Aula 7

Professores:

Anselmo Montenegro
Esteban Clua

Conteúdo:

- OpenGL

OpenGL e APIs Gráficas?

- Conceitos Básicos

- OpenGL x DirectX

- GLUT

- Renderização baseada em estados acumulativos

OpenGL - Conceitos Básicos

- Rendering Context

- Loop de renderização

- Back Buffer x Front Buffer

- Função de Renderização

OpenGL - Limitações

- 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.

OpenGL - Conceitos Básicos

1. Inicialização

Criação da Janela e do rendering context, estados iniciais, inicialização de outros componentes

2. Laço Principal

Entrada de dados, Física, IA, Renderização

3. Finalização

Liberação de recursos

OpenGL - Algumas regras de sintaxe

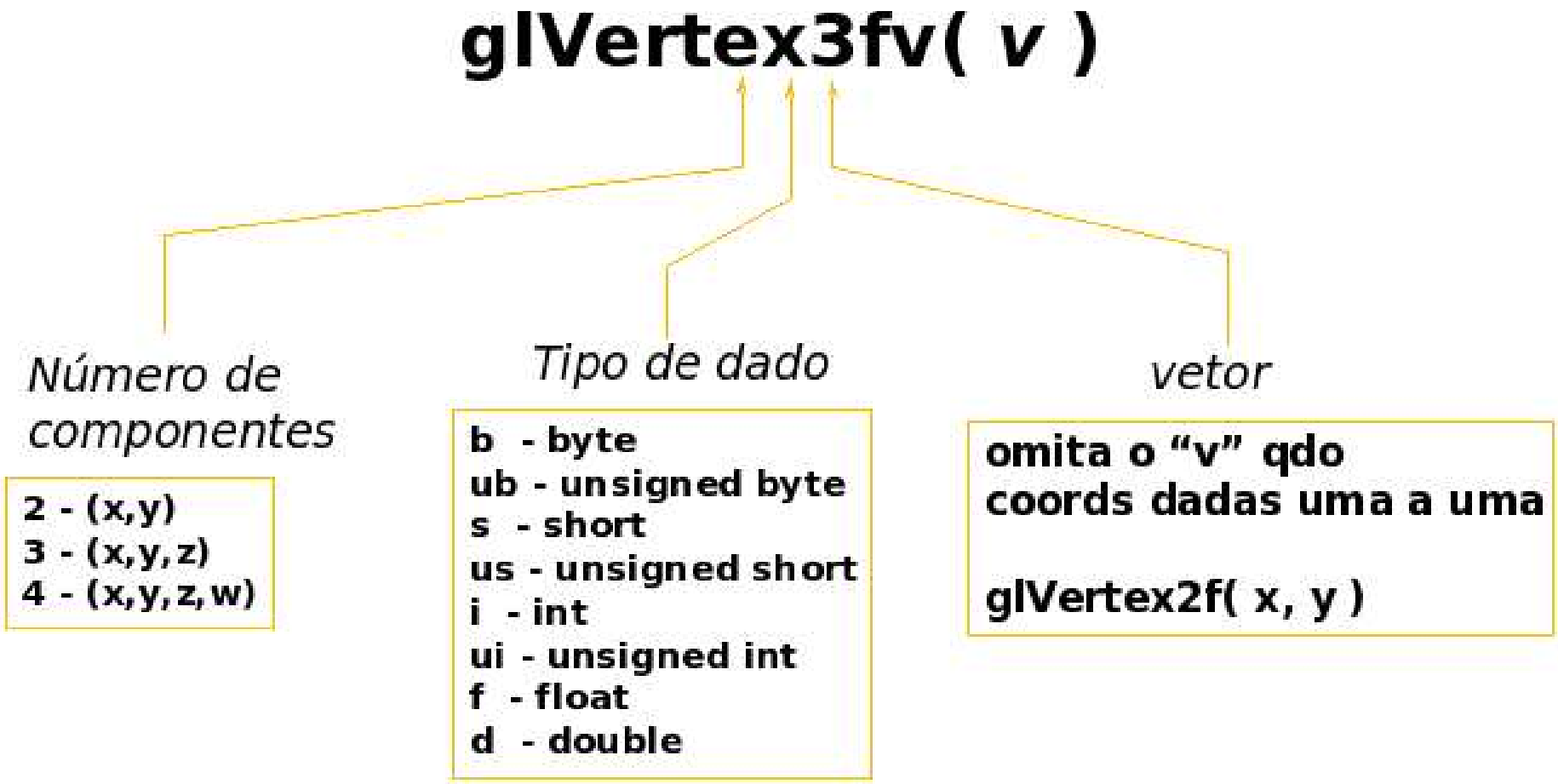
Todos os comandos começam com o sufixo gl (Ex.: glClearColor).

As palavras nos nome dos comandos começam com letras maiúsculas (Ex.: glColor()).

O sufixo indica o número e o tipo dos argumentos (Ex.: glVertex2i(1,3)).

As constantes começam com GL_ (Ex.:GL_COLOR_BUFFER_BIT).

Iniciando uma aplicação



Sufixos e Tipos dos Argumentos

Sufixo	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

OpenGL como máquina de Estados

A OpenGL funciona como uma máquina de estados.

Os estados correntes permanecem ativos até que sejam modificados.

Exemplo: a cor de desenho corrente é aplicada a qualquer primitiva geométrica até que seja modificada.

OpenGL como máquina de Estados

Existem vários estados:

- Cor de desenho corrente.
- Transformações de visualização e projeção.
- Padrões de linhas e polígonos.
- Modo de desenho dos polígonos.
- Posição e característica das fontes de luz.
- Propriedades dos materiais associados aos objetos.
- etc.

OpenGL como máquina 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().

APIs relacionadas

- GLU (OpenGL Utility Library)
  - Parte do padrão OpenGL.
  - NURBS, trianguladores, quádricas, etc.
- 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 official do OpenGL.

GLUT

- Biblioteca para criação de interfaces gráficas simples para programas gráficos baseados em OpenGL.
- Fornece um conjunto de primitivas para desenho de objetos mais complexos como quádricas e etc.

Headers OpenGL / GLUT

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

- Já inclui automaticamente os headers do OpenGL:

```
#include <GL/gl.h>
#include <GL/glu.h>
```

Se GLUT não for usado, os headers OpenGL têm que ser incluídos explicitamente, junto com os de outra camada de interface.

Há APIs para construção de interfaces gráficas (GUI) construídas sobre o GLUT cujos headers incluem os do GLUT.

- Por exemplo, o pacote GLUI requer:

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

(Já inclui `glut.h`)

Callbacks

Callbacks são rotinas que serão chamadas para tratar eventos.

Para uma rotina callback ser efetivamente chamada ela precisa ser registrada através da função.

```
glutXxxFunc (callback)
```

Onde Xxx designa uma classe de eventos e *callback* é o nome da rotina.

Por exemplo, para registrar uma callback de desenho chamada Desenho, usa-se

```
glutDisplayFunc (Desenho);
```

Callbacks de Desenho

É a rotina chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada (ex.: janela estava obscurecida por outra que foi fechada)

Todo programa 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! */
}
```

Callbacks de Redimensionamento

```
glutReshapeFunc (Reshape);
```

Chamada sempre que a janela é redimensionada, isto é, teve seu tamanho alterado.

Tem a forma

```
void reshape (int width, int height){...}
```

width/height são a nova largura/altura da janela (em pixels)

Obs: Se uma rotina de redimensionamento não for especificada, o GLUT usa uma rotina de redimensionamento "default" que simplesmente ajusta o *viewport* para usar toda a área da janela.

Callbacks

Outras callbacks comumente usadas

- Eventos de teclado

```
void keyboard(unsigned char key, int x, int y)
```

- Eventos de mouse

```
void mouse(int button,int state,int x,int y)
void motion(int x, int y)
void passiveMotion(int x, int y)
```

- Chamada continuamente quando nenhum outro evento ocorre

```
void idle(void)
```

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 e removidas.

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.

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 costme configurar variáveis de estado do OpenGL que não mudarão no decorrer do programa. Por exemplo:

- Cor do fundo
- Tipo de sombreamento de desejado

Exemplo uma aplicação

```
void init (void)
{
  /* selecionar cor de fundo (preto) */
  glClearColor (0.0, 0.0, 0.0, 0.0);

  /* inicializar sistema de viz. */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE
      | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Exemplo uma aplicação

```
void display(void)
{
  /* Limpar todos os pixels  */
  glClear (GL_COLOR_BUFFER_BIT);

  /* Desenhar um polígono branco (retângulo) */
  glColor3f (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();

  /* Não esperar! */
  glFlush ();
}
```

Rendering

```
void RenderScene()
{
        // Configura câmera.
        glLoadIdentity();
        gluLookAt(0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
        SetupCamera();

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glEnable(GL_ATRIBUTO_XXX); glDisable(GL_ATRIBUTO_YYY);

        glBegin(GL_TRIANGLES);
                glColor3f(1.0, 0.0, 0.0);
                glVertex3f(2.0, 0.0, 0.0);
                glColor3f(0.0, 1.0, 0.0);
                glVertex3f(0.0, 2.0, 0.0);
                glColor3f(0.0, 0.0, 1.0);
                glVertex3f(0.0, 0.0, 2.0);
        glEnd();

        ... // Muda estados, plota mais polígonos...

        // Depois de renderizar tudo 'realiza o swap de buffers.
        SwapBuffers();
}
```

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.:

```
glMaterial
glNormal
glTexCoord
```

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.

Conclusão: Antes de emitir um vértice, assegurar-se que cor, material, normal, etc têm o valor certo.

Primitivas de Desenho

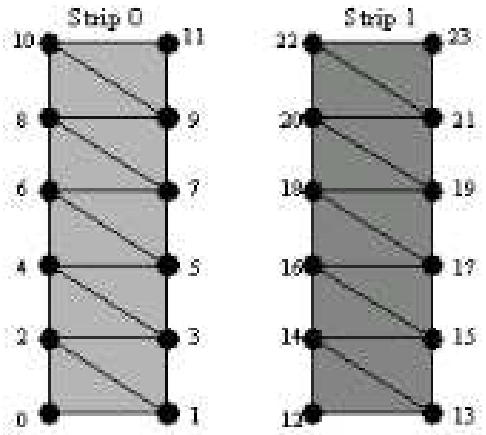
Valor	**Significado**
GL_POINTS	Pontos individuais
GL_LINES	Pares de vértices interpretados como segmentos de reta individuais.
GL_LINE_STRIP	Serie de segmentos de reta conectados.
GL_LINE_LOOP	Igual ao anterior. Ultimo vertice conectado a primeiro
GL_TRIANGLES	Triplas de vértices interpretados como 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.

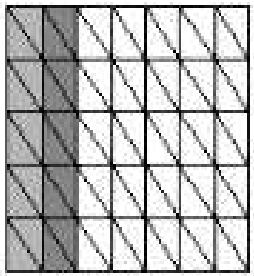
Triangle Strips

Idéia fundamental: minimizar volume de vértices e consequentemente, minimizar cálculos de iluminação, normais, clipping, etc.

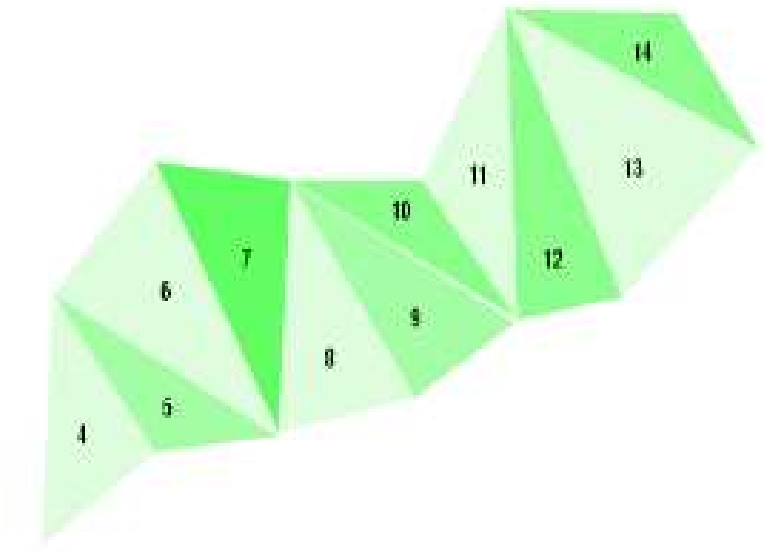
Triangle Strips

Strips: É possível descrever um triângulo com menos de 3 vértices?





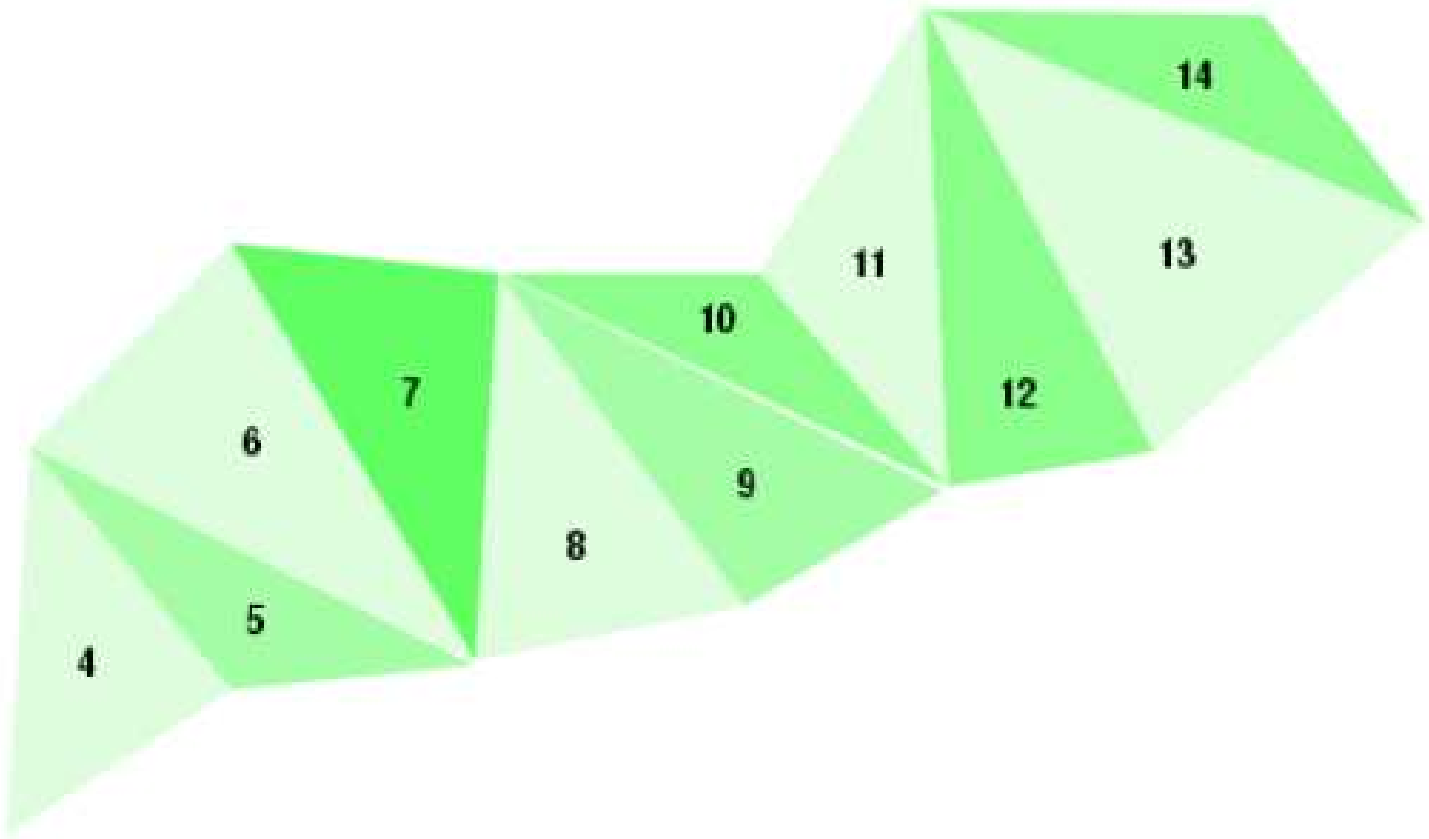
Problema



Para n triângulos, n+2 vértices

Cada Triangulo: V_i, V_{i+1}, V_{i+2}

Triangle Strips



Triangle Strips

```
...

for (int x = 0; x < 3; x++)
        {
        glBegin(GL_TRIANGLE_STRIP);
        for (int z = 0; z < 3; z++)
        {
                glVertex3f(x, 0.0, z);
                glVertex3f((x+1.0), 0.0, z);
                glVertex3f(x, 0.0, (z+1.0));
                glVertex3f((x+1.0), 0.0, (z+1.0));
        }
```

Aula 7

Professores:

Anselmo Montenegro
Esteban Clua

Conteúdo:

- OpenGL