Appendix VI

Graphics Programing Using C

INTRODUCTION

C is mostly favored by programmers for systems programming and developing embedded and real-time applications. However, C also possesses graphics programming capabilities. The header file *graphics.h* comprises of functions that can be used to develop graphical shapes, animations, etc.

In this appendix, we will start with graphics programming using C.

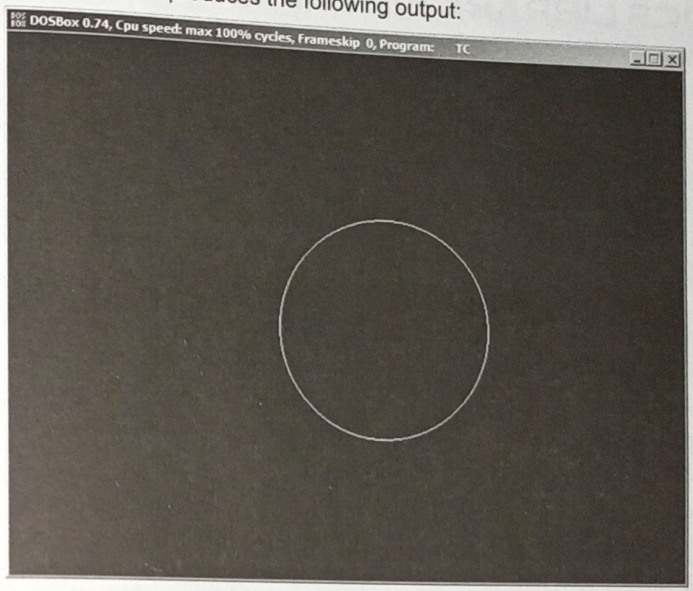
SAMPLE PROGRAM 1: DRAWING A CIRCLE

Consider a simple graphics program given in Figure VI.1.

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

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

  circle(350,250,100);

  getch();
  closegraph();
}
```

Fig. VI.1 *A program to draw a circle*

The program when executed produces the following output:



Let us analyze the program instructions to see how the above output was produced. The program begins with file inclusion directives for including the following two header files:

- **graphics.h**: It comprises of graphics library functions which are used for drawing graphics on the output screen.
- **conio.h**: It comprises of functions for performing console input/output operations.

The main function begins with the following instructions for initializing the graphics mode:

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

Here, *gd* and *gm* are declared as integer variables representing graphics driver and graphics mode respectively. *gd* is assigned *DETECT* value, which is a macro defined in the *graphics.h* header file. The *initgraph* function is subsequently called to initialize the graphics i.e. choose an appropriate graphics driver and set the maximum screen resolution possible. The arguments of *initgraph* function include the following:

- Address of *gd* variable
- Address of *gm* variable
- Directory path of BGI files (the path may vary as per the location of C compiler)

Once the graphics driver and mode are set, the environment is ready for you to draw graphical shapes using functions in the graphics library. In the above program, *circle* function was called to draw a circle on the output screen. The first two arguments of the *circle* function are the x and y coordinates representing the center of the circle. The third argument represents the radius of the circle.

Once the graphical shape has been drawn, it is important to restore the graphics system. The *closegraph* function does just that by closing the graphics mode and restoring it to the mode it was when *initgraph* function was called.

Note *initgraph* and *closegraph* functions are used in all programs where graphical shapes are drawn by calling the graphics library functions.

GRAPHICS LIBRARY FUNCTIONS

Table VI.1 lists some of the key graphics functions included in header file *graphics.h*.

Table. VI.1 *Graphics functions*

Function	Purpose	Format
arc	Draws an arc	*void arc(int x, int y, int stangle, int endangle, int radius);* – *x, y* are the coordinates of the center of the arc – *stangle* and *endangle* are the respective starting and ending angles of the arc – *radius* is the radius of the arc
bar	Draws a 2-dimensional, filled-in, rectangular bar	*void bar(int left, int top, int right, int bottom);* – *left* is the X coordinate of the top left corner of the bar – *top* is the Y coordinate of the top left corner of the bar – *right* is the X coordinate of the bottom right corner of the bar – *bottom* is the Y coordinate of the bottom right corner of the bar
line	Draws a straight line	*void line(int x1, int y1, int x2, int y2);* – *x1, y1* are the coordinates of the starting point of the line – *x2, y2* are the coordinates of the ending point of the line
rectangle	Draws a rectangle	void rectangle(int left, int top, int right, int bottom); – *left* is the X coordinate of the top left corner of the rectangle – *top* is the Y coordinate of the top left corner of the rectangle – *right* is the X coordinate of the bottom right corner of the rectangle – *bottom* is the Y coordinate of the bottom right corner of the rectangle
getcolor	Returns integer code of current drawing color	*int getcolor();*
setcolor	Changes the current drawing color to the specified color code	*void setcolor(int color);* – *color* is the integer value representing color code
getbkcolor	Returns integer code of current background color	*int getbkcolor();*
setbkcolor	Changes the current background color to the specified color code	*void setcolor(int color);* – *color* is the integer value representing color code

Function	Purpose	Format
getx	Returns the X coordinate of current cursor position	*int getx();*
gety	Returns the Y coordinate of current cursor position	*int gety();*
getmaxx	Returns the maximum X coordinate value for the current graphics mode and driver	*int getmaxx();*
getmaxy	Returns the maximum Y coordinate value for the current graphics mode and driver	*int getmaxy();*
getpixel	Returns the color code of the pixel present at specific coordinate location	*int getpixel(int x, int y);* – *x, y* are the coordinate values
putpixel	Sets the color code of the pixel present at specific coordinate location	*int putpixel(int x, int y, int color);* – *x, y* are the coordinate values – *color* is the color code for the pixel

EXAMPLE PROGRAMS

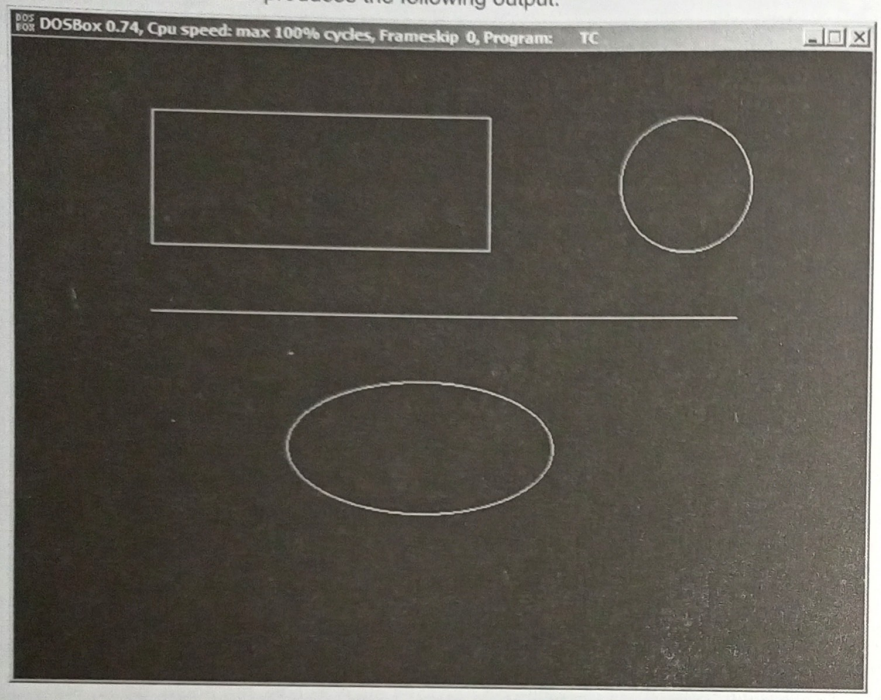
Program for Drawing Basic Shapes

Figure VI.2 shows the program for drawing basic shapes such as line, rectangle, circle, etc.

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

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

   circle(500,100,50);
   rectangle(100,50,350,150);
   ellipse(300, 300,0,360, 100,50);
   line(100,200,540,200);

   getch();
   closegraph();
}
```

Fig. VI.2 *Program for drawing basic shapes*

The program when executed produces the following output:



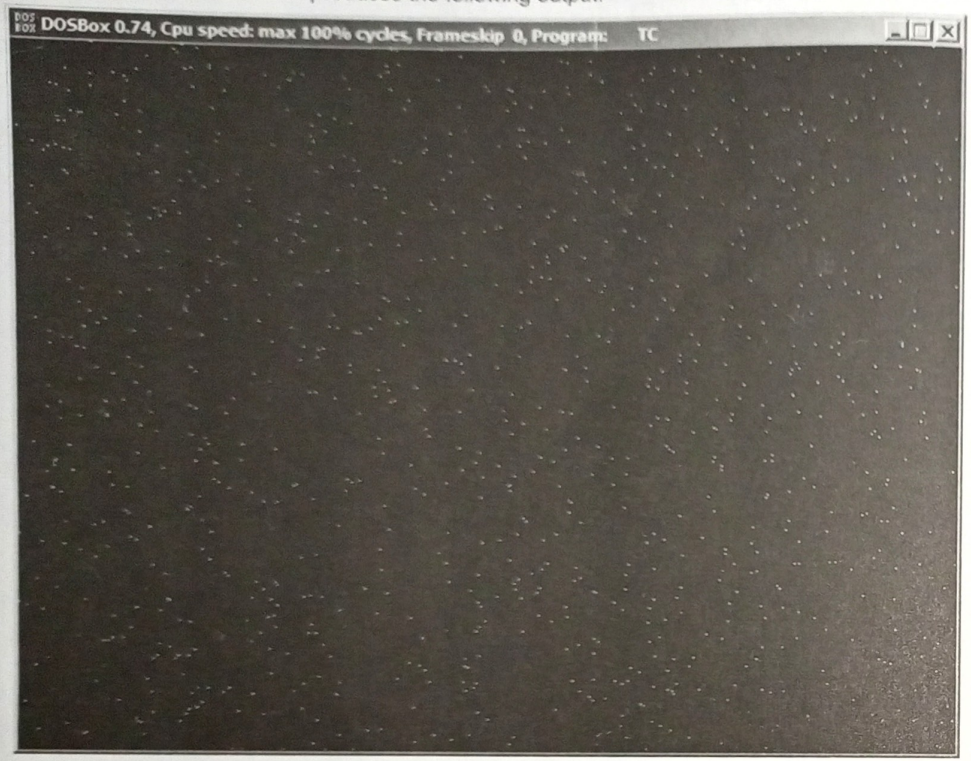
Program for Generating Graphics Effects

Figure VI.3 shows the program for depicting space filled with stars. The program uses *random()* function (part of *stdlib.h* header file) for generating random numbers.

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

void main()
{
   int x,y;
   int i=0;
   int gd=DETECT, gm;
   initgraph(&gd, &gm, "C:\Turbo\TC\BGI" );

   while(i<1000)
   {
   x=rand()%644;
   y=rand()%479;
   putpixel(x,y,15);
   setcolor(random(16));
   i++;
   }

   getch();
   closegraph();
}
```

Fig. VI.3 *Program for depicting space filled with stars*

The program when executed produces the following output:



Program for Drawing Lines of Different Colors

Figure VI.4 shows the program for drawing straight lines in different colors.

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

void main()
{
   int gd=DETECT, gm;
   int i,x;
   initgraph(&gd, &gm, "");

   x=0;
   for(i=0;i<=15;i++)
   {
   setcolor(i);
   line(100,x,540,x);
   x=x+25;
   }

   getch();
   closegraph();
}
```

Fig. VI.4 *Program for drawing lines in different colors*

The program when executed produces the following output: