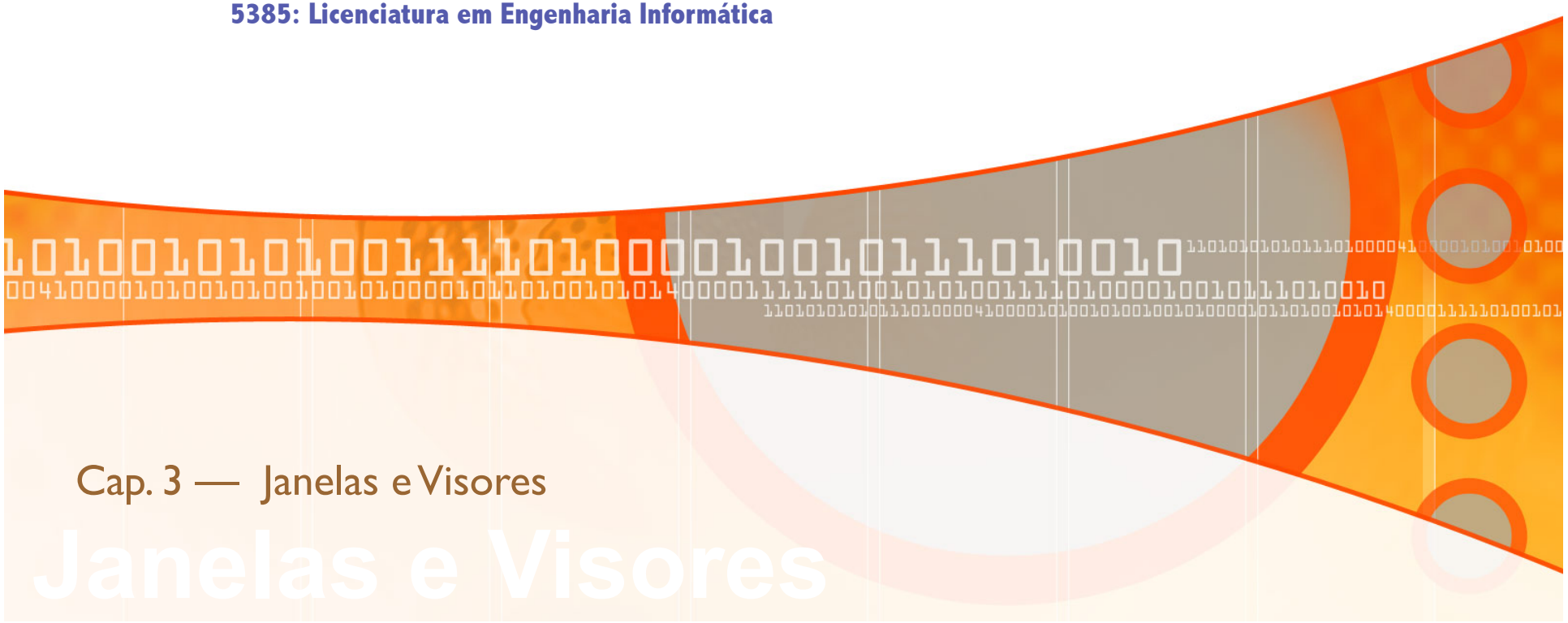
Computação Gráfica

5385: Licenciatura em Engenharia Informática



Cap. 3 — Janelas e Visores

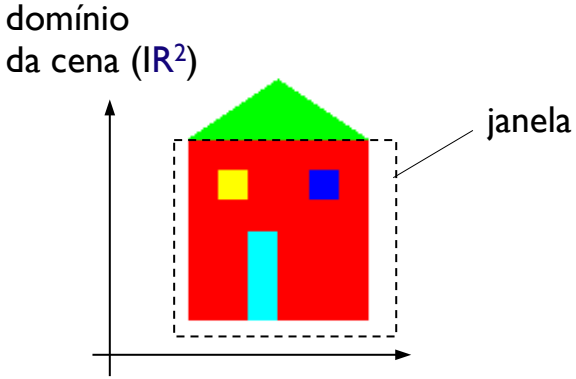
Janelas e Visores

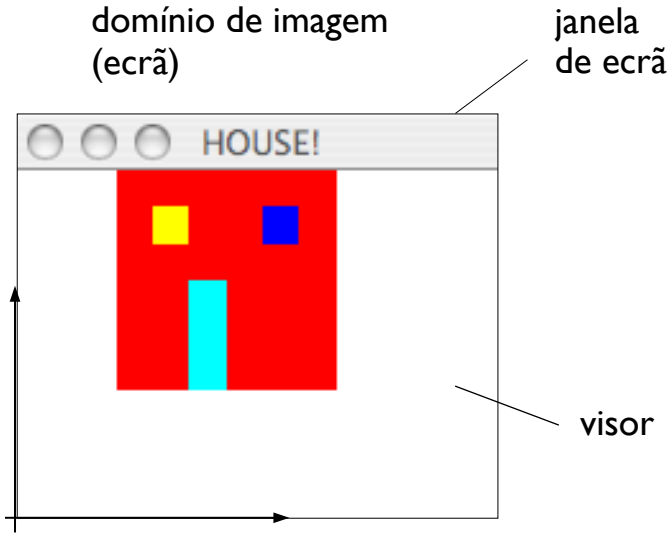
Sumário

…:

- Definições básicas em 2D:
  - sistema de coordenadas globais no domínio da cena
    - Janela = é parte do domínio da cena (por exemplo, $\mathbb{R}^2$)
  - sistema de coordenadas de ecrã no domínio de imagem
    - Visor = é parte do domínio de imagem (numa janela de desktop)
- Transformação janela-visor (*window-viewport*)
- Transformação janela-visor em OpenGL
- *Graphics pipeline* (ou *rendering pipeline*)
- Mapeamento de uma cena em vários visores
- Efeitos de zooming e panning através da transformação janela-visor
- Estratégias para a manutenção automática das proporções na transformação janela-visor
- Exemplos em OpenGL e exercícios com discussão de ideias.

Definições





Sistema de Coordenadas Globais (ou Domínio da Cena)

- É o espaço em que se encontram os objectos geométricos.
- É neste espaço que o modelo de aplicação é definido; por exemplo IR^2.
- É neste espaço que a **geometria** do objecto é definida.

Janela do Domínio da Cena (ou Subdomínio de Cena)

- Rectângulo que define a parte do domínio da cena que pretendemos visualizar.

Sistema de Coordenadas do Imagem (Domínio de Imagem)

- Espaço no qual a imagem é mostrada; por exemplo **800x600** pixéis.
- Espaço no qual a **imagem rasterizada** do objecto é definida.

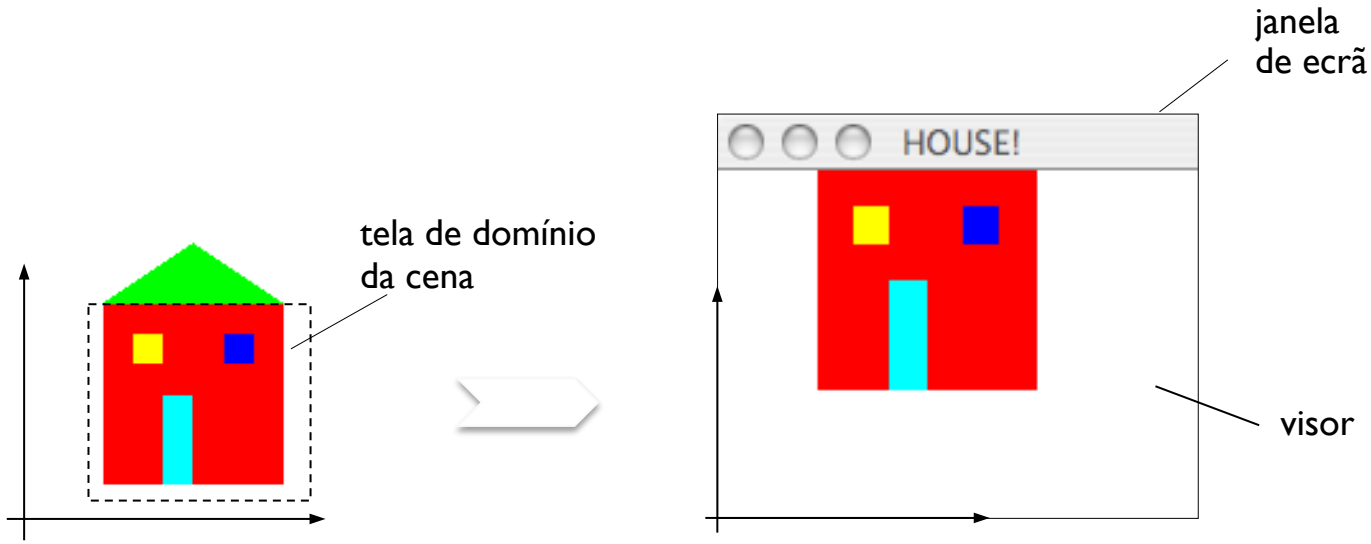
Visor de Imagem (Subdomínio de Imagem)

- Representação visual do sistema de coordenadas do ecrã para sistemas de saída baseados em janelas (sistema de coordenadas move-se com a janela de ecrã)

Definições (cont.)

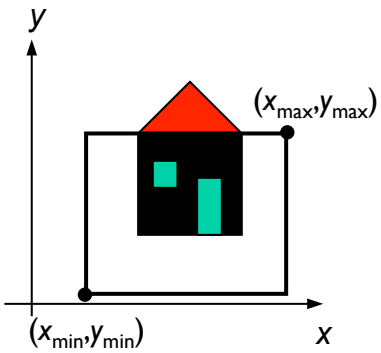
Transformações de visualização (*viewing transformations*)

- Processo de mapeamento de uma janela do domínio de cena (*world coordinates*) para um visor (*screen coordinates*) .
- Portanto, em princípio, a mesma cena pode ser mapeada para diferentes visores (ou *viewports*) dentro do ecrã ou duma janela de ecrã.

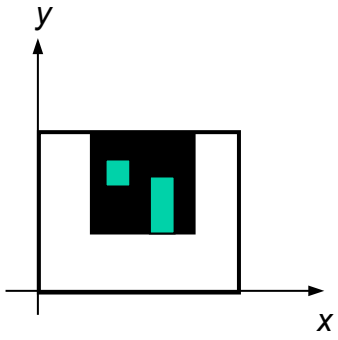


Transformação janela-visor

Dada uma tela e um visor, qual é a matriz de transformação que mapeia um ponto da tela em coordenadas globais num pixel do visor em coordenadas de ecrã? Esta matriz pode ser dada pela composição de 3 transformações:

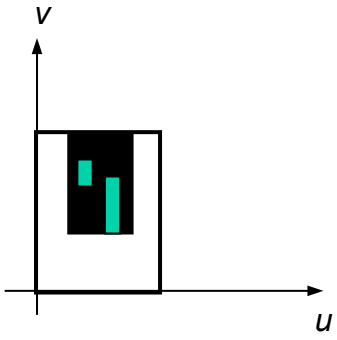


janela em coordenadas globais (ou de domínio de cena)



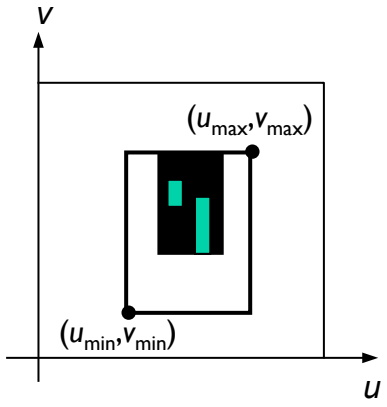
janela transladada para a origem

$$T(-x_{\min},-y_{\min})$$



tela com tamanho igual ao tamanho do visor

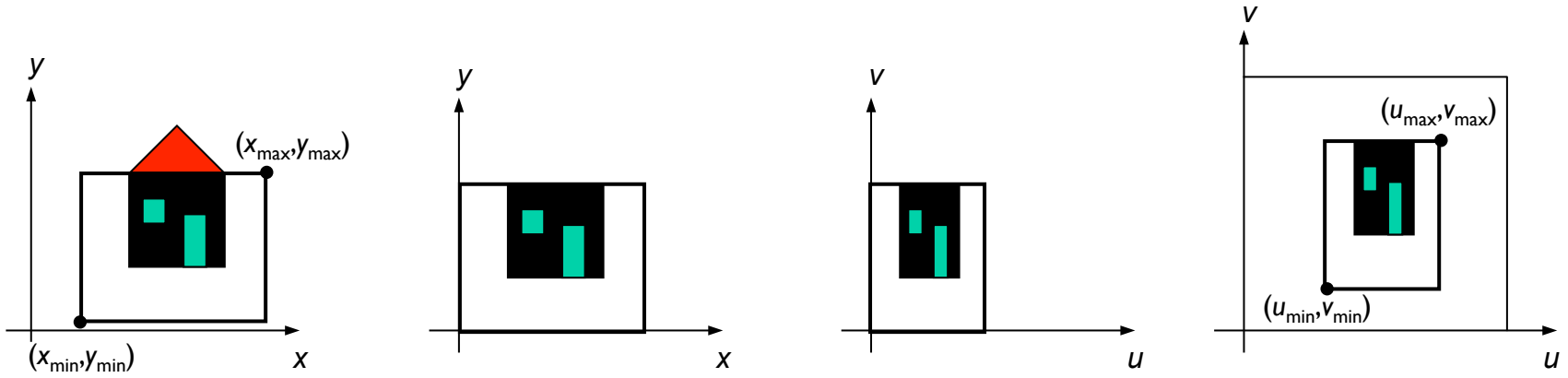
$$S(\frac{u_{\max}-u_{\min}}{x_{\max}-x_{\min}},\frac{v_{\max}-v_{\min}}{y_{\max}-y_{\min}})$$



visor transladado por (u_{min},v_{min}) para a posição final

$$T(u_{\min},v_{\min})$$

Transformação janela-visor: representação matricial

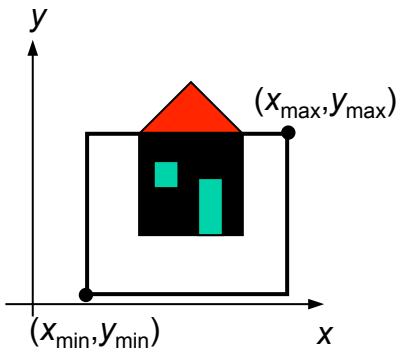


$$T(-x_{\min},-y_{\min}) \qquad S(\frac{u_{\max}-u_{\min}}{x_{\max}-x_{\min}},\frac{v_{\max}-v_{\min}}{y_{\max}-y_{\min}}) \qquad T(u_{\min},v_{\min})$$

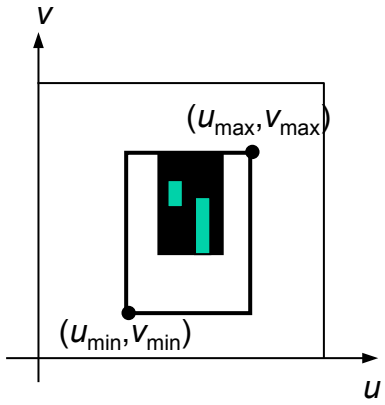
$$M_{wv} = T(u_{\min},v_{\min}).S(\frac{u_{\max}-u_{\min}}{x_{\max}-x_{\min}},\frac{v_{\max}-v_{\min}}{y_{\max}-y_{\min}}).T(-x_{\min},-y_{\min})$$

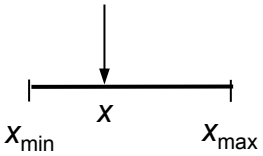
$$= \begin{bmatrix} 1 & 0 & u_{\min} \\ 0 & 1 & v_{\min} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \frac{u_{\max}-u_{\min}}{x_{\max}-x_{\min}} & 0 & 0 \\ 0 & \frac{v_{\max}-v_{\min}}{y_{\max}-y_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_{\min} \\ 0 & 1 & -y_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

Transformação janela-visor: como é efetuada?



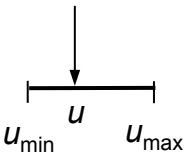
Mantendo a proporcionalidade no mapeamento de (x,y) para (u,v)

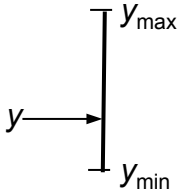




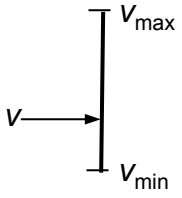
translação *variação de tamanho* *translação*

$$\frac{x - x_{min}}{x_{max} - x_{min}} = \frac{u - u_{min}}{u_{max} - u_{min}} \Leftrightarrow u = (x - x_{min}).\frac{u_{max} - u_{min}}{x_{max} - x_{min}} + u_{min}$$

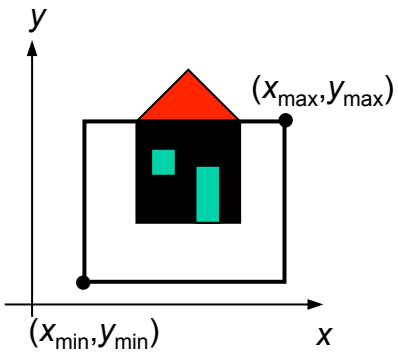




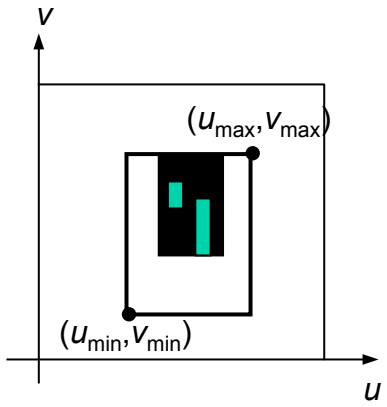
$$\frac{y - y_{min}}{y_{max} - y_{min}} = \frac{v - v_{min}}{v_{max} - v_{min}} \Leftrightarrow v = (y - y_{min}).\frac{v_{max} - v_{min}}{y_{max} - y_{min}} + v_{min}$$



Transformação janela-visor: exemplo



tela(10.0,5.0,40.0,30.0)



visor(100,50,250,300)

$$u = (x - 10.0).\frac{250 - 100}{40.0 - 10.0} + 100 \qquad \lambda_x = \frac{250 - 100}{40.0 - 10.0} = 5.0$$

$$v = (y - 5.0).\frac{300 - 50}{30.0 - 5.0} + 50 \qquad \lambda_y = \frac{300 - 50}{30.0 - 5.0} = 10.0$$

Transformação janela-visor: em OpenGL

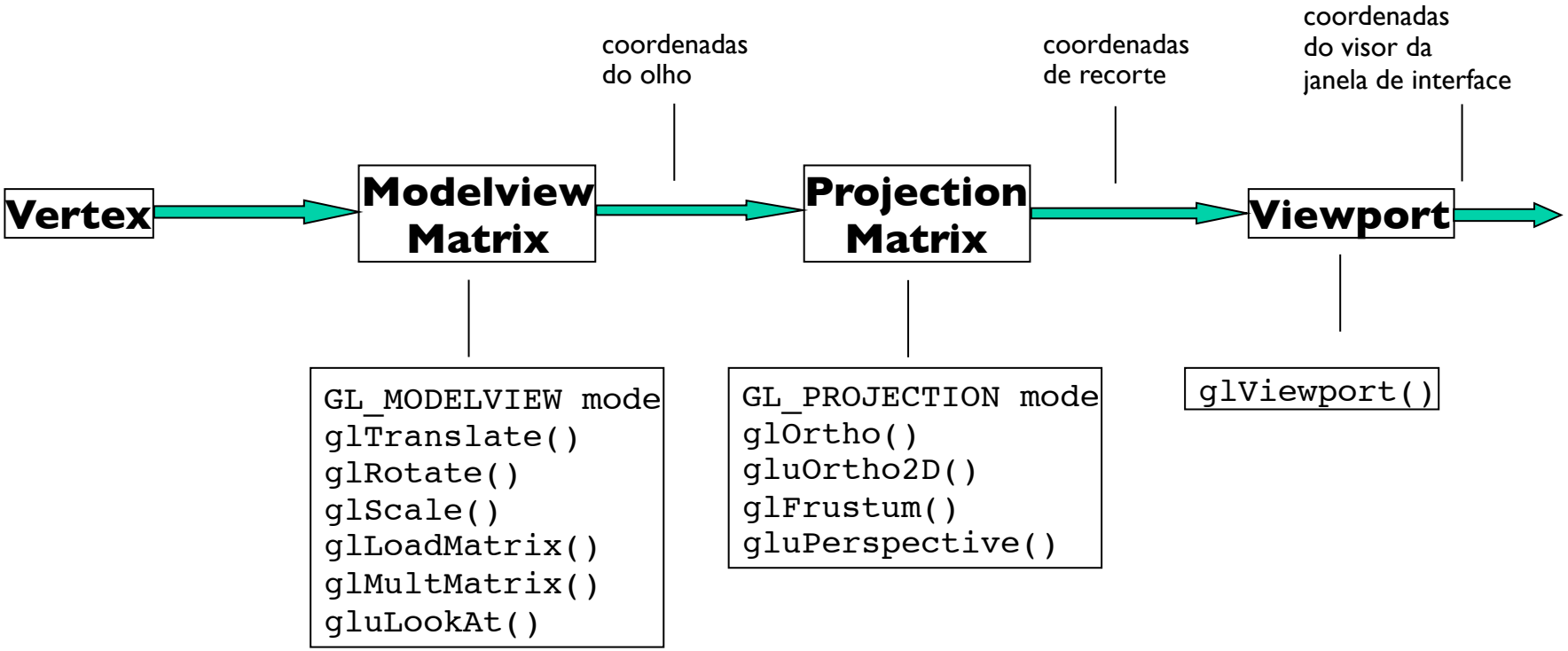
`gluOrtho2D(left, right, bottom, top)`

- Define um rectângulo de visualização ortogonal 2D ou *tela* de domínio de cena. É definida por dois planos verticais *left* e *right* e dois planos horizontais *bottom* e *top*.
- A tela é (-1,1,-1,1) por defeito.
- Define uma matriz de projecção ortogonal 2D.
- Define ainda a transformação tela-visor, o que requer a definição do visor através da seguinte função.

`glViewport(x, y, width, height)`

- Define o visor na *janela de ecrã*, onde x,y especificam o canto inferior esquerdo e width, height as suas dimensões.
- Por defeito, o visor ocupa a área gráfica total da janela de ecrã.
- Podem existir vários visores dentro da janela de ecrã.

Graphics pipeline em OpenGL



EXEMPLOS EM OPENGL

- Visor por omissão
- Um único visor
- Dois visores

Exemplo 1: visor por omissão

- Como já foi referido, se a função **glViewport(**x, y, width, height**) NÃO** é EXPLICITAMENTE usada no programa, o visor por omissão é toda a área gráfica da janela de ecrã.
- Veja-se o exemplo do próximo programa que retrata esta situação. O visor por omissão tem a área 500x500 e é definido por **glutInitWindowSize**(500,500) no programa principal.

```
/* Using the default viewport */
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h
#include <stdlib.h>
```

```
void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );

        // Sets up the PROJECTION matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
        // also sets up world window
     gluOrtho2D(0.0,50.0,-10.0,40.0);

        // Draw BLUE rectangle
     glColor3f( 0, 0, 1 );
     glRectf(0.0,0.0,10.0,30.0);

        // display rectangles
     glutSwapBuffers();
}    // end of draw()
```

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}
```

```
int main(int argc,  char ** argv)
{
   glutInit(&argc, argv);
   // Double Buffered RGB display
   glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
   // Set window size
   glutInitWindowSize( 500,500 );
   // Default viewport
   glutCreateWindow("Default viewport");
   // the display callback
   glutDisplayFunc(draw);
   // the keyboard callback
   glutKeyboardFunc(keyboard);
   // Start the Main Loop
   glutMainLoop();
   return 0;
}
```

Exemplo 2: 1 visor

- Um visor é EXPLICITAMENTE definido pela função **glViewport(**x,y, width,height**).**
- O visor pode ou não ocupar toda a área gráfica da janela de ecrã. Podem existir vários visores na janela de ecrã simultaneamente.
- A janela de ecrã e os seus visores são definidos antes da tela de domínio de cena, sendo esta última definida através de **gluOrtho2D**(left, right, bottom, top) porque esta função também define a transformação tela-visor.

```
/* Using a single viewport */
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
#include <stdlib.h>

void draw(){
   // Make background colour yellow
   glClearColor( 100, 100, 0, 0 );
   glClear ( GL_COLOR_BUFFER_BIT );
   // Sets up viewport spanning the
   // left-bottom quarter of the
   // interface window
   glViewport(0,0,250,250);
   // Sets up the PROJECTION matrix
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   // also sets up world window
   gluOrtho2D(0.0,50.0,-10.0,40.0);
   // Draw BLUE rectangle
   glColor3f( 0, 0, 1 );
   glRectf(0.0,0.0,10.0,30.0);
   // display rectangles
   glutSwapBuffers();
}  // end of draw()
```

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
   glutInit(&argc, argv);
   // Double Buffered RGB display
   glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
   // Set window size
   glutInitWindowSize( 500,500 );
   // Default viewport
   glutCreateWindow("Default viewport");
   // the display callback
   glutDisplayFunc(draw);
   // the keyboard callback
   glutKeyboardFunc(keyboard);
   // Start the Main Loop
   glutMainLoop();
   return 0;
}
```

Exemplo 3: 2 visores

```
/* Using two viewports */
#include <GLUT/glut.h>

void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );
        // Sets up FIRST viewport
     glViewport(0,0,250,250);
        // Sets up the PROJECTION matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     // also sets up world window
     gluOrtho2D(0.0,50.0,-10.0,40.0);
     // Draw BLUE rectangle
     glColor3f( 0, 0, 1 );
     glRectf(0.0,0.0,10.0,30.0);
        // Sets up SECOND viewport
     glViewport(250,250,250,250);
        // Sets up the PROJECTION matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     // also sets up world window
     gluOrtho2D(0.0,50.0,-10.0,40.0);
     // Draw RED rectangle
     glColor3f( 1, 0, 0 );
     glRectf(0.0,0.0,10.0,30.0);
        // display rectangles
     glutSwapBuffers();
}
```

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
     if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
   glutInit(&argc, argv);
   // Double Buffered RGB display
   glutInitDisplayMode(GLUT_RGB |GLUT_DOUBLE);
   // Set window size
   glutInitWindowSize( 500,500 );
   // two viewports
   glutCreateWindow("Two viewports");
   // the display callback
   glutDisplayFunc(draw);
   // the keyboard callback
   glutKeyboardFunc(keyboard);
   // Start the Main Loop
   glutMainLoop();
   return 0;
}
```

Transformação janela-visor: efeitos relevantes

Zooming:

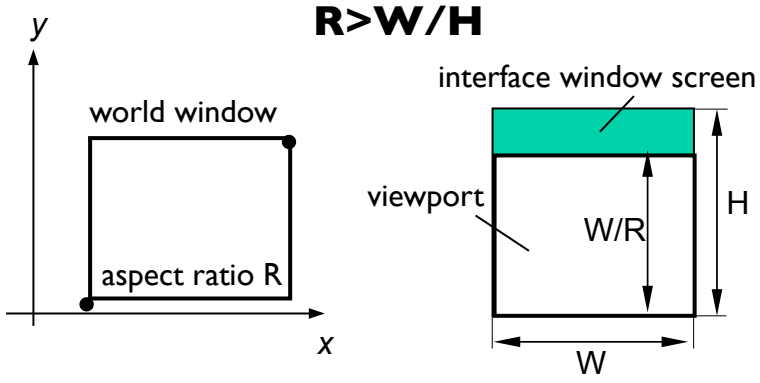
- Quando a janela aumenta de tamanho e o visor mantém o seu tamanho, a cena renderizada no visor aparece mais pequena;
- Da mesma forma, quando a janela diminui de tamanho, a cena representada no visor surge ampliada.

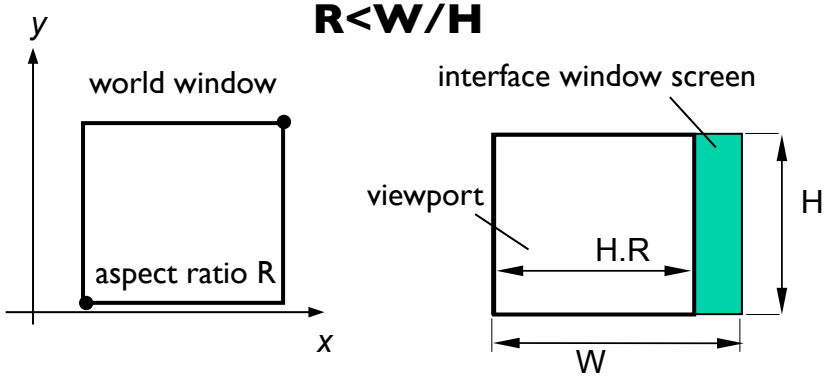
Panning:

- Quando se move a janela no domínio da cena, o que aparece no visor é uma parte diferente da cena.

Ativação automática do visor sem distorção de imagem

- Qual a maior imagem não-distorcida que ocupa o ecrã?
- R = Razão de aspecto da tela do domínio de cena
- Duas situações são possíveis:

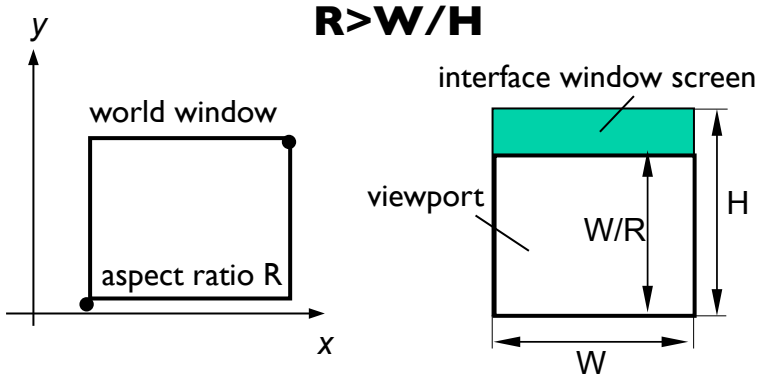


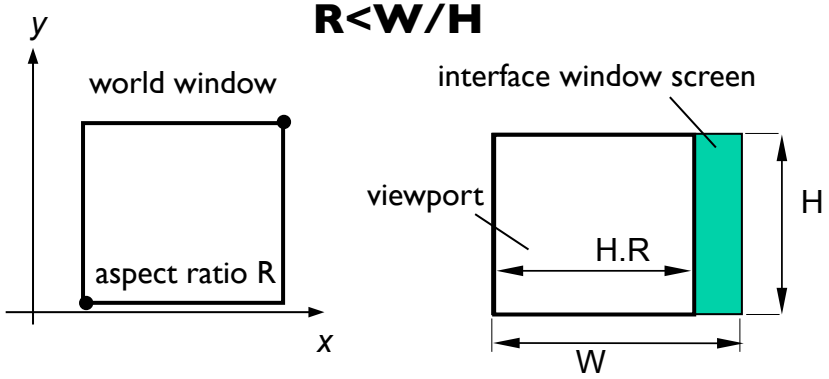


- A tela é pequena em altura mas ajustada à largura do visor da janela de ecrã, mas algum espaço sobrará em cima/baixo.
- Portanto, no máximo, o visor terá largura W e altura W/R.

- A tela é alta a estreita comparada com a janela de ecrã.
- O visor com a mesma razão de aspecto R ocupará toda área gráfica da janela de ecrã em altura, mas sobrará algum espaço à esquerda/direita.
- Portanto,no máximo, o visor terá largura H.R e altura H.

Ativação automática do visor sem distorção de imagem (cont.)



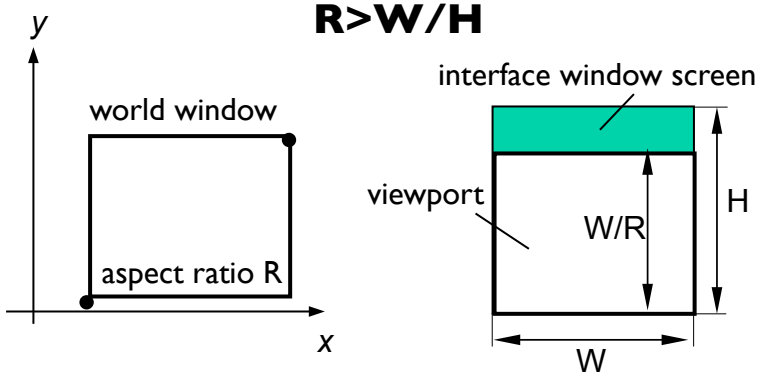



```
glViewport(0,0,W,W/R);
```

```
glViewport(0,0,H*R,H);
```

Exemplo 1: janela baixa

- Se a tela tem razão de aspecto R=2.0 e o visor da janela de ecrã tem altura H=200 e largura W=360, então W/H=1.8.
- Portanto, estamos no primeiro caso, e o visor é activado com 180 pixéis de altura e 360 pixéis de largura.

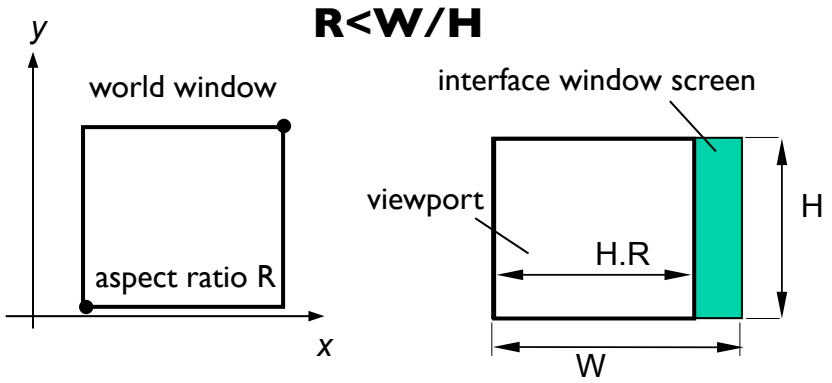



```
glViewport(0,0,W,W/R);

glViewport(0,0,360,360/2);
```

Exemplo 2: janela alta

- Se a tela tem razão de aspecto R=1.6 e o visor da janela de ecrã tem H=200 e W=360, então W/H=1.8.
- Portanto, estamos no segundo caso, e o visor é activado com 200 pixéis de altura e 320 pixéis de largura.




```
glViewport(0,0,H*R,H);

glViewport(0,0,320,200);
```

Estratégia para a manutenção automática das proporções na transformação janela-visor

Estratégia:

- O utilizador aumenta ou diminui o tamanho dum visor com w pixéis de largura e h pixéis de altura através do arrastamento para fora ou para dentro do canto inferior direito da janela de ecrã.
- Para evitar distorção, há que mudar o tamanho da tela do domínio de cena em conformidade.
- Para isso, assume-se *a priori* que a tela da cena é um quadrado cujos lados têm comprimento L.
- Uma solução possível é mudar a tela da cena sempre que o ecrã da janela de ecrã for alterada. Assim, a callback Glvoid reshape(GLsizei w, GLsizei h) tem de ser alterada por forma a incluir o código seguinte:

```
if (w <= h)
    gluOrtho2D(-L, L, -L * h/w, L * h/w);
else
    gluOrtho2D(-L * w/h, L * w/h, -L, L);
```

Exemplo: manutenção das proporções na transformação janela-visor

```
// Keyboard method to allow ESC key to quit
GLvoid reshape(GLsizei w, GLsizei h)
{
     GLfloat L = 100.0f;

     if (h == 0)               // prevent a divide by zero
     h=1;

     glViewport(0,0,w,h);           // set viewport to window dimensions

     glMatrixMode(GL_PROJECTION);  // reset projection matrix stack
     glLoadIdentity();// establish clipping volume (left, right, bottom, top, near, far)

     if (w <= h)
     gluOrtho2D(-L, L, -L * h/w, L * h/w);
     else
     gluOrtho2D(-L * w/h, L * w/h, -L, L);

     glMatrixMode(GL_MODELVIEW);   // reset model-view matrix stack
     glLoadIdentity();
}
```

Sumário:

…:

- Definições básicas em 2D:
  - sistema de coordenadas globais no domínio da cena
    - Janela = é parte do domínio da cena (por exemplo, $\mathbb{R}^2$)
  - sistema de coordenadas de ecrã no domínio de imagem
    - Visor = é parte do domínio de imagem (numa janela de desktop)
- Transformação janela-visor (*window-viewport*)
- Transformação janela-visor em OpenGL
- *Graphics pipeline* (ou *rendering pipeline*)
- Mapeamento de uma cena em vários visores
- Efeitos de zooming e panning através da transformação janela-visor
- Estratégias para a manutenção automática das proporções na transformação janela-visor
- Exemplos em OpenGL e exercícios com discussão de ideias.