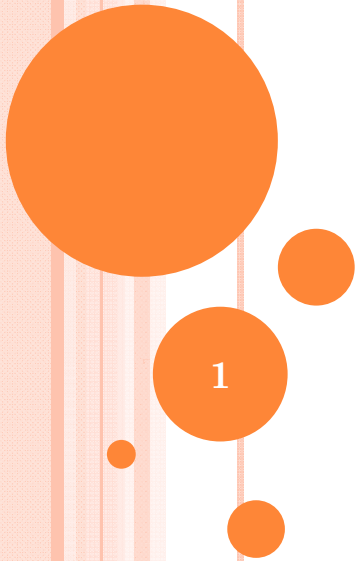


Introdução à OpenGL

Pedro Henrique Bugatti

Roteiro

Desmistificando OpenGL

- O que é OpenGL
- O que OpenGL faz?
- Onde entra OpenGL no aparato de geração de gráficos?
- Bibliotecas auxiliares

Programando Aplicativos Gráficos

- Preparando ambiente Dev C++
- O que é GLUT?
- Estrutura de um aplicativo gráfico com GLUT
- Comandos básicos de OpenGL

Objetivo

- Nas aulas será realizada um pequena introdução à biblioteca OpenGL
  - Gama extensa de funções
- Cabe a vocês aprofundarem os conceitos aqui apresentados
- Explorando tais conceitos por vocês mesmos

HISTÓRICO

- Silicon Graphics (SGI) em 1982 implementa pipeline em Hardware (aumento velocidade). Inicia o desenvolvimento de uma biblioteca gráfica denominada GL para facilitar programação de aplicações interativas e 3D;

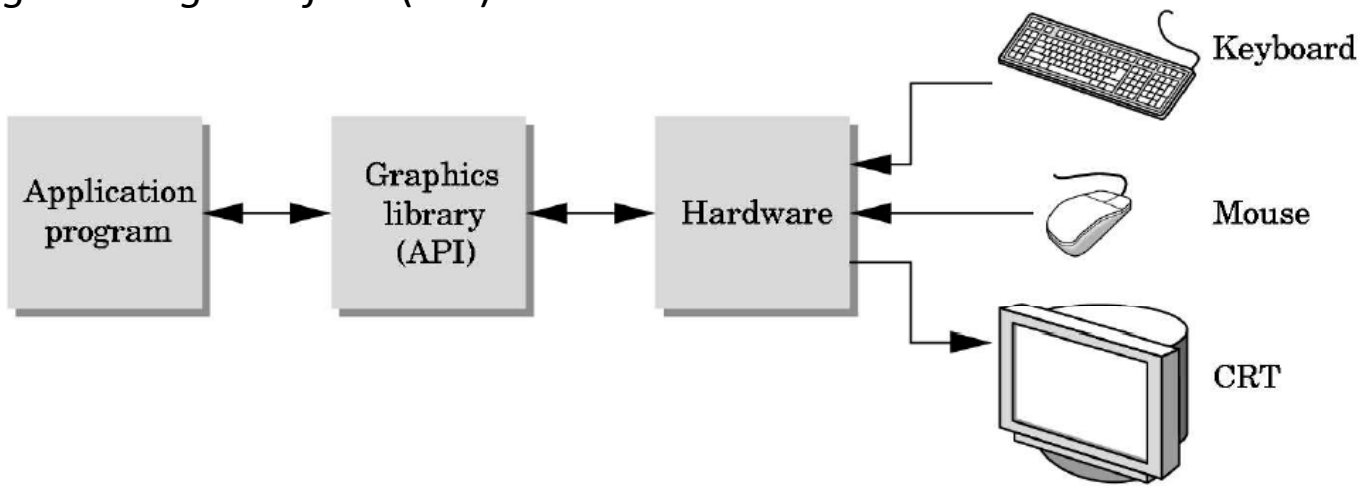
- O sucesso da GL levou ao desenvolvimento da OpenGL em 1992, uma API:
  - Independente de plataforma;
  - Fácil de usar
  - Apresenta facilidade para extrair todo o potencial do hardware

HISTÓRICO
OPENGL (OPEN GRAPHICS LIBRARY)

- Especificação atualmente gerenciada por um consórcio independente formado em 1992:
  - ARB (*Architecture Review Board)*
- formado por empresas líderes da área como *Sun*, *3DLabs*, *Apple*, *NVIDIA*, *SGI*, entre outras
- Consórcio responsável pela aprovação de novas funcionalidades, versões e extensões da OpenGL.
- para informações atualizadas consulte: www.opengl.org

Interface de Programação

Programadores enxergam o sistema gráfico através de uma interface *Application Programming Interface* (API)



Definição:

OpenGL nada mais é do que uma interface de software (API) para o hardware gráfico que é independente de dispositivos.

Objetivos OpenGL

- Encapsular a complexidade de interfaces de diferentes *hardwares;*
- Encapsular as diferentes capacidades dos hardwares, fazendo com que todas implementações suportem o mesmo conjunto de funcionalidades.

OpenGL

- Na verdade, OpenGL é uma biblioteca de rotinas gráficas e de modelagem, bi (2D) e tridimensional (3D), extremamente portável e rápida.

- A maior vantagem na sua utilização é a rapidez, uma vez que usa algoritmos cuidadosamente desenvolvidos e otimizados pela *Silicon Graphics, Inc.*, líder mundial em Computação Gráfica e Animação.

- É importante ressaltar que:
  - OpenGL não é uma linguagem de programação, mas sim uma poderosa e sofisticada API (*Application Programming Interface*) para criação de aplicações gráficas 2D e 3D.

  - Seu funcionamento é semelhante ao de uma biblioteca C, uma vez que fornece uma série de funcionalidades.

OpenGL

- Além do desenho de primitivas gráficas, tais como linhas e polígonos, OpenGL dá suporte a iluminação, colorização, mapeamento de textura, transparência, animação, entre muitos outros efeitos

OpenGL

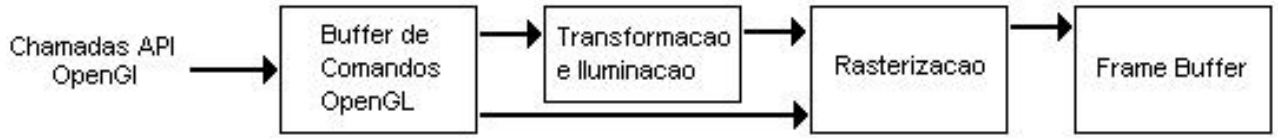
- Núcleo OpenGL: fornece um conjunto de comandos muito importantes para a modelagem e visualização de objetos geométricos;

- OpenGL Utility Library (GLU): criada para fornecer funcionalidades adicionais ao núcleo da open-gl;

- GLUT:
  - fornece uma API portável para a criação de janelas e interações com dispositivos de I/O.

PIPELINE OPENGL

- A palavra *pipeline* é usada para descrever um processo que pode ter dois ou mais passos distintos.



- Versão simplificada do *pipeline* OpenGL.
  - Quando uma aplicação faz chamadas às funções API OpenGL, tais comandos são colocados em um *buffer*. Este *buffer* é preenchido com comandos, vértices, dados de textura, etc. Quando este *buffer* é "esvaziado", os comandos e dados são passados para o próximo estágio
  - Após a etapa de aplicação das transformações geométricas (translação, rotação, cisalhamento, etc) e da iluminação, é feita a rasterização, isto é, é gerada a imagem a partir dos dados geométricos, dados de cor e textura. A imagem final, então, é colocada no *frame buffer*, que é a memória do dispositivo gráfico. Isto significa que a imagem é exibida no monitor

OpenGL – Funções

- Fornece diversas funções que especificam o que é necessário para formar uma imagem:

- Objetos, Posição da Câmera/Observador, Fonte de Luz, Material, Dispositivos de Interação, etc.

- Importante ressaltar que:

  - OpenGL é uma máquina de estados!

OpenGL – Máquina de Estados

- Sendo uma máquina de estados
  - É possível colocá-la em vários estados (ou modos) que não são alterados, a menos que uma função seja chamada para isto.
  - Por exemplo, a cor corrente utilizada para colorir os objetos é uma variável de estado que pode ser definida como branco. Todos os objetos, então, são desenhados com a cor branca, até o momento em que outra cor corrente é especificada.
- Dessa forma, suas funções são de dois tipos:
  - Execução de Primitivas:
    - Objetos são processados para atualizar a visualização;
  - Controle dos Estados:
    - Mudança de estados;
    - Alteração de valores de atributos;

OpenGL – Máquina de Estados

- OpenGL mantém uma série de variáveis de estado, tais como estilo (ou padrão) de uma linha, posições e características das luzes, e propriedades do material dos objetos que estão sendo desenhados.
  - Muitas delas referem-se a modos que podem ser habilitados ou desabilitados com os comandos *glEnable()* e *glDisable()*.

- Cada variável de estado possui um valor inicial (*default*) que pode ser alterado. As funções que utilizadas para saber o seu valor são:
  - *glGetBooleanv()*
  - *glGetDoublev()*
  - *glIsEnabled()*
  - .....

  - Dependendo do tipo de dado, é possível saber qual destas funções deve ser usada

OpenGL – Máquina de Estados

```
int luz;

//Habilita luz - GL_LIGHTING é a variável de estado
glEnable(GL_LIGHTING);

// retorna 1 (verdadeiro)
luz = glIsEnabled(GL_LIGHTING);

//Desabilita luz
glDisable(GL_LIGHTING);

// retorna 0 (falso)
luz = glIsEnabled(GL_LIGHTING);
```

Padronização de Nomes das Funções

- Todos os nomes das funções OpenGL seguem uma convenção que indica de qual biblioteca a função faz parte e, freqüentemente, quantos e que tipos de argumentos a função tem.

- Todas as funções OpenGL possuem o seguinte formato:

<PrefixoBiblioteca>
<ComandoRaiz>
<ContadorArgumentosOpcional>
<TipoArgumentosOpcional>

Padronização de Nomes das Funções

- void glColor3f(Glfloat red, Glfloat green, Glfloat blue)

gl – prefixo da biblioteca gl

Color – objetivo da função

3 – contador de argumentos (opcional)

f - tipo dos argumentos (no caso, ponto flutuante) (opcional)

- Por exemplo, a função *glColor3f* possui *Color* como raiz. O prefixo *gl* representa a biblioteca *gl*, e o sufixo *3f* significa que a função possui três valores de ponto flutuante como parâmetro.

Padronização de Nomes das Funções

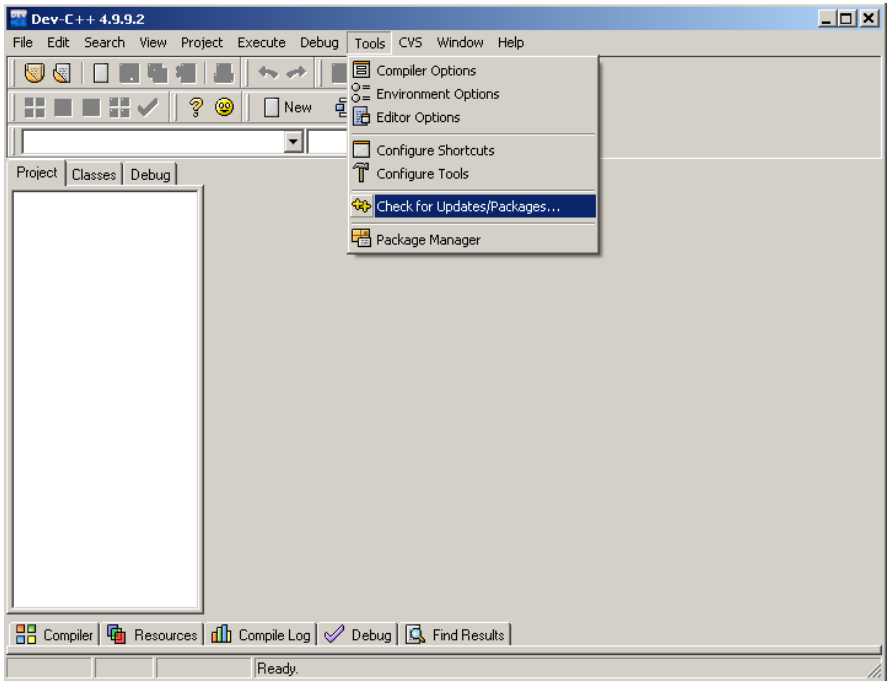
- Outras funções semelhantes:
  - glColor3i
  - glColor3d
  - glColor4f //um componente a mais para especificar opacidade

  - Variações da função do exemplo anterior, *glColor3f*, podem receber três valores inteiros como parâmetro (*glColor3i*), três *doubles* (*glColor3d*) e assim por diante

  - Algumas versões da *glColor* também recebem quatro argumentos. Neste caso, um dos argumentos é usado para especificar o componente alfa (transparência)

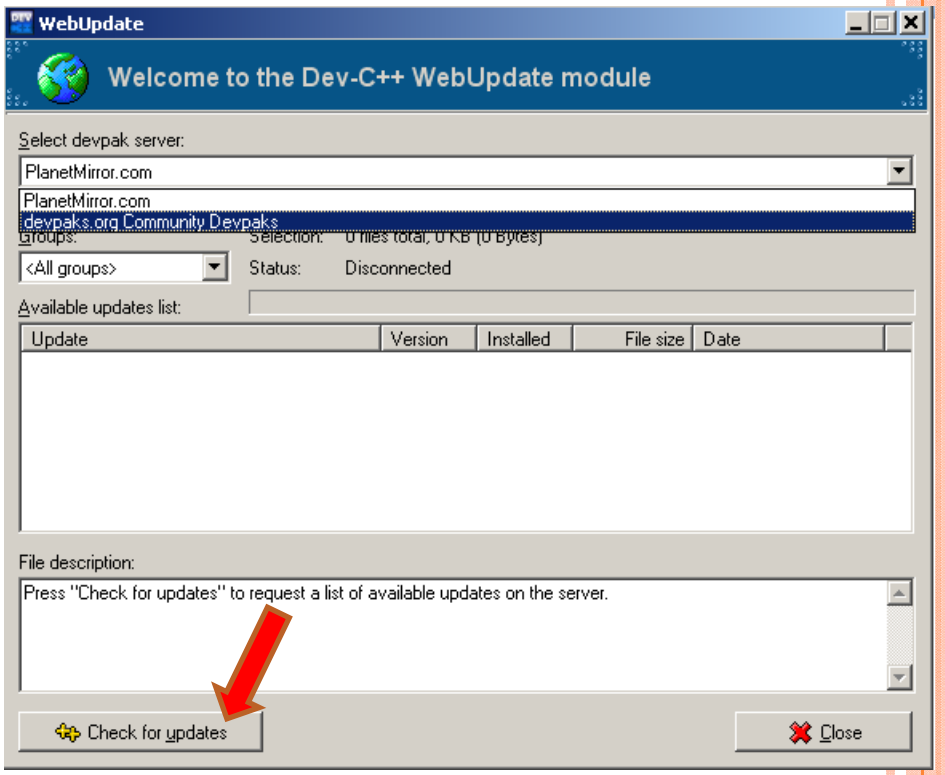
INSTALAÇÃO DA OPENGL – DEVC++

1. No menu “Tools”, selecione “Check for Updates”.

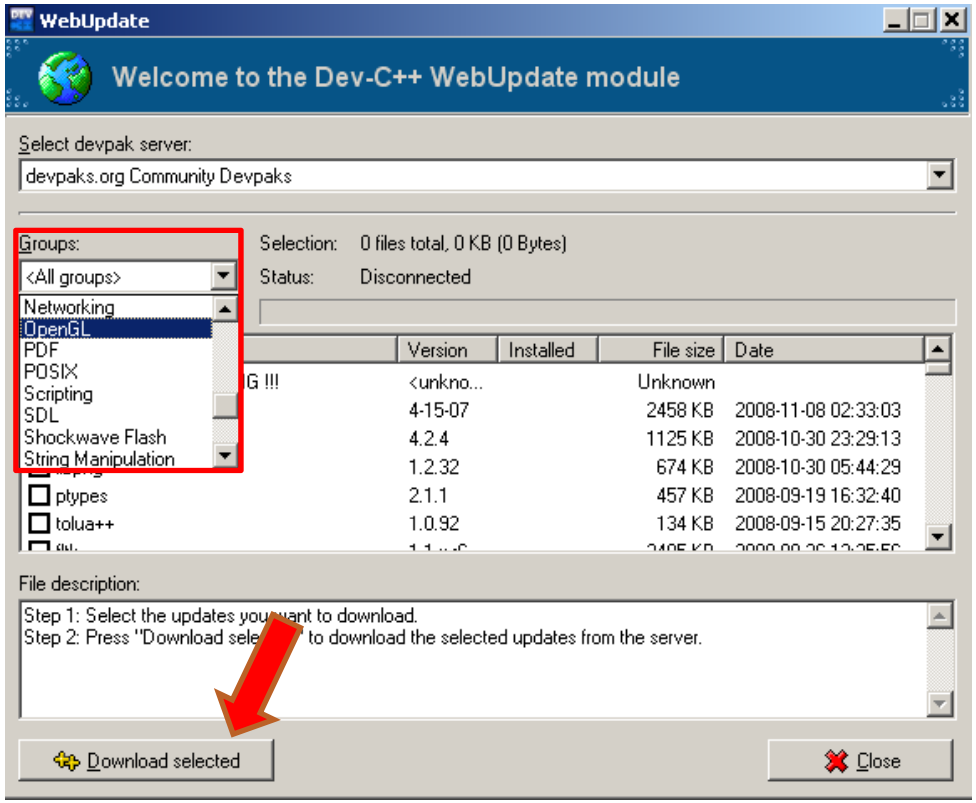


2. Em “Select devpack server” escolher “devpaks.org Community DevPaks”;

3. Clicar em “Check for updates”



INSTALAÇÃO DA OPENGL – DEVC++



4. No Menu "Groups", selecione "OpenGL";

5. Marcar a opção "glut" e "OpenGL" na lista de updates, e clicar no botão "Download selected".

Uma janela deve ser aberta automaticamente para instalar as bibliotecas Open-GL e a GLUT.

OpenGL – Includes

- Maioria das funções e constantes da OpenGL é definida nos arquivos:
  - gl.h
  - glu.h
  - glut.h
  - `#include <GL/gl.h>`
  - `#include <GL/glu.h>`
  - `#include <GL/glut.h>`

GLUT – GL Utility Toolkit

- Antes de começar a programar em OpenGL vamos apresentar a GLUT para facilitar o desenvolvimento de interfaces onde os objetos OpenGL serão exibidos...

GLUT – GL Utility Toolkit

- Biblioteca criada para facilitar o desenvolvimento de interfaces

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

- Biblioteca capaz de:
  - Tratar eventos de teclado, mouse;
  - Criar e gerenciar janelas;
  - Gerenciar desenhos de objetos em OpenGL;

- Para utilizar a GLUT:
  - Incluir arquivo de cabeçalho glut.h;
  - Incluir biblioteca glut32.lib;

- Para executar
  - Incluir glut32.dll;

GLUT – GL UTILITY TOOLKIT

- glutInitDisplayMode: inicializa o uso da biblioteca e define o modo de operação :
  - GLUT_DOUBLE: determina o uso de dois buffers de cores, um para a imagem corrente na tela e outro para a imagem que está sendo construída. Para forçar a troca de imagem na tela é usado a rotina glutSwapBuffers;
  - GLUT_SINGLE: determina o uso de um buffer de cor. O comando que atualiza a imagem na tela é glFlush;
  - GLUT_DEPTH: determina o uso de um buffer de profundidade (z-buffer);
  - GLUT_RGB/ GLUT_RGBA: define que as cores estarão no formato RGB e o A indica o nível de transparência;
  - GLUT_INDEX: indica o uso de índices e armazenamento das cores em uma tabela de cores.

GLUT – GL Utility Toolkit

- Algumas funções de Inicialização da GLUT:
  - glutInitWindowPosition(int x,int y):
    - posicionamento da janela do canto esquerdo da superior da janela;
  - glutInitWindowSize(int width,int height)
    - tamanho da janela;
  - int glutCreateWindow(char * string)
    - criação da janela (retorna idwim da mesma)
  - glutDestroyWindow(int idwin):
    - Destrói uma janela criada anteriormente.
  - glutMainLoop();
    - Inicia o processamento e aguarda interações do usuário.

GLUT – GL Utility Toolkit

- A GLUT é baseada em eventos. É necessário associar funções de tratamento aos eventos:
  - glutDisplayFunc(fredesenho)
    - registra função que será chamada para o redesenho da tela (chamada também ao iniciar a janela);
  - glutReshapeFunc(falter_tam)
    - registra função que será chamada quando houver mudança no tamanho da janela;
  - glutKeyboardFunc(ftrata_tecl)
    - registra função que será chamada quando uma tecla for pressionada;
  - glutMouseFunc(fmouse_f)
    - registra função que será chamada quando houver mudança na posição do mouse e algum botão do mesmo for pressionado ou liberado.

GLUT Exemplo

- exemplo01
- explorando o exemplo

```
#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();
}

// 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(void)
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Primeiro Programa");
    glutDisplayFunc(Desenha);
    Inicializa();
    glutMainLoop();
}
```

GLUT - EXEMPLO

- *Este programa simples contém quatro funções da biblioteca GLUT (prefixo glut), e três funções OpenGL (prefixo gl). O conteúdo deste programa é descrito detalhadamente a seguir.*

- ***glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);*** *avisa a GLUT que tipo de modo de exibição deve ser usado quando a janela é criada.*

- Neste caso os argumentos indicam a criação de uma janela single-buffered (GLUT_SINGLE) com o modo de cores RGBA (GLUT_RGB). O primeiro significa que todos os comandos de desenho são feitos na janela de exibição. Uma alternativa é uma janela double-buffered, onde os comandos de desenho são executados para criar uma cena fora da tela para depois rapidamente colocá-la na view (ou janela de visualização). Este método é geralmente utilizado para produzir efeitos de animação. O modo de cores RGBA significa que as cores são especificadas através do fornecimento de intensidades dos componentes red, green e blue separadas

GLUT - EXEMPLO

- ***glutCreateWindow("Primeiro Programa");*** é o comando da biblioteca GLUT que cria a janela. Neste caso, é criada uma janela com o nome "Primeiro Programa". Este argumento corresponde a legenda para a barra de título da janela.

- ***glutDisplayFunc(Desenha);*** estabelece a função "Desenha" previamente definida como a função callback de exibição. Isto significa que a GLUT chama a função sempre que a janela precisar ser redesenhada. Esta chamada ocorre, por exemplo, quando a janela é redimensionada ou encoberta. É nesta função que se deve colocar as chamadas de funções OpenGL, por exemplo, para modelar e exibir um objeto.

- ***Inicializa();*** não é uma função OpenGL nem GLUT, é apenas uma convenção utilizada no exemplo. Nesta função são feitas as inicializações OpenGL que devem ser executadas antes do rendering. Muitos estados OpenGL devem ser determinados somente uma vez e não a cada vez que a função “Desenha” é chamada.

GLUT - EXEMPLO

- ***glutMainLoop();*** é a função que faz com que comece a execução da "máquina de estados" e processa todas as mensagens específicas do sistema operacional, tais como teclas e botões do mouse pressionados, até que o programa termine.

- ***glClearColor(0.0f, 0.0f, 1.0f, 1.0f);*** é a função que determina a cor utilizada para limpar a janela de visualização.

  Seu protótipo é: void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alfa);. GLclampf é definido como um float na maioria das implementações de OpenGL. O intervalo para cada componente red, green, blue é de 0 a 1. O componente alfa é usado para efeitos especiais, tal como transparência.

- ***glClear(GL_COLOR_BUFFER_BIT);*** "limpa" um buffer particular ou combinações de buffers, onde buffer é uma área de armazenamento para informações da imagem. Os componentes RGB são geralmente referenciados como color buffer ou pixel buffer. Existem vários tipos de buffer, mas por enquanto só é necessário entender que o color buffer é onde a imagem é armazenada internamente e limpar o buffer com glClear remove o desenho da janela.

GLUT - EXEMPLO

- ***glFlush();*** *faz com que qualquer comando OpenGL não executado seja executado. Neste primeiro exemplo tem apenas a função glClear*

- explorando o exemplo
  - Modifiquem a cor de fundo da janela de visualização
  - .....

- Exemplo Hello.c
  - Explorem o exemplo
    - glutInitWindowSize (250, 250);
    - glutInitWindowPosition (100, 100);
    - glutCreateWindow ("Exemplo 1");
    - glColor3f (1.0, 1.0, 1.0);
    - Baseado no código apresentado - Desenhem um retângulo de cor azul

Modelagem de Objetos

- Matrizes de trabalho da OpenGL:
  - Imagem Corrente (GL_MODELVIEW);
  - Projeção (GL_PROJECTION);
  - Textura (GL_TEXTURE);

- void **glMatrixMode**(GLenum *mode*)
  - define matriz de trabalho para as próximas operações;
  - *mode* especifica qual matriz é a alvo das subseqüentes operações.
  - Três valores são aceitos:
    - **GL_MODELVIEW**,
    - **GL_PROJECTION**,
    - **GL_TEXTURE**.

- O valor inicial é **GL_MODELVIEW**.

MODELAGEM DE OBJETOS

- ***glMatrixMode(GL_MODELVIEW);*** avisa a OpenGL que todas as futuras alterações, tais como operações de escala, rotação e translação, irão afetar os modelos da cena, ou em outras palavras, o que é desenhado.

- Exemplos

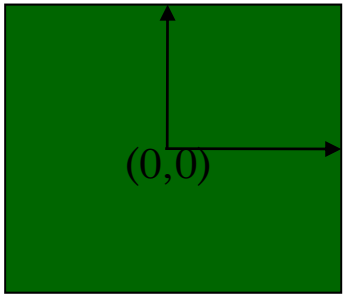
```
glMatrixMode(GL_PROJECTION);

glMatrixMode(GL_MODELVIEW);
```

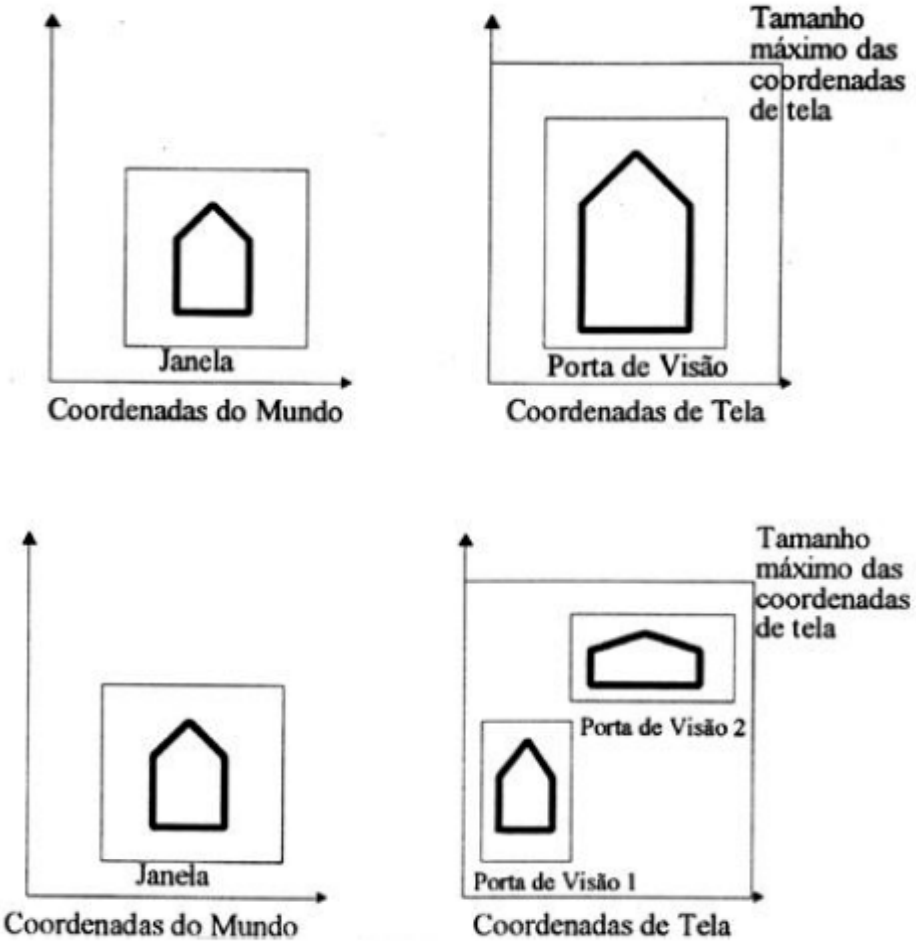
Orientação da Tela

Orientação bidimensional da tela em OpenGL

Quando não se usa a transformação de porta de visão, as coordenadas da tela variam de -1 a 1.



Transformação de Porta de Visão



Transformação de Porta de Visão

Define um retângulo de pixel na janela na qual a imagem final é desenhada.

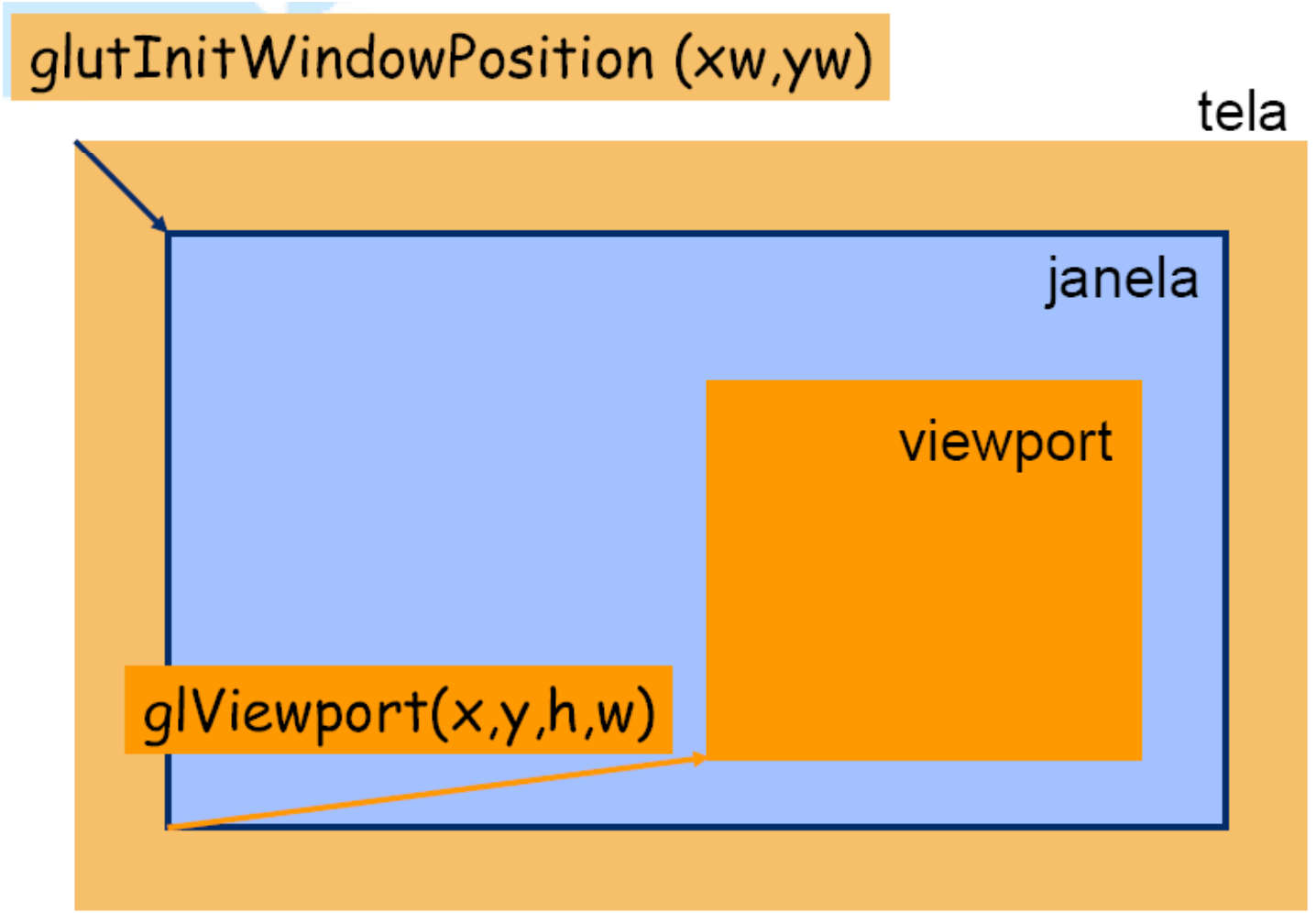
void glViewport(GLint x, y, GLsizei width, height);

- x, y especifica o canto inferior-esquerdo da viewport;
- width, height são a largura e altura (número de pixels) do viewport.

Os valores de viewport iniciais por padrão são:
(0, 0, winWidth, winHeight)

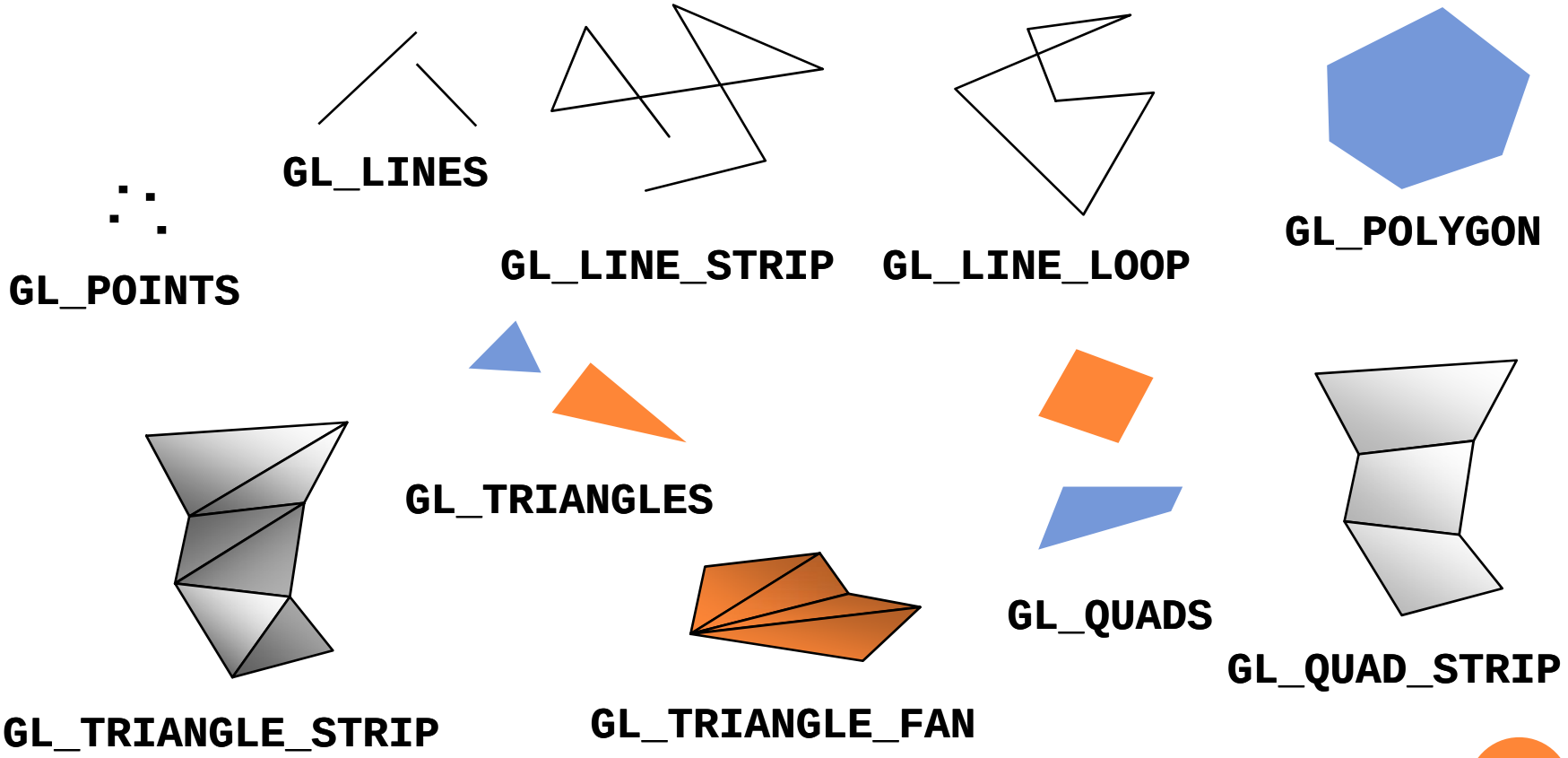
onde winWidth e winHeight são o tamanho da janela.

Transformação de Porta de Visão



Primitivas Geométricas OpenGL

- Todas as primitivas estão especificadas pelos vértices



PRIMITIVAS GEOMÉTRICAS OPENGL

- Com apenas algumas primitivas simples, tais como pontos, linhas e polígonos, é possível criar estruturas complexas.

- Em outras palavras, objetos e cenas criadas com OpenGL consistem em simples primitivas gráficas que podem ser combinadas de várias maneiras

- OpenGL fornece ferramentas para desenhar pontos, linhas e polígonos, que são formados por um ou mais vértices

- Neste caso, é necessário passar uma lista de vértices, o que pode ser feito entre duas chamadas de funções OpenGL:

  glBegin()
  glEnd()

- O argumento passado para *glBegin()* determina qual objeto será desenhado.

PRIMITIVAS GEOMÉTRICAS OPENGL

- Para desenhar três pontos pretos é utilizada a seguinte seqüência de comandos:

```
glBegin(GL_POINTS);
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex2i(100, 50);
    glVertex2i(100, 130);
    glVertex2i(150, 130);
glEnd();
```

PRIMITIVAS GEOMÉTRICAS OPENGL

- Para desenhar outras primitivas, basta trocar *GL_POINTS*, que exibe um ponto para cada chamada ao comando *glVertex*, por:
  - GL_LINES: exibe uma linha a cada dois comandos *glVertex*;
  - GL_LINE_STRIP: exibe uma seqüência de linhas conectando os pontos definidos por *glVertex*;
  - GL_LINE_LOOP: exibe uma seqüência de linhas conectando os pontos definidos por *glVertex* e ao final liga o primeiro como último ponto;
  - GL_POLYGON: exibe um polígono convexo preenchido, definido por uma seqüência de chamadas a *glVertex*;
  - GL_TRIANGLES: exibe um triângulo preenchido a cada três pontos definidos por *glVertex*;
  - GL_TRIANGLE_STRIP: exibe uma seqüência de triângulos baseados no trio de vértices v0, v1, v2, depois, v2, v1, v3, depois, v2, v3, v4 e assim por diante;

PRIMITIVAS GEOMÉTRICAS OPENGL

- GL_TRIANGLE_FAN: exibe uma seqüência de triângulos conectados baseados no trio de vértices v0, v1, v2, depois, v0, v2, v3, depois, v0, v3, v4 e assim por diante;
- GL_QUADS: exibe um quadrado preenchido conectando cada quatro pontos definidos por *glVertex*;
- GL_QUAD_STRIP: exibe uma seqüência de quadriláteros conectados a cada quatro vértices; primeiro v0, v1, v3, v2, depois, v2, v3, v5, v4, depois, v4, v5, v7, v6, e assim por diante

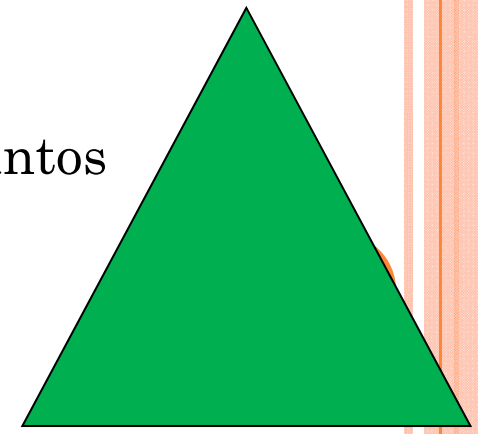
Primitivas geométricas OpenGL

- Desenhando um Triângulo:

```
glBegin(GL_TRIANGLES);
        glVertex3f(x1,y1,z1);
        glVertex3f(x2,y2,z2);
        glVertex3f(x3,y3,z3);
glEnd();
```

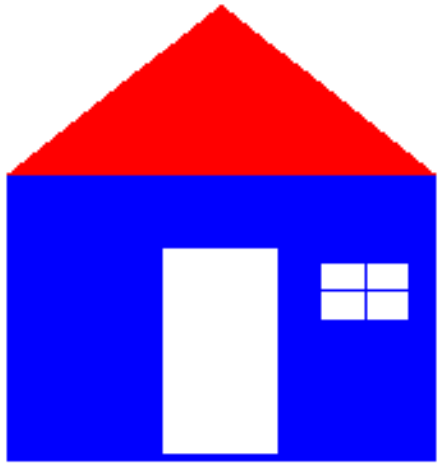
- Exercício 1:
  - Desenhe um triângulo de cor verde

- Exercício 2:
  - Desenhe um triângulo cujos vértices sejam os cantos inferiores da janela e o cento da borda superior.



PRIMITIVAS GEOMÉTRICAS OPENGL

- Exercício 3:
  - Desenhem um casa utilizando as primitivas apresentadas

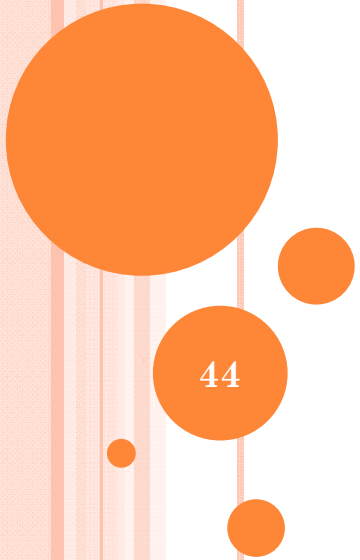


GL_TRIANGLES – telhado

GL_QUADS – porta, janela, corpo da casa

GL_LINES – linhas da janela

- Exercício 4:
  - Explorem o exercício 3 criando uma casa tridimensional



Introdução ao OpenGL

Pedro Henrique Bugatti