Introdução ao OpenGL

Profa. M. Cristina
Profa. Rosane

OpenGL

- *Application Programming Interface* (API)
  - Coleção de rotinas que o programador pode chamar do seu programa
  - Modelo de como estas rotinas operam em conjunto para gerar gráficos
  - Programador ´enxerga´ apenas a interface
  - Não precisa lidar com aspectos específicos do hardware ou idiossincracias de software no sistema gráfico residente
  - Oferece suporte para gerar e exibir cenas 3D complexas, e também para gráficos 2D simples

OpenGL

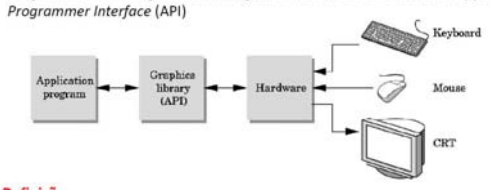
- Ambiente p/ escrever e executar programas gráficos
  - Monitor (´tela´) + biblioteca de software + placa gráfica
    - para desenhar primitivas gráficas na tela
- API pode ser vista como uma ´caixa preta´
  - Interface de software para o hardware gráfico que é independente de dispositivos (monitor, placa gráfica)

OpenGL

- API pode ser vista como uma ´caixa preta´, eescrita em termos das funções que disponibiliza
  - Entradas
    - chamadas a funções da biblioteca feitas pelo programa do usuário
    - Informações obtidas dos dispositivos de entrada (mouse, teclado...)
    - ...
  - Saídas: gráficos exibidos no monitor
  - Objetivo: encapsular a complexidade de interfaces de diferentes hardwares; encapsular as diferentes capacidades dos hardwares
    - Sistema gráfico suporta o mesmo conjunto de funcionalidades, mesmo que seja necessário emular algumas em software

OpenGL

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



OpenGL

- Independente de plataforma
- Fácil de usar
- Permite à aplicação extrair o melhor potencial do hardware

Terminologia

- Rendering: processo de criar imagens a partir de modelos geométricos
- Modelos (objetos): objetos definidos em termos de primitivas geométricas simples, como pontos, retas ou polígonos
  - Especificados pelos seus vértices
- Pixel: menor elemento visível no hardware de exibição
- Imagem renderizada: os pixels apresentados na tela. A informação em cada pixel é mantida em memória, organizada em *bitplanes*.

Terminologia

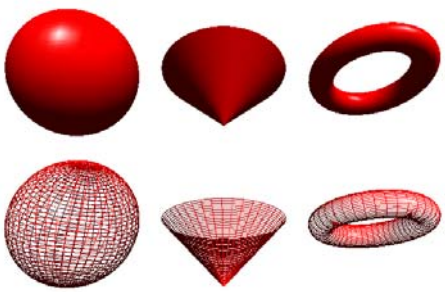
- Framebuffer: memória que retém as necessárias para o display controlar a cor, intensidade e opacidade de todos os pixels na tela
- Bitplane: área da memória que retém um bit de informação para cada pixel na tela

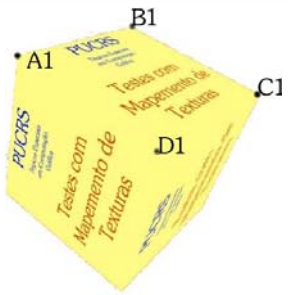
O que o OpenGL não faz

- Não inclui comandos para tratar janelas e entrada de dados
  - Feito pela GLUT
- Não inclui comandos de alto nível para descrever modelos geométricos complexos
  - Modelos de superfícies descritos por malhas de polígonos
- Não implementa algoritmos complexos de *rendering*, como *ray tracing* e radiosidade
  - Implementa o *scanline*

O que o OpenGL faz

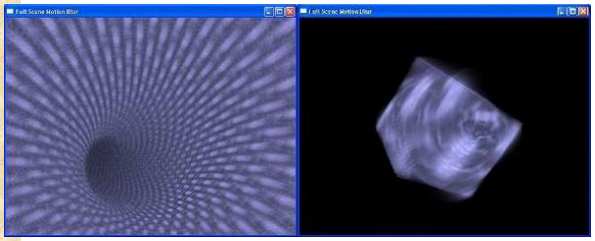
- Calcula transformações geométricas e projeções (paralela, perspectiva)
  - Define um modelo de câmera virtual
- Implementa modelo de iluminação para *smooth shading*
- Implementa diversos recursos para aumentar o realismo das cenas
  - Mapeamento de texturas, *fog*, *motion-blur*





http://www.inf.pucrs.br/~pinho/CG/Aulas/OpenGL/Texturas/MapTextures.html

http://gpwiki.org/index.php/OpenGL:Tutorials:Tutorial_Framework:Light_and_Fog



http://www.codeproject.com/KB/openGL/MotionBlur.aspx

O que o OpenGL faz

- Implementa *clipping* e algoritmos de rasterização
- Suporta linguagem para implementação de shaders: rotinas executadas no hardware para processamento de vértices e fragmentos
- Funções executadas no hardware: alto desempenho

OpenGL – como funciona

- Fornece funções que permitem especificar os elementos necessários para gerar uma imagem:
  - Objetos, Posição da Câmera/Observador, Fonte de Luz, Materiais, Dispositivos de Interação, Capacidade do Sistema, etc.
- Programa OpenGl é uma máquina de estados, com 2tipos de funções
  - Execução de primitivas: objetos são processados para atualizar a cena
  - Controle dos estados: mudança de estados e alteração dos valores dos atributos (ex., cor, espessura de primitiva...)

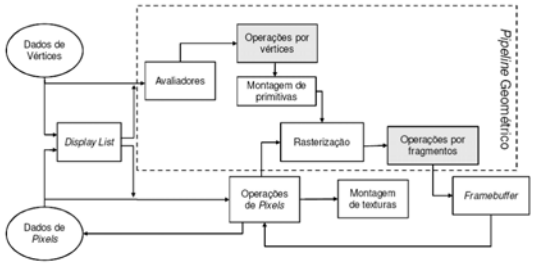
OpenGL – como funciona

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)

- Outras funções semelhantes: glColor3i, glColor3d, glColor4f

OpenGL - Pipeline de Rendering



OpenGL - Pipeline Geométrico

Bibliotecas relacionadas

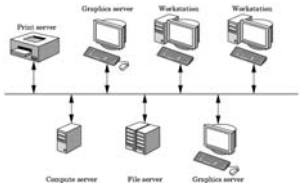
- **GLU:** Definição de curvas e superfícies NURBs (*Non - Uniform Rational B - Spline*); Funções utilitárias para definir parâmetros de visualização
- **GLUT:** Definição de janelas e tratamento de entrada de dados; criação de interfaces gráficas
- **GLUI: c**riação de interfaces gráficas mais elaboradas

API Open GL

- Programa
  - Em geral, trabalha com um sistema de janelas (´window system´) (Fig. 2.1)
  - Inicializações: modo de exibição (´display mode´), janela de desenho e sistema de coordenadas de referência (associado à janela)
- API oferece centenas de funções...
  - diferentes funcionalidades
  1. Funções primitivas: o que
  2. Funções de atributos: como
  3. ...

X Window Input

- X Window System: modelo cliente-servidor para rede de *workstations*
  - **Cliente**: programa OpenGL
  - **Servidor Gráfico**: bitmap display c/ dispositivo apontador e teclado



Modos de Input

- Dispositivos de entrada incluem um *trigger* que sinaliza para o sistema operacional a ocorrência de um evento relevante
  - Botão no mouse pressionado/solto
  - Tecla pressionada/solta
  - …
- Quando ativados, dispositivos de entrada retornam informação (*measure*) para o sistema
  - Mouse retorna informação de posição
  - Teclado retorna código ASCII

Modo Request

- Input fornecido ao programa somente quando o usuário ativa o dispositivo
- Típico para entrada via teclado
  - Pode apagar (backspace), editar, corrigir, etc. – até que a tecla enter (return) é pressionada (o trigger)

Modo Event

- Sistemas em geral tem mais de um dispositivo de entrada – cada qual pode ser ativado arbitrariamente pelo programador
- Cada trigger dispara um evento, cuja ´medida´ vai para uma fila de eventos, a qual é acessada pelo programa

Tipos de Eventos

- Window: resize, expose, iconify
- Mouse: click one or more buttons
- Motion: move mouse
- Keyboard: press/release a key
- Idle: ausência de eventos
  - Define o que deve ser feito se não há eventos na fila

Callbacks

- Interface de programação para entrada ´event-driven
- Define uma *callback function* para cada tipo de evento que o sistema gráfico reconhece
- Essa função, fornecida pelo programador, é executada quando o evento ocorre
- Exemplo GLUT:
  `glutMouseFunc(mymouse)`

mouse callback function

GLUT callbacks

GLUT reconhece um subconjunto de eventos – os que são reconhecidos por sistemas de janelas (Windows, X, Macintosh)

- **`glutDisplayFunc`**
- **`glutMouseFunc`**
- **`glutReshapeFunc`**
- **`glutKeyboardFunc`**
- **`glutIdleFunc`**
- **`glutMotionFunc, glutPassiveMotionFunc`**

GLUT Event Loop

- A última linha em um programa **`main.c`** que usa a GLUT deve ser
  `glutMainLoop();`
- Isso coloca o programa em um loop infinito de tratamento de eventos
- A cada passo desse loop de eventos, a GLUT
  - Verifica os eventos na fila
  - Para cada evento, executa a função callback apropriada (caso tenha sido definida)
  - Se nenhuma callback foi definida para o evento, ele é ignorado

Programação em ambiente de janelas

- Direcionada a eventos (*event-driven*)
  - programa responde a eventos: clique do mouse, tecla pressionada, redimensionamento da janela
- Fila de eventos
  - mensagem informa a ocorrência de um evento, política FIFO de tratamento
  - Programa organizado como coleção de *callback functions*
    - Cada tipo de evento associado a uma *callback* que é ativada quando ele ocorre
  - Modelo de programação diferente do ´procedimental seqüencial´...
    - *Forever*: ´não faça nada até que um evento ocorra, quando isso acontece, trate o evento (ative sua *callback*)...´

OpenGL Utility Toolkit

- Biblioteca para gerenciamento de eventos
  - Ex. glutMouseFunc(myMouse);
  - Registra função myMouse p/ tratar ocorrência de evento do mouse
  - Código da função definido pelo programador


```
#include <gl/glut.h>
void main( )
{
    Inicializações
    Cria janela
    glutDisplayFunc(myDisplay);
    glutReshapeFunc(myReshape);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKeyboard);
    Outras inicializações
    glutMainLoop(); /* entra loop infinito */
}
Definição de todas as callback functions aqui...
```

Esqueleto de um programa OpenGL (*event driven*)

Código para abrir um janela para desenho e desenhar um ponto (principal)

```
int main(int argc, char **argv)
{
    cont=0;
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowSize(400,400); /* Antes do glutCreateWIndow */
    glutInitWindowPosition(1,1);
    glutCreateWindow("Ponto");
    gluOrtho2D(0,399,399,0); /* Apos CreateWindow */
    glutDisplayFunc(redesenha);
    glutMainLoop();
}
```


Código para abrir um janela para desenho e desenhar um ponto (desenho do ponto)

```
void redesenha()
{
    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_POINTS);
        glVertex2f(200.0,200.0);
    glEnd();
    glFlush();
}
```


Quantas vezes uma callback é chamada?
Ex. redesenha

```
int cont; /*inicializada no prog. Principal */
void redesenha()
{
    glClearColor(0.0,0.0,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glColor3f(1.0,0.0,0.0);
    itoa(cont++,buf,10);
    glutSetWindowTitle((const char*)buf );
    glBegin(GL_POINTS);
        glVertex2f(200.0,200.0);
    glEnd();
    glFlush();
}
```

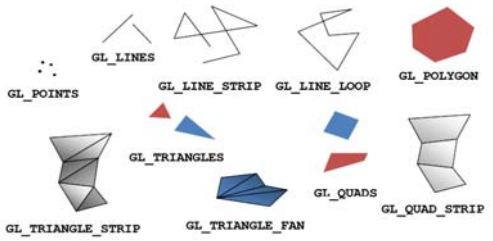
Quando cont é incrementado?

- V. instruções de compilação/execução

Primitivas

- Traçado requer um sistema de referência para posicionamento espacial
  - Em CG trabalha-se com diversos sistemas de coordenadas... Inicialmente, adotamos um muito simples
    - Associado ao sistema de coordenadas da janela de desenho
    - Distâncias medidas em pixels
    - Zero no canto superior esquerdo da janela
- Para desenho de primitivas 2D: GLOrtho2D define transformação necessária para a exibição
  - (janela 640 x 480)

Primitivas básicas: especificadas por seus vértices



Várias janelas/ “janela corrente”

- No código anterior, toda a definição de propriedade e todo traçado ocorria na única janela.
- Múltiplas janelas:
  - usa-se o identificador de janela
  - int w = glutCreateWindow (“nome_janela”);
  - glutSetWindow(w);
  - w passa a ser a “janela corrente”

Exemplo

- Múltiplas primitivas em múltiplas janelas

Primitivas

- Desenho de primitivas
  - Diversos objetos: GL_POINTS, GL_POINTS, GL_LINES, GL_POLYGON, etc.
  - Para descrever objeto, usuário informa a lista de vértices

```
glBegin(GL_POINTS);
        glVertex2i(100, 50);  // draw three points
        glVertex2i(100, 130);
        glVertex2i(150, 130);
glEnd();
```

  - Informação sobre cada vértice encaminhada para um *´pipeline* gráfico´, no qual passa por várias etapas de processamento

Tipos de Dados

- OpenGL suporta um conjunto fixo de tipos de dados.

sufixo	tipo	tipo C	nome
b	inteiro 8 bits	signed char	GLbyte
s	inteiro 16 bits	short	Glshort
i	inteiro 32 bits	int/long	GLint
f	float 32 bits	float	GLfloat
...	...	...	...

Tipos de Dados

```
void drawDot(int x, int y)          //Perigo!!: sistema passa int...
{
    glBegin(GL_POINTS);
        glVertex2i(x, y);        // função c/ sufixo i ´espera´ inteiro 32 bits
    glEnd();
}

void drawDot(GLint x, GLint y)   // código seguro... compilação associa os
{                                // tipos adequadamente (GL.h)
    glBegin(GL_POINTS);
        glVertex2i(x, y);
    glEnd();
}
```

Função genérica para traçar um ponto: cuidado com portabilidade!!

Estados do OpenGL

- OpenGL rastreia diversas variáveis de estado
  - Tamanho atual de um ponto, cor de fundo da janela, cor do desenho
  - O valor corrente permanece ativo até que seja alterado
    - Tamanho de ponto: glPointSize(3.0)
    - Cor de desenho: glColor3f(red, green, blue)
    - Cor de fundo: glClearColor(red, green, blue, alpha)
    - Limpar janela: glClear(GL_COLOR_BUFFER_BIT)

Mais primitivas: linhas

- GL_LINES: extremidades c/o argumentos

```
glBegin(GL_LINES);
    glVertex2i(40, 100);
    glVertex2i(202, 96);
glEnd();
```

- glLineWidth(4.0) // default é 4.0

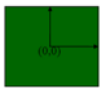
Mais primitivas: linhas

- Encapsulando p/ conveniência

```
void drawLineInt(GLint x1, GLint y1, GLint x2, GLint y2)
{
    glBegin(GL_LINES);
        glVertex2i(x1, y1);
        glVertex2i(x2, y2);
    glEnd();
}
```


Espaço de trabalho

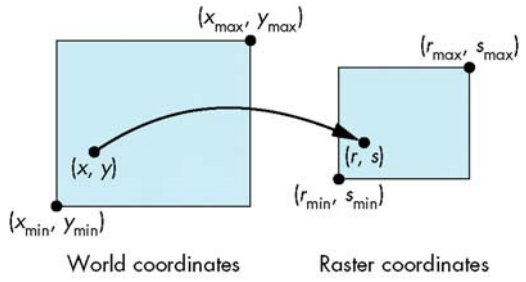
- Exemplos até agora: primitivas desenhadas diretamente no espaço da tela
- Situação usual: aplicativo usa um sistema de coordenadas (´do mundo´), desenho é apresentado no sistema de referência da tela



Sistemas de Coordenadas

- É necessário aplicar uma transformação à descrição da cena para passar de um sistema para outro
- ´mundo´ é infinito, o sistema de referência da tela é finito e discreto: depende da resolução.
- Mapeamento ´window-viewport´

Mapeamento ´window-viewport´



Fonte: E. Angel, Interactive Computer Graphics

Mapeamento ´window-viewport´

- Define área de exibição: glViewport(Glint x, Glint y, Glsizei width, Glsizei height);
  - (x,y): coord´s canto inferior esquerdo
  - width,height: altura e largura em pixels
- Define *window* de interesse no ´mundo´: gluOrtho2D(Gldouble left, Gdouble right, Gldouble bottom, Gldouble top);
  - left, bottom: coord´s canto inferior esquerdo
  - right, top: coord´s canto superior direito

Mapeamento ´window-viewport´

- Note que a janela de exibição (*glutCreateWindow*) pode ter sua área compartilhada por múltiplas *viewports*
- exemplo

Interação com Mouse e Teclado

- glutMouseFunc(myMouse), registra a fç de tratamento do evento de pressionar/soltar botão do mouse
- glutMotionFunc(myMovedMouse), registra a fç de tratamento do evento de mover o mouse com um botão pressionado
- glutKeyboardFunc(myKeyboard), registra a fç de tratamento do evento tecla pressionada

Interação com Mouse e Teclado

- void myMouse(int button, int state, int x, int y)
- Button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
- State: GLUT_UP, GLUT_DOWN
- x, y: posição do mouse no momento da ocorrência do evento
  - Posição do pixel em relação ao sistema de coordenadas com origem no canto superior esquerdo da janela

Interação com Mouse

- Ex. posicionar pontos com o mouse

```
void myMouse(int button, int state, int x, int y)
{
    if(button == GLUT_LEFT_BUTTON && state ==
    GLUT_DOWN)
        drawDot(x, screenHeight – y);
    else if(button == GLUT_RIGHT_BUTTON && state ==
    GLUT_DOWN)
        exit(-1)
}
```

Movimento do Mouse

- void myMovedMouse(int x, int y)

```
void myMovedMouse(int MouseX, int MouseY)
{
   gLint x = mouseX;
   gLint y = ScreenHeight - mouseY;
   gLint brushSize = 20;
   gLRecti(x,y, x + brushSize, y + brushSize);
   glFlush();
}
```


Interação com Teclado

- void myKeyboard(unsigned int key, int x, int y)
- Key: valor ASCII da tecla pressionada

Bibliografia

- Computer Graphics Using OPEN GL, F.S. Hill, Prentice-Hall 2001
- E. Angel, Interactive Computer Graphics, 3a. Edição, Adison Wesley, 2003
- OpenGL: uma abordagem prática e objetiva – Ed. Novatec, Marcelo Cohen e Isabel Harb Manssour