To start with graphics programming, Turbo C is a good choice. Even though DOS has its own limitations, it is having a large number of useful functions and is easy to program. To implement graphics algorithms, to give graphical display of statistics, to view signals from any source, we can use Turbo C/C++ graphics.

There are two ways to view the display screen in Turbo C graphics model:

- The Text Mode
- The Graphics Mode.

The Text Mode

In the Text Mode, the entire screen is viewed as a grid of cells, usually 25 rows by 80 columns. Each cell can hold a character with certain foreground and background colors (if the monitor is capable of displaying colors). In text modes, a location on the screen is expressed in terms of rows and columns with the upper left corner corresponding to (1,1), the column numbers increasing from left to right and the row numbers increasing vertically downwards.

The Graphics Mode

In the Graphics Mode, the screen is seen as a matrix of pixels, each capable of displaying one or more color. The Turbo C Graphics coordinate system has its origin at the upper left hand corner of the physical screen with the x-axis positive to the right and the y-axis positive going downwards.

Turbo C has a good collection of graphics libraries. If you know the basics of C, you can easily learn graphics programming.

To run the program, you need ***graphics.h*** header file, ***graphics.lib*** library file and Graphics driver (**BGI** file) in the program folder. The **B**orland **G**raphics **I**nterface, also known as BGI, is a graphics library bundled with several Borland compilers for the DOS operating systems since 1987.

These files are part of Turbo C package. In all our programs we used 640x480 VGA monitor. So all the programs are according to that specification. You need to make necessary changes to your programs according to your screen resolution. For VGA monitor, graphics driver used is EGAVGA.BGI.

Graphics mode Initialization

First of all we have to call the ***initgraph()*** function that will initialize the graphics mode on the computer. ***initigraph()*** has the following prototype.

Syntax:

```
void initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver);
```

1) initgraph

It initializes the graphics system by loading the graphics driver from disk (or validating a registered driver) then putting the system into graphics mode. **initgraph()** also resets all graphics settings (color, palette, current position, viewport, etc.) to their defaults, then resets graphresult to 0.

2) graphdriver

Integer that specifies the graphics driver to be used. You can give graphdriver a value using a constant of the graphics_drivers enumeration type which is listed in ***graphics.h***. Normally we use value as "0" (requests auto-detect).

3) graphmode

Integer that specifies the initial graphics mode (unless *graphdriver = DETECT). If graphdriver = DETECT, initgraph sets graphmode to the highest resolution available for the detected driver.

4) pathtodriver

Specifies the directory path where initgraph looks for graphics drivers (*.BGI) first. Like as (CGA.BGI or ATT.BGI or HERC.GBI or EGAVGA.BGI)

Drawing a Circle:

The circle command takes a X coordinate which means Vertical axis and Y coordinate which means Horizontal axis. And the last one is the radius of the circle.

Syntax:

circle(x coordinate , y coordinate , radius);

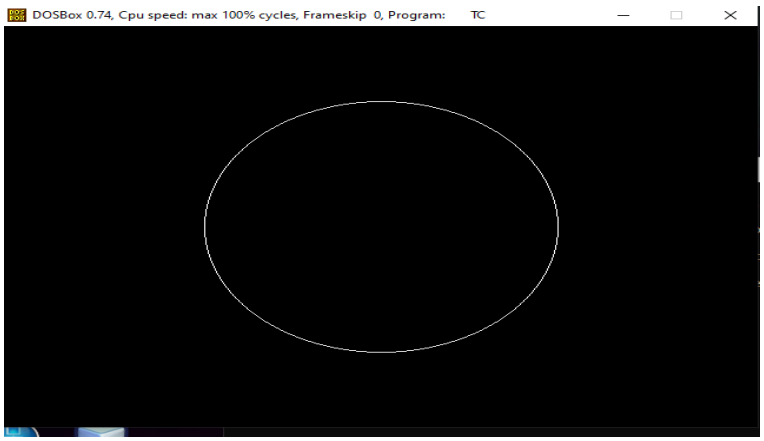
Example

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

void main()
{
  int gd=DETECT, gm;

  initgraph(&gd, &gm, "c:\\tc\\bgi " ); // initialization of graphic mode
  circle(320,240,150);

  getch();
  closegraph();  // Restore original screen mode
}
```



This program draws a circle in the current drawing color with its center at (320,240) and the radius (150) given by radius.

Below is the descriptions of some graphics functions used in programs.

Function	Description
initgraph	It initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode.
getmaxx	It returns the maximum X coordinate in current graphics mode and driver.
getmaxy	It returns the maximum Y coordinate in current graphics mode and driver.
outtextxy	It displays a string at a particular point (x,y) on screen. (x,y,"any text")
circle	It draws a circle with radius **r** and center at (x, y).
rectangle	It draws a rectangle on screen. It takes the coordinates of top left and bottom right corners. **(x1,y1,x2,y2)**
setcolor	It changes the current drawing color. Default color is white. Each color is assigned a number, like BLACK is 0 ,Blue 1, Green 2, Cyan 3 and RED is 4 and so on. Here we are using color constants defined inside ***graphics.h*** header file.
setbkcolor	The **setbkbolor** function sets the current background color to the specified color value, or to the nearest physical color if the device cannot represent the specified color value.
setfillstyle	It sets the current fill pattern and fill color.
floodfill	It is used to fill a closed area with current fill pattern and fill color. It takes any point inside closed area and color of the boundary as input.
closegraph	It unloads the graphics drivers and sets the screen back to text mode.

Let us study more about shapes latter. Here is some idea about colors. There are 16 colors declared in ***graphics.h*** as listed below.

Color Name	Number
BLACK:	0
BLUE	1
GREEN:	2
CYAN:	3
RED:	4
MAGENTA:	5
BROWN:	6
LIGHTGRAY:	7
DARKGRAY	8
LIGHTBLUE:	9
LIGHTGREEN:	10
LIGHTCYAN:	11
LIGHTRED	12
LIGHTMAGENTA:	13
YELLOW:	14
WHITE:	15

To use these colors, use functions setcolor(), setbkcolor() and setfillstyle(). setcolor() function sets the current drawing color. If we use setcolor(RED); and draw any shape, line or text after that, the drawing will be in red color. You can either use color as defined above or number like setcolor(4);. setbkcolor() sets background color for drawing. setfillstyle() sets fill pattern and fill colors. After calling setfillstyle(), if we use functions like floodfill(), fillpoly(), bar etc, shpes will be filled with fill color and pattern set using setfillstyle(). These function declarations are as follows.

setcolor Declaration:

setcolor sets the current drawing color to color, which can range from 0 to 15.

Syntax

```
void far setcolor(int color);
```

setbkcolor Declaration:

setbkcolor sets the background to the color specified by color.

Syntax

```
void far setbkcolor(int color);
```

setfillstyle Declaration:

setfillstyle sets the current fill pattern and fill color.

Syntax

```
void far setfillstyle(int pattern, int color);
```

The parameter pattern in setfillstyle() is as follows:

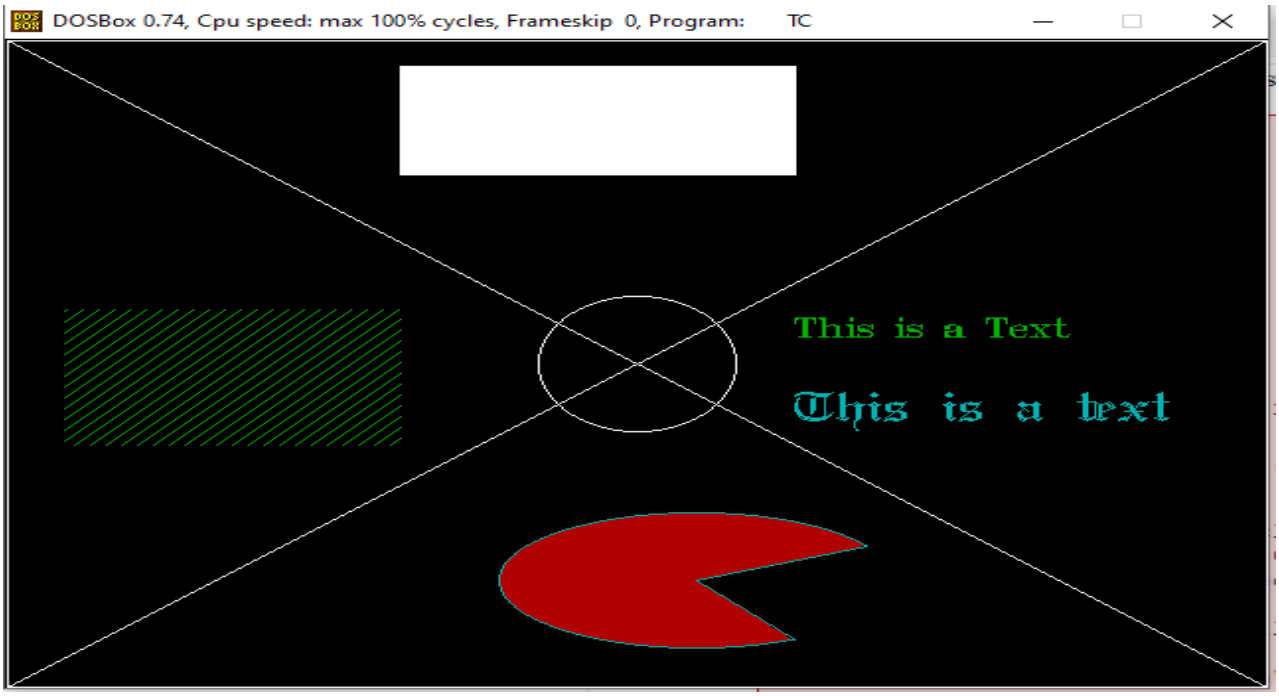
Names	Value	Means Fill With pattern
EMPTY_FILL	0	Background color
SOLID_FILL	1	Solid fill
LINE_FILL	2	---
LTSLASH_FILL	3	///
SLASH_FILL	4	///, thick lines
BKSLASH_FILL	5	\\\, thick lines
LTBKSLASH_FILL	6	\\\
HATCH_FILL	7	Light hatch
XHATCH_FILL	8	Heavy crosshatch
INTERLEAVE_FILL	9	Interleaving lines
WIDE_DOT_FILL	10	Widely spaced dots
CLOSE_DOT_FILL	11	Closely spaced dots
USER_FILL	12	User-defined fill pattern

How to use these graphic functions?

Here are a few sample programs and examples of using these graphic functions to draw some shapes and images in Turbo C/C++.

Some Shapes:

These are command takes an X coordinate which means Horizontal axis and Y coordinate which means Vertical axis.

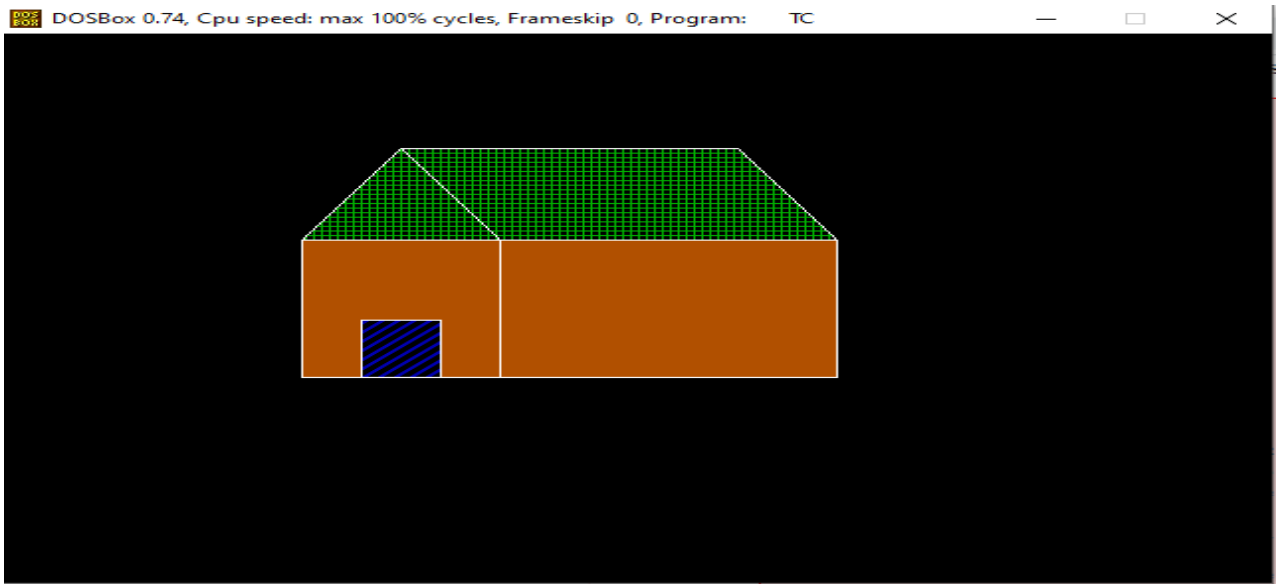


```
#include<graphics.h>
#include<conio.h>

void main()
{
   int gd=DETECT, gm;

   initgraph(&gd, &gm, "c:\\tc\\bgi " );
   rectangle(1,1,639,479);
   circle(320,240,50);
   line(1,1,639,479);
   line(639,1,1,479);
   bar(200,20,400,100);
   setfillstyle(3,2);
   bar(30,200,200,300);
   setcolor(2);
   settextstyle(1,0,2);
   outtextxy(400,200,"This is a Text");
   settextstyle(4,0,4);
   setcolor(3);
   outtextxy(400,250,"This is a text");
   setfillstyle(SOLID_FILL,RED);
   sector(350,400,30,300,100,50);
   getch();
   closegraph();
}
```

Example: (C program to draw a house and color it using Turbo C/C++ graphics)



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

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

    /* Draw Hut */
  setcolor(WHITE);
  rectangle(150,180,250,300);
  rectangle(250,180,420,300);
  rectangle(180,250,220,300);

   line(200,100,150,180);
   line(200,100,250,180);
   line(200,100,370,100);
   line(370,100,420,180);

   /* Fill colours */
 setfillstyle(SOLID_FILL, BROWN);
 floodfill(152, 182, WHITE);
 floodfill(252, 182, WHITE);
 setfillstyle(SLASH_FILL, BLUE);
 floodfill(182, 252, WHITE);
 setfillstyle(HATCH_FILL, GREEN);
 floodfill(200, 105, WHITE);
 floodfill(210, 105, WHITE);

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

Exercise:

Theory Question:

1) How many types of view screen Mode in Turbo C/C++.
2) How many rows and column in text screen mode in Turbo C/C++.
3) Define five graphics functions with syntax.
4) What is difference between the circle and ellipse?

Practical's Questions

1) Create a smiley emoticon using graphics functions in Turbo C/C++ with Printout.
2) Create a colored rectangle with your name written inside it in Turbo C/C++ with printout.
3) Create a smiley emoticon using graphics functions in Turbo C/C++. With printout.
4) Make your own creative image of anything using graphic functions in C/C++.

Objective MCQ's:

1) Computer screen Mode has _____ types.
   a) 1
   b) 2
   c) 3
   d) 4
2) VGA screen maximum resolution has _____
   a) 320x200
   b) 320x300
   c) 640x480
   d) 640,350
3) BGI stand for_____
   a) Bigger graphics interface
   b) Beginner graphics interface
   c) Borland graphics interlink
   d) Borland graphics interface
4) Red color number is _____
   a) 1
   b) 2
   c) 3
   d) 4
5) Which function for use circle.
   a) Rectangle
   b) Ellipse
   c) Circle
   d) Line