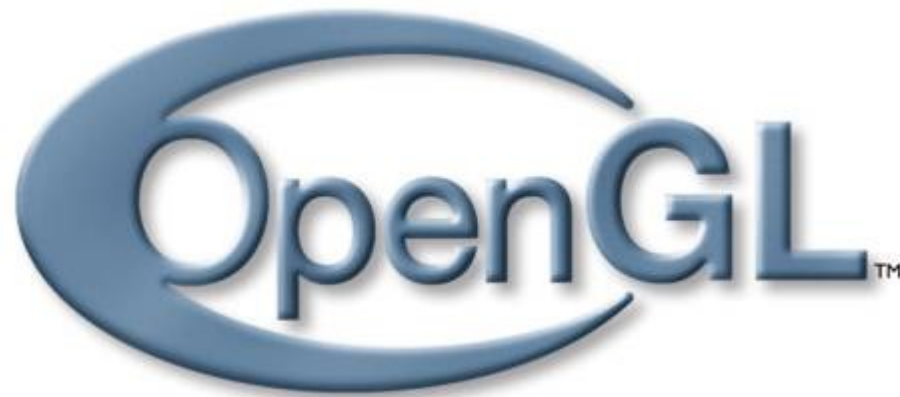


Unidade 6 - Introdução ao OpenGL



IME 04-10842
Computação Gráfica
Professor Guilherme Mota
Professor Gilson Costa

Definição



Padrão aberto de arquitetura para computação gráfica que especifica hardware e uma API de software .

Exemplos de OpenGL



<http://www.rage.com/>

Exemplos de OpenGL



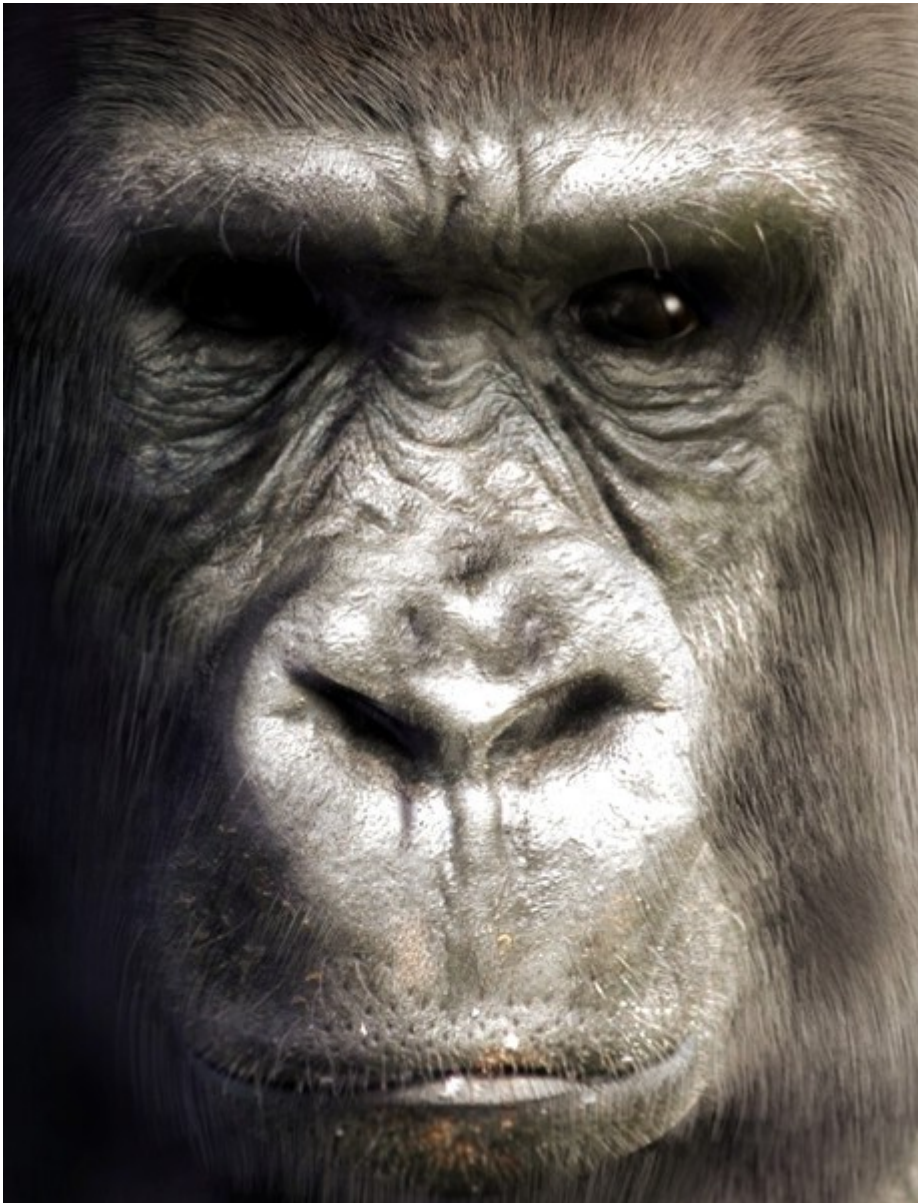
http://en.wikipedia.org/wiki/Quake_4

Exemplos de OpenGL

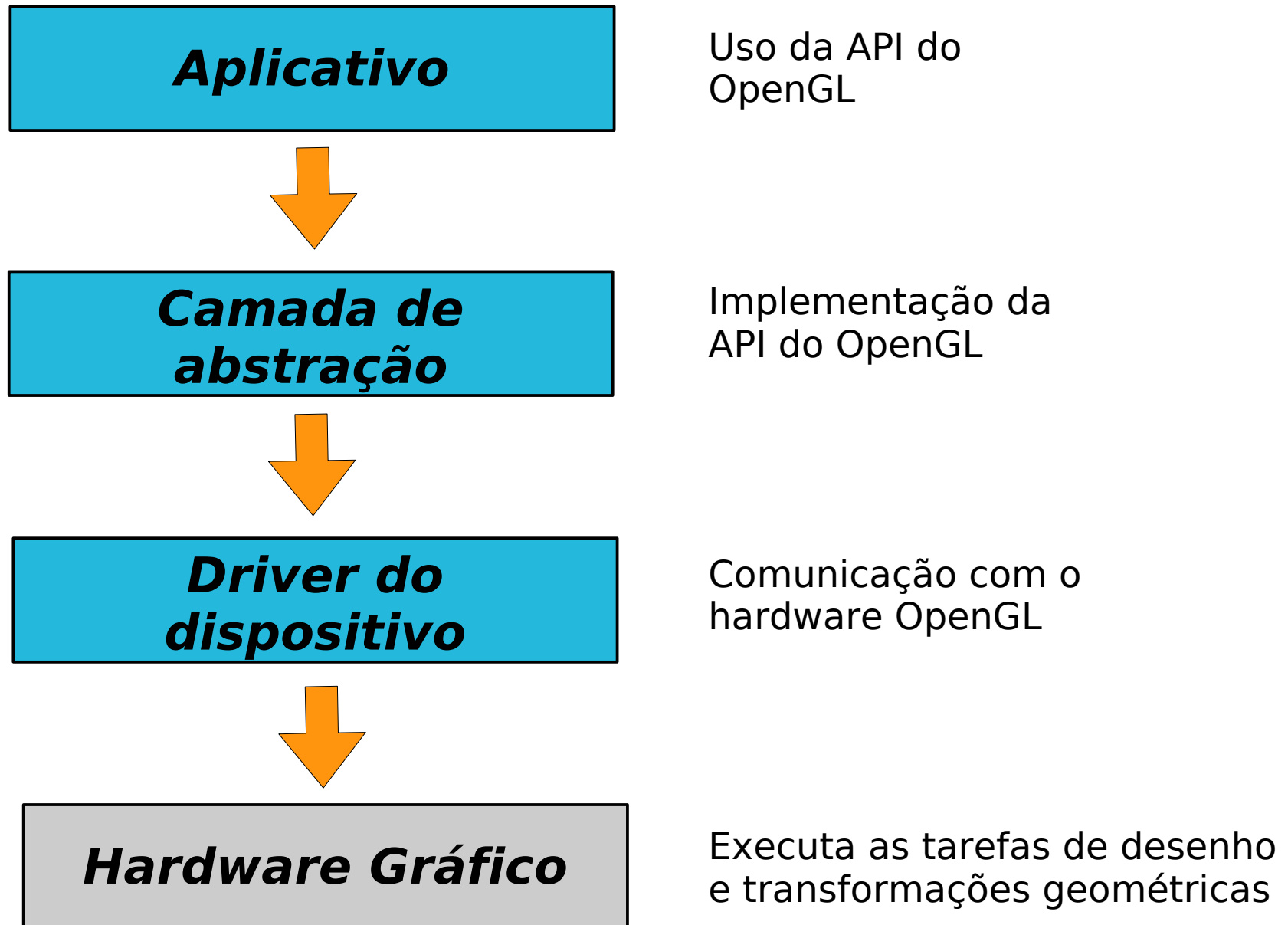


<http://www.aerofly.de/>

Exemplos de OpenGL



Pipeline de Software



Arquitetura

Arquitetura

- Possui primitivas e operações para a geração e manipulação de dados vetoriais e matriciais.
- Capaz de gerar imagens de alta qualidade.
- Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível).
- Independente de plataforma.

Arquitetura

- Não gerencia janelas nem trata eventos produzidos por dispositivos de interação.
- Não possui comandos de alto nível para especificação de objetos 3D complexos.
- Objetos complexos devem ser construídos a partir de primitivas geométricas simples.

Arquitetura

- Cria descrições matemáticas de objetos a partir de primitivas geométricas (pontos, linhas e polígonos) e imagens/bitmaps.
- Organiza os objetos no espaço 3D e seleciona o ponto de vista adequado para a cena

Arquitetura

- Calcula as cores dos objetos por:
 - Atribuição direta.
 - Modelos de iluminação.
 - Mapeamentos de texturas.
 - Combinações.
- Converte as descrições matemáticas + cores em pixels (rasterização).

Sintaxe OpenGL

Código OpenGL

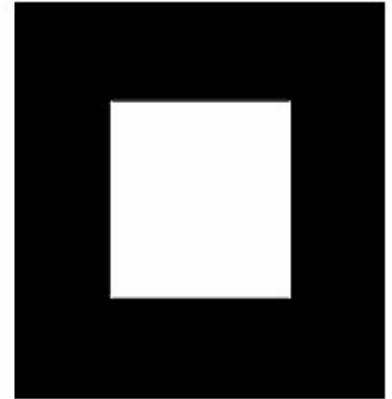
```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```



Sintaxe das funções da API

- Todas as funções começam com o prefixo `gl` (Ex.: `glClearColor()`).
- Padrão lower camel case (Ex.: `glColor()`).
- O sufixo indica a quantidade e o tipo dos argumentos (Ex.: `glVertex2i(1, 3)`).
- Constantes: `GL_COLOR_BUFFER_BIT`.

Sintaxe das funções da API

`glVertex3fv(v)`

Número de componentes

2 - (x,y)
3 - (x,y,z)
4 - (x,y,z,w)

Tipo de dado

b - byte
ub - unsigned byte
s - short
us - unsigned short
i - int
ui - unsigned int
f - float
d - double

vetor

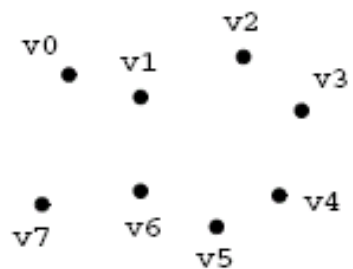
“v” é necessário quando
coords são passadas
por um array

Sufixos e tipos dos argumentos

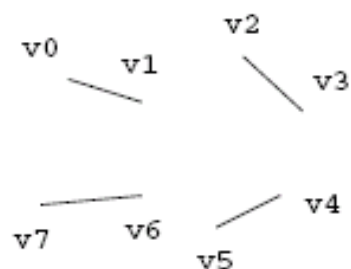
| Tipo | Tipo | C | OpenGL |
|-------------|---------------------------|----------------|----------------------------|
| b | Inteiro 8-bits | signed char | GLbyte |
| s | Inteiro 16-bits | short | GLshort |
| i | Inteiro 32-bits | long | GLint, GLsizei |
| f | Ponto-flutuante 32-bit | float | GLfloat, GLclampf |
| d | Ponto-flutuante 64-bit | double | GLdouble, GLclampd |
| ub | Caractere s/ sinal 8-bit | unsigned char | GLubyte, GLboolean |
| us | Caractere s/ sinal 16-bit | unsigned short | GLushort |
| ui | Caractere s /sinal 32-bit | unsigned long | GLuint, GLenum, GLbitfield |

Primitivas de Desenho OpenGL

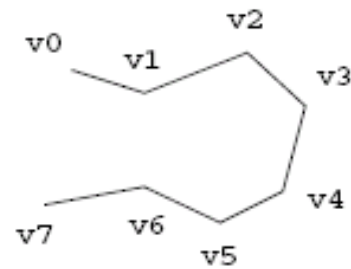
Primitivas de Desenho



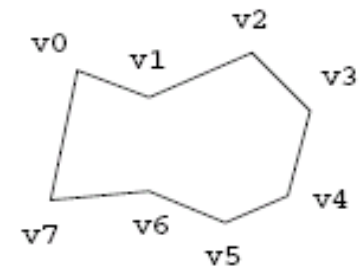
GL_POINTS



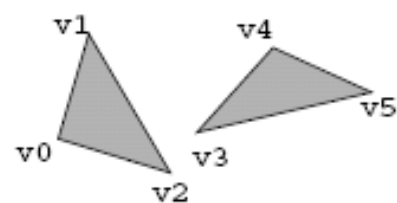
GL_LINES



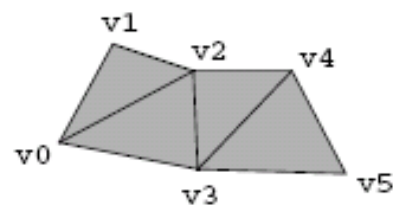
GL_LINE_STRIP



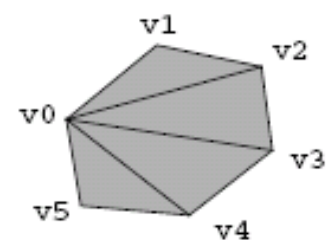
GL_LINE_LOOP



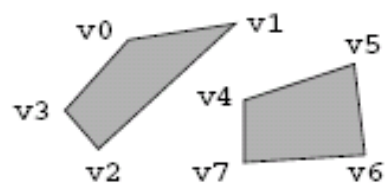
GL_TRIANGLES



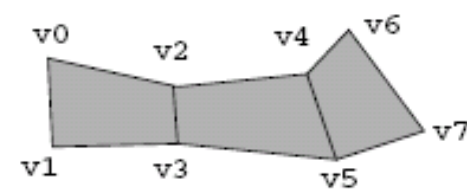
GL_TRIANGLE_STRIP



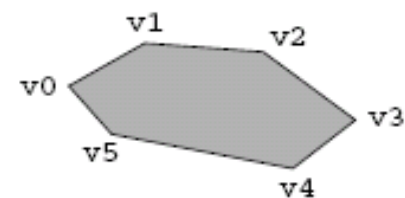
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

OpenGL- Primitivas de desenho

- `glBegin (PRIMITIVA) ;`
 - especificação de vértices, cores, coordenadas de textura, propriedades de material
- `glEnd () ;`
- Entre `glBegin ()` e `glEnd ()` apenas alguns comandos podem ser usados. Ex.:
 - `glVertex`
 - `glMaterial`
 - `glNormal`
 - `glTexCoord`

OpenGL - Primitivas de desenho

- Uma vez emitido um vértice (`glVertex`), este é desenhado com as propriedades (cor, material, normal, coordenadas de textura etc) registradas nas variáveis de estado correspondentes.
- Atenção: Antes de emitir um vértice, assegurar-se que cor, material, normal, etc têm o valor certo.

Primitivas de Desenho

| Valor | Significado |
|-------------------|---|
| GL_POINTS | Pontos individuais |
| GL_LINES | Pares de vértices interpretados como segmentos de reta individuais. |
| GL_LINE_STRIP | Serie de segmentos de reta conectados. |
| GL_LINE_LOOP | Igual ao anterior. Ultimo vertice conectado a primeiro |
| GL_TRIANGLES | Triplas de vértices interpretados como triângulos. |
| GL_TRIANGLE_STRIP | Cadeia triângulos conectados. |
| GL_TRIANGLE_FAN | Leque de triângulos conectados. |
| GL_QUADS | Quadrupla de vértices interpretados como quadriláteros. |
| GL_QUAD_STRIP | Cadeia de quadriláteros conectados. |
| GL_POLYGON | Borda de um polígono convexo simples. |

OpenGL como máquina de estados

OpenGL como máquina de estados

- Uma aplicação OpenGL funciona como uma máquina de estados.
- Os estados correntes permanecem ativos.
- Exemplo: a cor de desenho é aplicada a qualquer primitiva geométrica até que a cor corrente seja modificada.

OpenGL como maquina de estados

- Existem várias variáveis de estados, por exemplo:
 - cor de desenho corrente
 - transformações de visualização e projeção
 - padrões de linhas e polígonos
 - modo de desenho dos polígonos
 - atributos das fontes de luz
 - propriedades de reflexão e textura dos materiais associados aos objetos

OpenGL como maquina de estados

- Vários estados se referem a modos que estão habilitados ou desabilitados.
- Estes estados são modificados através dos comandos `glEnable()` e `glDisable()`.
- Exemplo: `glEnable(GL_LIGHTING)`.

OpenGL como maquina de estados

- Alguns comandos para ler um estado:
 - `glGetBooleanv()`, `glGetDoublev()`,
`glGetFloatv()`, `glGetIntegerv()`,
`glPointerv()` ou `glIsEnabled()`.
- Comandos para salvar um estado:
 - `glPushAttrib()` e
`glPushClientAttrib()`.
- Comandos para restaurar um estado:
 - `glPopAttrib()` e `glPopClientAttrib()`.

OpenGL como maquina de estados

- Exemplo: GL_LIGHTING

```
int luz;  
  
glEnable(GL_LIGHTING); //Habilita iluminação  
  
luz = glIsEnabled(GL_LIGHTING); // retorna 1  
    (verdadeiro)  
  
glDisable(GL_LIGHTING); //Desabilita luz  
  
luz = glIsEnabled(GL_LIGHTING); // retorna 0 (falso)
```

APIs complementares

API's relacionadas

- GLU (OpenGL Utility Library)
 - Parte do padrão OpenGL
 - NURBS, trianguladores, quádricas, mapeamento, mipmaps, superfícies quadráticas, transformação e posicionamento da câmera e primitivas de desenho adicionais
- AGL, GLX, WGL
 - Camadas entre o OpenGL os diversos sistemas de janelas
- GLUT (OpenGL Utility Toolkit)
 - API portátil de acesso aos sistemas de janelas
 - Encapsula e esconde as camadas proprietárias
 - Não é parte oficial do OpenGL
 - <http://www.opengl.org/resources/libraries/glut/spec3/spec3.html>

GLUT - OpenGL Utility Toolkit

- API permite a implementação de aplicativos simples com uso do OpenGL
- Independente do sistema de janelas
- Permite a implementação de aplicativos multiplataforma
- Projetada para o desenvolvimento de programas de porte pequeno e médio
- *Bindings* para diversas linguagens de programação
- É um software gratuito e não *open source*
 - *freeglut* (Licença MIT)
- Fornece um conjunto de primitivas para desenho de objetos mais complexos como quádricas e etc.

GLUT Callbacks

GLUT - Callbacks

- *Callbacks* são trechos de código (rotinas) chamadas para tratar eventos síncronos ou assíncronos
- Na linguagem C são implementadas a partir de ponteiros para funções, instrumento que permite que funções sirvam de argumento para outras. Exemplo:

```
<type> callerFunc(<type> (*argFunc) (<argList>))  
{  
    /* Código de callerFunc */  
}
```

- Nas chamadas à `callerFunc` precisa ser fornecida a função `argFunc`
- A assinatura da função passada como parâmetro precisa ser idêntica à da função `argFunc`

GLUT - *Callbacks*

- As *callbacks* do GLUT são monitoradas e chamadas a partir do `GlutMainLoop`
- A API do GLUT possui diversas *callbacks* predefinidas
- Para uma rotina *callback* ser ativada ela precisa ser registrada através da função de registro

- `glut<Callbackname>Func ((<type> (*callBackFunc) (<argList>)))`

- `<Callbackname>` revela a classe do evento
 - `callBackFunc` é o nome da rotina de Callback

- Exemplo, função de registro da *callback* de desenho:

```
void glutDisplayFunc(void (*func) (void)) ;
```

GLUT - Callbacks

`glutDisplayFunc`

`glutOverlayDisplayFunc`

`glutReshapeFunc`

`glutKeyboardFunc`

`glutMouseFunc`

`glutMotionFunc`

`glutPassiveMotionFunc`

`glutVisibilityFunc`

`glutEntryFunc`

`glutSpecialFunc`

`glutSpaceballMotionFunc`

`glutSpaceballRotateFunc`

`glutSpaceballButtonFunc`

`glutButtonBoxFunc`

`glutDialsFunc`

`glutTabletMotionFunc`

`glutTabletButtonFunc`

`glutMenuStatusFunc`

`glutIdleFunc`

`glutTimerFunc`

GLUT - Callback de desenho

`glutDisplayFunc (void (*func) (void));`

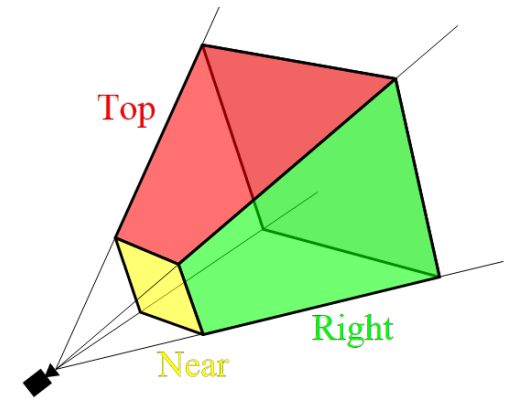
- Chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada
- Todo programa OpenGL/GLUT precisa ter uma
- Exemplo:

```
void display ( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glBegin( GL_TRIANGLE_STRIP );
        glVertex3fv( v[0] );
        glVertex3fv( v[1] );
        glVertex3fv( v[2] );
        glVertex3fv( v[3] );
    glEnd();
    glutSwapBuffers(); /* Usamos double-buffering! */
}
```

GLUT - Callback de redimensionamento

```
glutReshapeFunc ((void (*func) (int width,  
int height)) ;
```

- Chamada sempre que a janela é redimensionada
- `width` e `height` são a nova largura/altura da janela (em pixels)
- Os valores fornecidos servem para o recálculo do *frustum*
- Há uma rotina *default* que simplesmente ajusta o *viewport* para usar toda a área da janela gráfica



GLUT - *Callback* de Teclado

```
void glutKeyboardFunc(void (*func)  
(unsigned char key, int x, int y));
```

- Chamada sempre que um caracter é emitido
- `key` indica o caracter e `x` e `y` a posição relativa do mouse no momento do envio
- `glutGetModifiers` pode ser chamado para determinar as teclas pressionadas
- Para desabilitar esta *callback* é preciso passar `NULL` para `glutKeyboardFunc`
- Não há *callback* de teclado *default*

GLUT - *Callback* de Temporização

```
void glutTimerFunc(unsigned int msec, void  
(*func)(int value), value);
```

- Registra uma *callback* de temporização
- Após msec milissegundos, a chamada para a *callback* se dará assim que for possível
- value tem o mesmo valor em (*func) e glutTimerFunc
- Podem ser registradas múltiplas *callbacks* de temporização.
- Uma vez registrada, não é possível cancelar uma *callback* de temporização. Contudo, pode-se ignorá-la em função de value

GLUT - Outras Callbacks

- Eventos de mouse
 - `void glutMouseFunc(int button, int state, int x, int y)`
 - `void glutMotionFunc(int x, int y)`
 - `void glutPassiveMotionFunc(int x, int y)`
- Chamada continuamente quando nenhum outro evento ocorre
 - `void glutIdleFunc(void)`

Inicialização GLUT

Programa OpenGL/GLUT - Inicialização

- Inicialização do GLUT
 - `glutInit (int* argc, char** argv)`
- Estabelece contato com sistema de janelas.
 - Em X, opções de linha de comando são processadas.

Programa OpenGL/GLUT - Inicialização

- Inicialização da(s) janela(s):
`glutInitDisplayMode (int modo)`
- Estabelece o tipo de recursos necessários para as janelas que serão criadas.
- Modo é um “ou” bit-a-bit de constantes:
 - GLUT_RGB cores dos pixels serão expressos em RGB.
 - GLUT_DOUBLE bufferização dupla (ao invés de simples).
 - GLUT_DEPTH buffer de profundidade (z-buffer).
 - GLUT_ACCUM buffer de acumulação.
 - GLUT_ALPHA buffer de cores terá componente alfa.
 - GLUT_RGBA cores dos pixels serão expressos em RGBA.

Programa OpenGL/GLUT - Inicialização

- `glutInitWindowPosition (int x, int y)`
 - Estabelece a posição inicial do canto superior esquerdo da janela a ser criada.
- `glutInitWindowSize (int width, height)`
 - Estabelece o tamanho (em pixels) da janela a ser criada.

Programa OpenGL/GLUT - Inicialização

- Criação da(s) janela(s):

```
int glutCreateWindow (char* nome)
```

- Cria uma nova janela primária (top-level)
- Nome é tipicamente usado para rotular a janela
- O número inteiro retornado é usado pelo GLUT para identificar a janela

Programa OpenGL/GLUT - Inicialização

- Outras inicializações
- Após a criação da janela é costumeiro configurar variáveis de estado do OpenGL que não mudarão no decorrer do programa. Por exemplo:
 - Cor do fundo
 - Tipo de sombreamento de desejado

Programa OpenGL/GLUT - Laço principal

- Depois de registradas as callbacks, o controle é entregue ao sistema de janelas:

```
glutMainDisplayLoop (void)
```

- Esta rotina na verdade é o “despachante” de eventos.
- Ela nunca retorna.

Exemplo 1

Janela glut

Exemplo 1 (Janela glut)

```
// PrimeiroPrograma.c - Isabel H. Manssour
// Um programa OpenGL simples que abre uma janela GLUT
// Este código está baseado no Simple.c, exemplo
// disponível no livro "OpenGL SuperBible",
// 2nd Edition, de Richard S. e Wright Jr.

#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    //Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);

    //Executa os comandos OpenGL
    glFlush();
}
```

Exemplo 1 (Janela glut)

```
// Inicializa parâmetros de rendering
void Inicializa(void)
{
    // Define a cor de fundo da janela de visualização como preta
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}

// Programa Principal
int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Primeiro Programa");
    glutDisplayFunc(Desenha);
    Inicializa();
    glutMainLoop();
}
```

Exemplo 2

Desenho 2D

Exemplo 2 (Desenho 2D)

```
#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    //Limpa a janela de visualização com a cor branca
    glClearColor(1, 1, 1, 0);
    glClear(GL_COLOR_BUFFER_BIT);
    //Define a cor de desenho: vermelho
    glColor3f(1, 0, 0);
    //Desenha um triângulo no centro da janela
    glBegin(GL_TRIANGLES);
        glVertex2f(-0.5, -0.5);
        glVertex2f(0.0, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();
    //Executa os comandos OpenGL
    glFlush();
}
```

Exemplo 2 (Desenho 2D)

```
//Função callback chamada para gerenciar eventos de teclas
void Teclado(unsigned char key, int x, int y)
{
    if (key == 27)
        exit(0);
}

// Inicializa parâmetros e variáveis
void Inicializa(void)
{
    // Define a janela de visualização 2D
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
}
```


Exemplo 2 (Desenho 2D)

```
// Programa Principal
int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutCreateWindow("Segundo Programa");
    glutDisplayFunc(Desenha);
    glutKeyboardFunc(Teclado);
    Inicializa();
    glutMainLoop();
    return 0;
}
```

Exemplo 2 (Desenho 2D)

Experimente:

- Altere no código a cor de cada um dos vértices do triângulo: coloque cores diferente para cada um.
- Em tempo de execução, altere o tamanho da janela criada pelo programa.

Exemplo 3

Window-Viewport

Exemplo 3 (Window-Viewport)

```
#include <windows.h>
#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);
    // Especifica que a cor corrente é verde
    glColor3f(0.0f, 1.0f, 0.0f);
    // Desenha um quadrado preenchido com a cor corrente
    glBegin(GL_QUADS);
        glVertex2i(100,150);
        glVertex2i(100,100);
        glVertex2i(150,100);
        glVertex2i(150,150);
    glEnd();
    // Executa os comandos OpenGL
    glFlush();
}
```

Exemplo 3 (Window-Viewport)

```
#include <windows.h>
#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // Limpa a janela de visualização com a cor de fundo especificada
    glClear(GL_COLOR_BUFFER_BIT);
    // Especifica que a cor corrente é verde
    glColor3f(0.0f, 1.0f, 0.0f);
    // Desenha um quadrado preenchido com a cor corrente
    glBegin(GL_QUADS);
        glVertex2i(100,150);
        glVertex2i(100,100);
        glVertex2i(150,100);
        glVertex2i(150,150);
    glEnd();
    // Executa os comandos OpenGL
    glFlush();
}
```

Exemplo 3 (Window-Viewport)

```
// Inicializa parâmetros de rendering
void Inicializa (void)
{
    // Define a cor de fundo da janela de visualização como preta
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}
// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h)
{
    // Evita a divisao por zero
    if(h == 0) h = 1;
    // Especifica as dimensões da Viewport
    glViewport(0, 0, w, h);
    // Inicializa a matriz de projeção
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // Estabelece World Window (left, right, bottom, top)
    if (w <= h)
        gluOrtho2D (0.0f, 250.0f, 0.0f, 250.0f*h/w);
    else
        gluOrtho2D (0.0f, 250.0f*w/h, 0.0f, 250.0f);
}
```

Exemplo 3 (Window-Viewport)

```
// Programa Principal
int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 350);
    glutInitWindowPosition(10, 10);
    glutCreateWindow("Quadrado");
    glutDisplayFunc(Desenha);
    glutReshapeFunc(AlteraTamanhoJanela);
    Inicializa();
    glutMainLoop();
}
```

Exemplo 3 (Window-Viewport)

Experimente:

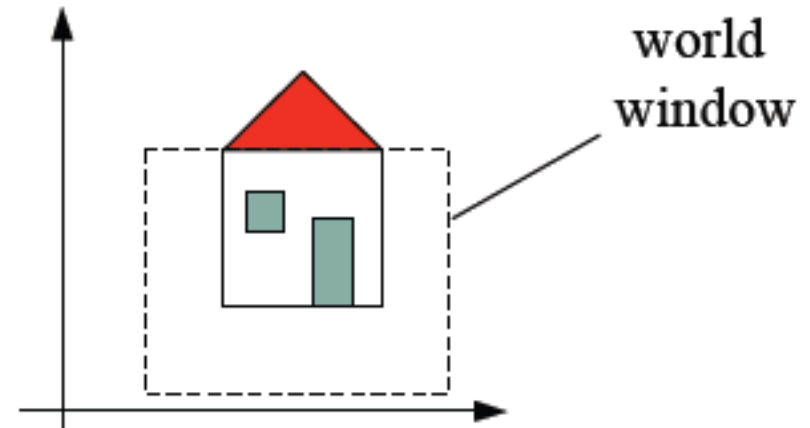
- Altere no código a cor de cada um dos vértices do quadrado.
- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Por que o redimensionamento da janela provoca um efeito diferente em relação ao Exemplo 2?

Window ViewPort

Mapeamento Window-Viewport

Sistema de Coordenadas Globais (**World Coordinate System**)

- Domínio da Cena: espaço dos objetos.
- Espaço em que se encontram os objetos geométricos.
- Espaço onde a geometria do objeto e o modelo de aplicação é definido, e.g., \mathbb{R}^2 .



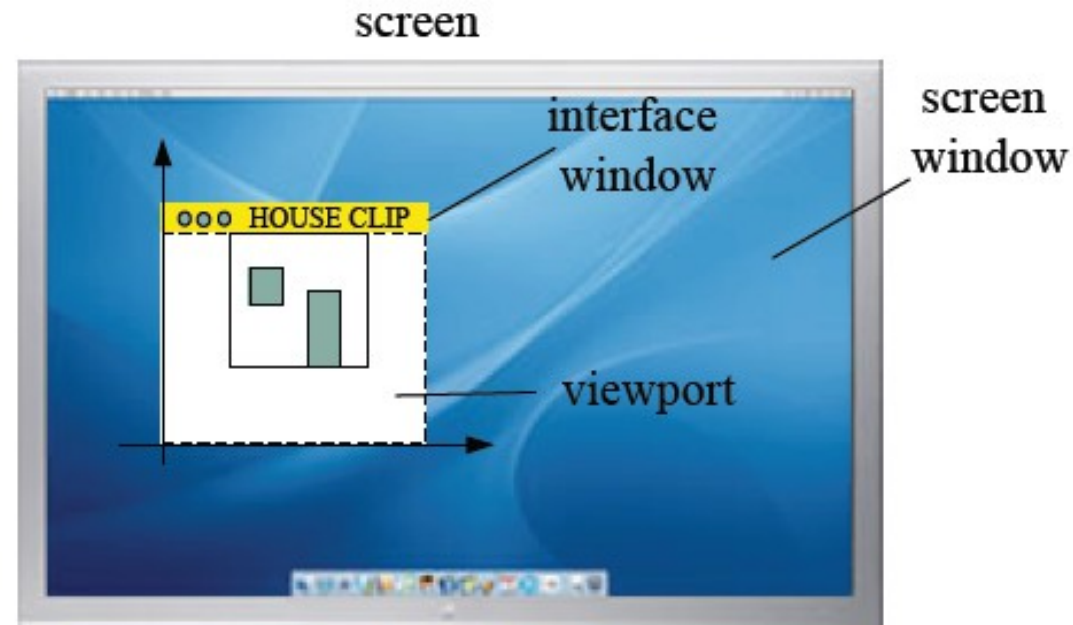
Subdomínio da Cena (**World Window**)

- Retângulo que define a parte da cena (world) que se pretende visualizar.

Mapeamento Window-Viewport

Sistema de Coordenadas da Imagem (**Screen Coordinate System**)

- Domínio de Imagem
- Espaço no qual a imagem é mostrada; e.g., 800x600 pixels.
- Espaço no qual a imagem rasterizada do objeto é definida.



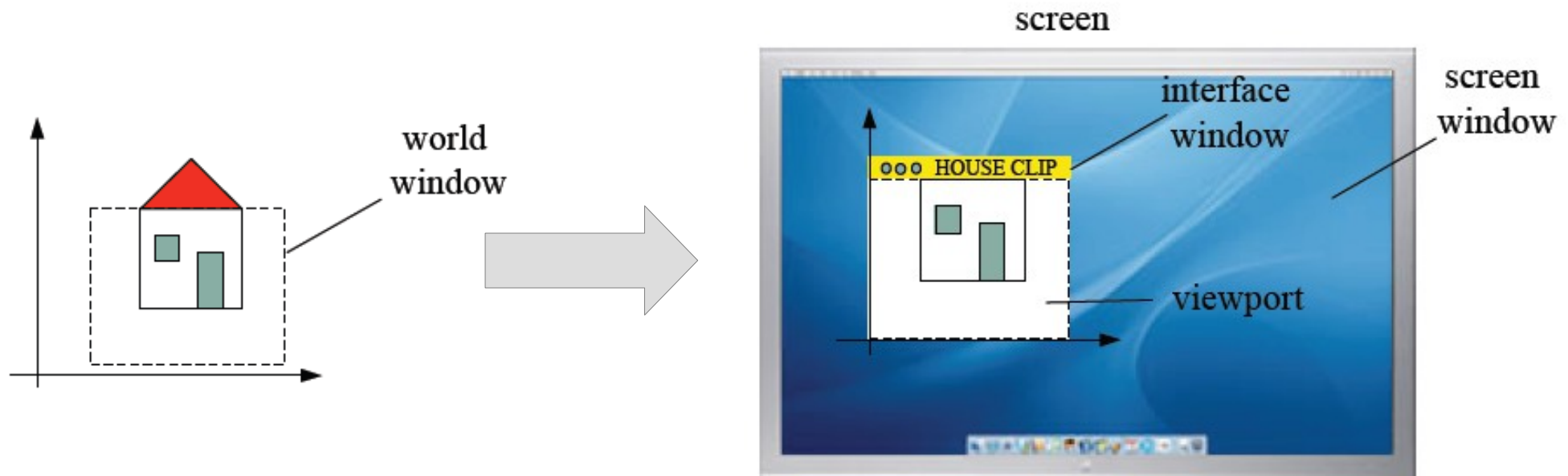
Subdomínio de Imagem (**Viewport**)

- Sistema de coordenadas da tela (janela do sistema de janelas).
- Sistema de coordenadas se move com a janela.

Mapeamento Window-Viewport

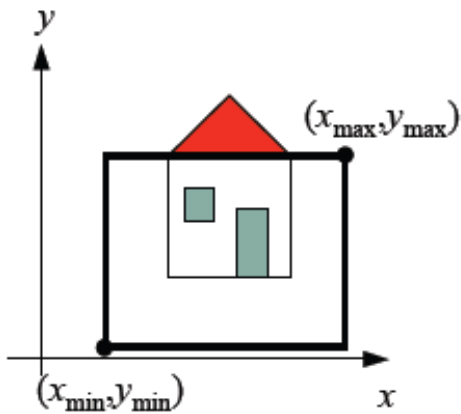
Transformação de Visualização: Window-Viewport

- Mapeamento de uma janela no domínio da cena (World coordinates) para o viweport (screen coordinates).

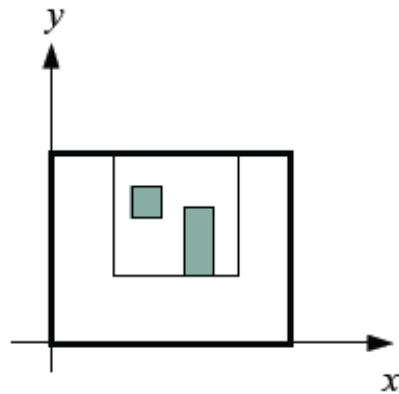


Mapeamento Window-Viewport

Transformação de Visualização: Window-Viewport

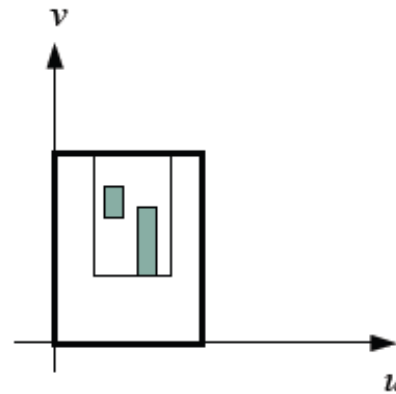


Window
em coordenadas
do mundo



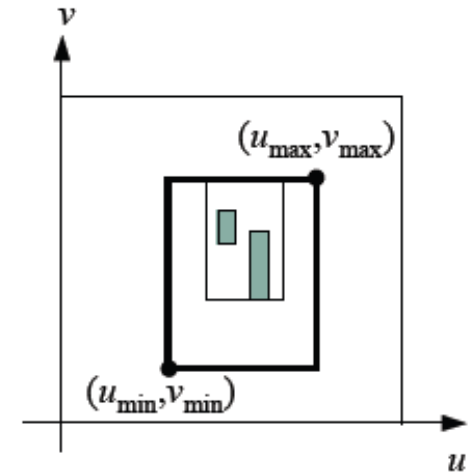
Window
transladada
para a origem

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



Window
escalada para o
tamanho do
Viewport

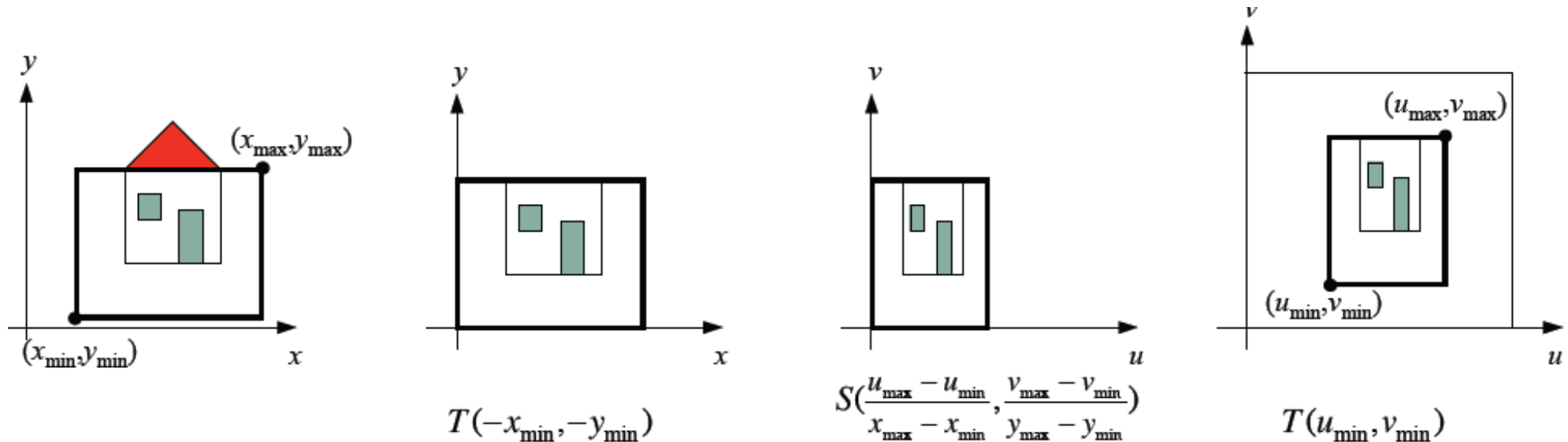
$$S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right)$$



Viewport
transladado para
a posição final

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

Mapeamento Window-Viewport



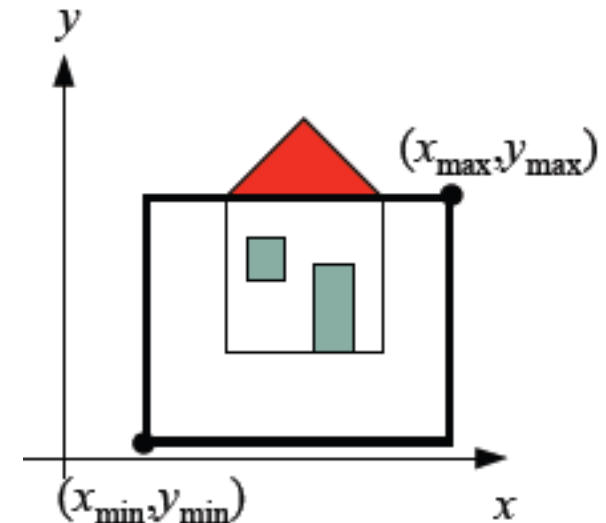
$$M_{wv} = T(u_{\min}, v_{\min}) \cdot S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right) \cdot T(-x_{\min}, -y_{\min})$$

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

Mapeamento Window-Viewport

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

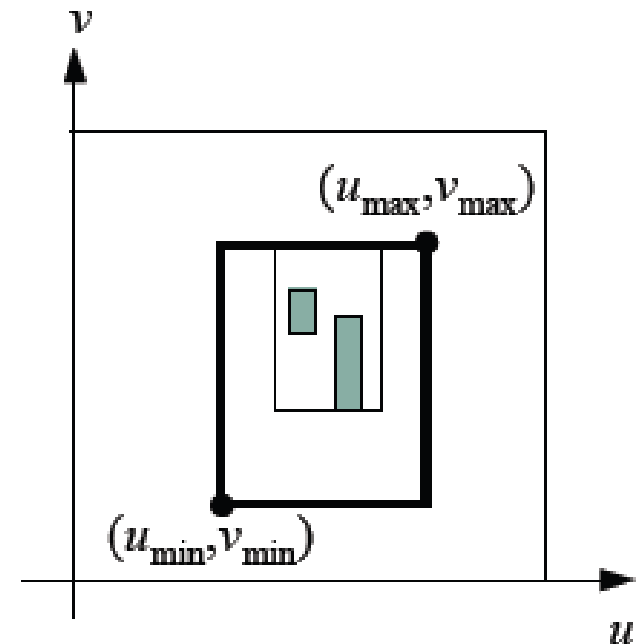
- Define um retângulo de visualização ortogonal 2D (**Window**): parte visível da cena.
- Define dois planos verticais (left, right) e dois planos horizontais (bottom, up).
- Os valores (left, right, bottom, top) são (-1, 1, -1, 1) por default.
- Define uma matriz de projeção ortogonal 2D.
- Define ainda a transformação Window-Viewport, em combinação com a função `glViewport`.



Mapeamento Window-Viewport

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

- Define as dimensões do **Viewport** numa janela do sistema de janelas.
- Onde x, y indicam o canto inferior esquerdo e $width, height$ a extensão do Viewport.
- Por default o Viewport ocupa toda a janela.
- Podem existir vários Viewports dentro de uma janela.



Mapeamento Window-Viewport

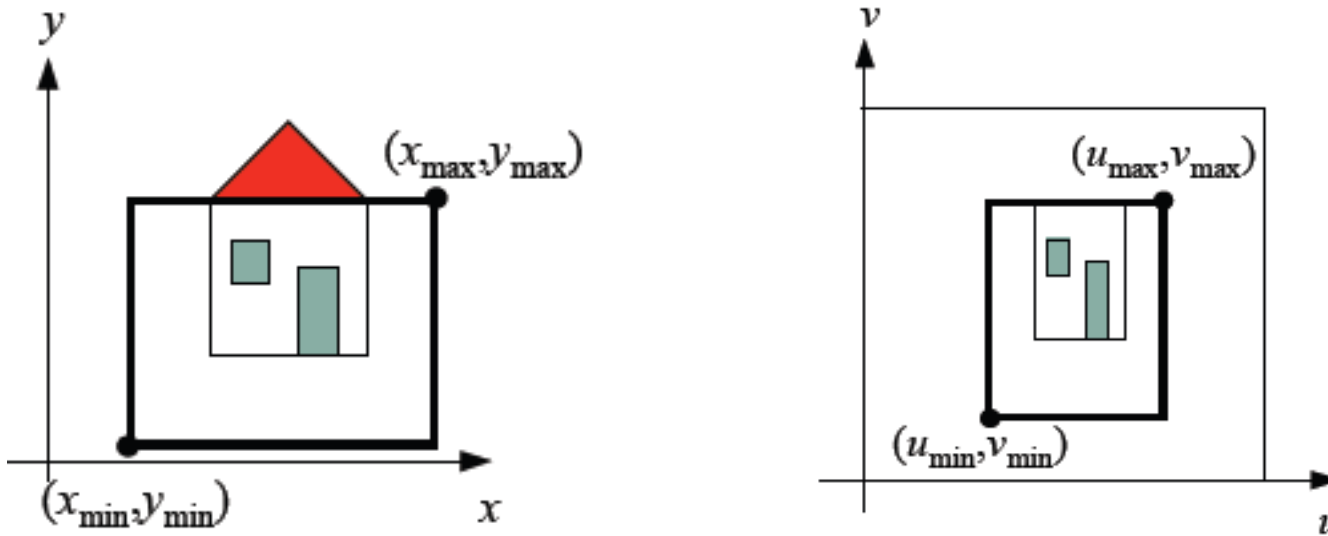
Manutenção automática das proporções na transformação de visualização:

- Para evitar distorção quando a janela na tela é redimensionada.
- Inserir na função callback de redimensionamento de janela, e.g., `AlteraTamanho`, (registrada através de `glutReshapeFunc`):

```
void AlteraTamanho(GLsizei w, GLsizei h)
{
    ...
    glViewport(0, 0, w, h);
    // recalcula a matriz de projeção
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        gluOrtho2D (xmin, xmax, ymin, ymaxf*h/w);
    else
        gluOrtho2D (xmin, xmax*w/h, ymin, ymaxf);
    ...
}
```

- `xmin`, `xmax`, `ymin`, `ymax` definem os planos que delimitam a Window.

Mapeamento Window-Viewport



```
void AlteraTamanho(GLsizei w, GLsizei h)
...
glViewport(0, 0, w, h);
// recalcula a matriz de projeção
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w <= h)
    gluOrtho2D (xmin, xmax, ymin, ymaxf*h/w);
else
    gluOrtho2D (xmin, xmax*w/h, ymin, ymaxf);
...
```

Exemplo 4

Interface-Movimento

Exemplo 4 (Interface-Movimento)

```
#include <GL/glut.h>

// Funções callback
void desenha(void);
void movimento(int x, int y);
void botao(int button, int state, int x, int y);
void alteraTamanhoJanela(GLsizei w, GLsizei h);

// Variáveis globais
float largura, altura; // tamanho da world window e do viewport
float rx, ry; // posição inicial do retângulo vermelho

int main(int argc, char *argv[])
{
    largura = 600;
    altura = 600;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glClearColor(0, 0, 0, 1);
```

Exemplo 4 (Interface-Movimento)

```
glutPositionWindow(200, 100);  
glutReshapeWindow(largura, altura);  
  
glMatrixMode(GL_PROJECTION);  
glLoadIdentity();  
gluOrtho2D(0, largura, 0, altura);  
  
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
  
glutDisplayFunc(desenha);  
glutIdleFunc(desenha);  
glutMotionFunc(movimento);  
glutMouseFunc(botao);  
glutReshapeFunc(alteraTamanhoJanela);  
  
glutMainLoop();  
  
return 0;  
}
```

Exemplo 4 (Interface-Movimento)

```
// Função callback chamada para fazer o desenho
void desenha(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1, 1, 0); // desenha em amarelo
    glRectf(100, 100, 300, 200); // desenha retângulo amarelo
    glColor3f(1, 0, 0); // desenha em vermelho
    glRectf(rx, ry, 100 + rx, 20 + ry); // desenha retângulo vermelho

    GlutSwapBuffers(); // buffer de cor duplo
}
// Função callback para o mouse se movimentando com botão pressionado
void movimento(int x, int y)
{
    rx = x; ry = altura - y;
}
// Função callback chamada quando o botão do mouse é pressionado
void botao(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        rx = x; ry = altura - y;
}
```

Exemplo 4 (Interface-Movimento)

```
// Função callback chamada quando o tamanho da janela é alterado
void alteraTamanhoJanela(GLsizei w, GLsizei h)
{
    largura = w;
    altura = h;

    // Especifica as dimensões da Viewport
    glViewport(0, 0, largura, altura); // comando não necessário:
                                     //comportamento default
    // Inicializa o sistema de coordenadas do mundo
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Dimensões da World Window (left, right, bottom, top)
    gluOrtho2D(0, largura, 0, altura);
}
```

Exemplo 4 (Interface-Movimento)

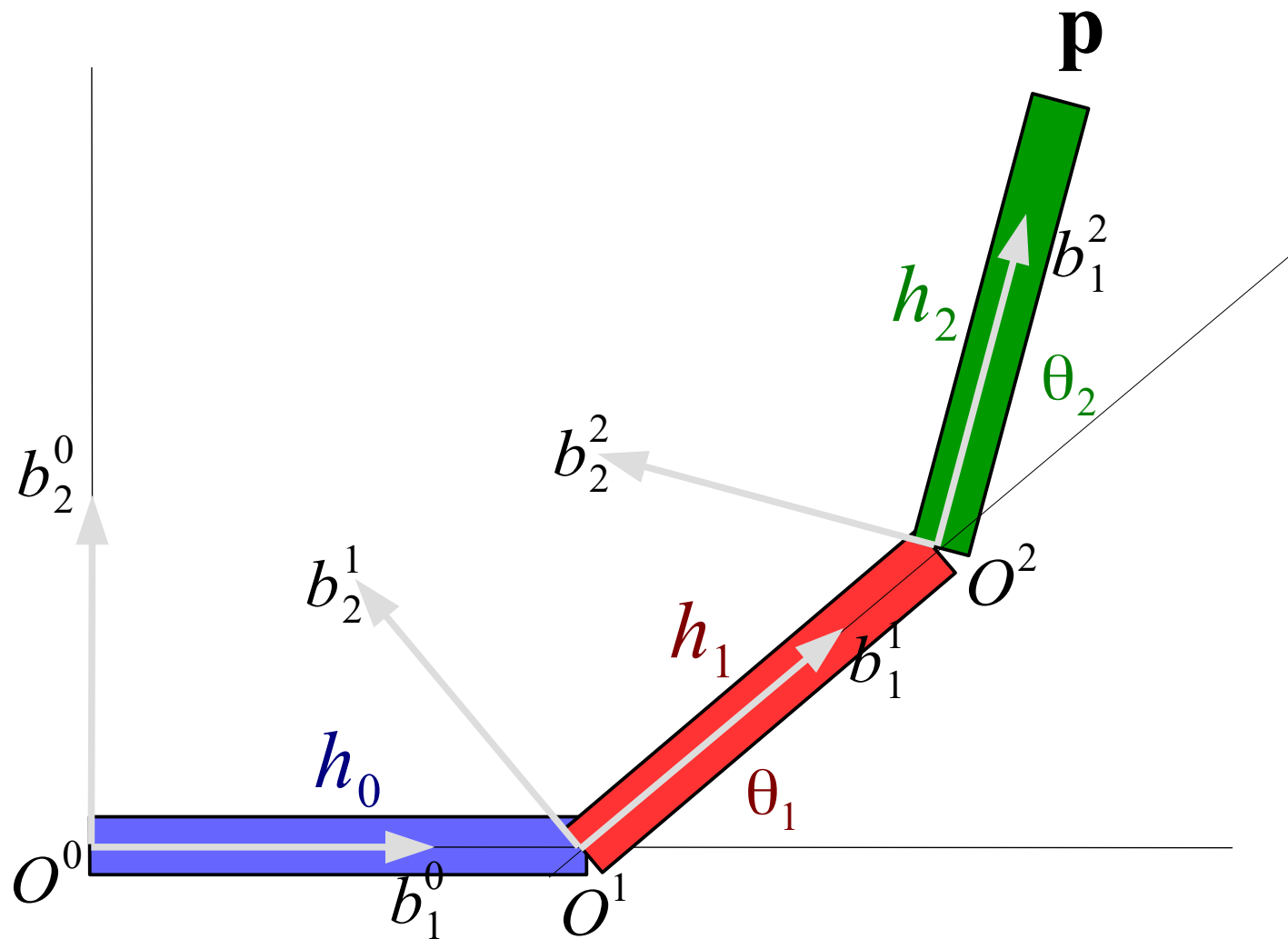
Experimente:

- Dê um clique do mouse para movimentar o retângulo vermelho.
- Clique e arraste (com o botão esquerdo pressionado).
- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Por que o redimensionamento da janela provoca um efeito diferente em relação aos exemplos 2 e 3?

Exemplo 5

Animação-Timer

Exemplo 5 (Animação-Timer)



Exemplo 5 (Animação-Timer)

```
// Exemplo Braço Mecânico 2D

#include <GL/freeglut.h>

#define PI 3.141592654
#define tAnima 5.0
#define deltaT 33

float theta1, theta2, thetaMax, thetaMin, hMax, h0, h1, h2, incremento;

void draw(void);
void keyboard(unsigned char key, int x, int y);
void setTimer(int value);
void reshapeWindow(GLsizei w, GLsizei h);
```

Exemplo 5 (Animação-Timer)

```
// Exemplo Braço Mecânico 2D

#include <GL/freeglut.h>

#define PI 3.141592654
#define tAnima 5.0
#define deltaT 33

float theta1, theta2, thetaMax, thetaMin, hMax, h0, h1, h2, incremento;

void draw(void);
void keyboard(unsigned char key, int x, int y);
void setTimer(int value);
void reshapeWindow(GLsizei w, GLsizei h);
```

Exemplo 5 (Animação-Timer)

```
int main(int argc, char ** argv)
{
    h0 = 6; h1 = 4; h2 = 2;
    hMax = h0 + h1 + h2;
    theta1 = theta2 = 0;
    thetaMax = PI/2;
    thetaMin = -PI/2;
    incremento = 0.1;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutCreateWindow("Braço Mecânico 2D");
    glutPositionWindow(200, 100);
    glutReshapeWindow(200, 400);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, hMax, -hMax, hMax);
```

Exemplo 5 (Animação-Timer)

```
    glutDisplayFunc(draw);
    glutKeyboardFunc(keyboard);
    glutTimerFunc(deltaT, setTimer, 1);
    glutReshapeFunc(reshapeWindow);

    glutMainLoop();
    return 0;
}
void draw(void)
{
    glClearColor(1, 1, 1, 1);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glLineWidth(20);
    /* haste 0 */
    glBegin(GL_LINES);
        glColor3f(0, 0, 1);
        glVertex2f(0.0, 0.0);
        glVertex2f(h0, 0.0);
    glEnd();
}
```

Exemplo 5 (Animação-Timer)

```
/* haste 1 */
glTranslatef(h0, 0, 0);
glRotatef(180.0*theta1 / PI, 0.0, 0.0, 1.0);
glBegin(GL_LINES);
    glColor3f(1, 0, 0);
    glVertex2f(0.0, 0.0);
    glVertex2f(h1, 0.0);
glEnd();
/* haste 2 */
glTranslatef(h1, 0, 0);
glRotatef(180.0*theta2 / PI, 0.0, 0.0, 1.0);
glBegin(GL_LINES);
    glColor3f(0, 1, 0);
    glVertex2f(0.0, 0.0);
    glVertex2f(h2, 0.0);
glEnd();
glutSwapBuffers();
}
```

Exemplo 5 (Animação-Timer)

```
void keyboard(unsigned char key, int x, int y)
{
    if (key == 27) exit(0);
}
void setTimer(int value)
{
    theta1 = theta1 + incremento;
    if ((theta1>=thetaMax) || (theta1<=thetaMin)) incremento *= -1;
    theta2 = theta1 * 2;
    glutPostRedisplay();
    glutTimerFunc(deltaT, setTimer, 1);
}
void reshapeWindow(GLsizei w, GLsizei h)
{
    if (h == 0) h = 1;
    glViewport(0, 0, h/2, h);
}
```


Exemplo 5 (Animação-Timer)

Experimente:

- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Modifique o código para fazer com que os ângulos θ_1 e θ_2 sejam alterados pelo teclado.

Fim