Graphics in C Programming

Graphics programming in C used to drawing various geometrical shapes (rectangle, circle eclipse etc), use of mathematical function in drawing curves, coloring an object with different colors and patterns and simple animation programs like jumping ball and moving cars.

functions of graphics.h

C graphics using graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in Turbo C compiler you can make graphics programs, animations, projects, and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them. Following is a list of functions of graphics.h header file. Every function is discussed with the arguments it needs, its description, possible errors while using that function and a sample C graphics program with its output.

1. First graphics program (Draw a line)

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

void main(void) {
   int gdriver = DETECT, gmode;
   int x1 = 200, y1 = 200;
   int x2 = 300, y2 = 300;
   clrscr();

   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
   line(x1, y1, x2, y2);
   getch();
   closegraph();
}
```

2. Explanation of Code:

The first step in any graphics program is to include graphics.h header file. The graphics.h header file provides access to a simple graphics library that makes it possible to draw lines, rectangles, ovals, arcs, polygons, images, and strings on a graphical window.

The second step is initializing the graphics drivers on the computer using initgraph method of graphics.h library.

```
void initgraph(int *graphicsDriver, int *graphicsMode, char *driverDirectoryPath);
```

It initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode. It also resets or initializes all graphics settings like color, palette, current position etc, to their default values. Below is the description of input parameters of initgraph function.

- **graphicsDriver :** It is a pointer to an integer specifying the graphics driver to be used. It tells the compiler that what graphics driver to use or to automatically detect the drive. In all our programs we will use DETECT macro of graphics.h library that instruct compiler for auto detection of graphics driver.
- **graphicsMode :** It is a pointer to an integer that specifies the graphics mode to be used. If *gdriver is set to DETECT, then initgraph sets *gmode to the highest resolution available for the detected driver.
- **driverDirectoryPath :** It specifies the directory path where graphics driver files (BGI files) are located. If directory path is not provided, then it will search for driver files in current working directory directory. In all our sample graphics programs, you have to change path of BGI directory accordingly where you Turbo C++ compiler is installed.

We have declared variables so that we can keep track of starting and ending point.

```
int x1=200, y1=200;

int x2=300, y2=300;
```

No, We need to pass just 4 parameters to the line function.

```
line(x1,y1,x2,y2);
```

line Function Draws Line From (x1,y1) to (x2,y2) .

```
Syntax : line(x1,y1,x2,y2);
```

Parameter Explanation

- x1 - X Co-ordinate of First Point
- y1 - Y Co-ordinate of First Point
- x2 - X Co-ordinate of Second Point
- y2 - Y Co-ordinate of Second Point

At the end of our graphics program, we have to unloads the graphics drivers and sets the screen back to text mode by calling closegraph function.

3. Colors in C Graphics Programming

There are 16 colors declared in graphics.h header file. We use colors to set the current drawing color, change the color of background, change the color of text, to color a closed shape etc (Foreground and Background Color). To specify a color, we can either use color constants like setcolor(RED), or their corresponding integer codes like setcolor(4). Below is the color code in increasing order.

CONSTANT	VALUE	BACKGROUND?	FOREGROUND?
BLACK	0	Yes	Yes
BLUE	1	Yes	Yes
GREEN	2	Yes	Yes
CYAN	3	Yes	Yes
RED	4	Yes	Yes
MAGENTA	5	Yes	Yes
BROWN	6	Yes	Yes
LIGHTGRAY	7	Yes	Yes
DARKGRAY	8	NO	Yes
LIGHTBLUE	9	NO	Yes
LIGHTGREEN	10	NO	Yes
LIGHTCYAN	11	NO	Yes
LIGHTRED	12	NO	Yes
LIGHTMAGENTA	13	NO	Yes
YELLOW	14	NO	Yes
WHITE	15	NO	Yes
BLINK	128	NO	*

PROGRAM

```
#include <graphics.h>

#include <conio.h>


int main() {

   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\TC\\BGI");

   bar(120, 120, 250, 250);

   getch();

   closegraph();

   return 0;

}
```

Background color

```
#include<graphics.h> /* header file */
#include<conio.h>

main()
{
 /* the following two lines are the syntax for writing a particular
 program in graphics. It's explanation is given after the program.*/

   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\TC\\BGI");


   setbkcolor (GREEN);

   getch();
   closegraph();
   return 0;
}
```

Output

bar()

Syntax

void bar(int left, int top, int right, int bottom);

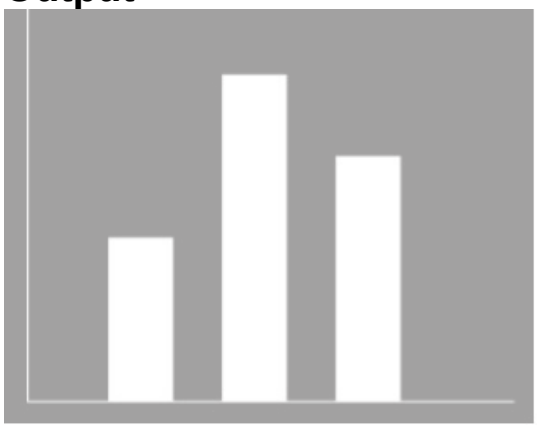
The bar() function is used to draw a bar (of bar graph) which is a 2-dimensional figure. It is filled rectangular figure. The function takes four arguments that are the coordinates of (X, Y) coordinates of the top-left corner of the bar {left and top } and (X, Y) coordinates of the bottom-right corner of the bar {right and bottom}.

PROGRAM

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


int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   bar(120, 120, 250, 250);
   getch();
   closegraph();
   return 0;
}
```

Output



What is BGI?

Borland Graphics Interface (BGI) is a graphics library that is bundled with several Borland compilers for the DOS operating systems since 1987.

Write a Program to draw basic graphics construction like line, circle, arc, ellipse and rectangle.

```
#include<graphics.h>
#include<conio.h>
void main()
{
    intgd=DETECT,gm;
    initgraph (&gd,&gm,"c:\\tc\\bgi");
    setbkcolor(GREEN);
    printf("\t\t\t\n\nLINE");
    line(50,40,190,40);
    printf("\t\t\n\n\n\nRECTANGLE");
    rectangle(125,115,215,165);
    printf("\t\t\t\n\n\n\n\n\n\nARC");
    arc(120,200,180,0,30);
    printf("\t\n\n\n\nCIRCLE");
    circle(120,270,30);
    printf("\t\n\n\n\nECLIPSE");
    ellipse(120,350,0,360,30,20);
    getch();
}
```

Output: