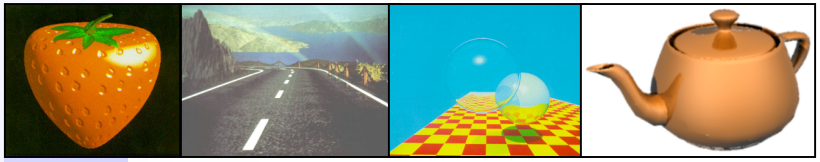
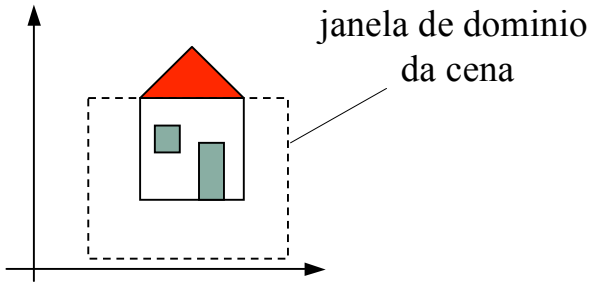


Cap. 4: Janelas, Visores & Recorte Gráfico

Sumário

- Definições básicas: sistema de coordenadas globais, sistema de coordenadas do ecrã; janela de domínio de cena; janela de interface e visores de saída gráfica.
- Transformação janela-visor
- Transformação janela-visor OpenGL
- Serializador do processo de renderização
- Serializador do processo de renderização 2D em OpenGL
- Recorte de segmentos de recta: algoritmo de Cohen-Sutherland
- Recorte de polígonos
- Recorte no serializador OpenGL

Definições

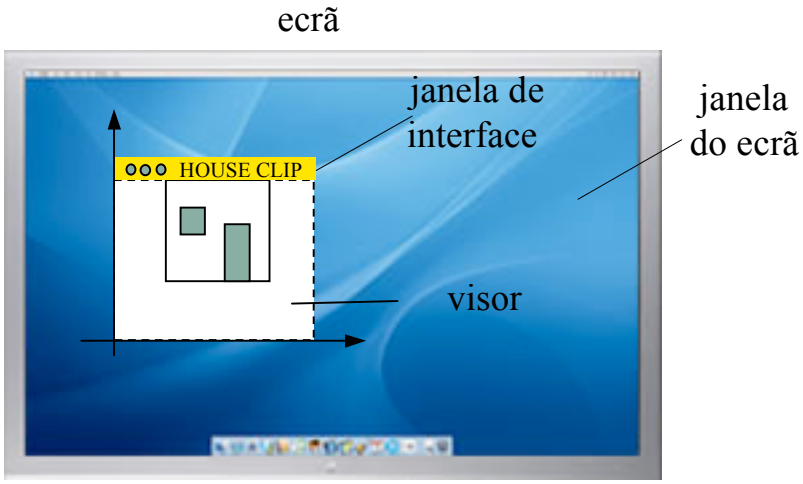


Sistema de Coordenadas Globais (Espaço do Objecto ou Domínio da Cena)

- É o referencial do espaço em que se encontram os objectos geométricos.
- É neste espaço que o modelo de aplicação é definido; por exemplo $\mathbf{R}^2$.
- É neste espaço que a **geometria** do objecto é definida.

Janela de Domínio da Cena (Subespaço do Domínio de Cena)

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



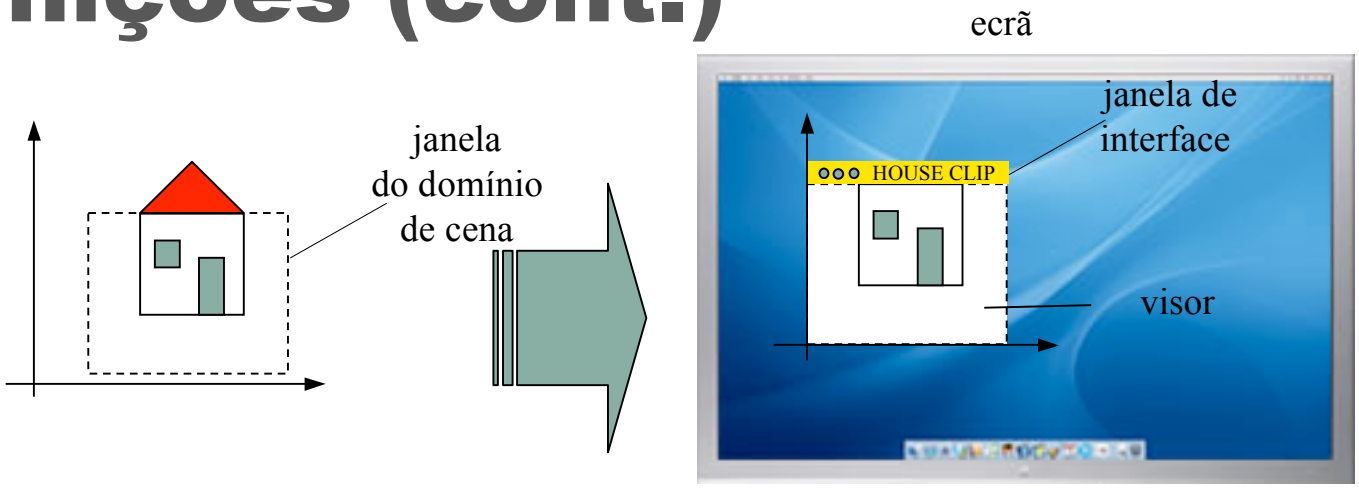
Sistema de Coordenadas do Ecrã (Espaço de Imagem)

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

Janela de Interface (Subespaço 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 interface)

Definições (cont.)



Transformações de visualização (viewing Transformations)

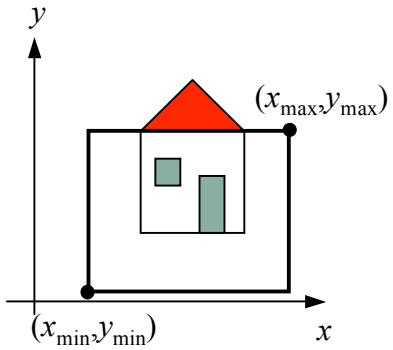
- Processo de mapeamento de uma janela do domínio de cena (*world coordinates*) para um visor (*screen coordinates*) .

Visor (Subespaço de Imagem)

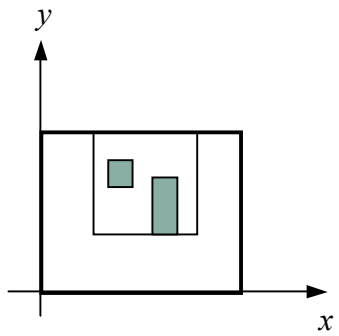
- Um rectângulo no ecrã rasterizado (ou janela de interface) que define onde a imagem irá aparecer, ou na totalidade do ecrã ou numa janela de interface.
- Portanto, em princípio, a mesma imagem pode ser replicada em diferentes visores (ou *viewports*) dentro do ecrã ou duma janela de interface.

Transformação Janela-Visor

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

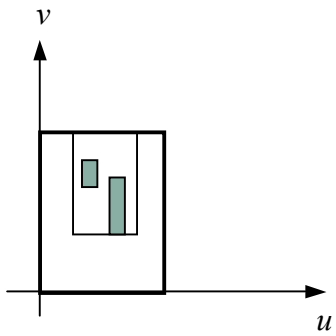


janela em coordenadas globais (ou de domínio de cena)



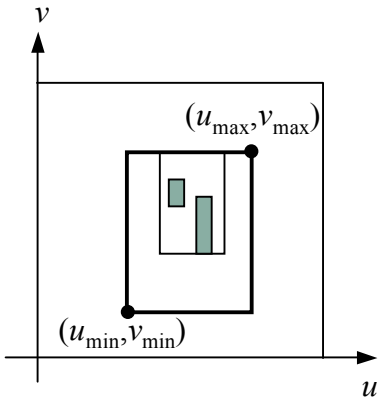
janela transladada para a origem

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



janela com tamanho alterado igual ao tamanho do visor

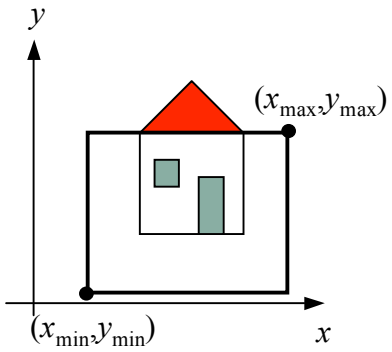
$$S(\frac{u_{max}-u_{min}}{x_{max}-x_{min}},\frac{v_{max}-v_{min}}{y_{max}-y_{min}})$$

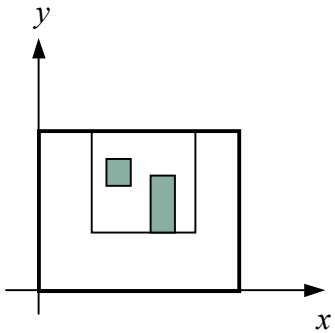


visor transladado por (u_{min},v_{min}) para a posição final

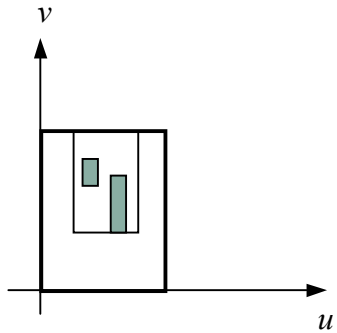
$T(u_{min},v_{min})$

Transformação Janela-Visor: representação matricial

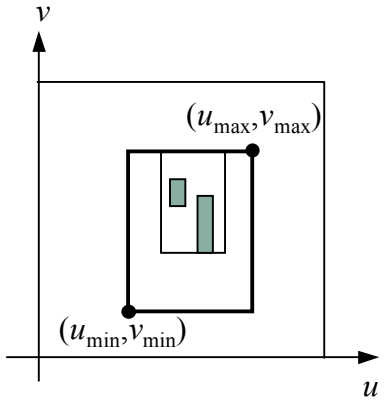




$T(-x_{min}, -y_{min})$



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

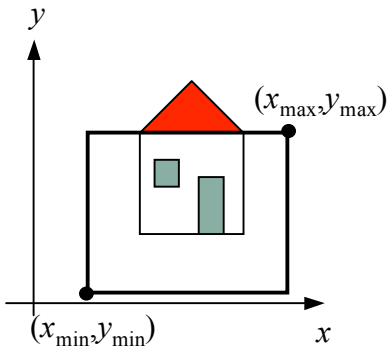


$T(u_{min}, v_{min})$

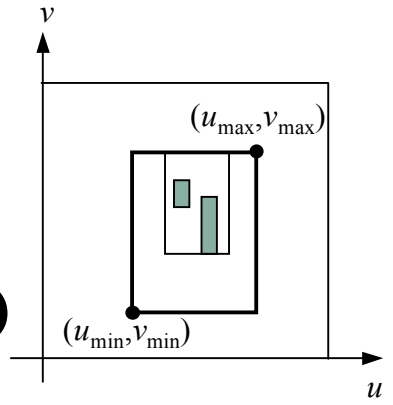
$$M_{wv} = T(u_{min}, v_{min}).S(\frac{u_{max} - u_{min}}{x_{max} - x_{min}}, \frac{v_{max} - v_{min}}{y_{max} - y_{min}}).T(-x_{min}, -y_{min})$$

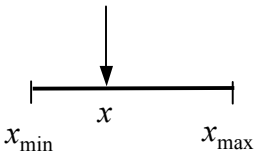
$$= \begin{bmatrix} 1 & 0 & u_{min} \\ 0 & 1 & v_{min} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \frac{u_{max} - u_{min}}{x_{max} - x_{min}} & 0 & 0 \\ 0 & \frac{v_{max} - v_{min}}{y_{max} - y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_{min} \\ 0 & 1 & -y_{min} \\ 0 & 0 & 1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

Transformação Janela-Visor: como é feita?



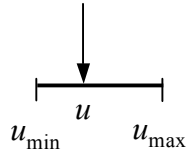
Mantendo a <u>proporcionalidade</u> no mapeamento de (x,y) para (u,v)

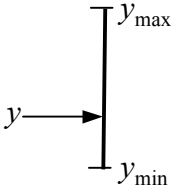




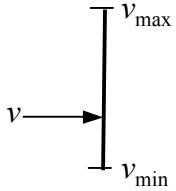
translação *variação de tamanho* *translação*

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

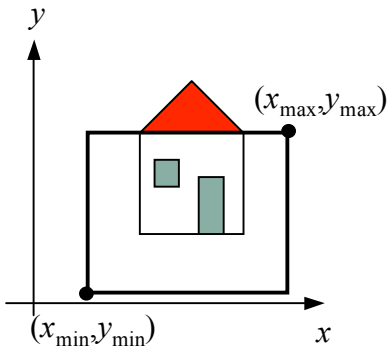


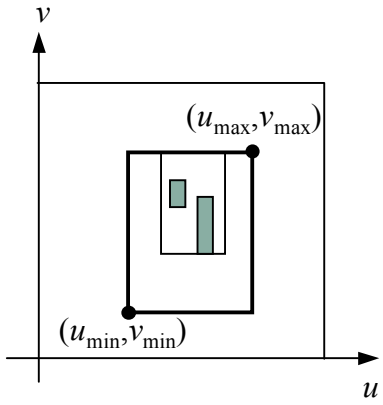


$$\frac{y - y_{min}}{y_{max} - y_{min}} = \frac{v - v_{min}}{v_{max} - v_{min}} \Leftrightarrow v = (y - y_{min}).\frac{v_{max} - v_{min}}{y_{max} - y_{min}} + v_{min}$$



Transformação Janela-Visor: exemplo





janela(10.0,2.0,40.0,30.0)

visor(100,50,250,300)

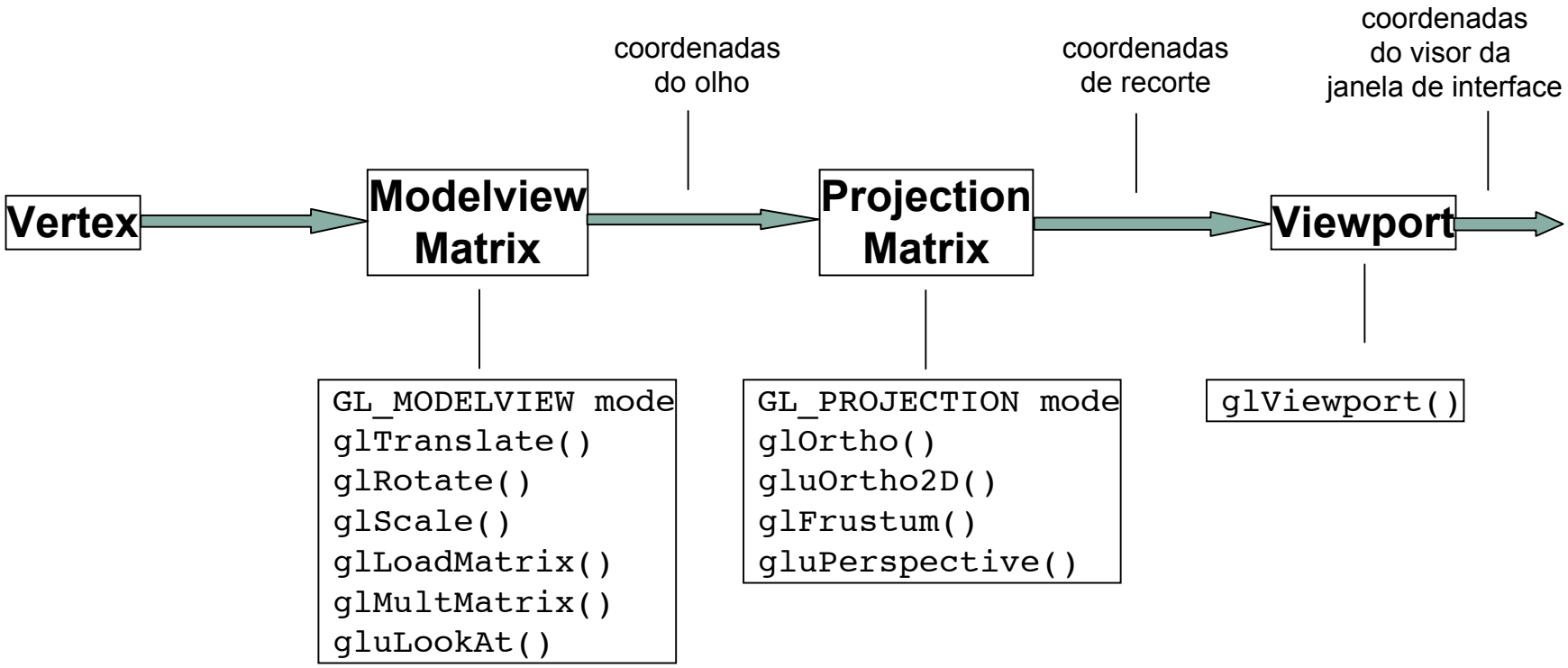
$$u = (x - 10.0).\frac{250 - 100}{40.0 - 10.0} + 100 \qquad \lambda_x = \frac{250 - 100}{40.0 - 10.0} = 5.0$$

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

Transformação Janela-Visor: em OpenGL

- `gluOrtho2D(left, right, bottom, top)`
  - Define uma região de visualização ortogonal 2D ou *janela* de domínio de cena. É definida por dois planos verticais de recorte `left` e `right` e dois planos horizontais de recorte `bottom` e `top`.
  - A janela é (-1,1,-1,1) por defeito.
  - Define uma matriz de projecção ortogonal 2-D.
  - Define ainda a transformação janela-visor, o que requer a definição do visor através da seguinte função:
- `glViewport(x, y, width, height)`
  - Define o visor na *janela de interface*, 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 interface.
  - Podem existir vários visores dentro da janela de interface.

Serializador de Transformações OpenGL



Exemplos em OpenGL

- Visor por defeito
- Um único visor
- Dois visores

Exemplo 1:
visor por defeito

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

Exemplo 1:
visor por defeito

```
/* * WV-defaultViewport.cc - Using the default viewport * Abel Gomes  */
#include <OpenGL/gl.h>        // Header File For The OpenGL Library
#include <OpenGL/glu.h>       // Header File For The GLu Library
#include <GLUT/glut.h>        // Header File For The GLut Library
#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();
     gluOrtho2D(0.0,50.0,-10.0,40.0);  // also sets up world window

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

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

Exemplo 1:
visor por defeito (cont.)

```
// 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 );
     glutCreateWindow(”Default viewport spans the whole interface window");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     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 interface.
- Podem existir vários visores na janela de interface simultaneamente.
- Note-se que a janela de interface e os seus visores são definidos antes da janela de domínio de cena, sendo esta últim definida através de `gluOrtho2D(left, right, bottom, top)` porque esta função também define a transformação janela-visor.

Exemplo 2:
1 visor

```
/* * WV-singleViewport.cc - Using a single viewport * Abel Gomes  */
#include <OpenGL/gl.h>        // Header File For The OpenGL Library
#include <OpenGL/glu.h>       // Header File For The GLu Library
#include <GLUT/glut.h>        // Header File For The GLut Library
#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();
     gluOrtho2D(0.0,50.0,-10.0,40.0);  // also sets up world window

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

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

Exemplo 2:
1 visor (cont.)

```
// 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 );
     glutCreateWindow(”Single viewport spans the left-bottom interface window quarter");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Exemplo 3:
2 visores

```
/* * WV-twoViewports.cc - Using two viewports * Abel Gomes  */
#include <OpenGL/gl.h>        // Header File For The OpenGL Library
#include <OpenGL/glu.h>       // Header File For The GLu Library
#include <GLUT/glut.h>        // Header File For The GLut Library
#include <stdlib.h>

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

          // Sets up FIRST viewport spanning the left-bottom quarter of the interface window
     glViewport(0,0,250,250);
          // Sets up the PROJECTION matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0.0,50.0,-10.0,40.0);  // also sets up world window

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

                              // continues on next page
```

Exemplo 3:
2 visores (cont.)

```
/*  rest of the function draw() */


          // Sets up SECOND viewport spanning the right-top quarter of the interface window
     glViewport(250,250,250,250);
          // Sets up the PROJECTION matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluOrtho2D(0.0,50.0,-10.0,40.0);  // also sets up world window

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

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

Exemplo 3:
2 visores (cont.)

```
// 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 );
     glutCreateWindow(”Two viewports spanning the left-bottom and right-top quarters");
          // Declare the display and keyboard functions
     glutDisplayFunc(draw);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Transformação Janela-Visor: nota importante

Quando a janela aumenta de tamanho,

a imagem no visor dimimui,

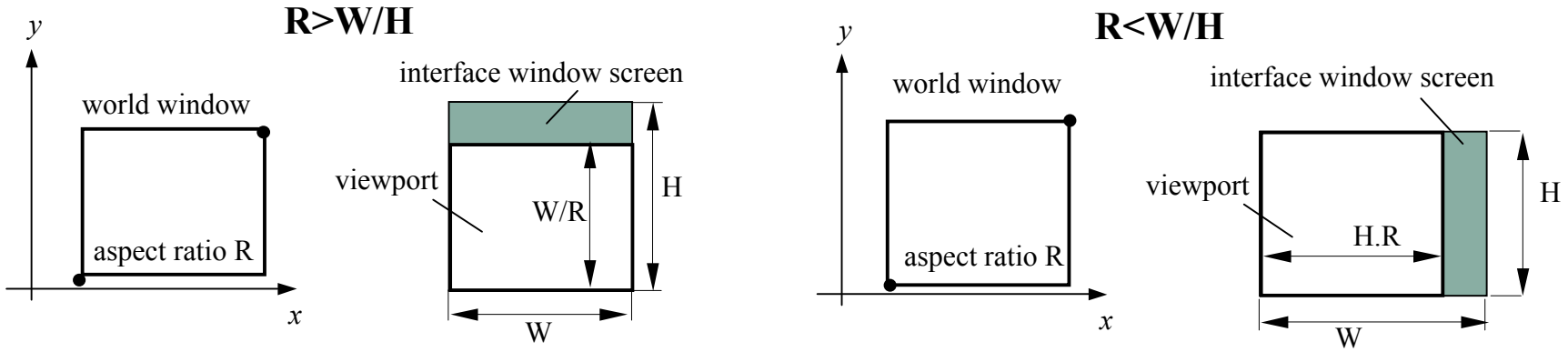
e vice-versa.

Transformação Janela-Visor: aplicações

- **Panning**
  - Mover a janela no domínio de cena
- **Zooming**
  - Redução/Aumento do tamanho da janela

Activação automática do visor sem distorção de imagem

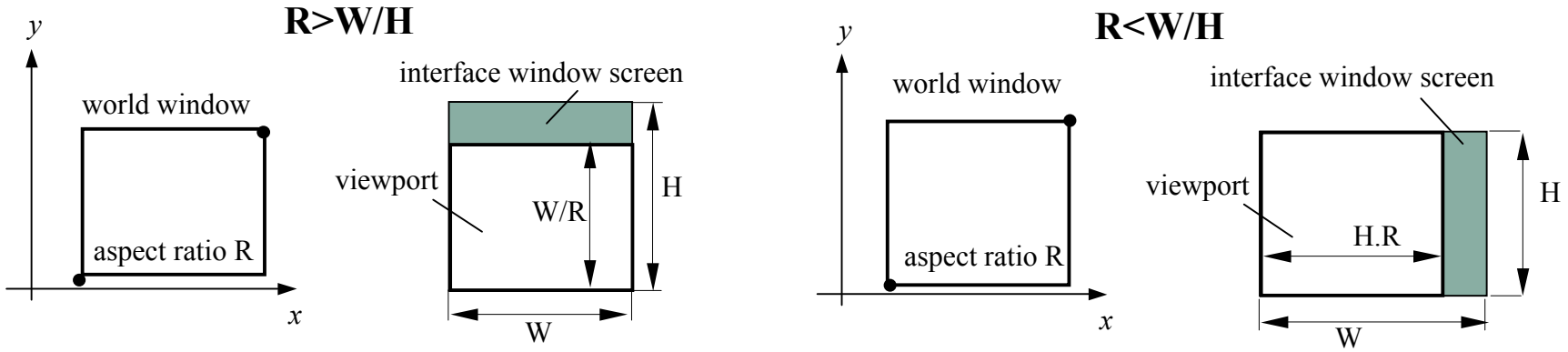
- Qual a maior imagem não-distorcida que ocupa o ecrã?
- R = Razão de aspecto da janela do domínio de cena
- Duas situações são possíveis:



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

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

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

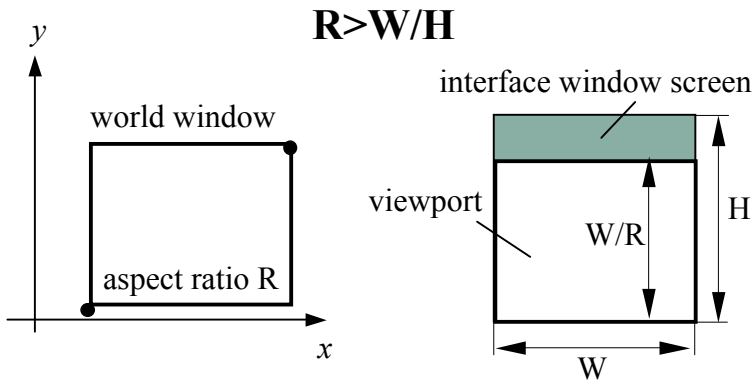



```
glViewport(0,0,W,W/R);
```

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

Exemplo 4:
janela baixa

- Se a janela tem razão de aspecto R=2.0 e o ecrã da janela de interface 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.

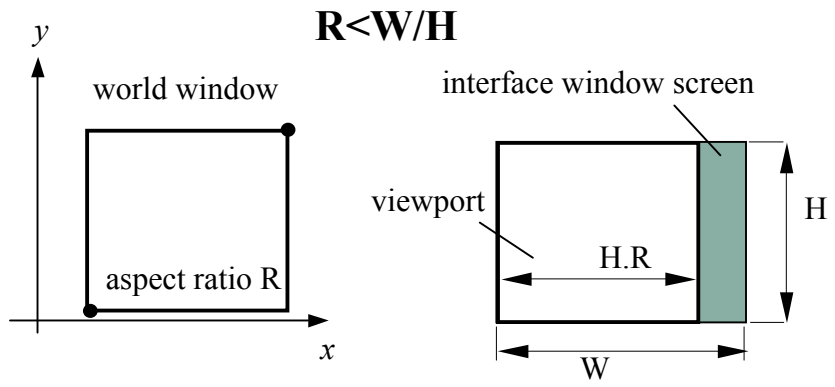



```
glViewport(0,0,W,W/R);

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

Exemplo 5: janela alta

- Se a janela tem razão de aspecto R=1.6 e o ecrã da janela de interface 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 de manutenção das proporções automaticamente na passagem da janela para o visor

- 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 interface.
- Para evitar distorção, há que mudar o tamanho da janela do domínio de cena em conformidade.
- Para isso, assume-se *a priori* que a janela da cena é um quadrado cujos lados têm comprimento `L`.
- Uma solução possível é mudar a janela da cena sempre que o ecrã da janela de interface 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 6:
activação automática do mapea-mento janela-visor sem distorção

```
/* Setting up window-viewport automatically without distortion
 * Abel Gomes  */
#include <OpenGL/gl.h>        // Header File For The OpenGL Library
#include <OpenGL/glu.h>       // Header File For The GLu Library
#include <GLUT/glut.h>        // Header File For The GLut Library
#include <stdlib.h>

void draw(){
          // Make background colour yellow
     glClearColor( 100, 100, 0, 0 );
     glClear ( GL_COLOR_BUFFER_BIT );
          // Draw house
     glColor3f( 0, 0, 1 );
     glRectf(0.0,0.0,30.0,30.0);
     glColor3f(1,0,0);
     glBegin(GL_TRIANGLES);
          glVertex3f(0.0,30.0,1.0);
          glVertex3f(30.0,30.0,1.0);
          glVertex3f(15.0,40.0,1.0);
     glEnd();
          // display house
     glutSwapBuffers();
}                                                  // end of draw()
```

Exemplo 6: (cont.)

```
// 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();
}
```

Exemplo 6: (cont.)

```
// 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 );
     glutCreateWindow(”Single viewport spans the left-bottom interface window quarter");
          // Declare the display, reshape, and keyboard functions
     glutDisplayFunc(draw);
     glutReshapeFunc(reshape);
     glutKeyboardFunc(keyboard);
          // Start the Main Loop
     glutMainLoop();
     return 0;
}
```

Recorte Gráfico

- Algoritmo de Sutherland-Hodgeman
- ....

FIM

Exemplo 1:
visor por defeito

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