COMPUTAÇÃO GRÁFICA

Primitivas Gráficas

CURSO DE CIÊNCIA DA COMPUTAÇÃO
UNIPAC BARBACENA
PROFESSOR NAIRON NERI SILVA

OpenGL

- O **OpenGL** (**Open G**raphics **L**ibrary) é uma API livre utilizada na computação gráfica, para desenvolvimento de aplicativos gráficos, ambientes 3D, jogos, entre outros.

- O OpenGL é um conjunto de algumas centenas de funções, que fornecem acesso a praticamente todos os recursos do *hardware* de vídeo.

OpenGL

- O OpenGL funciona como uma máquina de estados.

- Usando as funções da API, você pode ligar ou desligar vários aspectos dessa máquina, tais como a cor atual, incidência ou não de transparência, se cálculos de iluminação devem ser feitos e assim por diante.

- É importante conhecer cada um desses estados, pois não é incomum a obtenção de resultados indesejados simplesmente por deixar um ou outro estado definido de maneira incorreta.

OpenGL

- As funções do OpenGL são identificadas pelo prefixo “gl” e a primeira letra de cada palavra após esse prefixo é maiúscula.

- Como exemplo, podemos citar a função `glClear`, responsável por “limpar” a área de visualização da janela de renderização. Internamente utilizamos, por exemplo, a função `glDrawPixels`, responsável por escrever um bloco de *pixels* (a imagem) no *framebuffer*.

OpenGL – Primitivas

- Alguns trabalhos práticos a serem desenvolvidos exigirão conhecimento sobre criação de desenhos usando as primitivas do OpenGL.

- Em OpenGL, utilizamos primitivas geométricas. As primitivas são: pontos, linhas e polígonos, que se descrevem a partir de seus vértices.

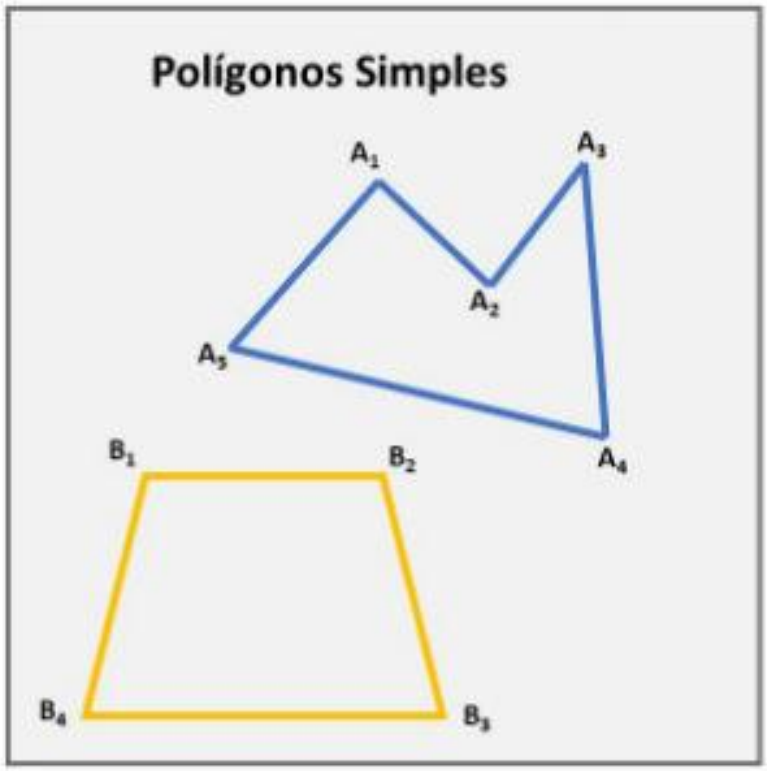
OpenGL – Primitivas

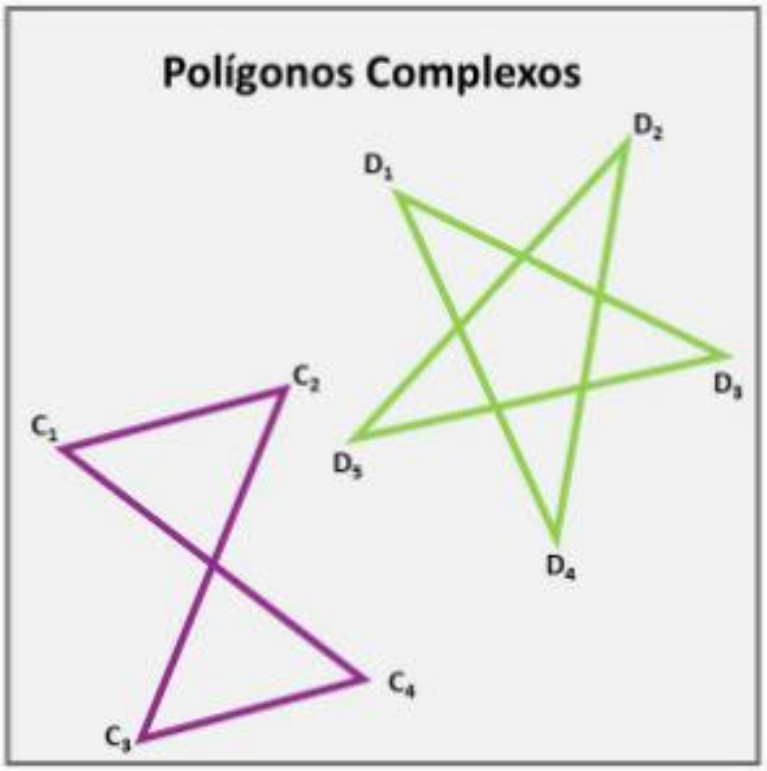
- **Pontos**: são conjuntos de números reais chamados vértices. Pode-se trabalhar com vértices com 2 ou 3 coordenadas.

- **Linhas**: as linhas em OpenGL são na realidade segmentos de reta. Os vértices definem os pontos extremos de cada segmento.

- **Polígonos**: área envolta por um conjunto finito e fechado de segmentos de retas. Em OpenGL, os polígonos devem ser simples (suas bordas não se sobrepõem) e convexos (dado quaisquer dois pontos dentro do polígono, o segmento que os une também esta em seu interior).

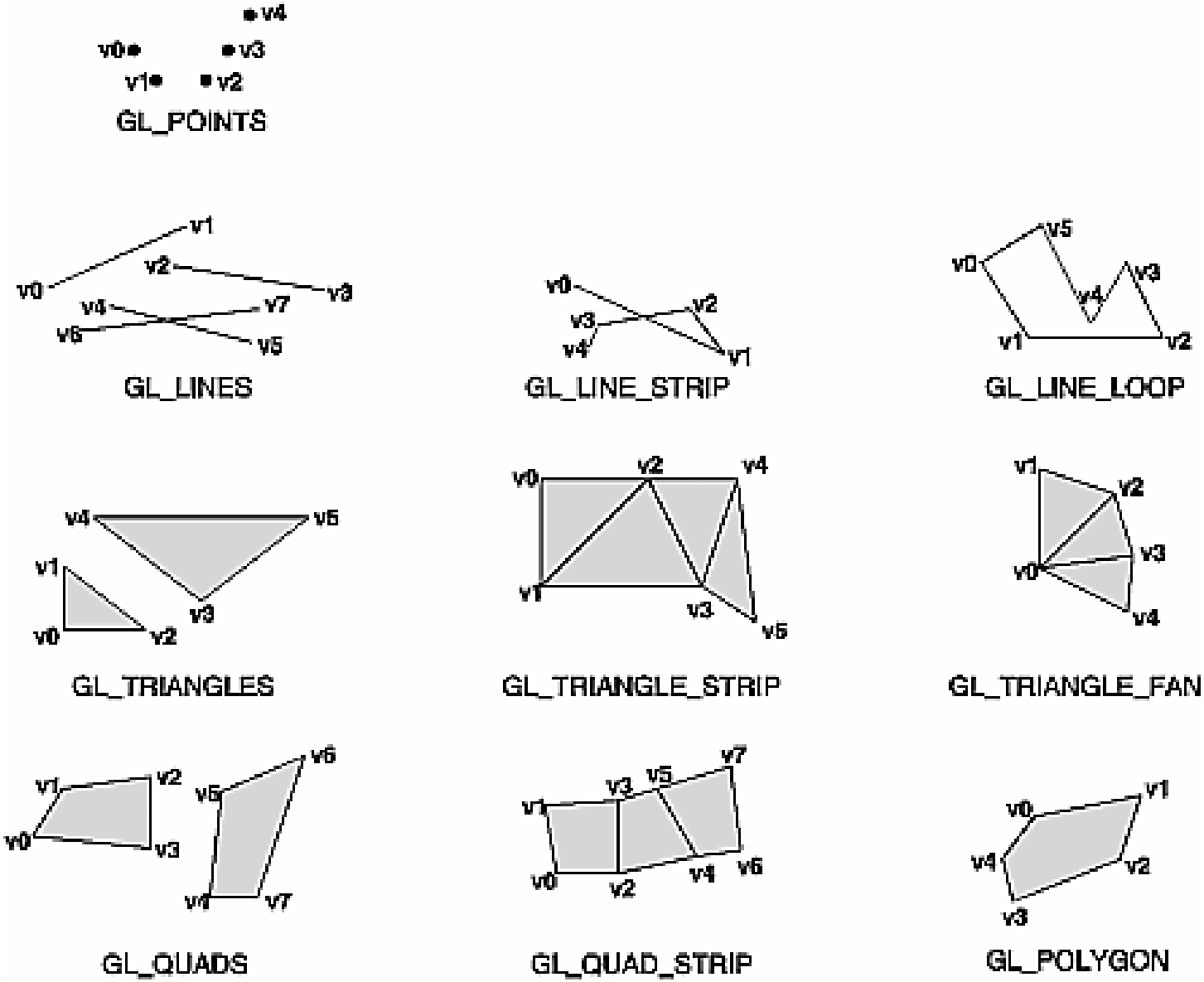
OpenGL – Primitivas





O OpenGL utiliza polígonos simples

OpenGL – Primitivas



OpenGL – Primitivas

- A criação de figuras no OpenGL deve ser feita somente dentro do par de instruções `glBegin()` e `glEnd()` como no exemplo abaixo:

```
glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(3.0, 2.0);
glEnd();
```

- No exemplo acima, foi definida uma cor para cada vértice da linha. A cor da linha será obtida através da interpolação entre essas duas cores.

OpenGL – Window-Viewport

- Existem algumas funções que governam a relação entre o domínio da cena (onde os objetos são definidos) e o domínio da imagem (espaço no qual a janela é renderizada). As principais funções são as seguintes:
  - `glOrtho` ou `gluOrtho2D`
  - `glViewport`
  - `glutReshapeWindow`
  - `glutInitWindowPosition`

OpenGL – Window-Viewport

- `gluOrtho2D(left, right, bottom, top)`
  - Define uma região de visualização ortogonal 2D ou tela de domínio de cena. É definida por dois planos verticais de recorte *left* e *right* e dois planos horizontais de recorte *bottom* e *top*.
  - A configuração *default desta função é* (-1,1,-1,1).
  - Define uma matriz de projeção ortogonal 2D.

OpenGL – Window-Viewport

- `glViewport(x, y, width, height)`
  - Define uma área de visualização na janela da aplicação, onde *x*, *y* especificam o canto inferior esquerdo e *width*, *height* as suas dimensões.
  - Se o comando *glViewport* não for definido, a área de visualização *default* ocupa a área gráfica total da janela de visualização (esta área é definida através de funções do *glut*).
  - Podem existir várias áreas de visualização dentro da janela de aplicação.

GLUT

- GLUT é um *toolkit* para gerenciamento de janelas e eventos para o OpenGL. A *API* é portável, fazendo com que os códigos escritos possam ser compilados em diversas plataformas como Windows, Linux, Mac etc.

- É uma API simples, fácil de usar e leve.

- Implementaremos nossos códigos utilizando o GLUT em C/C++.

Estado Inicial do GLUT

- Nos arquivos de teste a serem disponibilizados, várias funções de inicialização do GLUT são chamadas. Algumas desses funções são as seguintes:

  `glutInit(&argc, argv);`

  É usada para inicializar a biblioteca GLUT. Nos exemplos a serem utilizados, basta chamá-la com os argumentos *default.*

  `glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);`

  Estabelece os parâmetros iniciais do display. A configuração acima serve tanto para exibir imagens estáticas quanto animações (*Double Buffer*) e de uma forma geral não será modificada.

Manipulação de janelas

- Algumas funções do GLUT para manipulação de janelas:

`glutInitWindowSize(int width, int height );`

Estabele o tamanho inicial da janela de renderização em pixels. Esse tamanho pode ser modificado pela função `glutReshapeWindow`.

`glutInitWindowPosition(int x, int y );`

Estabele a posição inicial da janela de renderização em pixels (considere a origem no canto superior esquerdo do monitor).

`glutCreateWindow("Nome da janela");`

Cria a janela principal onde será exibida a imagem.

Atualização das janelas

- Existem funções específicas para definir se uma janela precisa ou não ser redesenhada e para atualizar essa janela:

```
void glutPostRedisplay(void);
```

Quando chamado, define que a janela corrente precisa ser redesenhada.

```
void glutSwapBuffers(void);
```

Atualiza a janela corrente fazendo um *swap* entre o *buffer* que foi desenhado e o *buffer* corrente (é necessário usar *double buffer* para chamar essa função).

Registro de *Callbacks*

- É necessário que registremos no *glut* funções específicas para *display*, teclado, mouse etc. Abaixo temos alguns exemplos dessas funções com seus respectivos argumentos:

*void glutDisplayFunc(void (*func)(void));*

Define a callback responsável pelas funções de desenho.

*void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));*

Define a callback para gerenciar eventos de teclado.

*void glutMouseFunc(void (*func)(int button, int state, int x, int y));*

Define a callback para gerenciar eventos de mouse.

*void glutIdleFunc(void (*func)(void));*

Define a callback para gerenciar tarefas em background quando nenhum outro evento está sendo executado (geralmente utilizado em animações).

Loop principal

- Todos os eventos do *glut* são executados em *loop*. É através da função `glutMainLoop( )` que esse *loop* tem início.

Interface Gráfica

- Os exemplos com interface gráfica fazem uso da biblioteca GLUI (*OpenGL Utility Interface*).

- Material interessante sobre gerenciamento de janelas em OpenGL

  http://www.di.ubi.pt/~agomes/cg/teoricas/03-janelas.pdf

Fonte dos Slides

Slides do material do Prof. Rodrigo Luis de Souza da Silva - UFJF