



# Computação Gráfica

André Perrotta (avperrotta@dei.uc.pt)

Hugo Amaro (hamaro@dei.uc.pt)

**TP-02: Introdução ao  
OpenGL e  
OpenFrameworks(OF)**

# OpenGL



- Open Graphics Library (1992)
  - Industry standard
  - 2D e 3D graphics API
    - Possibilita executar as operações do “render pipeline” através do uso de objetos e funções de alto nível.
    - Não possui funções de alto nível para criação, todos os objetos (forma e estética) devem ser descritos com informação de baixo nível (vértices, arestas, faces, propriedade dos materiais, etc.)
  - Multiplataforma
    - Elimina a necessidade de se ter de programar as aplicações diferentemente para cada tipo de hardware (OS + interface de saída gráfica)
  - C/C++
  - Só contém objetos e funções relativos aos processos e algoritmos da construção gráfica, funções “burocráticas” como criar janelas, permitir interação com rato e teclado, etc., devem ser realizadas com auxílio de outras bibliotecas
    - Glut, Glew, etc.

# OpenGL

- Um programa que utiliza OpenGL terá tipicamente as seguintes componentes:
  - Criação de uma janela utilizando código apropriado para inicializar e configurar
  - Inicialização de variáveis e parâmetros necessários
  - Criação dos objetos
  - Configuração da cena (luz, câmera, projeção)
  - Desenho dos objetos

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize(600, 480);
glutInitWindowPosition(0, 0);
glutCreateWindow(" Exemplo inicial ");
```

```
glClearColor(0., 0., 0., 1.0);
glShadeModel(GL_FLAT);
glEnable(GL_DEPTH_TEST);
glEnable(GL_NORMALIZE);
glEnable(GL_LIGHTING);
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glViewport(0, 0, wScreen, hScreen);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(120, (float)wScreen / hScreen, 0.1, 1000.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(2, 0, 2, 2, 0, 0, 0, 1, 0);
glEnable(GL_LIGHTING);
glColor3f(1., 1., 1.);
glPushMatrix();
glTranslatef(2, 0., 0.);
glRotated(theta, 0, 1, 0);
triangle();
glPopMatrix();
glutSwapBuffers();
```

# OpenGL

- Namespace
  - Todos os objetos e funções do OpenGL obedecem a mesma nomenclatura
    - prefixo | comando | contador de argumentos | tipo de variável do argumento
  - Exemplo:
    - glColor3f
      - Prefixo: gl
      - Comando: Color
      - Contador de argumentos: 3
      - Tipo: f (float)

# OpenFrameworks (OF)

- Framework (arquétipo) para aplicações multimédia baseadas em OpenGL.
  - Objetivo principal é o de facilitar a utilização de bibliotecas de programação de baixo nível (por exemplo: OpenGL) através de uma coleção de funções e ferramentas que simplificam e minimizam o código necessário para criar aplicações.
  - Facilitam a integração de várias funcionalidades provenientes de diferentes bibliotecas (computer vision, machine learning, dsp, hardware integration etc.)
  - Permitem “misturar” as funções de alto nível do framework com as funções originais (de mais baixo nível) das bibliotecas standard.

# OpenFrameworks (OF)

- É muito utilizado por profissionais da área (multimédia interativo, vídeo mapping, cenografia de shows e espetáculos...)
- É free e open source.
- É continuamente atualizado
- Tem comunidade engajada e prestativa (o fórum é bastante amigável e útil)
- Ponto de partida para quem quiser explorar a área do desenvolvimento de aplicações multimédia (entretenimento, cultura, games, etc.)

# Aplicação OF

- Todas as Apps feitas com OF começam com 3 ficheiros de código:
  - `main.cpp`
    - Configuração da janela
  - `ofApp.h`
    - Declarações da classe ofApp, suas variáveis e métodos
  - `ofApp.cpp`
    - Implementação da classe ofApp

# Aplicação OF: main.cpp

- No ficheiro main.cpp está o início do programa, que como todo e qualquer programa em C/C++ é a função main()
- Podemos configurar a resolução da janela em pixels e também a forma de exibição:
  - OF\_WINDOW -> janela
  - OF\_FULLSCREEN -> tela cheia com resolução do sistema
  - OF\_GAME\_MODE -> tela cheia com resolução fixa
  - É possível alterar entre o modo janela e fullscreen durante a execução:
    - ofToggleFullscreen();
- Após configurar a janela, o programa inicializa a rotina de execução de um ofApp

```
#include "ofMain.h"
#include "ofApp.h"
//=====
int main( ){
    ofGLWindowSettings settings;
    settings.setSize(1024, 768);
    settings.windowMode = OF_WINDOW;
    auto window = ofCreateWindow(settings);
    ofRunApp(window, make_shared<ofApp>());
    ofRunMainLoop();
}
```



# Aplicação OF: ofApp.h

- Declaração da classe “ofApp.h” e seus métodos obrigatórios
- Possui os métodos necessários para interação com rato, teclado, redimensionar a janela, etc.
- Declaração dos 3 métodos fulcrais das aplicações OF:
  - Setup()
  - Update()
  - Draw()

```
#pragma once
#include "ofMain.h"
class ofApp : public ofBaseApp{
public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y );
    void mouseDragged(int x, int y, int button);
    void mousePressed(int x, int y, int button);
    void mouseReleased(int x, int y, int button);
    void mouseEntered(int x, int y);
    void mouseExited(int x, int y);
    void windowResized(int w, int h);
    void dragEvent(ofDragInfo dragInfo);
    void gotMessage(ofMessage msg);
};
```

# Aplicação OF: ofApp.cpp

- Implementação dos métodos declarados em ofApp.h

```
1  #include "ofApp.h"
2
3  //-----
4  void ofApp::setup(){
5
6  }
7  //-----
8  void ofApp::update(){
9
10 }
11 //-----
12 void ofApp::draw(){
13
14 }
15 //-----
16 void ofApp::keyPressed(int key){
17
18 }
19
20 //-----
21 void ofApp::keyReleased(int key){
22
```

# Aplicação OF: setup/update/draw

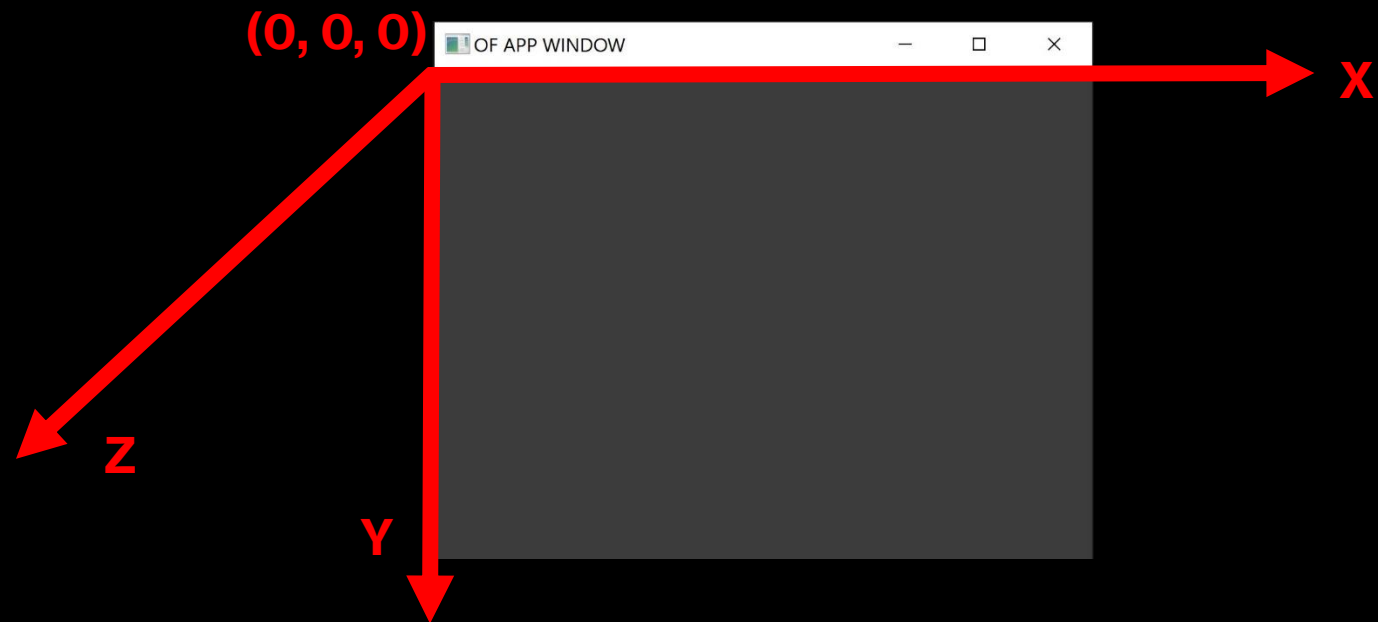
- void setup() -> executado apenas 1 vez quando a aplicação inicializa
- void update() -> executado em loop infinito
- void draw() -> executado em loop infinito
- update() e draw() são chamadas em série, ou seja, 1º executa update() e só ao fim é executado o draw()
  - A “velocidade” de execução (framerate) pode ser definida e controlada ou não (executa tão rápido quanto for possível).
- A ideia é separar algoritmos que não geram output visual dos algoritmos de desenho propriamente ditos.
  - Por exemplo: se tivermos uma “bola” a andar na tela com as variáveis X e VelX, onde em cada frame atualizamos  $X = X + \text{VelX}$ , antes de desenhar circle(X, Y, Z). Colocaremos a atualização da coordenada no update() e a chamada para desenho no draw()
- Isto não é obrigatório, se quiser, pode colocar tudo no draw ou update. Apenas ajuda a organizar.

# Aplicação OF: namespace

- Semelhante ao namespace do OpenGL, o namespace de OF vai ter um prefixo e comando:
  - Prefixo | comando
- Exemplo:
  - ofGetWidth
    - Prefixo: of
    - Comando: GetWidth (retorna a largura da tela em pixels)

# Aplicação OF: coordenadas (default)

- por defeito, a origem do Sistema de coordenadas é o canto superior esquerdo



# Aplicação OF: máquina de estados

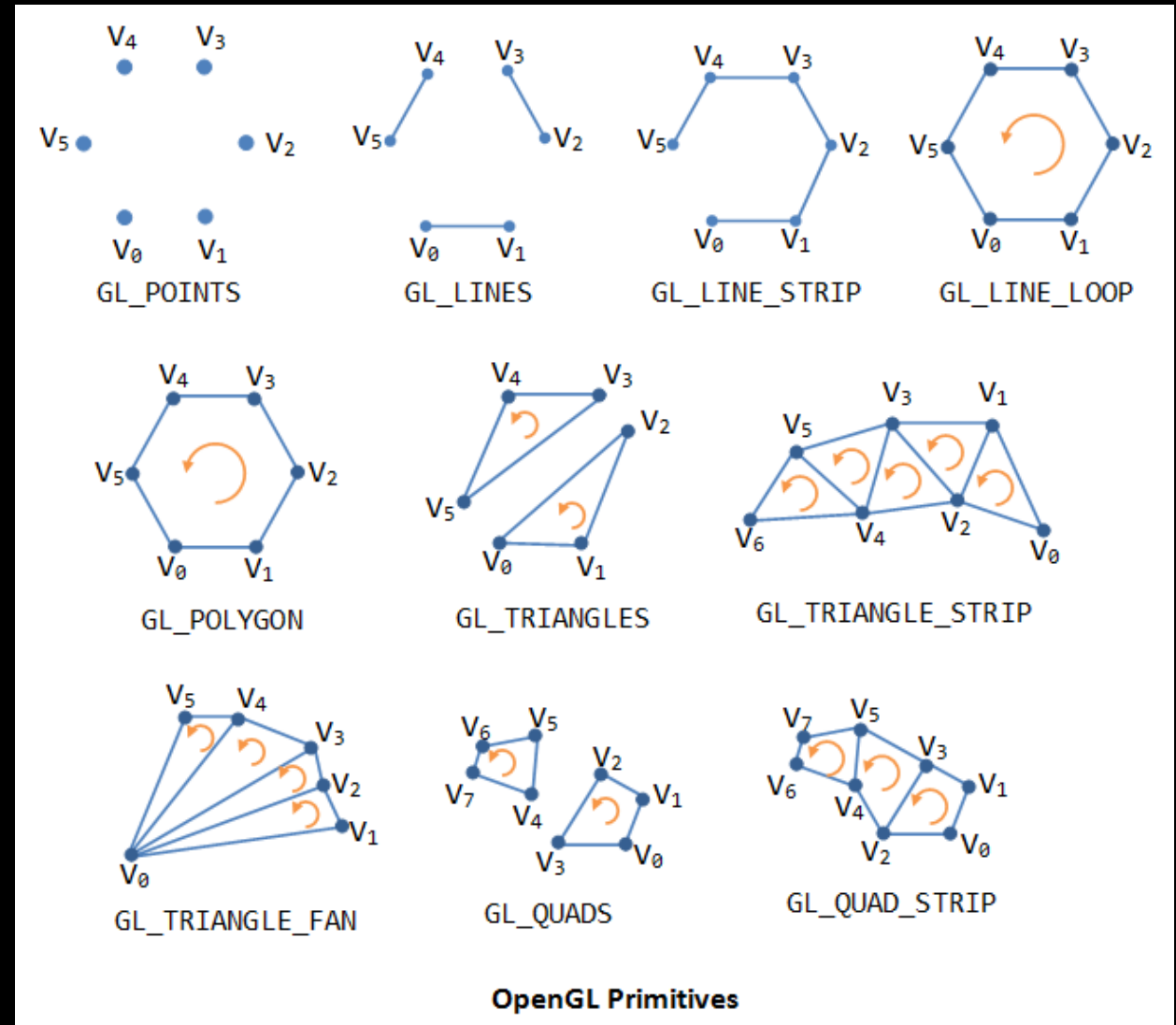
- Uma app OF, por ser baseada em conceitos e código OpenGL, funciona como uma máquina de estados (paradigma openGL)
  - Uma vez que determinamos uma configuração ou parâmetro, este passa a valer para todas ações futuras, a não ser que seja explicitamente renunciado ou modificado.
  - Por exemplo:
    - Uma vez que determinar a “cor” a ser utilizada para os desenhos, ela passa a valer para todo desenho que venha a seguir

# Primitivas de desenho

- Em OpenGL para construir objetos 2D (ou 3D), devemos ser capaz de definir:
  1. Os vértices do objeto
  2. Uma estratégia (dentre as possíveis) para que os vértices sejam conectados.
- A estratégia de ligação do vértices é chamada de:
  - Primitiva de desenho (drawing primitive)

# Primitivas de desenho

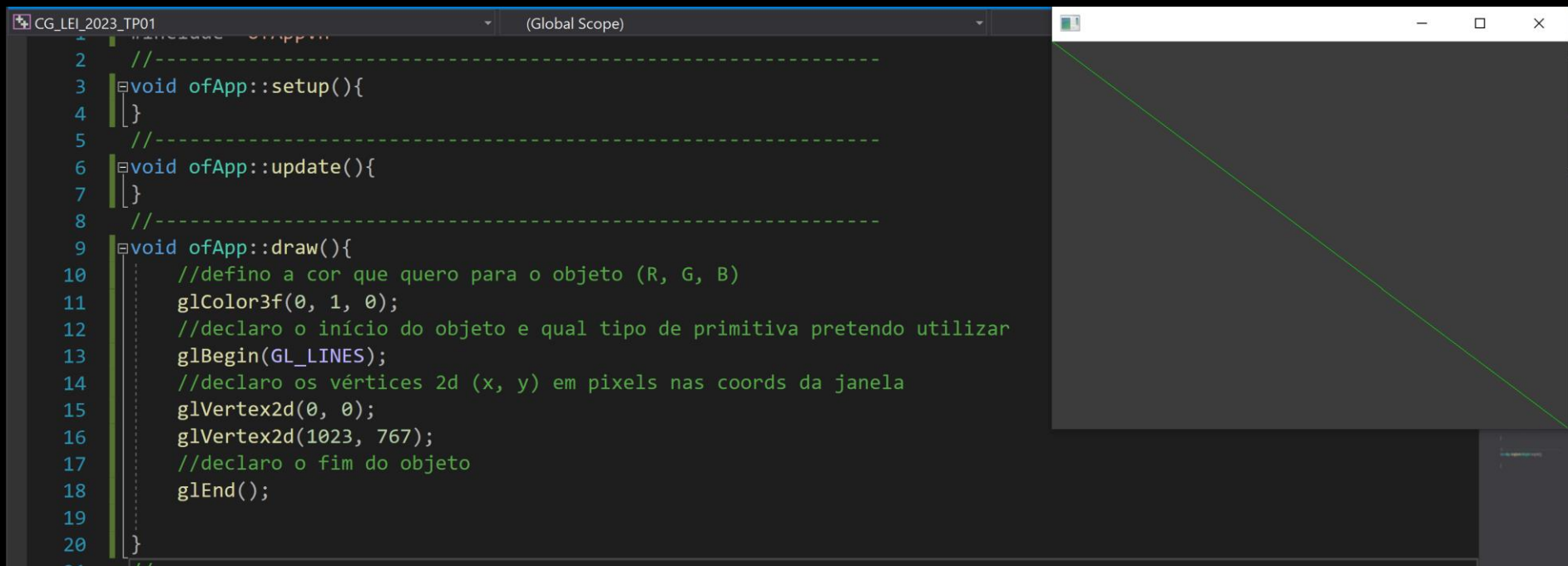
- Em OpenGL são possíveis as seguintes primitivas:





# Primitivas de desenho

- Exemplo em código para o desenho de uma linha reta que faz a diagonal do canto esquerdo superior até o direito inferior:



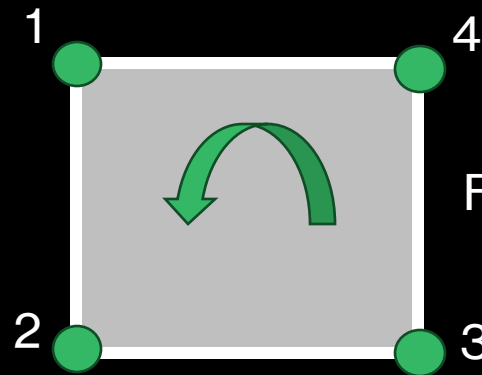
The image shows a code editor window on the left and a graphical window on the right. The code editor, titled 'CG\_LEI\_2023\_TP01', contains the following code:

```
1 //-----  
2  
3 void ofApp::setup(){  
4 }  
5 //-----  
6 void ofApp::update(){  
7 }  
8 //-----  
9 void ofApp::draw(){  
10     //defino a cor que quero para o objeto (R, G, B)  
11     glColor3f(0, 1, 0);  
12     //declaro o início do objeto e qual tipo de primitiva pretendo utilizar  
13     glBegin(GL_LINES);  
14     //declaro os vértices 2d (x, y) em pixels nas coords da janela  
15     glVertex2d(0, 0);  
16     glVertex2d(1023, 767);  
17     //declaro o fim do objeto  
18     glEnd();  
19  
20 }
```

The graphical window on the right displays a dark gray rectangle with a thin green diagonal line running from the top-left corner to the bottom-right corner, representing the output of the provided code.

# Primitivas de desenho

- Para além de determinar como os vértices serão conectados, a ordem em que determinamos os vértices também determina a orientação da face: **Face Normal**, que é fundamental para questões de visualização e iluminação.
- Por defeito, isto é determinado pela "regra da mão direita".
- Não faz diferença o vértice que escolhemos como sendo o primeiro, desde que a ordem esteja correta



Face aponta para "frente" (for a da tela)



# Dúvidas?