Introdução à Programação com a OpenGL

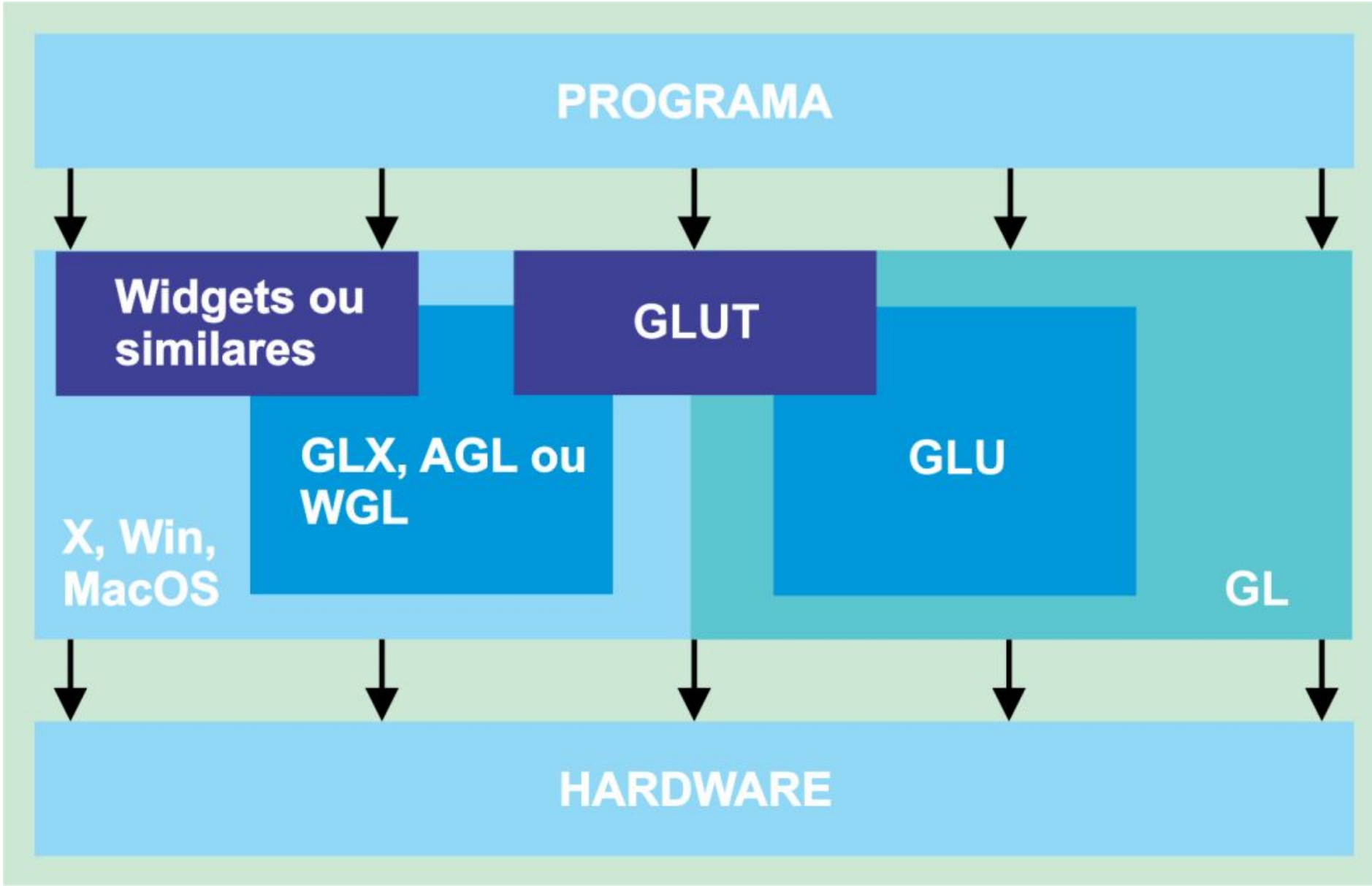
COMPUTAÇÃO GRÁFICA

AULA PRÁTICA 01

Prof. Dr. José Luiz de Souza Pio

OpenGL

- Uma API para geração de gráficos
  - 3D e 2D
  - Primitivas vetoriais e matriciais (imagens)
  - Capaz de gerar imagens de alta qualidade
  - Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível)
  - Independente de plataforma
  - Independente de sistema de janelas


PROGRAMA
Widgets ou similares
GLUT
GLX, AGL ou WGL
GLU
X, Win, MacOS
GL
HARDWARE

Biblioteca GLUT

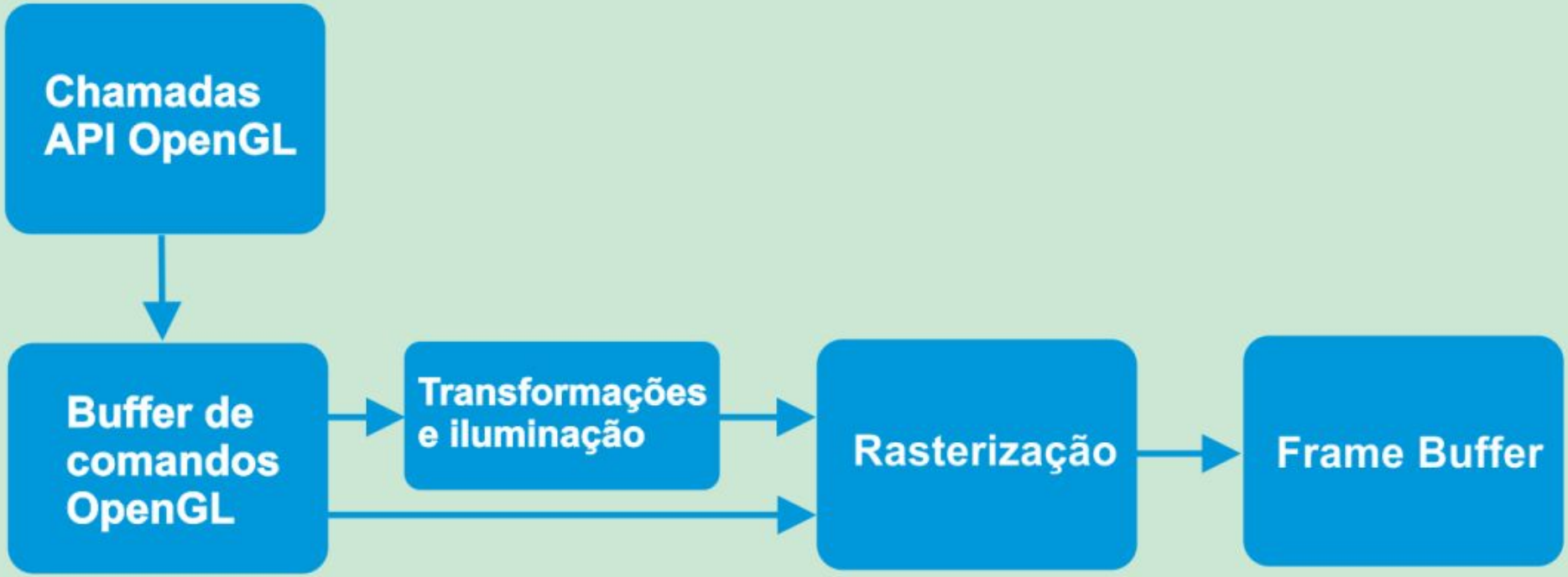
- Oferece recursos de manipulação de janelas, popup menu, tratamento de eventos de mouse e teclado e algumas primitivas gráficas 3D pré-definidas (como cubo, esfera, bule, entre outras).
- O principal objetivo da GLUT é a abstração do sistema operacional fazendo com que os aplicativos sejam multiplataforma.

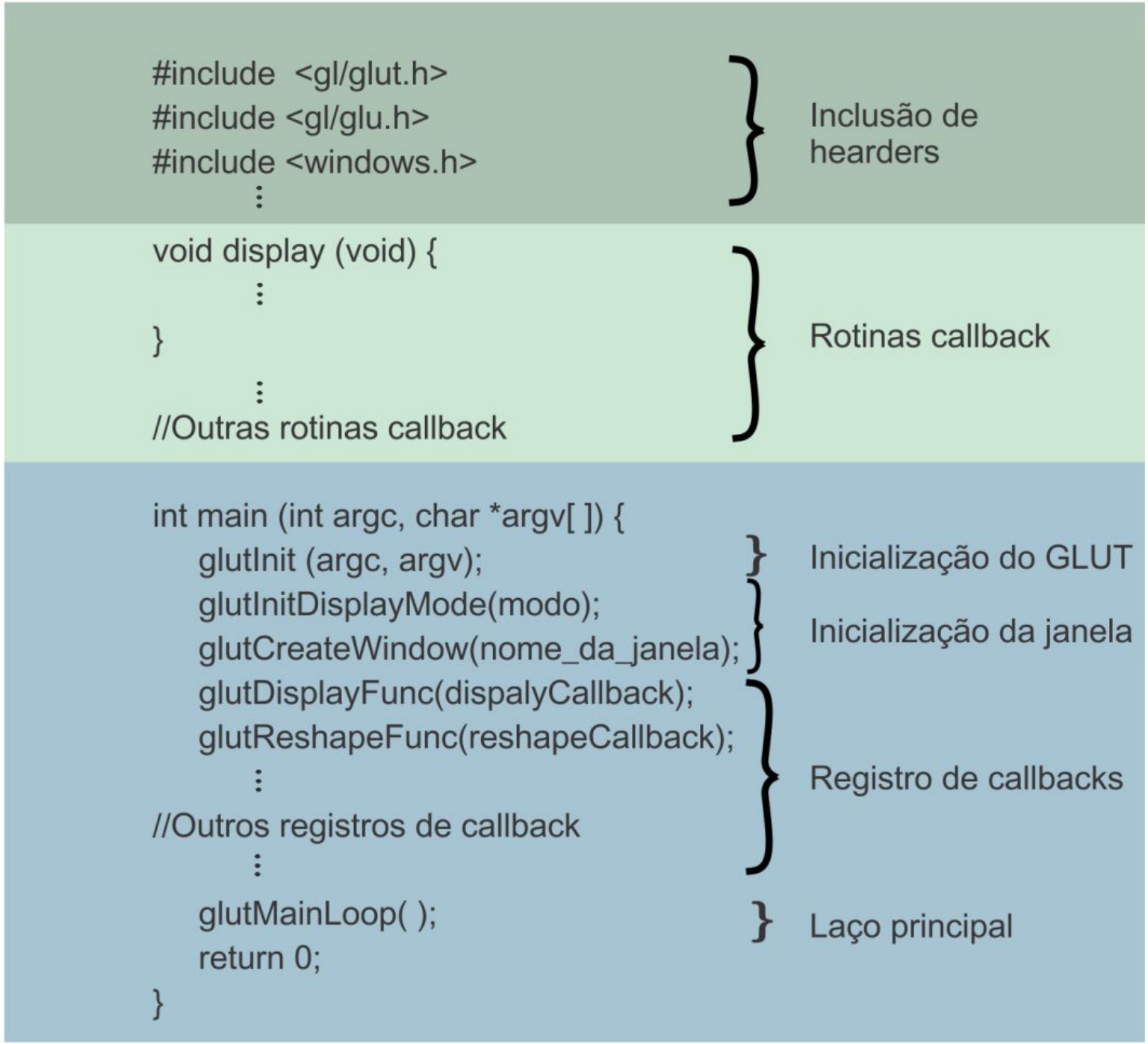
Desenhando com OpenGL

- OpenGL funciona como uma máquina de estados
- 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 um padrão em evolução
  - Mecanismo padronizado de extensões
  - Novas versões são estabelecidas por um comitê (ARB) de usuários e fabricantes

Tipos de Dados na OpenGL

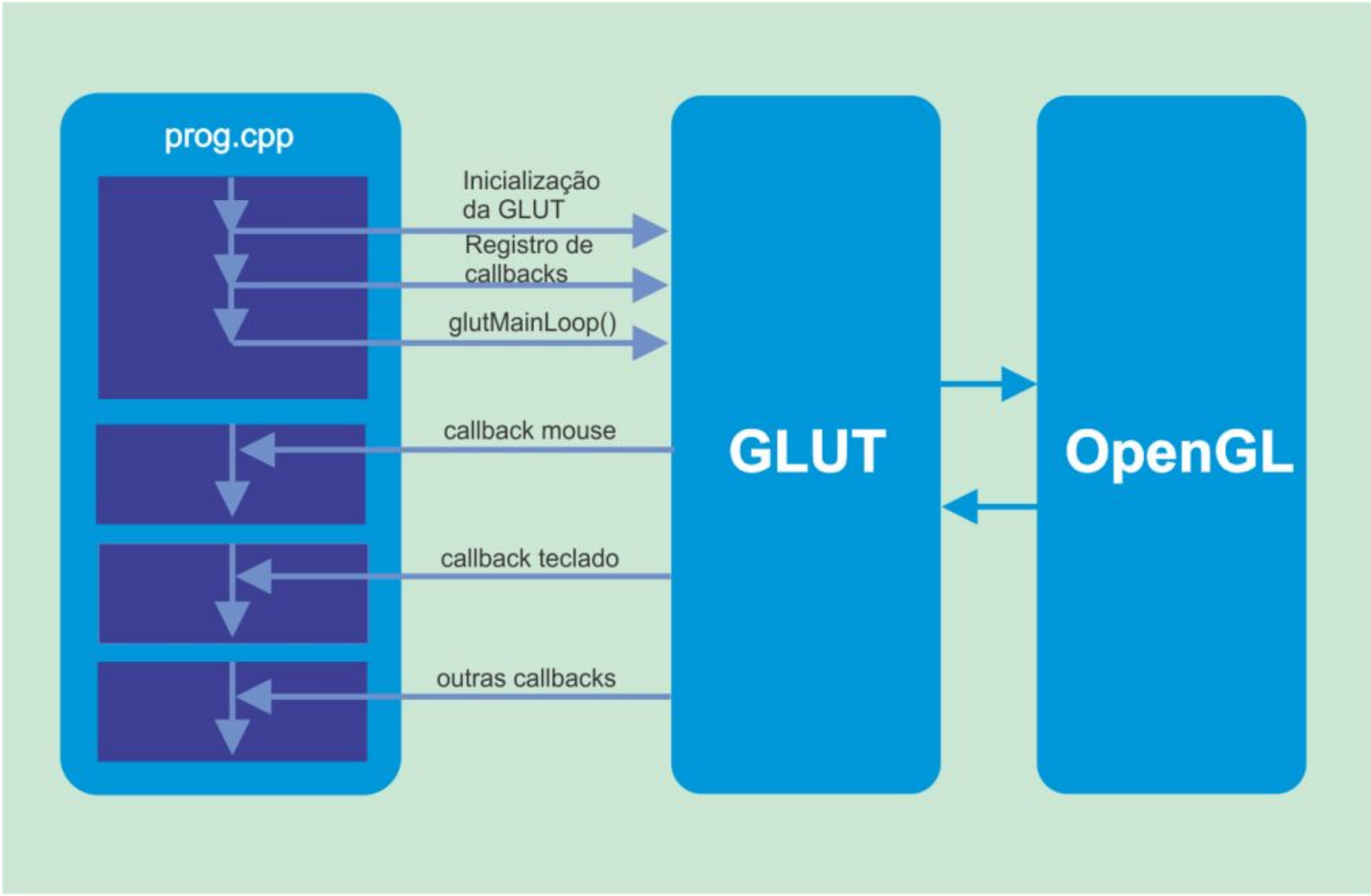
Tipo em OpenGL	Representação	Tipo em C	Sufixo
GLbyte	8-bit integer	signed char	b
GLshort	16-bit integer	short	s
GLint, GLsizei	32-bit integer	int ou long	i
GLfloat, GLclampf	32-bit float-point	float	f
GLdouble, GLclampd	64-bit floating-point	double	d
GLubyte, GLboolean	8-bit unsigned integer	unsigned char	ub
GLushort	16-bit unsig-ned integer	unsigned short	us
GLuint, GLenum, GLbitfield	32-bit unsigned integer	unsigned long ou unsigned int	ui


Chamadas
API OpenGL
Buffer de
comandos
OpenGL
Transformações
e iluminação
Rasterização
Frame Buffer


#include <gl/glut.h>
#include <gl/glu.h>
#include <windows.h>
Inclusão de hearders
void display (void) {
}
//Outras rotinas callback
Rotinas callback
int main (int argc, char *argv[ ]) {
glutInit (argc, argv);
Inicialização do GLUT
glutInitDisplayMode(modo);
glutCreateWindow(nome_da_janela);
Inicialização da janela
glutDisplayFunc(dispalyCallback);
glutReshapeFunc(reshapeCallback);
//Outros registros de callback
Registro de callbacks
glutMainLoop( );
return 0;
}
Laço principal

Loop de Eventos e Funções Callback

- As funções de callbacks são utilizadas para registrar na GLUT funções definidas pelo usuário (ponteiros para função) que vão tratar os eventos da aplicação.
- As funções de callback são aquelas executadas quando qualquer evento ocorre no sistema, tais como : redimensionamento de janela o desenho da mesma, entradas de usuários através de teclado, mouse, ou outro dispositivo de entrada, e ocorrência de animações.


prog.cpp
Inicialização
da GLUT
Registro de
callbacks
glutMainLoop()
callback mouse
callback teclado
outras callbacks
GLUT
OpenGL

```
/*simple.c */

#include <GL/glut.h>

void display()

{

    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();

    glFlush();

}


int main(int argc, char** argv)
{

    glutInit(&argc,argv);
    glutCreateWindow("simple");
    glutDisplayFunc(display);
    glutMainLoop();

}
```

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

Inicializa a GLUT e deve ser chamada Antes de qualquer função OpenGL

```
int glutCreateWindow(*char title)
```

Cria uma janela na posição padrão com 300 x 300 pixels

```
void glutDisplayFunc(void (*func) (void))
```

Função callback, display é chamada toda vez para redesenhar a figura

```
void glutMainLoop()
```

Faz o programa entrar no loop de Processamento de eventos. Este comando deve ser o último da função main().

```
/*simple.c */

#include <GL/glut.h>

void display()

{

    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();

    glFlush();

}


int main(int argc, char** argv)
{

    glutInit(&argc,argv);
    glutCreateWindow("simple");
    glutDisplayFunc(display);
    glutMainLoop();

}
```

```
void glVertex{234}{sifd}(TYPE xcoordinate, TYPE
    ycoordinate,....)
void glVertex{234}{sifd}v(TYPE *coordinates)
```

Entidade fundamental para especificar objetos Geométricos. Especifica a localização de um vértice

```
void glClear(GLbitfield mask)
```

Limpa o buffer

```
void glBegin(GLenum mode)
```

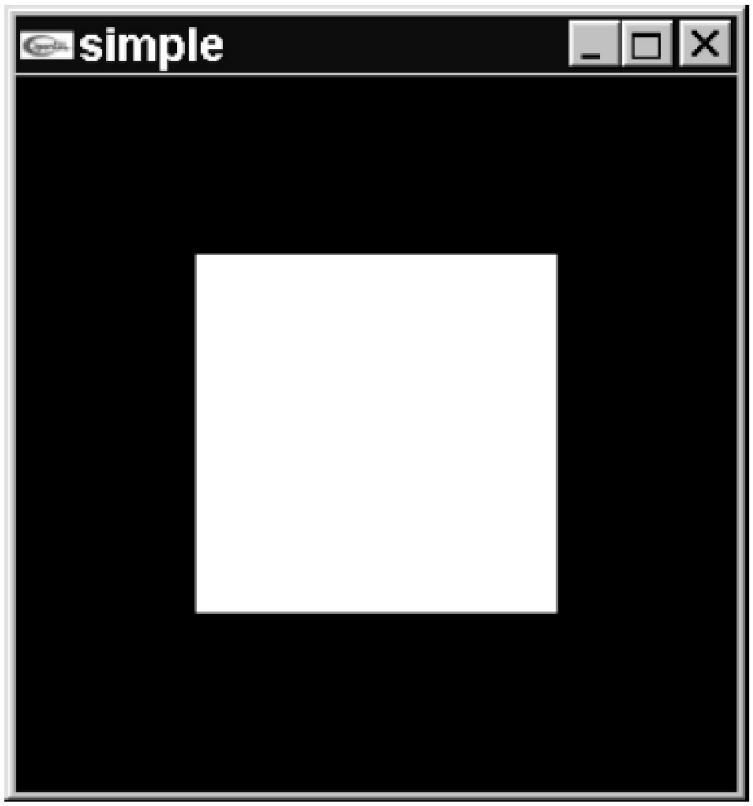
Especifica o início de um objeto e seu modo

```
void glEnd()
```

Finaliza a lista de vértices (não esqueça!)

```
void glFlush()
```

Força a execução dos comandos



1) O que fazer para se ter uma imagem com tamanho diferente?
2) O que fazer para a imagem aparecer em outra posição da tela?
3) Por que o retângulo branco ocupa a metade da área da tela?

Mudança dos padrões da GLUT

```
void glutInitDisplayMode(unsigned int mode)
```

Requests a display with the properties in `mode`. The values of `mode` are combined by using the logical `OR` of options, such as color model (`GLUT_RGB`, `GLUT_INDEX`) and buffering of color buffers (`GLUT_SINGLE`, `GLUT_DOUBLE`).

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

Specifies the initial height and width, in pixels, of the window on the screen.

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

Specifies the top-left corner of the window, measured in pixels, from the top-left corner of the screen.

Cores

```
void glColor3{b i f d ub us ui}(TYPE r, TYPE g, Type b)
void glColor3{b i f d ub us ui}v(TYPE *color)
void glColor4{b i f d ub us ui}(TYPE r, TYPE g, Type b,
    TYPE a)
void glColor4{b i f d ub us ui}(TYPE *color)
```

Specifies RGB and RGBA colors, using the standard types. If the `v` is present, the color is in an array pointed to by `color`.

```
void glClearColor(GLclampf r, GLclampf g, GLclampf b,
    GLclampf a)
```

Specifies the clear color (RGBA) used when clearing the color buffer.

Visualização 2D

```
void gluOrtho2D(GLdouble left, GLdouble right, GLdouble
     bottom, GLdouble top)
```

Specifies a two-dimensional rectangular clipping region whose lower-left corner is at (`left, bottom`) and whose upper-right corner is at (`right, top`)

```
void glMatrixMode(GLenum mode)
```

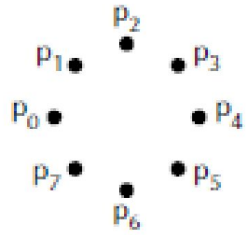
Specifies which matrix will be affected by subsequent transformation functions. The `mode` is usually `GL_MODELVIEW` or `GL_PROJECTION`.

```
void glLoadIdentity()
```

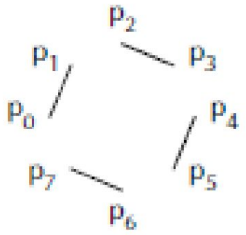
Initializes the current matrix to an identity matrix.

```
/* simple.c second version */
/* This program draws a white rectangle on a black background.*/


#include <GL/glut.h>        /* glut.h includes gl.h and glu.h*/


void display()

{
/* clear window */

    glClear(GL_COLOR_BUFFER_BIT);


/* draw unit square polygon */

    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5)
        glVertex2f(0.5, -0.5);
    glEnd();

/* flush GL buffers */

    glFlush();

}


void init()
{

/* set clear color to black */

    glClearColor(0.0, 0.0, 0.0, 0.0);

/* set fill color to white */

    glColor3f(1.0, 1.0, 1.0);

/* set up standard orthogonal view with clipping */
/* box as cube of side 2 centered at origin */
/* This is default view and these statements could be removed */

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

int main(int argc, char** argv)
{

/* Initialize mode and open a window in upper left corner of
/* screen */
/* Window title is name of program (arg[0]) */

    glutInit(&argc,argv)
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("simple");
    glutDisplayFunc(display);
    init();
    glutMainLoop();

}
```

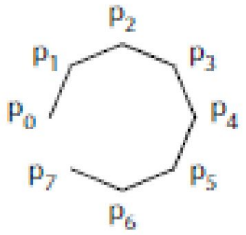
Aula Prática 01/2024
Objetivo: Uso da OpenGL para o traçado de figuras 2D.



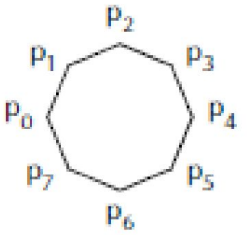
GL_POINTS



GL_LINES



GL_LINE_STRIP



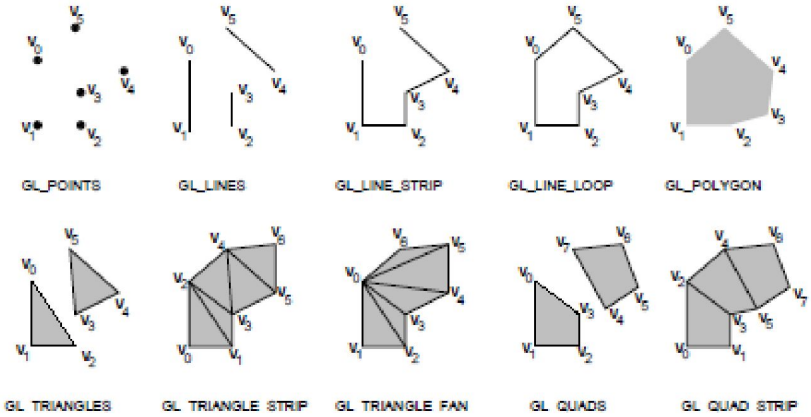
GL_LINE_LOOP

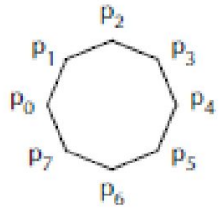
```
glBegin(GL_LINES);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
glEnd();
```

```
glBegin(GL_LINE_STRIP);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
glEnd();
```

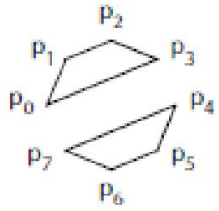
```
glBegin(GL_LINE_LOOP);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
glEnd();
```

```
glPointSize(2.0);
glBegin(GL_POINTS);
    glColor3f(1.0, 1.0, 1.0);
    glVertex2f(-0.5, -0.5);
    glColor3f(1.0, 0.0, 0.0);
    glVertex2f(-0.5, 0.5);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(0.5, 0.5);
    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(0.5, -0.5);
glEnd();
```

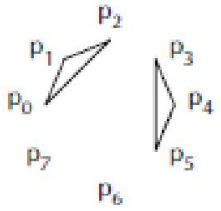




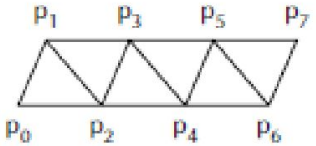
GL_POLYGON



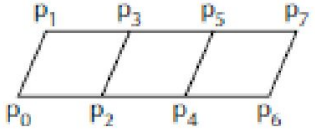
GL_QUADS



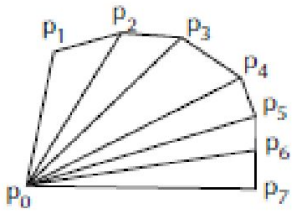
GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_QUAD_STRIP



GL_TRIANGLE_FAN

Tarefas:

1) Implementar o programa para o desenho do retângulo;
2) Alterar os parâmetros do programa e relatar seus efeitos;
3) Desenhar uma circunferência de raio 0.5 no centro da janela
    1) Usando GL_LINE_STRIP;
    2) Usando GL_POLYGON;

```
#include <GL/glut.h>
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
glEnd();
glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutCreateWindow("simple");
glutDisplayFunc(display);
glutMainLoop();
}
```