Chapter 15

Graphics Programming

Dr. Niroj Kumar Pani

nirojpani@gmail.com

Department of Computer Science Engineering & Applications

Indira Gandhi Institute of Technology

Sarang, Odisha

Chapter Outline…

- Introduction
- The 1st Graphics Program (Opening the Graphics Window)
- Working with Graphics
- Programming Examples
- Assignments

Introduction

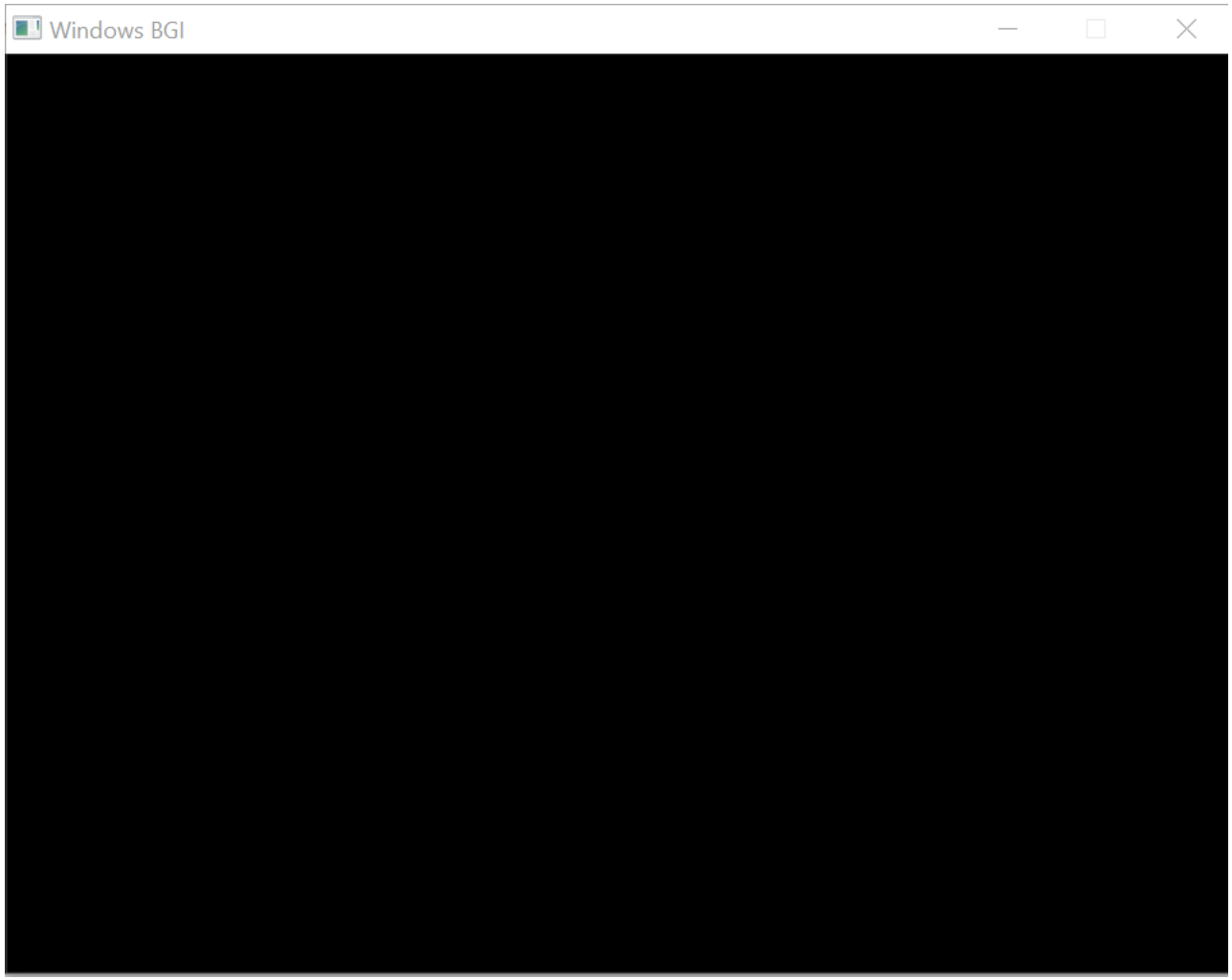
- In C, graphics programming is normally used to
  - Draw various geometrical shapes (e.g., rectangle, circle, eclipse etc.).
  - Color objects with different colors and patterns.
  - Do simple animation programs (e.g., jumping ball, moving cars etc.).

The 1st Graphics Program (Opening the Graphics Window)

- **Let's understand the basics of graphics programming through a simple program that just opens the graphics window (a blank window, no drawings).**

```
1   /* PR15_1.c: Program to open the graphics window. *

2   # include <graphics.h>
3   # include <stdio.h>
4   # include <conio.h>

5    int main(void)
6    {
7        int gDriver = DETECT, gMode;

8        /* Initialize (enter) graphics mode (from text mode). */
9        initgraph(&gDriver, &gMode, "");

10       /* Wait for a key to close graphics mode. */
11       getch();

12       /* Close graphics mode (enter text mode). */
13       closegraph();

14       return 0;
15   }
```

Output



Explanation (of the Code in Red Ink)

- **Line No. 2: The first step in any graphics program is to include the header file "`graphics.h`". It contains definitions of constants and prototypes of standard library functions required for graphics programming.**
- **Line Nos. 7 and 9: The second step is switching the system from text mode to graphics mode. This is done by calling the standard library function "`initghaph()`".**

[NOTE]: There are two modes in which the system can work: text and graphics mode. In text mode, only text display is possible (in terms of ASCII), while in graphics mode any text / figures can be displayed. The default mode of the system is the text mode. Therefore, before doing any graphics program we have to switch from the text mode to graphics mode and after doing the graphics program we have to switch back from the graphics mode to the text mode.

initgraph():

- It initializes the graphics system by loading the graphics driver and then switches the system to graphics mode.
- It also resets all graphics settings like color, palette, current position etc., to their default values.
- **Syntax:**

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

`initgraph` takes three parameters.

- **graphicsDriver:**
  - It's a pointer to an integer that specifies what graphics driver to be used, or it is to be auto detected (preferred).
  - To enable auto detection of the graphics driver, it is set to the `DETECT` macro (of `graphics.h` library), i.e., `"int gDriver = DETECT;"`.

- **`graphicsMode`:**
  - It's a pointer to an integer that specifies the graphics mode to be used.
  - If `gDriver` = `DETECT`, then `initgraph` automatically sets `gMode` to the highest resolution available for the detected driver.
- **`graphicsDriverPath`:**
  - It's a pointer to a string that specifies the directory path where `initgraph` looks for the graphics driver files (BGI files).

    (NOTE: Normally, the BGI files are in "c:\\turboc3\\bgi")
  - If the files are not there at the specified path, or the directory path is not provided (like we have done), then the driver files are searched in current working directory.

- **Line No. 13:** The **last step** in a graphics program is to **close the graphics mode** (by unloading the graphics driver) and set the screen back to text mode. This is done by calling the standard library function "**`closeghaph()`**".
  - **Syntax:**

```
void closegraph();
```

Working with Graphics

- **Working with graphics (e.g., writing text, drawing shapes, coloring etc.) is accomplished through the standard library functions available in "`graphics.h`". Some important functions are listed in Table 15.1.**

[Table 15.1: Standard Graphics Library Functions]

Function Name	Syntax / Example	Operation
`cleardevice()`	`cleardevice();`	Clears the screen in graphic mode. (Similar to `clrscr()` in text mode.)

`outtextxy()`	`outtextxy(int x, int y, char *textstring)`
	e.g.,
	`outtextxy(5, 10, "Hi");`
		Displays a line of text at position (x, y).

[Cont.]

Function Name	Syntax / Example	Operation

`getpixel()`	`getpixel(int x, int y);`
	e.g.,
	`getpixel(250, 350);`
		Gives location of point at position (x, y).

`putpixel()`	`putpixel(int x, int y, color);`
	e.g.,
	`putpixel(50, 100, RED)`
		Plots a point / pixel at position (x, y) with specified color (color specifications are given in Table 15.1).

`moveto()`	`moveto(int x, int y);`
	e.g.,
	`moveto(100, 150);`
		Moves the current point to new position (x, y).

`getmaxx()` `getmaxy()`	`x1= getmaxx();` `y1= getmaxy();`	Gives maximum values of x and y coordinates.

`line()`	`line(int x0, int y0, int x1, int y1);`
	e.g.,
	`line(10, 20, 50, 100);`
		Draws a line from position (x0, y0) to position (x1, y1).

`lineto()`	`lineto(int x, int y);`
	e.g.,
	`lineto(50, 150);`
		Draws a line in from current position to new position (x, y).

[Cont.]

Function Name	Syntax / Example	Operation

`circle()`	`circle(int x, int y, int radius);`
	e.g.,
	`circle(200, 100, 75);`
		Draws a circle centered at position (x, y) with given radius.

`arc()`	`arc(int x, int y, int startAangle, int endAngle, int radius );`
	e.g.,
	`arc(100, 100, 0, 90, 30);`
		Draws a circular arc centered at position (x, y) with given radius, start angle, and end angle.

`ellipse()`	`ellipse(int x, int y, int startAngle, int endAngle, int xRadius, int yRadius);`
	e.g.,
	`ellipse(150, 100, 0, 360, 75,50);`
		Draws an ellipse centered at position (x, y) with given start angle, end angle and semi major / minor radius on x and y axis.

[Cont.]

Function Name	Syntax / Example	Operation

`rectangle()`	`rectangle(int left, int top, int right, int bottom);`
	e.g.,
	`rectangle(100, 50, 150, 250);`
		Draws a rectangle. Left and top gives left top corner position of the rectangle and Right & bottom gives right bottom corner position of the rectangle.

`drawpoly()`	`drawpoly(int points, int polypoints);`
	e.g.,
	`int triangle[]={320, 100, 100, 250, 520, 250, 320, 100};`
	`drawpoly(4, triangle);`
		Draws outline of a polygon. The number of points are joined using the given coordinates.

`bar()`	`bar(int x0, int y0, int x1, int y1);`
	e.g.,
	`bar(100, 50, 150, 250);`
		Draws a filled bar with corner points (x0, y0) and (x1, y1)

[Cont.]

Function Name	Syntax / Example	Operation

`setcolor()`	`setcolr(int color);`
	e.g.,
	`setcolor(4);`
	`setcolor(RED);`
		Sets the drawing color of an object as per the specified color value. (Color specifications are given in Table 15.1).

`setfillstyle()`	`void setfillstyle(`
	`int pattern,`
	`int color);`
	e.g.,
	`void setfillstyle(1,`
	`RED);`
		Fills the shape with chosen color and pattern as per the specified color and pattern values. (Color specifications are given in Table 15.1).

[Table 15.2: Color Specifications (There are 16 colors declared in "graphics.h")]

Color Name (as 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

Programming Examples

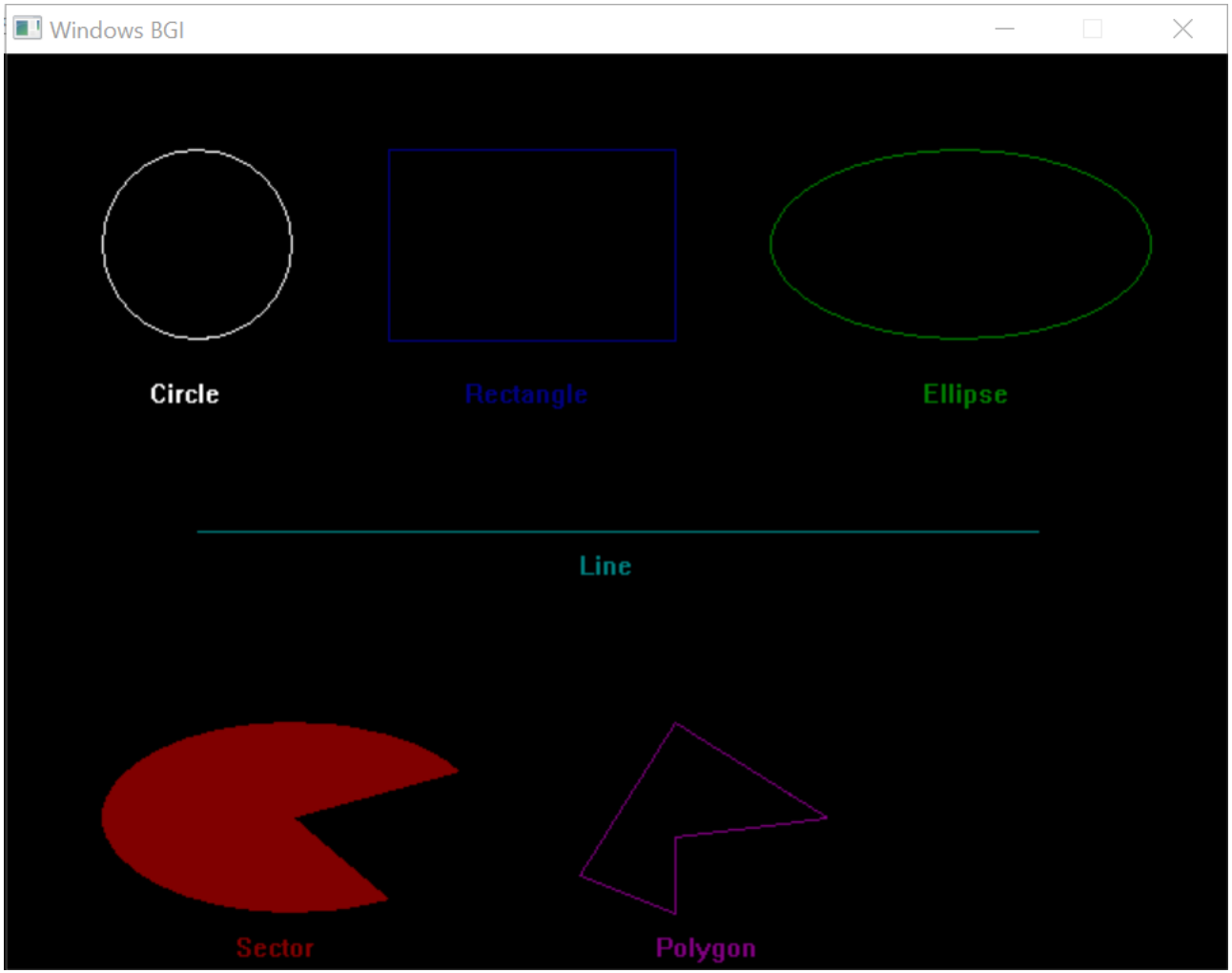
- **Programming Example 1:**

```
/* PR15_2.c: Program to draw some basic shapes with different colors and patterns. */

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

int main(void)
{
    int gDriver = DETECT, gMode;
    int poly[12]={350,450, 350,410, 430,400, 350,350, 300,430,m
    350,450 };

    initgraph(&gDriver, &gMode, "");

    /* Drawing a circle. */
    circle(100,100,50);
    outtextxy(75,170, "Circle");

    /* Drawing a rectangle. */
    setcolor(1); /* Same as setcolor(BLUE); */
    rectangle(200,50,350,150);
    outtextxy(240, 170, "Rectangle");
```

[Cont.]

```
    /* Drawing an ellipse. */
    setcolor(GREEN);
    ellipse(500, 100,0,360, 100,50);
    outtextxy(480, 170, "Ellipse");

    /* Drawing a line. */
    setcolor(3);
    line(100,250,540,250);
    outtextxy(300,260,"Line");

    /* Drawing a sector. */
    setcolor(4);
    setfillstyle(1, 4);
    sector(150, 400, 30, 300, 100,50);
    outtextxy(120, 460, "Sector");

    /* Drawing a polygon. */
    setcolor(5);
    drawpoly(6, poly);
    outtextxy(340, 460, "Polygon");

    getch();
    closegraph();

    return 0;
}
```

Output



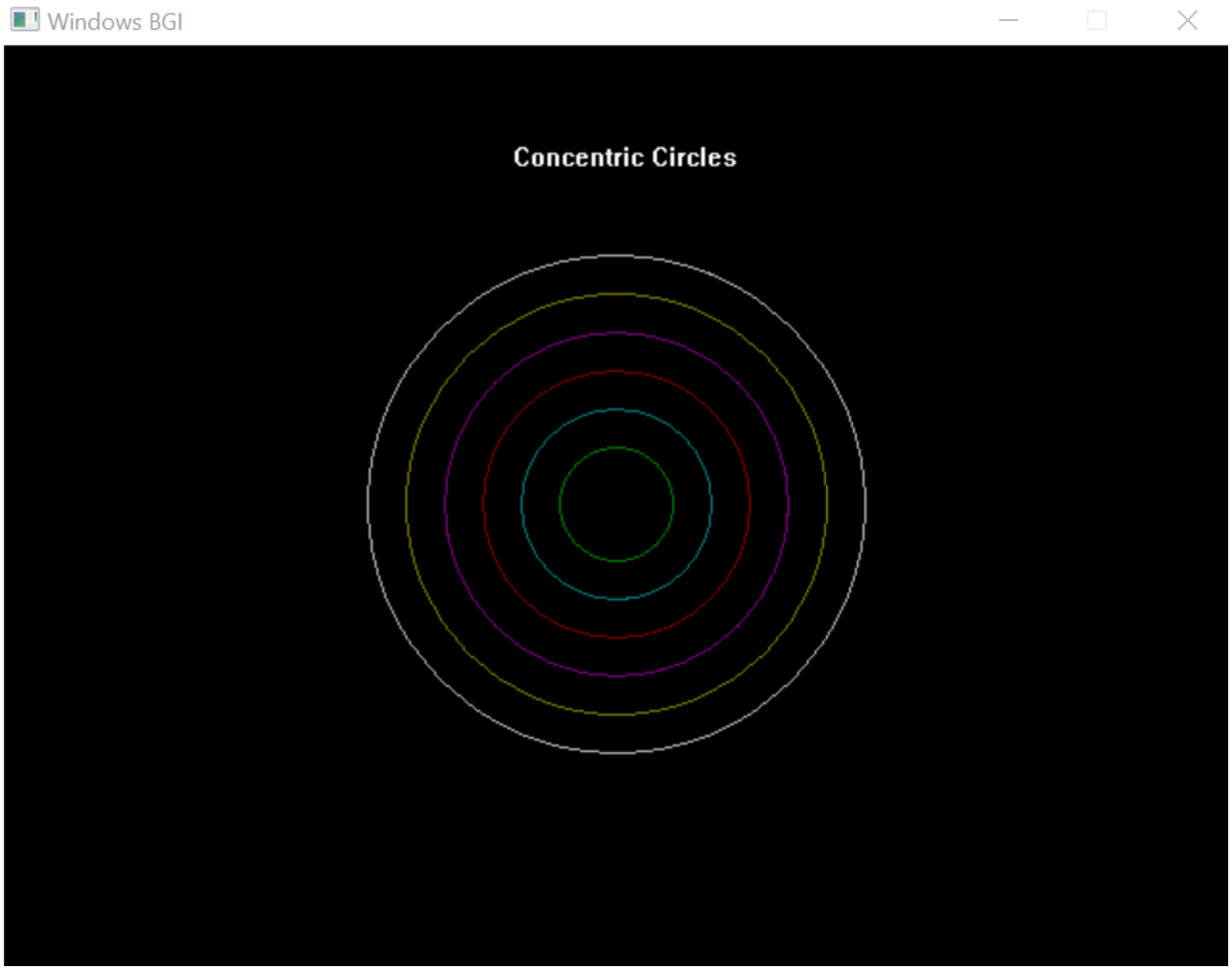
Programming Example 2:

```
/* PR15_2.c: Program to draw six concentric circles. */

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

int main(void)
{
    int gDriver = DETECT, gMode;
    int x ,y, i, radious = 30;

    initgraph(&gDriver, &gMode, "");

    /* Initialize center of circle with center of screen. */
    x = getmaxx()/2;
    y = getmaxy()/2;

    outtextxy(265, 50, "Concentric Circles");
```

[Cont.]

```
    /* Drawing six concentric circles of different colors. */
    for (i=0;i<=5;i++)
    {
        setcolor(i+2);
        circle(x, y, radious);
        radious = radious+20;
    }

    getch();
    closegraph();

    return 0;
}
```

Output



Web References

- https://developerinsider.co/graphics-graphics-h-c-programming/
- https://www.techcrashcourse.com/2015/08/c-graphics-programming-tutorial.html
- https://www.programmingsimplified.com/c-graphics-programming-tutorial

Assignments

Complete the experiments given in "Lab Manual - Section 18"**.**

End of Chapter 15