Transformações de *Viewing* 2D

SCC0250 - Computação Gráfica

Profa. Maria Cristina F. Oliveira

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

7 de outubro de 2023

Sumário

1. Janela de Recorte (Cena 2D)

2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto

3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D

Sumário

1. Janela de Recorte (Cena 2D)
2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto
3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D

Introdução

Viewing Pipeline 2D

Etapas para renderizar uma cena 2D, determinando qual região da cena será exibida na tela, e aonde

- A cena está especificada no **sistema de coordenadas do mundo** (*world coordinates*, ou WCS)
- Uma janela define a região da cena a ser renderizada, cujo conteúdo será mapeado para uma janela no dispositivo (*device coordinates*)
- Tal mapeamento envolve transformações geométricas (de escala e translação, no caso de janelas alinhadas aos eixos do WCS)
- Objetos e partes da cena que estão fora da janela serão descartadas (recorte)

Introdução

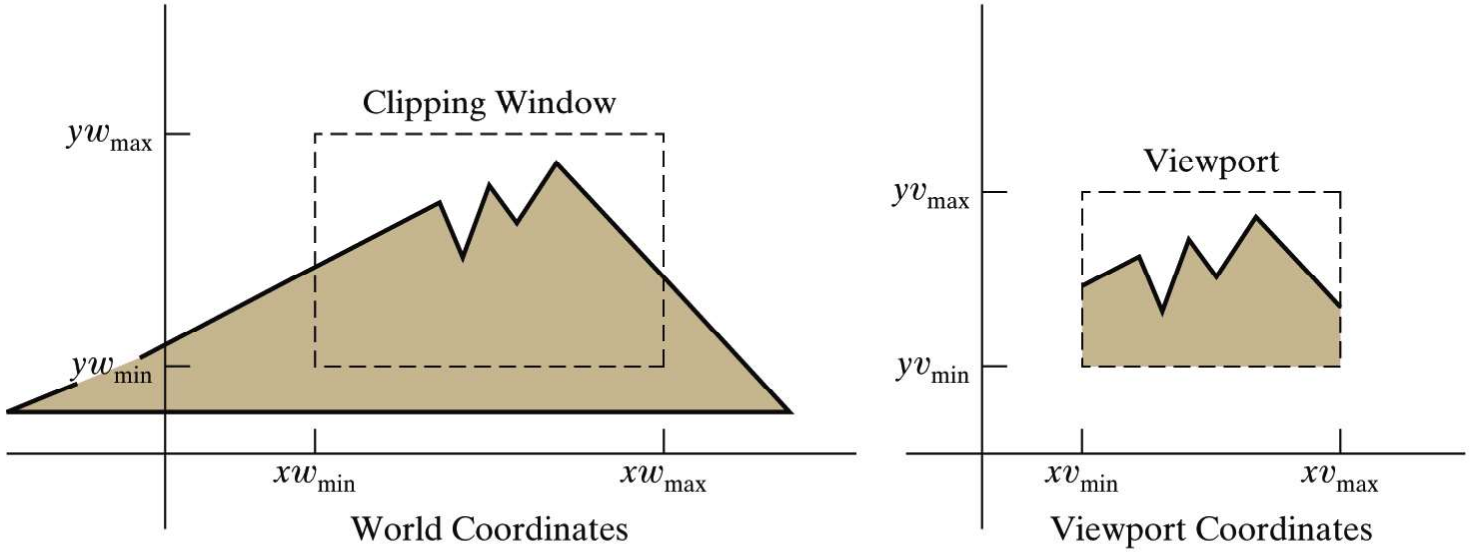
Janela de Recorte ou *Clipping Window*

- Define a região da cena 2D selecionada para ser exibida
- Tudo o que estiver fora dessa região será "recortado", i.e. descartado (não é renderizado)

Viewport

- O conteúdo da *Janela de Recorte* é transferido para o dispositivo de exibição usando outra "janela" intermediária, denominada **Viewport**
  - Objetos contidos na *Janela de Recorte* são mapeados para a **Viewport**, a qual por sua vez será mapeada para o dispositivo de exibição
  - Pode-se usar **Múltiplas Viewports** para exibir diferentes regiões da cena, em diferentes posições no dispositivo

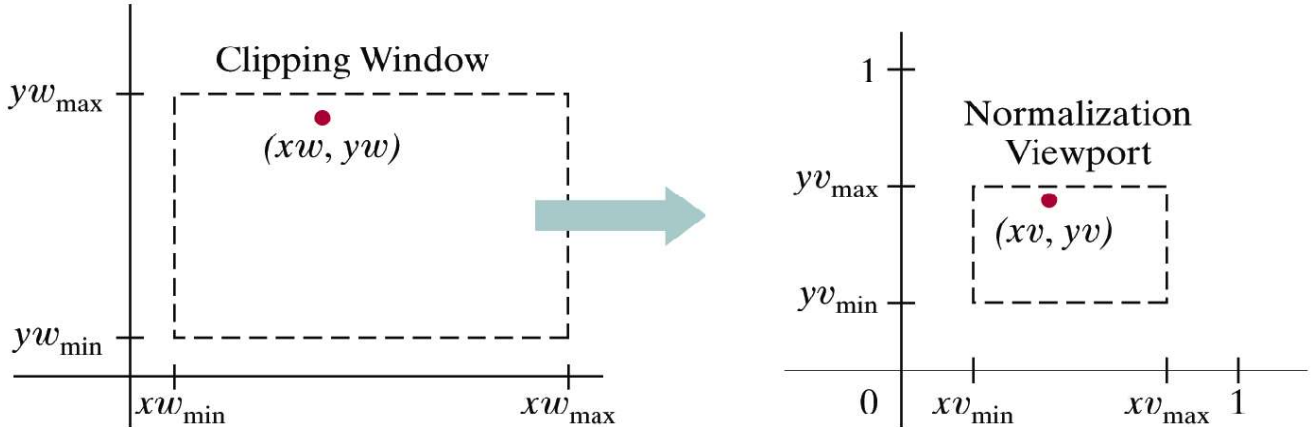
Janela de Recorte e *Viewport*



- Considerando uma *clipping window* e uma *viewport* definidas como na figura, i.e. retângulos de (xw_{min}, yw_{min}) a (xw_{max}, yw_{max}), e de (xv_{min}, yv_{min}) a (xv_{max}, yv_{max})
- Como mapear o conteúdo da *window* para a *viewport*, preservando as dimensões e posições relativas dos objetos na cena?

Mapeando a Janela de Recorte em uma Viewport

- Como mapear o conteúdo da *clipping window* para a *viewport* preservando as dimensões e posições relativas dos objetos na cena?



- Um ponto qualquer de coordenadas (xw, yw) será mapeado para um ponto de coordenadas (xv, yv)

Mapeando a Janela de Recorte em uma Viewport

- Para mapear as coordenadas (xw, yw) de um ponto na janela de recorte (WCS) nas coordenadas (xv, yv) correspondentes na *viewport*, a transformação é dada por:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

- Resolvendo para as coordenadas (xv, yv) na *viewport* tem-se

$$xv = S_x \cdot xw + t_x$$

$$yv = S_y \cdot yw + t_y$$

Mapeando a Janela de Recorte em uma Viewport

- Em que os fatores de escala são

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

- E os fatores de translação são

$$t_x = \frac{xw_{max} \cdot xv_{min} - xw_{min} \cdot xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max} \cdot yv_{min} - yw_{min} \cdot yv_{max}}{yw_{max} - yw_{min}}$$

Mapeando a Janela de Recorte em uma Viewport

- Essencialmente, a transformação é dada pela sequencia de transformações geométricas que mapeiam o retângulo da *Janela de Recorte* no retângulo da *viewport*
- Como:
  1. Escala a *Janela de Recorte* de modo que tenha o mesmo tamanho da *viewport*. Escala com pivô no ponto (xw_{min}, yw_{min})
  2. Translada a *Janela de Recorte* de modo que o ponto (xw_{min}, yw_{min}) fique posicionado em (xv_{min}, yv_{min})

Mapeando a Janela de Recorte em uma Viewport

- Assim, a matriz de escala é dada por

$$\mathbf{S} = \begin{bmatrix} s_x & 0 & xw_{min}(1 - s_x) \\ 0 & s_y & yw_{min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

- E a matriz de translação é dada por

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & xv_{min} - xw_{min} \\ 0 & 1 & yv_{min} - yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeando a Janela de Recorte em uma Viewport

- A matriz composta de transformação é dada por

$$\mathbf{M}_{window,viewport} = \mathbf{T} \cdot \mathbf{S}$$

- Igual a

$$\mathbf{M}_{window,normviewport} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Com os fatores s_x, s_y, t_x e t_y apresentados anteriormente

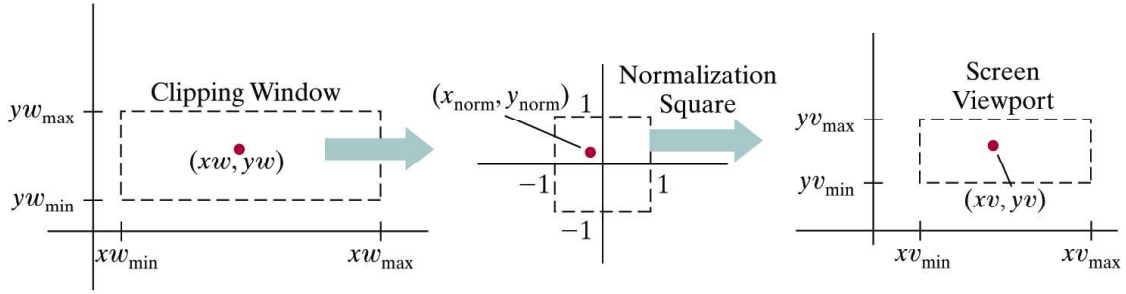
Mapeando a Janela de Recorte em uma Viewport

- Essa matriz de transformação implementa a viewing transformation 2D
- Nesse mapeamento, as posições relativas dos objetos na cena são preservadas
  - Um objeto contido na *Janela de Recorte* estará contido na *viewport*
  - Um objeto no centro da *Janela de Recorte* estará no centro da *viewport*

- As dimensões e proporções relativas dos objetos serão preservadas se *viewport* e *Janela de Recorte* tiverem a mesma razão de aspecto (mesma relação entre as dimensões horizontal e vertical)
  - Em outras palavras, a escala deve ser uniforme, i.e., $s_x = s_y$

Mapeando a Janela de Recorte em um Quadrado Normalizado

- Outra abordagem para obter a viewing transformation 2D consiste em mapear a *Janela de Recorte* para um quadrado normalizado, fazer o recorte da cena em relação a esse quadrado, e então transferir a descrição da cena para a *viewport* especificada no sistema de coordenadas da tela



- É simples fazer o recorte dos objetos (ou partes) fora dos limites $x = \pm 1$ e $y = \pm 1$
- É simples mapear os conteúdos do quadrado normalizado para a *viewport* no dispositivo

Mapeando a Janela de Recorte em um Quadrado Normalizado

- Para mapear o conteúdo da *Janela de Recorte* para o quadrado normalizado procedemos similarmente à transformação *window-viewport*
- Este é um caso particular em que $xv_{min} = yv_{min} = -1$ e $xv_{max} = yv_{max} = +1$

$$\mathbf{M}_{window,normsquare} = \begin{bmatrix} \frac{2}{xw_{max} - xw_{min}} & 0 & -\frac{xw_{max} + xw_{min}}{xw_{max} - xw_{min}} \\ 0 & \frac{2}{yw_{max} - yw_{min}} & -\frac{yw_{max} + yw_{min}}{yw_{max} - yw_{min}} \\ 0 & 0 & 1 \end{bmatrix}$$

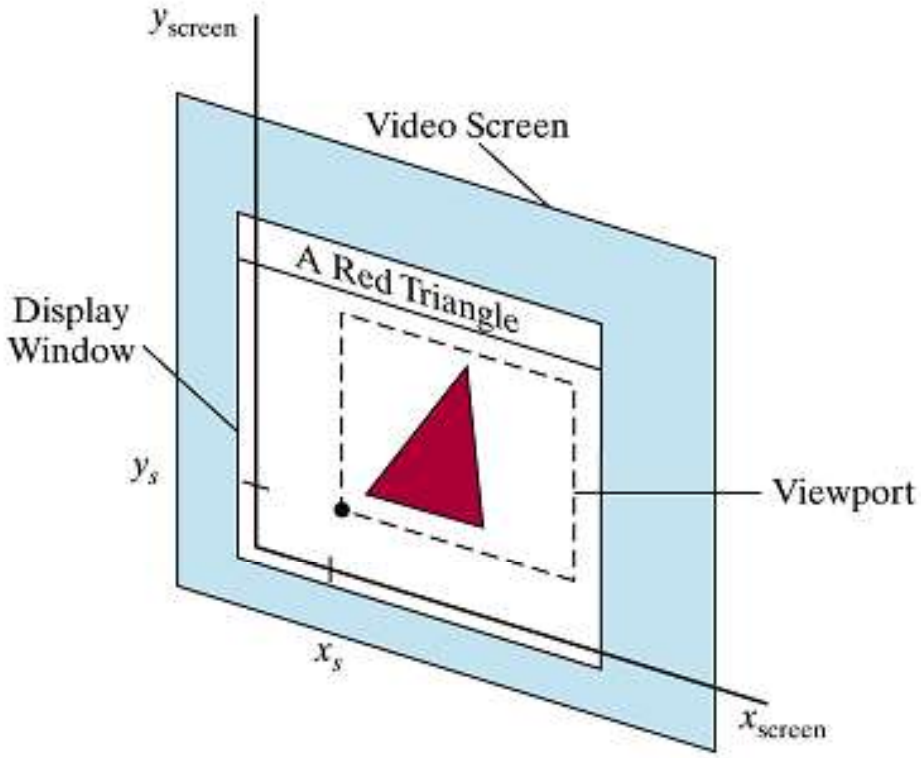
Mapeando a Janela de Recorte em um Quadrado Normalizado

- Analogamente, depois de aplicados os algoritmos de recorte, o conteúdo do quadrado normalizado de tamanho 2 é mapeado na *viewport*
- Neste caso, $xw_{min} = yw_{min} = -1$ e $xw_{max} = yw_{max} = -1$

$$\mathbf{M}_{normsquare,viewport} = \begin{bmatrix} \frac{xv_{max} - xv_{min}}{2} & 0 & \frac{xv_{max} + xv_{min}}{2} \\ 0 & \frac{yv_{max} - yv_{min}}{2} & \frac{yv_{max} + yv_{min}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Mapeando a Janela de Recorte em um Quadrado Normalizado

- O último passo consiste em posicionar a área da *viewport* na janela da tela



Sumário

1. Janela de Recorte (Cena 2D)
2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto
3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D

Modo de Projeção OpenGL

- Antes de definir a *Janela de Recorte* e a *viewport*, é necessário definir que a matriz em uso é a matriz de projeção

```
1 glMatrixMode(GL.GL_PROJECTION);
```

- Não esqueça que as transformações são cumulativas, então quando necessário carregar a matriz identidade

```
1 glLoadIdentity();
```

Definindo a Janela de Recorte

- A *Janela de Recorte* é definida por

```
1 glOrtho2D(GLfloat xwmin, GLfloat xwmax, GLfloat ywmin, GLfloat ywmax);
```

- Se a Janela de Recorte não for especificada, as coordenadas padrão serão $xw_{min} = yw = min = -1$ e $xw_{max} = yw_{max} = +1$
  - O processo de recorte ocorre em um quadrado normalizado entre -1 e 1

Definindo a Viewport

- A *viewport* é definida e posicionada por

```
1 glViewport(GLint xvmin, Glint yvmin, GLsizei vpWidth, GLsizei vpHeight);
```

- Todos os parâmetros são dados no sistema de coordenadas da tela, relativas a janela de visão
  - $(xvmin, yvmin)$: canto inferior esquerdo
  - $vpWidth$ e $vpHeight$: largura e altura da *viewport*

Sumário

1. Janela de Recorte (Cena 2D)
2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto
3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D

Exemplo

```
1  public class Renderer implements GLEventListener {
2
3    void desenha_objeto(GL gl) {
4      gl.glBegin(GL.GL_TRIANGLES); //desenha um triangulo
5        gl.glVertex2i(50, -50);
6        gl.glVertex2i(0, 50);
7        gl.glVertex2i(-50, -50);
8      gl.glEnd();
9
10     gl.glBegin(GL.GL_LINE_LOOP); //desenha um quadrado
11       gl.glVertex2i(-99, -99);
12       gl.glVertex2i(99, -99);
13       gl.glVertex2i(99, 99);
14       gl.glVertex2i(-99, 99);
15     gl.glEnd();
16   }
17
18   public void init(GLAutoDrawable drawable) {
19     GL gl = drawable.getGL();
20     gl.glClearColor(1.0f, 1.0f, 1.0f, 0.0f); //define cor de fundo
21     gl.glMatrixMode(GL.GL_PROJECTION); //carrega a matriz de projeo
22     gl.glLoadIdentity(); //l a matriz identidade
23
24     GLU glu = new GLU();
25     glu.gluOrtho2D(-100.0, 100.0, -100.0, 100.0); //define janela de recorte
26   }
27
28   ...
29 }
```

Codificando

```
1  public void display(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3    gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
4
5    gl.glMatrixMode(GL.GL_MODELVIEW); //matrix em uso: modelview
6    gl.glLoadIdentity();
7
8    gl.glViewport(10, 10, 200, 200); //define a viewport
9    gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
10   desenha_objeto(gl); //desenha o objeto
11
12   gl.glViewport(310, 10, 100, 100); //define a viewport
13   gl.glColor3f(0.0f, 1.0f, 0.0f); //altera o atributo de cor
14   gl.glRotatef(90, 0, 0, 1);
15   desenha_objeto(gl); //desenha o objeto
16
17   gl.glFlush(); //processa as rotinas OpenGL o mais rpido possvel
18 }
```

Codificando

```
1  public static void main(String[] args) {
2    //acelera o rendering
3    GLCapabilities caps = new GLCapabilities();
4    caps.setDoubleBuffered(true);
5    caps.setHardwareAccelerated(true);
6
7    //cria o painel e adiciona um ouvinte GLEventListener
8    GLCanvas canvas = new GLCanvas(caps);
9    canvas.addGLEventListener(new Renderer());
10
11   //cria uma janela e adiciona o painel
12   JFrame frame = new JFrame("Aplicao JOGL Simples");
13   frame.getContentPane().add(canvas);
14   frame.setSize(500, 275);
15   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17   //inicializa o sistema e chama display() a 60 fps
18   Animator animator = new FPSAnimator(canvas, 60);
19   frame.setLocationRelativeTo(null);
20   frame.setVisible(true);
21   animator.start();
22 }
```

Sumário

1. Janela de Recorte (Cena 2D)

2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto

3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D

Mantendo Razão de Aspecto

```
1  public class Renderer implements GLEventListener {
2
3    public void init(GLAutoDrawable drawable) {
4      GL gl = drawable.getGL();
5      gl.glClearColor(1.0f, 1.0f, 1.0f, 0.0f); //define cor de fundo
6    }
7
8    void desenha_objeto(GL gl) {
9      gl.glBegin(GL.GL_TRIANGLES); //desenha um triangulo
10     gl.glVertex2i(50, -50);
11     gl.glVertex2i(0, 50);
12     gl.glVertex2i(-50, -50);
13     gl.glEnd();
14
15     gl.glBegin(GL.GL_LINE_LOOP); //desenha um quadrado
16     gl.glVertex2i(-55, -55);
17     gl.glVertex2i(55, -55);
18     gl.glVertex2i(55, 55);
19     gl.glVertex2i(-55, 55);
20     gl.glEnd();
21   }
22
23   public void display(GLAutoDrawable drawable) {
24     GL gl = drawable.getGL();
25     gl.glClear(GL.GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
26
27     gl.glMatrixMode(GL.GL_MODELVIEW); //matrix em uso: modelview
28     gl.glLoadIdentity();
29
30     gl.glColor3f(1.0f, 0.0f, 0.0f); //altera o atributo de cor
31     desenha_objeto(gl); //desenha o objeto
32
33     gl.glFlush(); //processa as rotinas OpenGL o mais rpido possvel
34   }
35
36   ...
37 }
```

Preservando a Razão de Aspecto

```
1  public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height
       ) {
2    GL gl = drawable.getGL();
3
4    // Evita a diviso por zero
5    if (height == 0) {
6      height = 1;
7    }
8
9    // Inicializa o sistema de coordenadas
10   gl.glMatrixMode(GL.GL_PROJECTION);
11   gl.glLoadIdentity();
12
13   // Estabelece a janela de recorte
14   GLU glu = new GLU();
15   if (width <= height) {
16     glu.gluOrtho2D(-100.0f, 100.0f, (-100.0f * height) / width, (100.0f * height
           ) / width);
17   } else {
18     glu.gluOrtho2D((-100.0f * width) / height, (100.0f * width) / height, -100.0
           f, 100.0f);
19   }
20 }
```

Preservando a Razão de Aspecto

```
1  public static void main(String[] args) {
2    //acelera o rendering
3    GLCapabilities caps = new GLCapabilities();
4    caps.setDoubleBuffered(true);
5    caps.setHardwareAccelerated(true);
6
7    //cria o painel e adiciona um ouvinte GLEventListener
8    GLCanvas canvas = new GLCanvas(caps);
9    canvas.addGLEventListener(new Renderer());
10
11   //cria uma janela e adiciona o painel
12   JFrame frame = new JFrame("Aplicao JOGL Simples");
13   frame.getContentPane().add(canvas);
14   frame.setSize(500, 500);
15   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17   //inicializa o sistema e chama display() a 60 fps
18   Animator animator = new FPSAnimator(canvas, 60);
19   frame.setLocationRelativeTo(null);
20   frame.setVisible(true);
21   animator.start();
22 }
```

Sumário

1. Janela de Recorte (Cena 2D)

2. Programação OpenGL
   - Múltiplas Viewports
   - Mantendo Razão de Aspecto

3. Algoritmos de Recorte
   - Recorte de Ponto 2D
   - Recorte de Linha 2D
   - Recorte de Polígonos 2D
   - Recorte de Outras Primitivas 2D