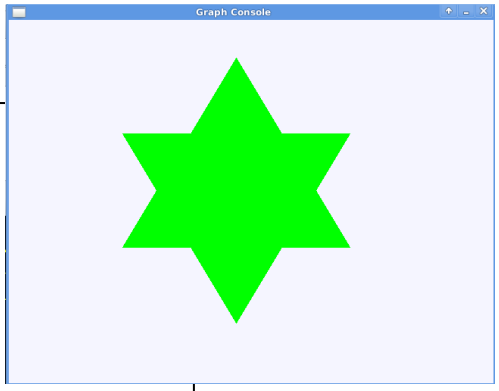
Introdução à biblioteca de suporte a Programação I

Durante o semestre será usada uma biblioteca que exporta, entre outras que veremos mais tarde, funções de desenho de formas gráficas, nomeadamente pontos, linhas, triângulos, rectângulos e círculos. O conjunto completo das funções de desenho é apresentado no final deste enunciado.

A seguir apresenta-se um programa que exemplifica a utilização da biblioteca o resultado da sua execução (o programa termina quando se fechar a consola gráfica criada):




```
// ficheiro de include necessário
#include "pg/graphics.h"

int main() {
        // chamada necessária antes da iniciação da biblioteca
        graph_init();

        // desenho de um triângulo dados os pontos dos vértices.
        // O penúltimo argumento indica a côr e o último que o
        // triângulo é desenhado "a cheio"
        graph_triangle(150, 300, 450, 300, 300,  50,  c_green, true);

        // desenho de um segundo triângulo dados os pontos dos vértices,
        // a cor e com desenho "a cheio"
        graph_triangle(150, 150, 450, 150, 300,  400, c_green, true);

        // chamada necessária para correcto refrescamento da consola gráfica
        graph_start();
        return 0;
}
```

Em geral, para compilar um programa que usa a biblioteca deverá usar-se o seguinte comando:

```
gcc –Wall –std=c99  –o <prog> <prog>.c –lpg
```

Definições e funções de desenho (excerto de pg/graphics.h)

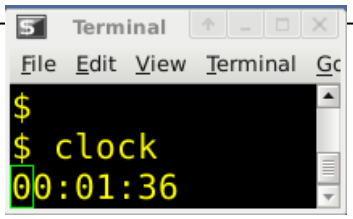
```
/* Graph Window Dimensions */
#define GRAPH_WIDTH  640
#define GRAPH_HEIGHT 480

/*cores pré-definidas */
const RGB c_white;
const RGB c_gray;
const RGB c_dgray;
const RGB c_cyan;
const RGB c_black;
const RGB c_orange;
const RGB c_lightblue;
const RGB c_red;
const RGB c_green;
const RGB c_blue;

 /* funções */
// returns an rgb color from their components
RGB graph_rgb(short r, short g, short b);
```

```
// initializes the graphics. Must be called first
int graph_init();

// draws a pixel at position with specified color
void graph_pixel(short x, short y, RGB color);

// draws a line at position with specified color
void graph_line(short x1, short y1, short x2, short y2, RGB color);

// draws a circle at position with specified color, filled or not
void  graph_circle(short x0, short y0, short radius, RGB color, bool toFill);

// draws a rect at position with specified color, filled or not
void graph_rect(short x0, short y0, short w, short h, RGB color, bool toFill);

// draws a ellipse at position with specified color, filled or not
void  graph_ellipse(short x0, short y0, short xr, short yr, RGB color, bool toFill);

// draws a round rectangle at position with specified color, filled or not
void graph_round_rect(short x0, short y0, short w, short h, RGB color, bool toFill);

// draws a triangle at position (x0,y0) with specified color, filled or not
int graph_triangle2(int x0, int y0, int a, int b, int c, RGB color, bool toFill);

// draws a triangle given his points (x0,y0), (x1,y1), (x2,y2)
// with specified color, filled or not
void graph_triangle(int x0, int y0, int x1, int y1, int x2, int y2, RGB color, bool toFill );

// start the event loop
void graph_start();

// stop the event loop
void  graph_exit();

//for text write
#define SMALL_FONT  1
#define MEDIUM_FONT 2
#define LARGE_FONT  3

// Writes a string starting at point x,y with "color" as text color
// and witth font size specified in fontsize (SMALL, MEDIUM, LARGE)
void graph_text( short x, short y, RGB color, char text[], int fontsize );

// image processing (just for 64 bits image)

// loads a bitmap or jpeg file in the graphic window given the filename (path),
// the topleft position (x,y) and dimensions (width, height)
bool graph_image(const char *path, int x, int y, int width, int height);

// save in a bitmap with name “path” the cuurent graphic window contents
bool graph_save_screen(const char path[]);

// sound functions (just for 64 bits ubuntu)

// play a wav file with name path. return false if file not exist
bool  sound_play(const char *path);

// play pause
void sound_pause();

// resume play
void sound_resume();

// stop file play
void sound_stop();
```

Eventos

Esta biblioteca permite registar funções (ditas de callback) que serão chamadas na ocorrência de um evento de teclado, de um evento de rato ou de um evento de temporização (por cada período de tempo previamente definido).

Como exemplo de um progra,ma que regista uma função de tratamento de evento de temporização, o programa seguinte mostra na consola de texto um relógio digital (HH:MM:SS):




```
#include <stdio.h>
#include “pg/graphics.h”

int hours, minuts, secs;

void myTimerHandler() {
        secs++;
        if (secs == 60) {
                secs = 0;
                minuts++;
                if (minuts == 60 ) {
                     minuts == 0;
                     hours = (hours + 1) % 24;
                 }
        }
        printf(“%02d:%02d:%02d\r”,  hours, minuts, secs);
        fflush(stdout); // esta função força os caracteres a serem escritos na consola
}

int main()  {
        graph_init();
        // regista a função “myTimerHandler”
        // para ser chamada em cada segundo
        graph_regist_timer_handler(myTimerHandler, 1000);

        graph_start();
        return 0;
}
```

Definições e funções relacionadas com eventos de teclado, rato e temporização

```
/*
 * constantes de eventos de rato
 */

// campo type de MouseEvent
#define MOUSE_MOTION_EVENT 1
#define MOUSE_BUTTON_EVENT 2

// campo button de MouseEvent
#define BUTTON_LEFT         1
#define BUTTON_RIGHT        2


// campo state de MouseEvent
#define BUTTON_PRESSED      1
#define BUTTON_RELEASED     2
#define BUTTON_CLICK        4
typedef struct MouseEvent  {
      int type;             /* mouse motion or mouse button event type */
      byte state;           /* button pressed, released or click */
      int button;           /* which button is pressed */
      ushort x, y;          /* mouse coordinates */
      short dx, dy;         /* delta move when mouse motion event */
} MouseEvent;

/*  KEYBOARD Events constants */
// campo state de KeyEvent
#define KEY_PRESSED 0
#define KEY_RELEASED1
#define MAX_NAME 30

typedef struct KeyEvent {
      int state;              /* key pressed or released */
      ushort keysym;          /* key code */
      char keyName[MAX_NAME]; /* key name */
} KeyEvent;

// assinatura das funções que tratam eventos
void KeyEventHandler(KeyEvent ke);
void MouseEventHandler(MouseEvent me);
void TimerEventHandler();

/* funções para registar as funções que processam eventos */

// Regist the keyboard event handler
void graph_regist_key_handler(KeyEventHandler ke);

// Regist the mouse event handler
void graph_regist_mouse_handler(MouseEventHandler me);

// Regist the timer event handler
void graph_regist_timer_handler(TimerEventHandler te, uint period);
```