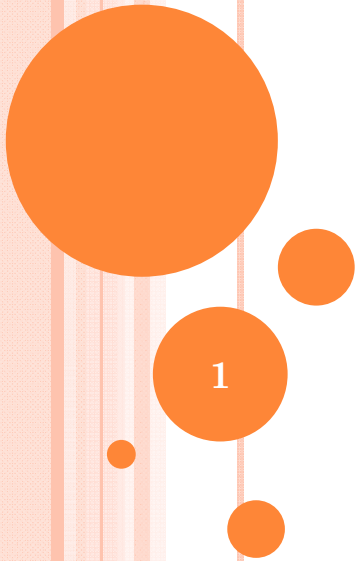


Introdução à OpenGL
Parte 2

Pedro Henrique Bugatti

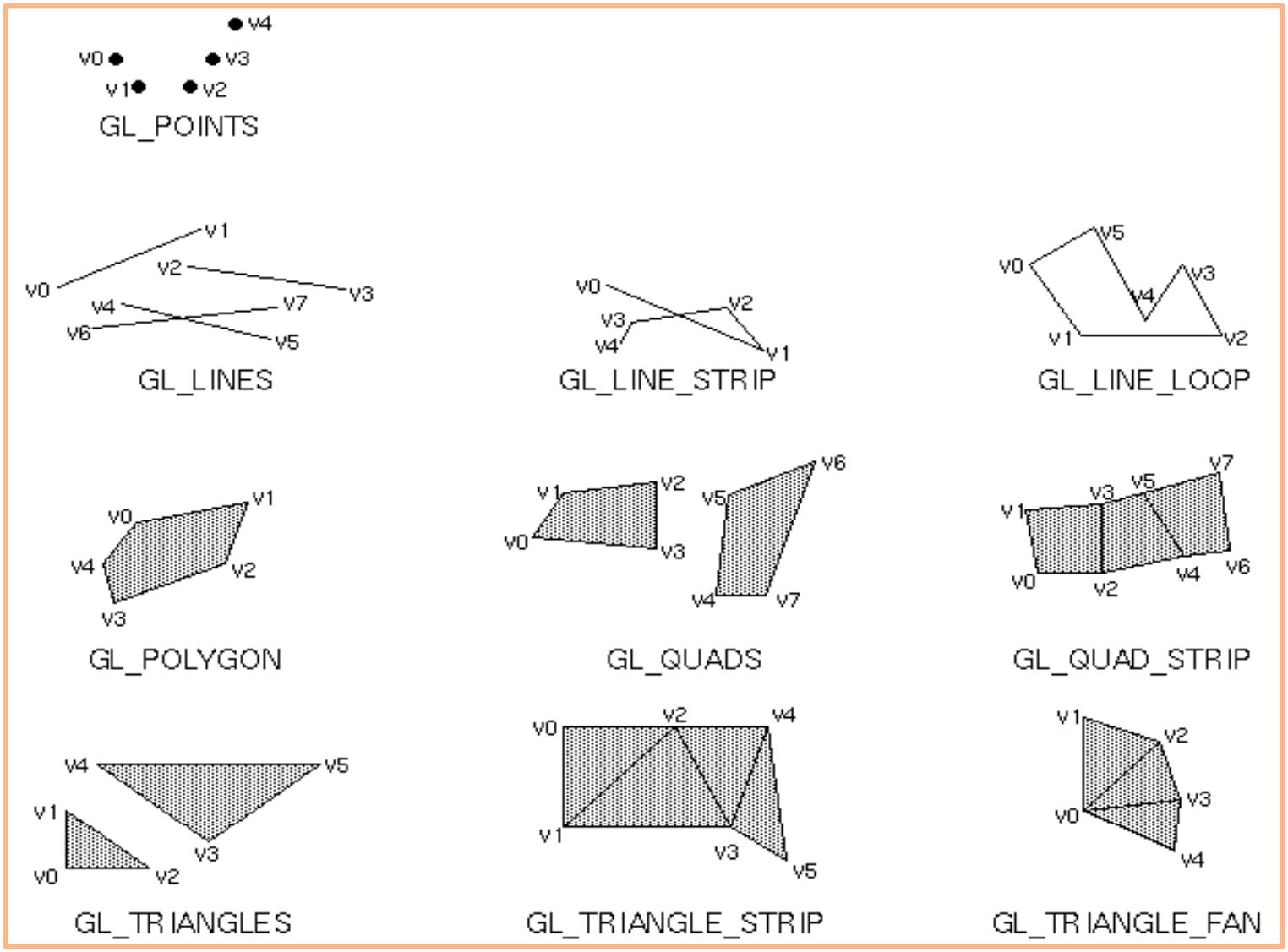
Roteiro

- Revisão Sucinta
- Transformações Geométricas
- Tratamento de Eventos – Funções *Callback*
- Animação

Revisão – 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.

Revisão – Primitivas de Desenho



Revisão – Sufixo e Tipo 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

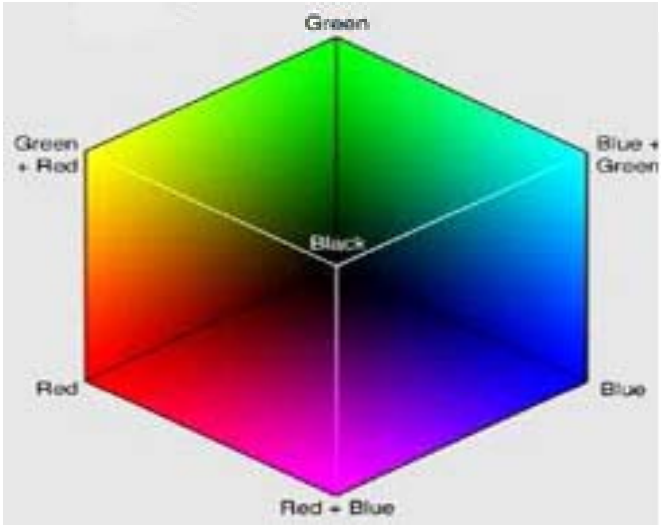
Revisão - Atributos

- Atributos são parte da OpenGL e determinam a aparência dos objetos:
  - **Cor**;
  - **Tamanho e espessura**;
  - **Modo de desenho de polígonos**:
    - Preenchido: cor sólida;
    - Não-Preenchido: somente mostra as arestas.

Revisão - Cor RGB

- Cada componente da cor é armazenado separadamente (usualmente usando 8 bits por componente)

- Os valores de cores variam de 0.0 (ausência do tom) a 1.0 (máxima presença)



Revisão – Exemplo de Rotina OpenGL

```
void desenha(){
   glClearColor(1,1,1,0); //determina cor de fundo
   glClear(GL_COLOR_BUFFER_BIT); // pinta o fundo
   glColor3f(1,0,0); // define cor do desenho
   glBegin(GL_TRIANGLES);
       glVertex2f(-0.5,-0.5);
       glVertex2f(0,0.5);
       glVertex2f(0,-0.5);
   glEnd();
   glFlush(); //força atualizar a tela
}
```

TRANSFORMAÇÕES GEOMÉTRICAS

- As transformações geométricas são usadas para manipular um modelo, isto é, através delas é possível mover, rotacionar ou alterar a escala de um objeto.

- A aparência final da cena ou do objeto depende muito da ordem na qual estas transformações são aplicadas.

- OpenGL é capaz de executar transformações de translação, escala e rotação por meio da multiplicação de matrizes

- Essas transformações em OpenGL é que elas podem ser combinadas em uma única matriz, de tal maneira que várias transformações geométricas possam ser aplicadas através de uma única operação

TRANSFORMAÇÕES GEOMÉTRICAS

- Isto ocorre porque uma transformação geométrica em OpenGL é armazenada internamente em uma matriz
- A cada transformação que é aplicada, esta matriz é alterada e usada para desenhar os objetos a partir daquele momento
- A cada nova alteração é feita uma composição de matrizes.
- Para evitar este efeito "cumulativo", é necessário utilizar as funções *glPushMatrix()* e *glPopMatrix()*, que salvam e restauram, respectivamente, a matriz atual em uma pilha interna da OpenGL.

TRANSFORMAÇÕES GEOMÉTRICAS

- glTranslatef(GLFloat tx,ty,tz)
  - Parâmetros: valores de translação aplicados aos eixos x,y,z

- glScaled(GLFloat tx,ty,tz)
  - Parâmetros: valores de escala aplicados aos eixos x,y,z

- glRotatef(GLfloat ângulo_em_graus,x,y,z)
  - Ângulo e eixo ao redor do qual será aplicada a rotação.

Exemplo:

Rotação de 60º. Sobre o eixo z:

glRotatef(60.0f, 0.0f, 0.0f, 1.0f);

Exemplo programa transforma

Exercício – Fazer em Aula

- Utilizando as primitivas geométricas apresentadas (linhas, triângulos, polígonos), crie 3 diferentes primitivas e aplique transformações geométricas (translação, rotação, escala).

- Sugestão: Utilize o exemplo *transforma.cpp* apresentado em aula

Tratamento de Eventos
Funções *Callback*

- As funções de *callback* são aquelas executadas quando qualquer evento ocorre no sistema, tais como :
  - redimensionamento de janela,
  - entradas de usuários através de teclado, mouse, ou outro dispositivo de entrada,
  - ocorrência de animações.

- Assim o desenvolvedor pode associar uma ação especifica à ocorrência de determinado evento.

Tratamento de Eventos
Funções *Callback*

GLUT oferece suporte a muitos diferentes tipos de ações de *callback* incluindo :

- **glutDisplayFunc()** – chamada quando um pixel na janela necessita ser atualizado.
- **glutReshapeFunc()** – chamado quando a janela é redimensionada.
- **glutKeyboardFunc()** – chamada quando uma tecla do teclado é pressionada.
- **glutMouseFunc()** – chamada quando o usuário pressiona um botão do mouse.
- **glutMotionFunc()** - chamada quando o usuário movimenta o mouse enquanto mantém um botão do mesmo pressionado.
- **glutPassiveMouseFunc()** – chamado quando o mouse é movimentado, independente do estado dos botões.
- **glutIdleFunc()** – uma função de callback chamada quando nada está acontecendo. Muito útil para animações.

TRATAMENTO DE EVENTOS
FUNÇÕES *CALLBACK*

- ***glutKeyboardFunc***

- Estabelece a função *callback* que é chamada pela GLUT cada vez que uma tecla que gera código ASCII é pressionada (por exemplo: a, b, A, b, 1, 2).

- Além do valor ASCII da tecla, a posição (x,y) do *mouse* quando a tecla foi pressionada também é retornada.

- Parâmetros de entrada da função *callback*:
  - ***(unsigned char key, int x, int y)***

Tratamento de Eventos
Funções *Callback*

- ***glutSpecialFunc***

- Estabelece a função *callback* que é chamada pela GLUT cada vez que uma tecla que gera código não-ASCII é pressionada, tais como Home, End, PgUp, PgDn, F1 e F2.

- Além da constante que identifica a tecla, a posição corrente (x,y) do *mouse* quando a tecla foi pressionada também é retornada.

- Parâmetros de entrada da função *callback*:
  - ***(unsigned char key, int x, int y)***.

- Os valores válidos para o primeiro parâmetro são:
  - GLUT_KEY_F1, GLUT_KEY_F2, ..., GLUT_KEY_F12
  - GLUT_KEY_LEFT, GLUT_KEY_UP, GLUT_KEY_RIGHT, GLUT_KEY_DOWN
  - GLUT_KEY_PAGE_UP, GLUT_KEY_PAGE_DOWN, GLUT_KEY_HOME, GLUT_KEY_END, GLUT_KEY_INSERT.

Tratamento de Eventos
Funções *Callback*

- ***glutMouseFunc***
- Estabelece a função *callback* que é chamada pela GLUT cada vez que ocorre um evento de *mouse*.
- Parâmetros de entrada da função *callback*:
  - ***(int button, int state, int x, int y)***.
- Três valores são válidos para o parâmetro *button*:
  - GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON e GLUT_RIGHT_BUTTON.
- O parâmetro *state* pode ser GLUT_UP ou GLUT_DOWN.
- Os parâmetros x e y indicam a localização do mouse no momento que o evento ocorreu.

Tratamento de Eventos
Funções *Callback*

Criar uma função para tratar o evento

```
static void myMouseFunc(int buttom, int state, int xm, int ym){
....
if(buttom==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
   int posx = x; int posy=y;
...
}
```

Buttom ➔GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, ou GLUT_RIGHT_BUTTON.

State ➔GLUT_UP ou GLUT_DOWN

Registrar a função para tratar o evento:

```
glutMouseFunc(myMouseFunc);
```

Tratamento de Eventos
Funções *Callback*

- Explore o programa *prog-callback*

  - Insira novas opções de "teclas" (detecção de eventos) para realização de diferentes transformações geométricas

  - Insira novos eventos para modificação de cores dos objetos

  - Utilize a função *glutMouseFunc()* apresentada para detectar eventos de clique de mouse, ao clicar o botão esquerdo do mouse transladar a imagem, ao clicar o botão direito rotacionar a imagem.

  - .....

ANIMAÇÃO

- É possível criar um laço que continuamente altera as coordenadas do objeto antes de chamar a função "Desenha".
  - Passando a impressão de movimento

- No entanto, a biblioteca GLUT fornece a possibilidade de registrar uma função *callback* que torna mais fácil o processo de fazer uma simples animação.

- A função *glutTimerFunc* pega o nome da função *callback* que deve ser chamada e o tempo que ela deve esperar antes de chamar a função.

Como Animar?

Opção 1: Criar uma função e a registrar para ser chamada sempre que o programa estiver ocioso

```
void idle(void)
{....
glutPostRedisplay();
}
```

Registrando a função idle:

```
glutIdleFunc(idle);
```

Como Animar?

Opção 2: Criar uma função e a registrar para ser chamada depois de um determinado intervalo.

```
glutTimerFunc(unsigned int msecs, void (*func)(int
   value), int value)

void animation(int value)
{
 ...
 glutPostRedisplay();
 glutTimerFunc(60,animation,1);
}
```

COMO ANIMAR?

- ***glutTimerFunc(33, Timer, 1);*** estabelece a função *Timer* previamente definida como a função *callback* de animação.

- Seu protótipo é: *void glutTimerFunc(unsigned int msecs, void (*func)(int value), int value);*.

- Esta função faz a GLUT esperar *msecs* milisegundos antes de chamar a função *func*. É possível passar um valor definido pelo usuário no parâmetro *value*. Como esta função é "disparada" apenas uma vez, para se ter uma animação contínua é necessário reinicializar o *timer* novamente na função *Timer*.

- ***void Timer(int value)*** é a função chamada pela *glutTimerFunc*

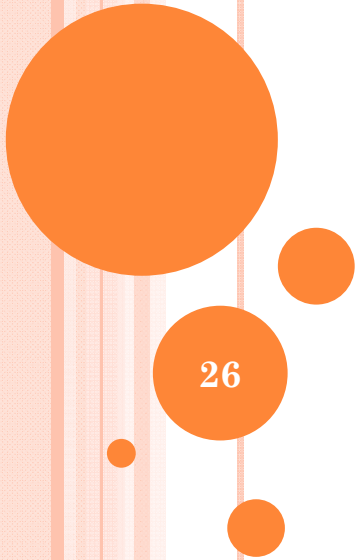
Exercício – Fazer em Aula

- Faça um quadrado inicialmente no centro da janela e o anime.
  - Para tanto, utilize as transformações geométricas apresentadas, em conjunto com os conceitos apresentados para realização de animação
  - Sugestão: utilize como base o exemplo ***anima.c***

Referências

- COHEN, Marcelo; MANSSOUR, Isabel Harb. **OpenGL: Uma abordagem prática e objetiva. São** Paulo: Novatec, 2006.

- ANGEL, E. **Interactive Computer Graphics: An interactive approach**. 4. ed. Addison-Wesley, 2006.



Introdução à OpenGL
Parte 2

Pedro Henrique Bugatti