Uso de Gráficos no ambiente Borland C++

Prof. Paulo André Castro
ITA - IEC

1. Introdução

O padrão ANSI da linguagem C (C ANSI) não define rotinas gráficas, entretanto elas são necessárias em vários programas profissionais. Por tal razão, foram criados independentemente vários conjuntos de rotinas gráficas como por exemplo o Microsoft C/C++ para DOS e o Borland Graphics Interface (BGI). Além destas, junto com sistemas operacionais baseados em janelas (Windows, X Windows, Mac OS, etc.) foram criadas bibliotecas gráficas para trabalhar com tais sistemas. Existem ainda outras bibliotecas gráficas avançadas, como o OpenGL, capazes de construir gráficos em 3D.

No contexto de um curso de introdução a computação, é interessante utilizar uma biblioteca gráfica simples em 2D, mas que forneça funções de desenhos formas geométricas, como retas, círculos, retângulos, etc. O restante deste documento, descreve como habilitar e utilizar o BGI no Borland C++. O livro Programação Orientada a Objetos com Turbo C++ do autor G. Perry (disponível na biblioteca do ITA) traz no seu apêndice F uma breve introdução ao uso de BGI em programas gráficos.

2. Biblioteca BGI no ambiente Borland C++

2.1. Uso de Projetos

Para trabalhar em modo gráfico será necessário utilizar o conceito de Projeto. Um projeto pode ser considerado como um container que armazena todos os elementos que irão compor um programa.

2.1.1. Criando um novo projeto.

- Clique no menu "File" e selecione "New", "Project...".
- Selecione a plataforma “DOS Standard” e selecione a opção BGI. Escolha um nome e local de gravação para o projeto. Você pode dar qualquer nome válido para um arquivo. O nome do projeto sera o nome do executável a ser gerado (ver figura 1).
- Após escolher o nome, clique "OK".

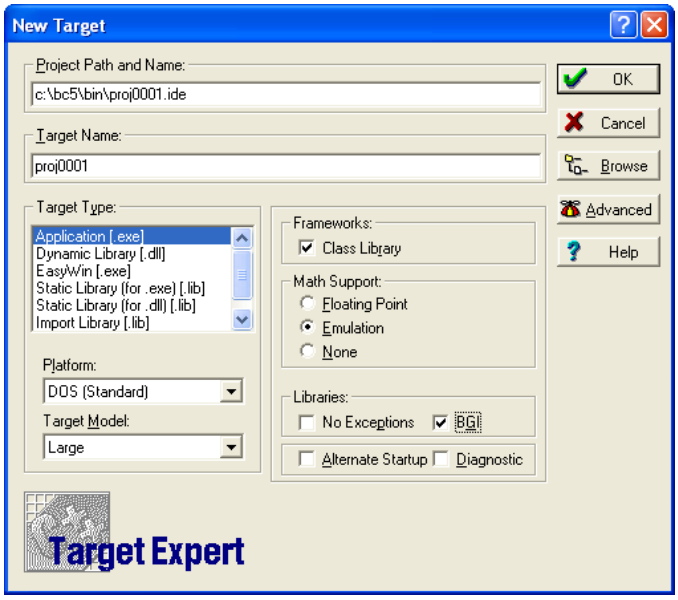

Figura 1. Criação de um projeto no Borland C++

3. Primeiros programas gráficos

Digite no arquivo fonte (.cpp) do projeto o código abaixo.

Código 1. Um programa gráfico simples

```
#include <stdio.h>
#include <graphics.h>

main () {

        int i, j, left, top, bottom, right;

        initwindow(700,500);

        left = 10; right = 600; top = 10; bottom = 200;
        for (i = top; i <= bottom; i++)
                for (j = left; j <= right; j++)
                        putpixel (j, i, MAGENTA);
        getch ( );
        closegraph ( );

}
```

Tabela. 1. Lista de cores disponíveis no BGI

Valor	Símbolo	Valor	Símbolo
0	BLACK	8	DARKGRAY
1	BLUE	9	LIGHTBLUE

2	GREEN	10	LIGHTGREEN
3	CYAN	11	LIGHTCYAN
4	RED	12	LIGHTRED
5	MAGENTA	13	LIGHTMAGENTA
6	BROWN	14	YELLOW
7	LIGHTGRAY	15	WHITE

Código 2. Uma alteração no código 1 para criar novas formas geométricas

```
#include <stdio.h>
#include <graphics.h>

main () {

        int i, j, left, top, bottom, right;

        detectgraph (&g_driver, &g_mode);
        initgraph (&g_driver, &g_mode, "C:\\BC5\\BGI"); //este deve ser o caminho da sua
                                                        //instalação do BC ++

        left = 10; right = 600; top = 10; bottom = 200;
        for (i = top; i <= bottom; i++)
                for (j = left; j <= right; j++)
                        putpixel (j, i, MAGENTA);
        getch ( );
        setcolor (YELLOW);
        rectangle (left, top, right, bottom);
        setcolor (LIGHTBLUE);
        circle (200, 200, 100);
        setcolor (WHITE);
        line (50, 350, 550, 50);
        left = 10; right = 600; top = 400;   bottom = 410;
        setfillstyle (1, GREEN);
        bar (left, top, right, bottom);
        getch ( );
        closegraph ( );

}
```

4. Lista das Funções Disponíveis

4.1. Principais Funções

Funções Gráficas Borland BGI

Protótipo da Função	Descrição
void far detectgraph(int far *graphdriver, int far *graphmode);	Determine o modo gráfico
void far initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver);	Inicializa o modo gráfico

void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);	Desenha elipse
void setfillstyle(int pattern, int color);	Seta estilo de preenchimento: SOLID_FILL, LINE_FILL, etc. e cor de preenchimento
void fillellipse(int x, int y, int xradius, int yradius);	Desenha elipse preenchida
void circle(int x, int y, int radius);	Desenha círculo com raio e centro definido.
void far bar(int left, int top, int right, int bottom);	Desenha retângulo preenchido.
void floodfill(int x, int y, int border);	Preenche uma área limitada pela cor definida em "border", com a cor de preenchimento.
void far rectangle(int left, int top, int right, int bottom);	Desenha retângulo não preenchido.
void setcolor(int color);	Seta cor corrente
void setbkcolor(int color);	Seta cor de fundo
Getmaxx() e getmaxy()	Retornam os maximos de x e y da tela
void line(int x1, int y1, int x2, int y2);	Desenha uma linha
void lineto(int x, int y);	Desenha uma linha da posição corrente para x,y
int kbhit()	Retorna 1 caso o usuário tenha digitado uma tecla, 0 caso contrário. Não bloqueia o programa.
int getch (void);	Retorna a tecla digitada pelo usuário.
void delay (int millisec);	Suspende a execução por um certo número de milisegundos.
void arc(int x, int y, int stangle, int endangle, int radius);	Desenha um arco de círculo
void outtextxy(int x, int y, char far *textstring);	Desenha um texto a partir da posicao x,y

4.2. Outras funções disponíveis na BGI

Para maiores informações sobre as funções abaixo e uma lista completa das funções BGI, consulte http://www.cs.colorado.edu/%7Emain/cs1300/doc/bgi/bgi.html. As funções marcadas com [WIN] são específicas da implementação BGIm, não fazendo parte da implementação padrão do BGI.

int getbkcolor (void);
int getch (void); [WIN
int getcolor (void);
int getcurrentwindow (void); [WIN]
int getdisplaycolor (int color); [WIN
int getgraphmode (void);
int getmaxcolor (void);
int getmaxmode (void);
int getmaxx (void);
int getmaxy (void);
int getpalettesize (void);
int getpixel (int x, int y);
int getx (void);
int gety (void);
int graphresult(void);
int installuserfont (char *name);
int kbhit (void); [WIN
int registerbgidriver (void (*driver)(void));
int registerbgifont (void (*font)(void));
int textheight (char *textstring);
int textwidth (char *textstring);
void arc (int x, int y, int stangle, int endangle, int radius);
void bar (int left, int top, int right, int bottom);
void bar3d (int left, int top, int right, int bottom, int depth, int topflag);
void circle (int x, int y, int radius);
void cleardevice (void);
void closegraph (int window=ALL_WINDOWS); [WIN

void delay (int millisec); [WIN
void drawpoly (int numpoints, int *polypoints);
void ellipse (int x, int y, int stangle, int endangle, int xradius, int yradius);
void fillellipse (int x, int y, int xradius, int yradius);
void fillpoly (int numpoints, int *polypoints);
void floodfill (int x, int y, int border);
void line (int x1, int y1, int x2, int y2);
void linerel (int dx, int dy);
void lineto (int x, int y);
void moverel (int dx, int dy);
void moveto (int x, int y);
void outtext (char *textstring);
void outtextxy (int x, int y, char *textstring);
void pieslice (int x, int y, int stangle, int endangle, int radius);
void putpixel (int x, int y, int color);
void rectangle (int left, int top, int right, int bottom);
void restorecrtmode (void);
void sector (int x, int y, int stangle, int endangle, int xradius, int yradius);
void setactivepage (int page);
void setallpalette (struct palettetype *palette);
void setaspectratio (int xasp, int yasp);
void setbkcolor (int color);
void setcolor (int color);
void setcurrentwindow (int window); [WIN]
void setfillpattern (char *upattern, int color);
void setfillstyle (int pattern, int color);
void setgraphmode (int mode);

void setlinestyle (int linestyle, unsigned upattern, int thickness);
void setpalette (int colornum, int color);
void setrgbpalette (int colornum, int red, int green, int blue);
void settextjustify (int horiz, int vert);
void settextstyle (int font, int direction, int charsize);
void setusercharsize (int multx, int divx, int multy, int divy);
void setviewport (int left, int top, int right, int bottom, int clip);
void setvisualpage (int page);
void setwritemode (int mode);
ype* getdefaultpalette (void);