LAB. 1

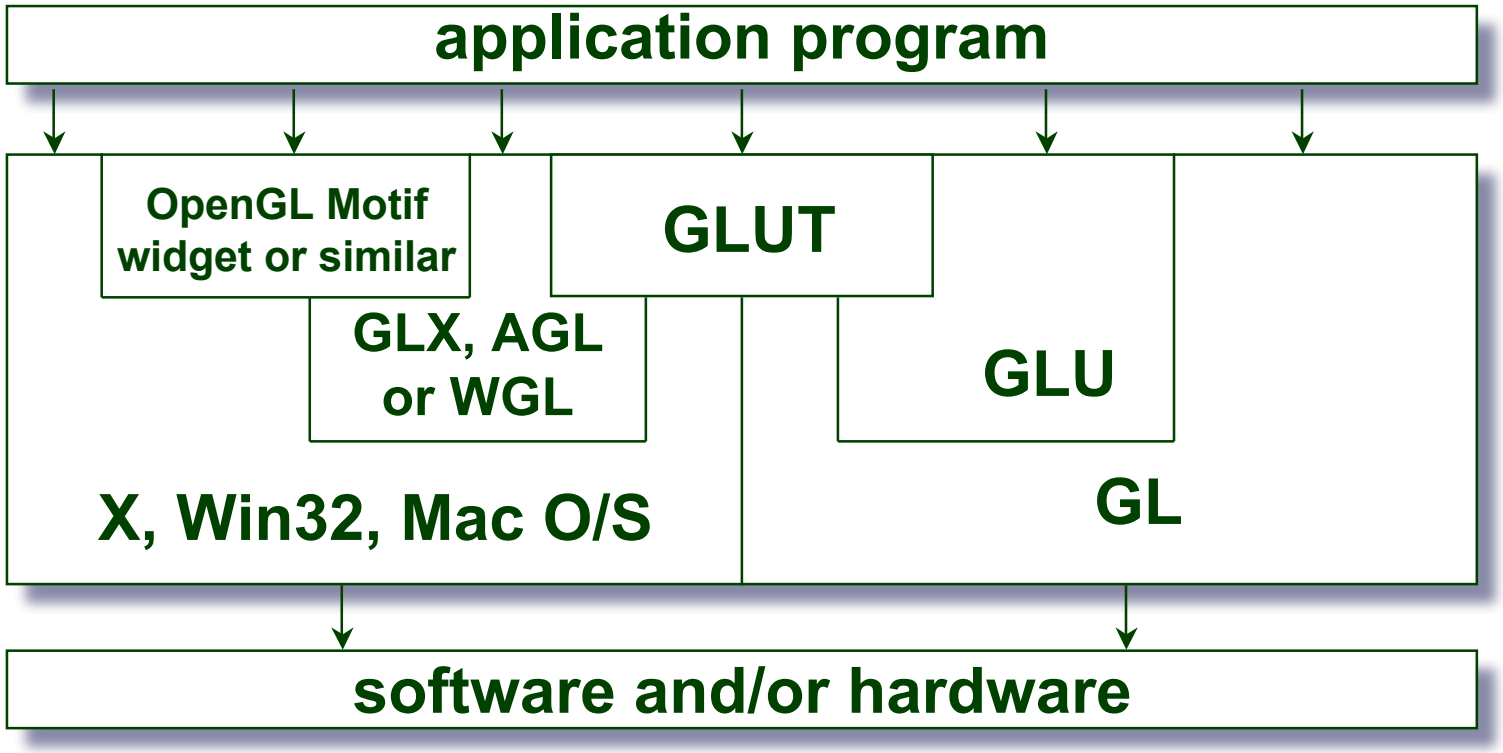
Introdução à OpenGL

OpenGL – O que é?

- É uma biblioteca de funções gráficas.

- É uma API (*Application Programming Interface*) gráfica
  - 2D e 3D
  - Primitivas vectoriais e rasterizadas (imagens)
  - Capaz de gerar imagens de alta qualidade
  - Normalmente implementada de forma a tirar partido da aceleração gráfica (se disponível)
  - Independente de plataforma (sistema operativo e hardware)
  - Independente de sistema de janelas

OpenGL – O que é?



Sistemas de Janelas

- Principal meio de interacção homem/máquina em ambientes de computação modernos.
- A tela do ecrã é vista como uma janela que contém outras janelas (eventualmente sobrepostas).
- Janelas são controladas por aplicações que têm a incumbência de mantê-las sempre actualizadas.
- Interacção com o utilizador e com o próprio sistema de janelas é comunicada à aplicação através de ***eventos***. Por exemplo:
  - Rato foi accionado
  - Janela foi redimensionada
- Eventos são tratados por rotinas ***callback*** da aplicação. Por exemplo:
  - Redesenhar o conteúdo da janela
  - Mover um objecto de um lado para outro da janela
- Cada sistema de janelas (SJ) possui uma API distinta:
  - MS Windows, X, Apple
  - Portabilidade: camada de interface com diversos SJ mas com API única (ex.: GLUT)

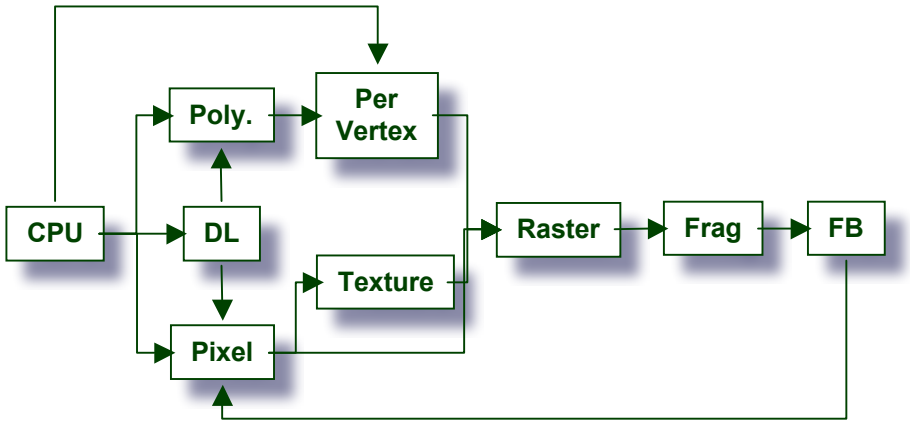
APIs Relacionadas

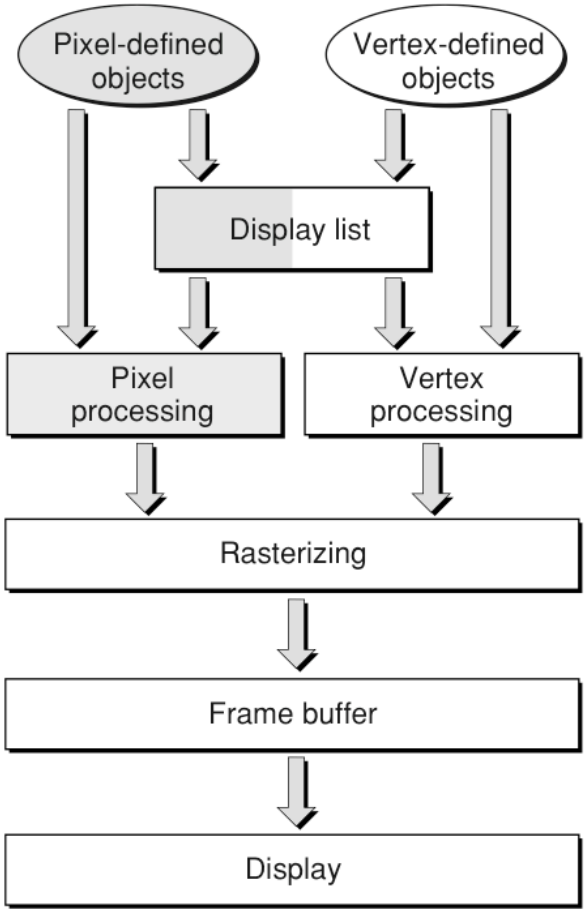
- **GLU** (OpenGL Utility Library)
  - Parte da norma OpenGL
  - NURBS, trianguladores, quádricas, etc.

- **AGL**, **GLX**, **WGL**
  - Camadas entre a OpenGL e os diversos sistemas de janelas

- **GLUT** (OpenGL Utility Toolkit)
  - API portátil de acesso aos sistemas de janelas
  - Encapsula e oculta as camadas proprietárias
  - Não é parte oficial da OpenGL

Arquitetura da OpenGL





Representação gráfica em OpenGL

- OpenGL funciona como uma máquina de estados
- OpenGL API tem rotinas para
  - desenhar primitivas geométricas e imagens
  - alterar variáveis de estado (ex.: cor, material, fontes de iluminação, etc)
  - consultar variáveis de estado
- OpenGL é uma norma em evolução
  - Mecanismo padronizado de extensões
  - Novas versões são estabelecidas por uma comissão (ARB) de utilizadores e fabricantes

Estrutura de um programa OpenGL/GLUT

```
#include <GLUT/glut.h>
/* Outros headers */

void display (void) {
    ...
}
/* Outras rotinas callback */

int main (int argc, char *argv[]) {
    glutInit (argc, argv);
    glutInitDisplayMode( modo );
    glutCreateWindow( nome_da_janela );
    glutDisplayFunc( displayCallback );
    glutReshapeFunc( reshapeCallback );
    /* Registro de outras rotinas callback */
    glutMainLoop();
    return 0;
}
```

Headers

Rotinas *callback*

Inicialização do GLUT

Inicialização da janela

Registo de *callbacks*

Lacete principal

Headers da OpenGL/GLUT

```
// headers files for Mac OS X
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>

// headers files for Windows XP
#include <GL/gl.h>
#include <GL/glu.h>
#include <GLUT/glut.h>
```

- Há APIs para construção de interfaces gráficas (GUI-*Graphical User Interfaces*) que funcionam sobre a GLUT. Isto significa que os seus headers incluem os da GLUT.

- Por exemplo, a utilização da GLUI num programa gráfico requer a inclusão de:

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

  a qual já inclui `glut.h`

GLUT – Registo de *callbacks*

- *Callbacks* são rotinas que são chamadas para tratar eventos.

- Para que uma rotina *callback* ser efectivamente invocada quando um dado evento ocorre, é preciso fazer previamente o seu registo através da função

```
glutXxxFunc (callback)
```

em que `Xxx` designa uma classe de eventos e *`callback`* é o nome da rotina

- Por exemplo, o registo da *callback* designada por `Desenho`, é feito do seguinte modo:

```
glutDisplayFunc (Desenho);
```

GLUT – *callback* de desenho

- É a rotina chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada (ex.: janela estava ocultada por outra que entretanto foi fechada)
- Todo programa GLUT precisa de 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(); /* double-buffering! */
}
```

GLUT – *callback* de redimensionamento

- Chamada sempre que a janela é redimensionada por alteração do seu tamanho
- Tem a seguinte sintaxe:

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

em que `width/height` especificam as novas largura/altura da janela (em pixels)

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

GLUT - *callbacks* de interacção humana com o computador

- Eventos do teclado:

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

- Eventos do rato:

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

- Evento de falta-de-evento:

```
void idle (void)
```

- Outros eventos…

Programa OpenGL/GLUT - Inicialização

- Inicialização da GLUT:

```
glutInit (int* argc, char** argv)
```

  - estabelece ligação com o sistema de janelas

Programa OpenGL/GLUT - Inicialização

- Inicialização da(s) janela(s):

```
glutInitDisplayMode (int modo)
```

  - em que `modo` é um OR bit-a-bit de constantes tais como:
    - `GLUT_RGB` **- modelo de cor**
    - `GLUT_DOUBLE` **- bufferização dupla**
    - `GLUT_DEPTH` **- buffer de profundidade (z-buffer)**
    - `GLUT_ACCUM` **- buffer de acumulação**
    - `GLUT_ALPHA` **- buffer de cores terá componente alfa**

```
glutInitWindowPosition (int x, int y)
```

  - estabelece a posição inicial do canto superior esquerdo da janela

```
glutInitWindowSize (int width, height)
```

  - estabelece o tamanho (em pixels) da janela

Programa OpenGL/GLUT - Inicialização

- Criação da(s) janela(s):

```
int glutCreateWindow (char* nome)
```

  - cria uma nova janela primária (*top-level*)
  - `nome` é a etiqueta que aparece na barra superior da janela
  - o número inteiro que é devolvido pela função é usado pela GLUT para identificar a janela

- Outras inicializações
  - Após a criação da janela, é costume configurar as variáveis de estado da OpenGL que não mudarão durante a execução do programa. Por exemplo:
    - cor do fundo
    - tipo de coloração

Programa OpenGL/GLUT – lacete principal de eventos

- Depois de registadas as callbacks, o controlo é entregue ao sistema de janelas:

```
glutMainLoop (void)
```

  - esta rotina na verdade é o “despachador” de eventos.

Exemplo

```
/* * quad.cc - Simple Quad * Adam Mills  */
#include <OpenGL/gl.h>          // Header File For The OpenGL Library
#include <OpenGL/glu.h>         // Header File For The GLu Library
#include <GLUT/glut.h>          // Header File For The GLut Library
#include <stdlib.h>
void quad(){
   glBegin(GL_QUADS);
        glVertex2f( 0.0f, 1.0f);  // Top Left
        glVertex2f( 1.0f, 1.0f);  // Top Right
        glVertex2f( 1.0f, 0.0f);  // Bottom Right
        glVertex2f( 0.0f, 0.0f);  // Bottom Left
   glEnd();
}
void draw(){
   glClearColor( 0, 0, 0, 0 );     // Make background colour black
   glClear ( GL_COLOR_BUFFER_BIT );
   glPushMatrix();                 // Push the matrix stack
   glColor3f( 0, 0, 1 );           // Set drawing colour to blue
   glTranslatef(-0.5, -0.5, 0.0); // Move the shape to middle of the window
   quad();                         // Call our Quad Method
   glPopMatrix();                        // Pop the Matrix
   glutSwapBuffers();              // display it
}
```

Exemplo (cont.)

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y){
    if(key==27) exit(0);
}

int main(int argc,  char ** argv)
{
    glutInit(&argc, argv);
    // Double Buffered RGB display
    glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
    // Set window size
    glutInitWindowSize( 500,500 );
    glutCreateWindow("Test");
    // Declare the display and keyboard functions
    glutDisplayFunc(draw);
    glutKeyboardFunc(keyboard);
    // Start the Main Loop
    glutMainLoop();
    return 0;
}
```

OpenGL – primitivas gráficas

```
glBegin ( PRIMITIVA );
```

especificação de vértices, cores, coordenadas de textura, propriedades de material

```
glEnd ();
```

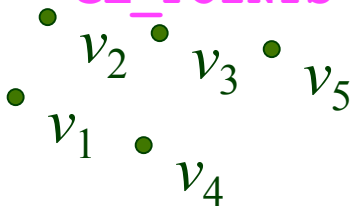
- Entre `glBegin()` e `glEnd()` apenas alguns comandos podem ser usados:
  - `glMaterial`
  - `glNormal`
  - `glTexCoord`

- Uma vez definido um vértice (`glVertex`), este é desenhado com as propriedades (cor, material, normal, coordenadas de textura, etc) registadas nas variáveis de estado.

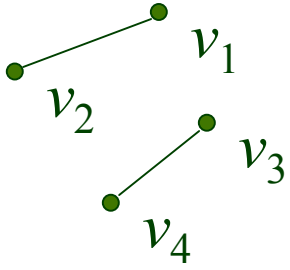
- Conclusão: Antes de definir um vértice, há que assegurar que a cor, o material, a normal, etc, têm o valor adequado.

OpenGL – primitivas gráficas

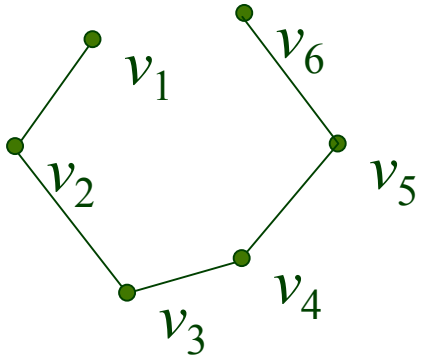
GL_POINTS



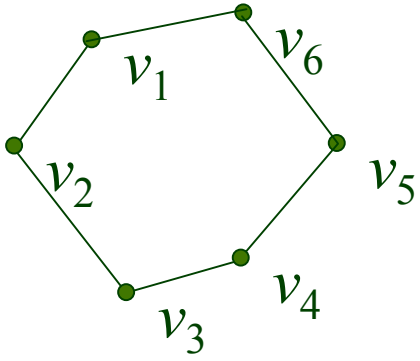
GL_LINES



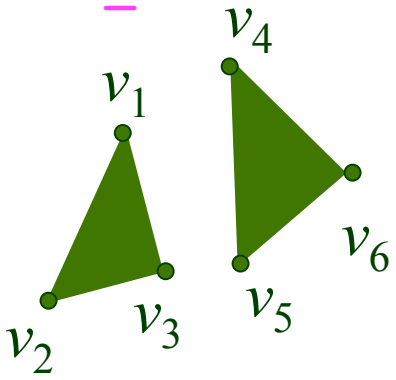
GL_LINE_STRIP



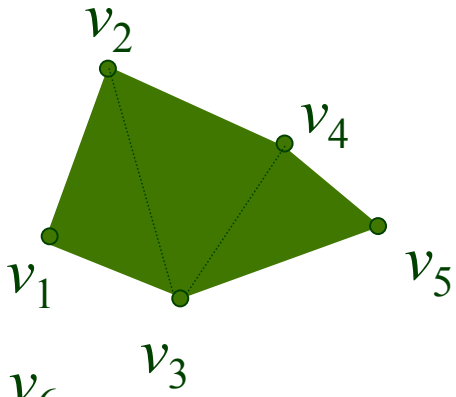
GL_LINE_LOOP



GL_TRIANGLES

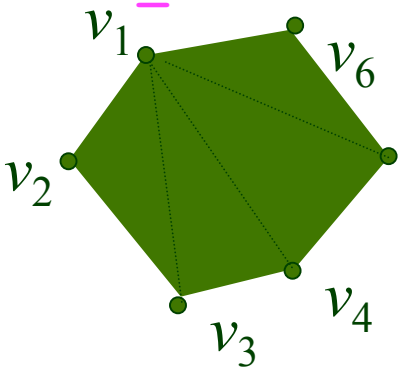


GL_TRIANGLE_STRIP

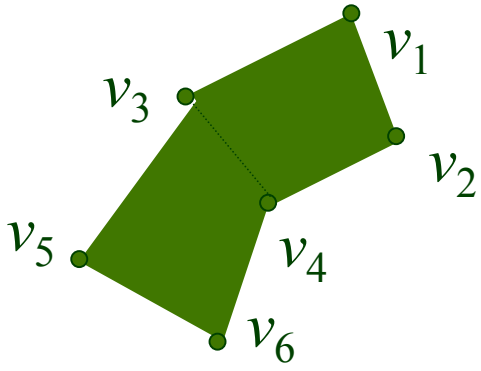


GL_POLYGON

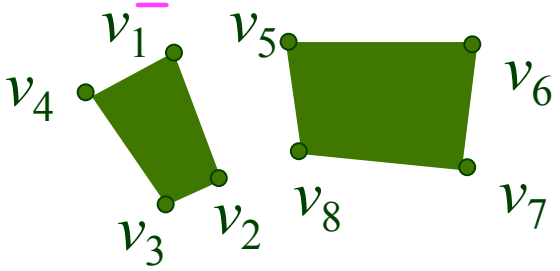
GL_TRIANGLE_FAN



GL_QUAD_STRIP



GL_QUADS



FIM