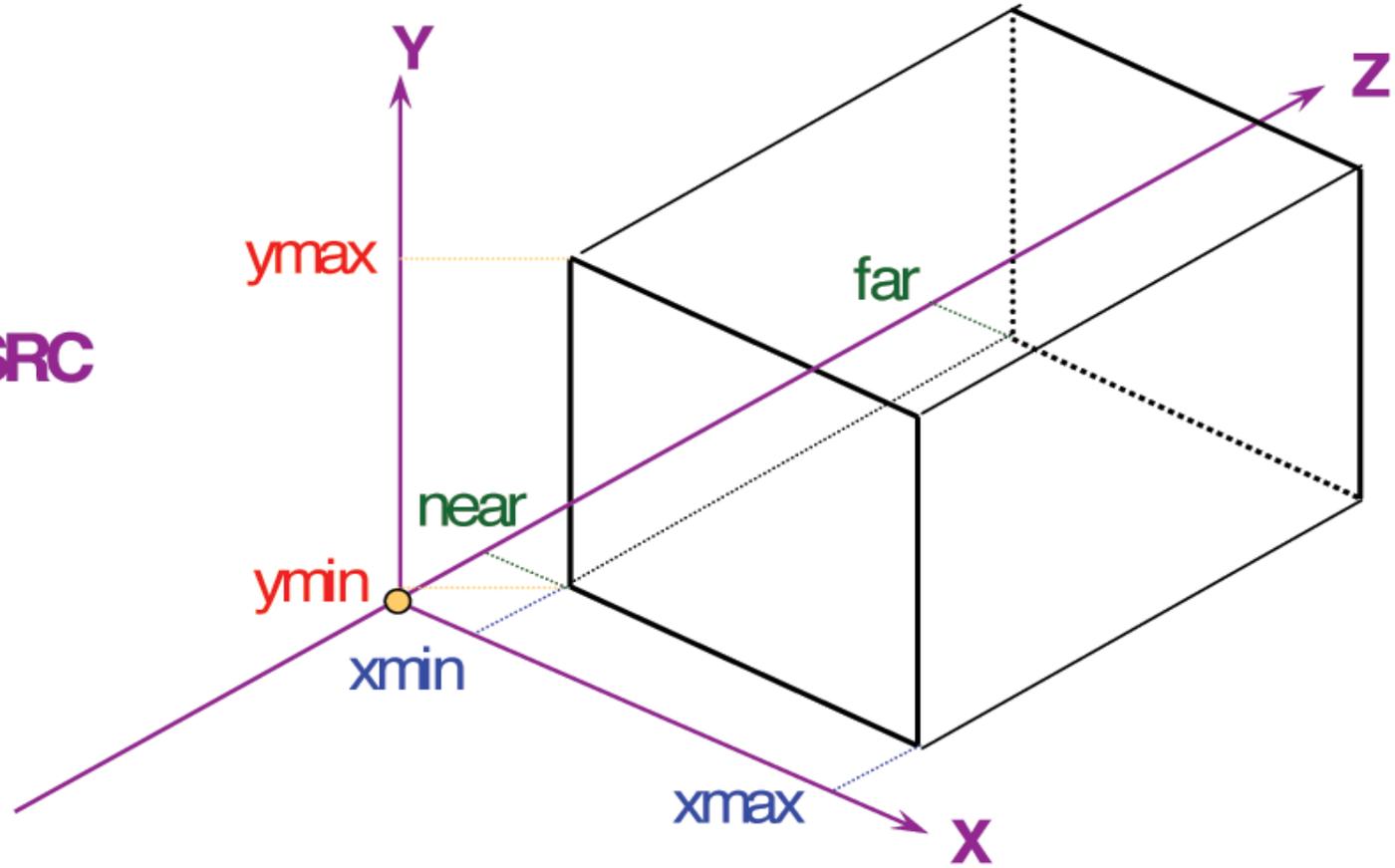

Front elevation
Elevation oblique
Plan oblique
Isometric
One-point perspective
Three-point perspective

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: glFrustrum
  - Determina o ângulo de visão gluPerspective

Definição do volume de visualização

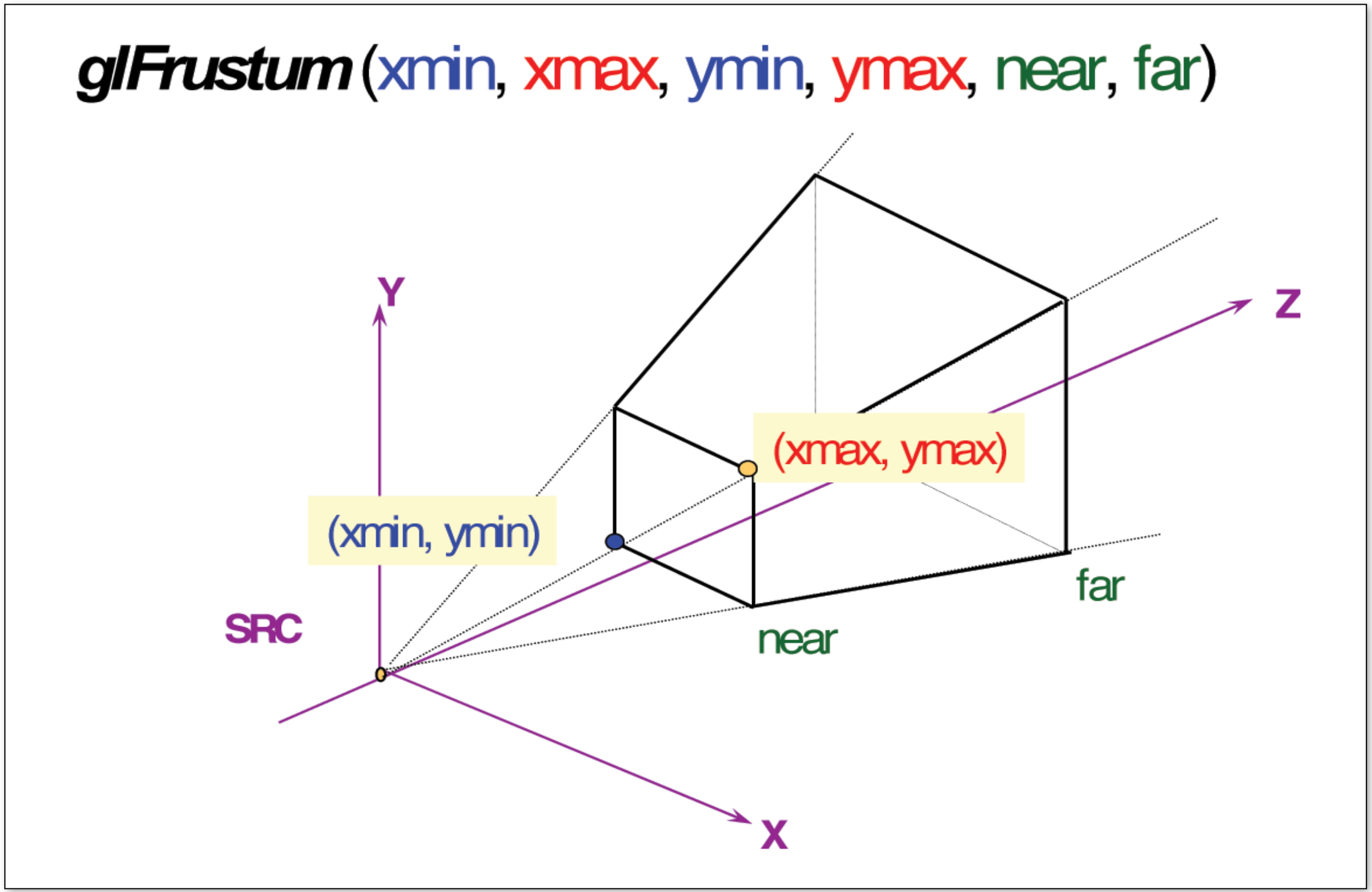
glFrustrum(left, right, bottom, top, near, far);

– não precisa ser simétrico

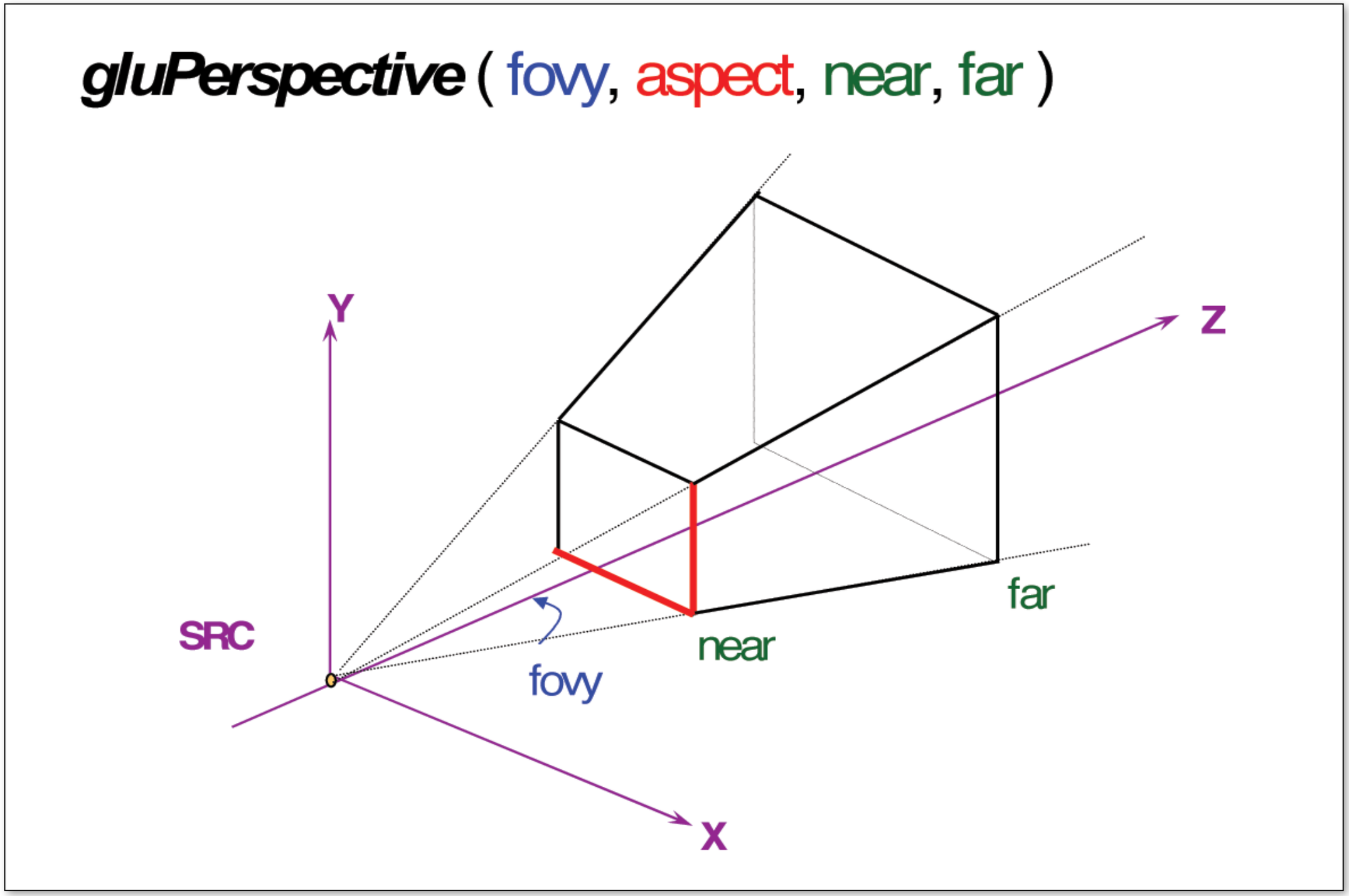
gluPerspective(fovy, aspect ratio, near, far);

– simétrico

glFrustrum


glFrustum (xmin, xmax, ymin, ymax, near, far)
Y
Z
(xmax, ymax)
(xmin, ymin)
far
near
SRC
X

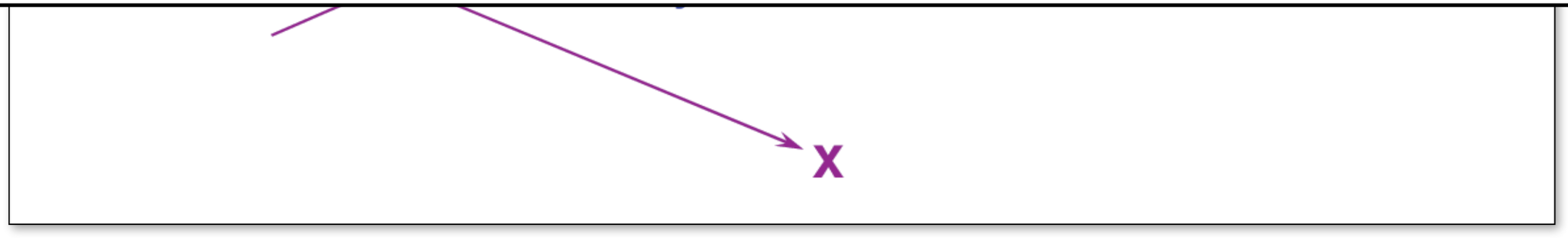
gluPerspective



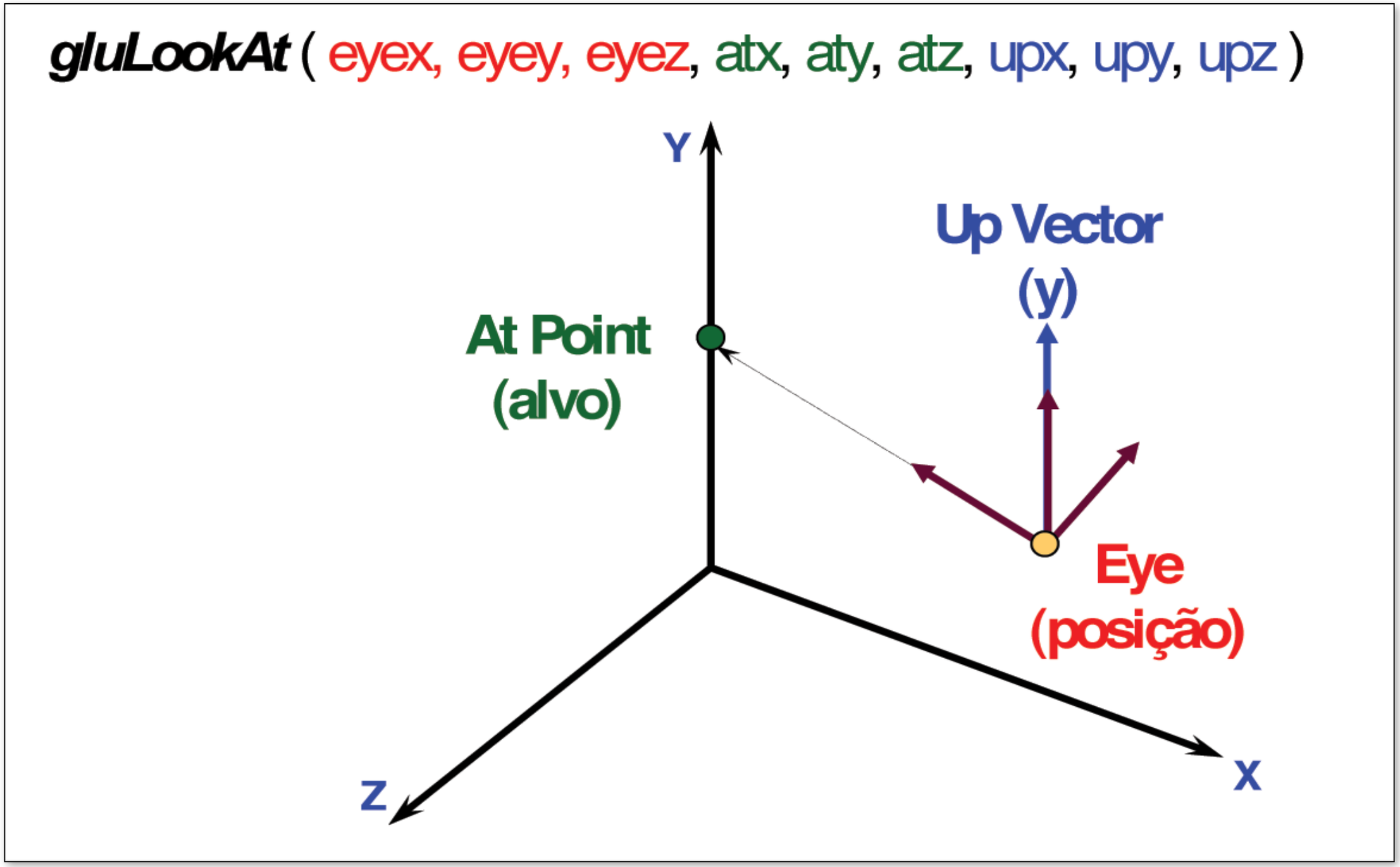
gluPerspective

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);
}
```



Projeção em perspectiva



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)

Exemplos

```
xEye = 20.0f;          yEye = 20.0f;          zEye = 20.0f;
xCenter = 0.0f;        yCenter = 0.0f;        zCenter = 0.0f;
```

```
public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    gl.glMatrixMode(GL.GL_PROJECTION);
    gl.glLoadIdentity();
        gl.glViewport(0, 0, width, height);

        glu.gluOrtho2D(-30.0f, 30.0f, -30.0f, 30.0f);
    glu.gluPerspective(60, width/height, 0.1, 100);                    // projecao Perpectiva 1 pto fu
        gl.glFrustum (-5.0, 5.0, -5.0, 5.0, 10, 100);                 // projecao Perpectiva 1 pto fuga 3D
    gl.glOrtho(-30.0f, 30.0f, -30.0f, 30.0f, -30.0f, 30.0f);    // projecao Ortogonal 3D

        Debug();
}
```

```
public void display(GLAutoDrawable drawable) {
        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

gl.glMatrixMode(GL.GL_MODELVIEW);
gl.glLoadIdentity();
        glu.gluLookAt(xEye, yEye, zEye, xCenter, yCenter, zCenter, 0.0f, 1.0f, 0.0f);

        drawAxis();
        gl.glColor3f(1.0f, 0.0f, 0.0f);
        drawCube(translacaoCubo1,escalaCubo1);

        gl.glColor3f(1.0f, 0.0f, 0.0f);
        drawCube(translacaoCubo2,escalaCubo2);

        gl.glFlush();
}
```

https://bitbucket.org/GCG-Disciplinas/gcg-cg/src/3c271c8dc88c3c31ff041fe2b06c3fc793814709/_java/N4/CG-N4_Camera_Enable/src/Main.java?at=master&fileviewer=file-view-default

Exemplos

```
private void drawCube(float translacao[], float escala[]) {
        if (eHMaterial) {
                gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, corRed, 0);
                gl.glEnable(GL.GL_LIGHTING);
        }

        gl.glPushMatrix();
                gl.glScalef(escala[0],escala[1],escala[2]);
                gl.glTranslated(translacao[0], translacao[1], translacao[2]);
                glut.glutSolidCube(1.0f);
        gl.glPopMatrix();

        if (eHMaterial) {
                gl.glDisable(GL.GL_LIGHTING);
        }
}
```

https://bitbucket.org/GCG-Disciplinas/gcg-cg/src/3c271c8dc88c3c31ff041fe2b06c3fc793814709/_java/N4/CG-N4_Camera_Enable/src/Main.java?at=master&fileviewer=file-view-default

```
public void init(GLAutoDrawable drawable) {
        glDrawable = drawable;
        gl = drawable.getGL();
        glu = new GLU();
        glut = new GLUT();
        glDrawable.setGL(new DebugGL(gl));

        gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        xEye = 20.0f;           yEye = 20.0f;           zEye = 20.0f;
        xCenter = 0.0f;         yCenter = 0.0f;         zCenter = 0.0f;

        ligarLuz();

    gl.glEnable(GL.GL_CULL_FACE);
    gl.glDisable(GL.GL_CULL_FACE);

    gl.glEnable(GL.GL_DEPTH_TEST);
    gl.glDisable(GL.GL_DEPTH_TEST);
}
```


https://bitbucket.org/GCG-Disciplinas/gcg-cg/src/3c271c8dc88c3c31ff041fe2b06c3fc793814709/_java/N4/CG-N4_Camera_Enable/src/Main.java?at=master&fileviewer=file-view-default

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/