Introdução ao OpenGL

Profa. M. Cristina
Profa. Rosane

OpenGL

- *Application Programming Interface* (API)
  - Coleção de rotinas que o programador pode chamar
  - 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

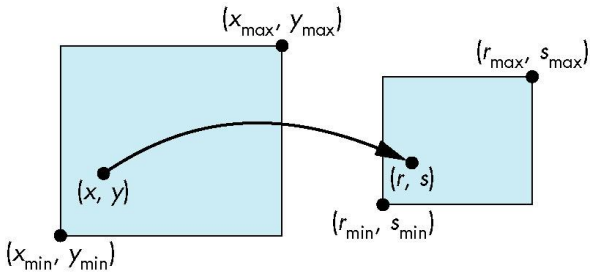
OpenGL

- Ambiente p/ escrever e executar programas gráficos
  - Monitor (´tela´) + biblioteca de software
    - para desenhar primitivas gráficas na tela
- API pode ser vista como uma ´caixa preta´
  - Entradas:
    - chamadas a funções da biblioteca feitas pelo programa do usuário
    - Medidas fornecidas por dispositivos de entrada
    - ...
  - Saídas:
    - Os gráficos exibidos no monitor
  - Descrita em termos das funções que disponibiliza

Sistemas de Coordenadas

- OpenGL permite desenhar gráficos de modo independente do dispositivo
- Usuário especifica elementos de interesse no Sistema de Coordenadas do Usuário, ou Sistema de Coordenadas do Mundo (ponto flutuante)
- Os elementos são traçados no sistema de coordenadas do dispositivo, ou sistema de coordenadas da tela (inteiro)
- OpenGL faz o mapeamento de forma transparente para o usuário

Sistemas de Coordenadas



Fonte: E. Angel, Interactive Computer Graphics

5

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. ...

6

Primitivas de desenho

- Sistemas gráficos oferecem primitivas
  - Ex.: setPixel(x,y,color), moveTo(x,y), Line(x1,y1, x2,y2), ...
- Programação independente do dispositivo
  - Mesmo programa compilado e executado em diferentes ambientes operacionais e sistemas gráficos
    - produzindo saídas (quase) idênticas
- OpenGL
  - Garante portabilidade: basta instalar as bibliotecas na nova máquina; aplicação não precisa ser alterada

7

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*)...´

8

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

9

```
#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*)

10

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();
}
```

11

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();
}
```

12

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?

13

Várias janelas/ "janela corrente"

- No código anterior, toda definição de propriedade e todo traçado ocorre na única janela.
- Mais de uma janela:
  - usa-se o identificador de janela
  - int w = glutCreateWindow ("nome_janela");
  - glutSetWindow(w);
  - w passa a ser a "janela corrente"

14

Código para abrir criar duas janelas e desenhar o ponto em cada uma delas.

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

15

Código para abrir criar duas janelas e desenhar o ponto em cada uma delas (cont).

```
glutInitWindowSize(400,400); /* Antes do glutCreateWIndow */
glutInitWindowPosition(1,1);
int t = glutCreateWindow("Reta");
gluOrtho2D(0,399,399,0);
glutDisplayFunc(redesenha);

//glutSetWindow(t); /*desnecessário já que t é corrente */
glutDisplayFunc(redesenha);

glutSetWindow(w);
glutDisplayFunc(redesenha);

glutMainLoop();
}
```

16

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
    - Distâncias medidas em pixels
    - Zero no canto superior esquerdo da janela
- Primitivas básicas
  - Pontos, linhas, poli-linhas, polígonos
  - Definidos em termos de **vértices**

17

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

18

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
...	...	...	...

19

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!!

20

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)

21

Sistema de Coordenadas

- Sistema simples: transformações entre diversos sistemas definidas por meio de matrizes
  - GLOrtho2D define transformação necessária
  - (janela 640 x 480)

```
void myInit(void)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
```

22

Exemplos

- Programa completo p/ desenhar 3 pontos na tela
  - myInit: inicializações p/ setar sistema de coord´s, tamanho do ponto, cor de fundo, cor de desenho
  - myDisplay: função de desenho
  - glFlush: em sistemas que operam em rede, dados são bufferizados na máquina hospedeira e enviados para o display remoto qdo o buffer enche ou um glFlush é executado.
  - Exercícios
    - executar programa no seu sistema, alterar propriedades (número de pontos, tamanho do ponto, cor de fundo, cor de desenho)

23

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
- Fig. 2.17 (Angel)

24

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();
}
```

25

Poli-linhas e Polígonos

- Poli-linha: GL_LINE_STRIP, GL_LINE_LOOP
  - seqüência de linhas conectadas... fechada ou não
  - fig. 2.18
  - Ex. desenhar seqüência de poli-linhas armazenada em arquivo
    - figs. 2.21 e 2.22 (Angel)
  - Ex. casa ´hardwired´ vs. parametrizada
    - figs. 2.23, 2.24 e 2.25 (Angel)

26

Poli-linhas e Polígonos

- Rotina p/ desenhar poli-linha

```
Class GLintPointArray{
    const int MAX_NUM = 100;
    public:
        int num;
        GLintPoint pt[MAX_NUM];
          };
void drawPolyLine(GlintPointArray poly, int closed)
{       glBegin(closed ? GL_LINE_LOOP : GL_LINE_STRIP);
            for(int i = 0; i < poly.num; i++)
                glVertex2i(poly.pt[i].x, poly.pt[i].y);
        glEnd();
        glFlush();
}
```

27

Outras primitivas

- GL_TRIANGLES
- GL_QUADS
- GL_TRIANGLE_STRIP
- GL_QUAD_STRIP

28

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

29

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

30

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)
}
```

31

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();
}
```

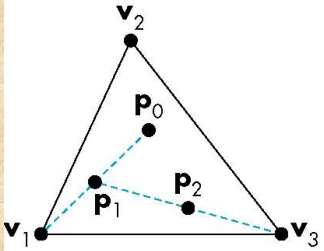
32

Interação com Teclado

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

33

Sierpinski Gasket



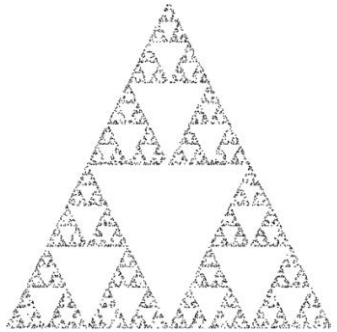
Fonte: E. Angel, Interactive Computer Graphics

v. Código gasket.c

1. Escolhe ponto qqr dentro do triângulo
2. Seleciona qqr um dos vértices
3. Determina ponto intermediário entre os dois anteriores
4. Exibe esse ponto
5. Substitui o ponto inicial por esse
6. Retorna ao passo 2

34

Sierpinski Gasket



Fonte: E. Angel, Interactive Computer Graphics

v. Código gasket.c

35

Bibliografia

- Computer Graphics Using OPEN GL, F.S. Hill, Prentice-Hall 2001
- E. Angel, Interactive Computer Graphics, 3a. Edição, Adison Wesley, 2003

36