Graphics

Computer Graphics

- Computer Graphics is one of the most powerful and interesting aspect of computers. There are many things we can do in graphics apart from drawing figures of various shapes.
- All video games, animation, multimedia predominantly works using computer graphics.

Graphics in C

- There is a large number of functions in C which are used for putting pixel on a graphic screen to form lines, shapes and patterns.
- The Default output mode of C language programs is "Text" mode. We have to switch to "Graphic" mode before drawing any graphical shape like line, rectangle, circle etc.

Basic color Function

- textcolor() function
- textbackground() function
- setbkcolor() function
- setcolor() function

textcolor() function

Declaration:

void textcolor(newcolor);

Remarks:

- This function works for functions that produce text-mode output directly to

  the screen (console output functions).
- **textcolor** selects a new character color in text mode.
- This functions does not affect any characters currently on the screen.

textbackground() function

Declaration:

void textbackground(newcolor);

Remarks:

textbackground selects the background color for text mode.

If you use symbolic color constants, the following limitation apply to the background colors you select:

- **You can only select one of the first eight colors (0--7).**

NOTE: **If you use the symbolic color constants, you must include conio.h.**

setcolor() function

Declaration:

void setcolor(color);

Remarks:

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

setbkcolor() function

Declaration:

void setbkcolor(color);

Remarks:

setbkcolor sets the background to the color specified by color.

Example 1.

txtcolor() & textbackground() functions

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
Void main()
{
textcolor(4);          OR          textcolor(4+BLINK)
textbackground(3);
cprintf("NFC-IEFR INSTITUTE ");
getch();
}
```

OUTPUT

NFC-IEFR INSTITUTE

Graphics in C

- There are lot of library functions in C language which are used to draw different drawing shapes.
- For eg.
  - line(x1, y1, x2, y2);
  - putpixel(x, y);
  - circle(xCenter, yCenter, radius);
  - rectangle(x1, y1, x2, y2);
  - ellipse(xCenter, yCenter, start, end, Xradius, Yradius);
  - arc(xCenter, yCenter, start, end, radius);

Graphics in C
Example 2

```
#include<graphics.h>
void main(void)
{
   int dr=DETECT, md;
    initgraph(&dr,&md,”c:\\tc\\bgi”);
    line(0 , 0, 640, 480);
    getch();
   closegraph();
}
```

Dissecting initgraph(…) Function

- The initgraph function is used to switch the output from text mode to graphics mode.
- The initgraph function takes three arguments.

```
initgraph(&dr , &md , ”c:\\tc\\bgi” );
```

graphics Driver Type

Initial graphics mode

Directory path of graphics driver

Graphics Drivers

Constant	Value
DETECT	0 (requests auto detection)
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMONO	5
IBM8514	6
HERCMONO	7
ATT400	8
VGA	9
PC3270	10

Graphics Mode

driver	graphics_modes	Value	Column x Row	Colors
CGA	CGAC0	0	320 x 200	4 colors
	CGAC1	1	320 x 200	4 colors
	CGAC2	2	320 x 200	4 colors
	CGAC3	3	320 x 200	4 colors
	CGAHI	4	640 x 200	2 colors
EGA	EGALO	0	640 x 200	16 colors
	EGAHI	1	640 x 350	16 colors
VGA	VGALO	0	640 x 200	16 colors
	VGAMED	1	640 x 350	16 colors
	VGAHI	2	640 x 480	16 colors

Directory path of graphics driver

- The third argument to initgraph() is the pathname for the graphics drivers. This driver is a file like cga.bgi or egavga.bgi.
- cga.bgi file is used to run programs in CGA modes.
- egavga.bgi file is used to run programs in EGA or VGA modes.
- Other Drivers are available for other display standards such as Hercules and IBM 8514.
- In the current version of Turbo C, these driver files are located in the subdirectory \tc\bgi. So this is the pathname used in the arguments to initgraph().

line() function

- **line draws a line between two specified points**

Syntax:

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

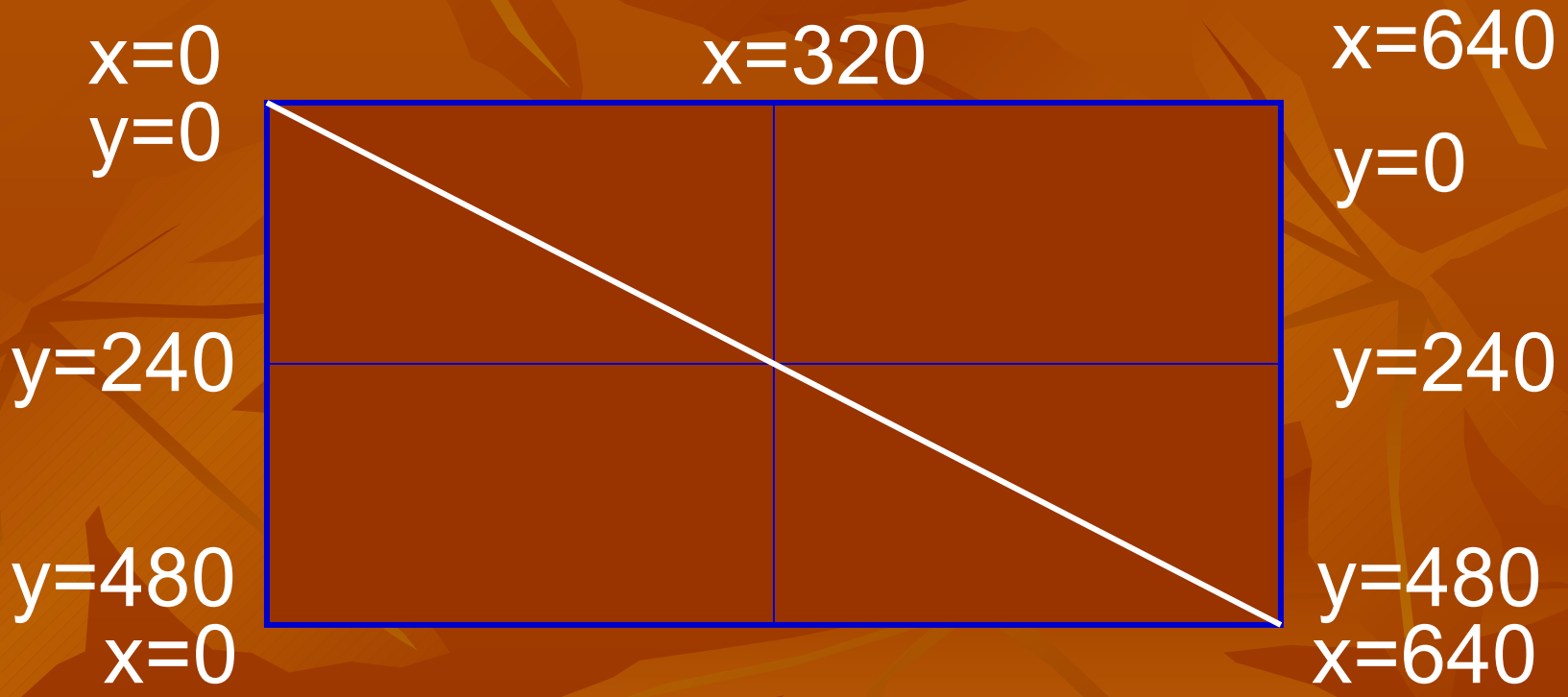
Remarks

line draws a line from (x1, y1) to (x2, y2) using the current color, line style, and thickness.

line() function

- e.g:

  line(0, 0, 640, 480);



setlinestyle() function

Sets the current line style and width or pattern

Syntax:

setlinestyle (linestyle, upattern, thickness);

Remarks:

setlinestyle sets the style for all lines drawn by line, lineto, rectangle, drawpoly

Line Styles

Line Style	Int Value	Pattern
SOLID_LINE	0	
DOTTED_LINE	1	
CENTER_LINE	2	
DASHED_LINE	3	
USERBIT_LINE	4	**User Defined**

upattern, thickness

- U pattern
  - User can define its own pattern.
  - 0 should be used if using predefined pattern, other wise any integer number representing user pattern
- Thickness
  - Thickness of the line in pixels

rectangle() function

Draws a rectangle (graphics mode)

syntax:

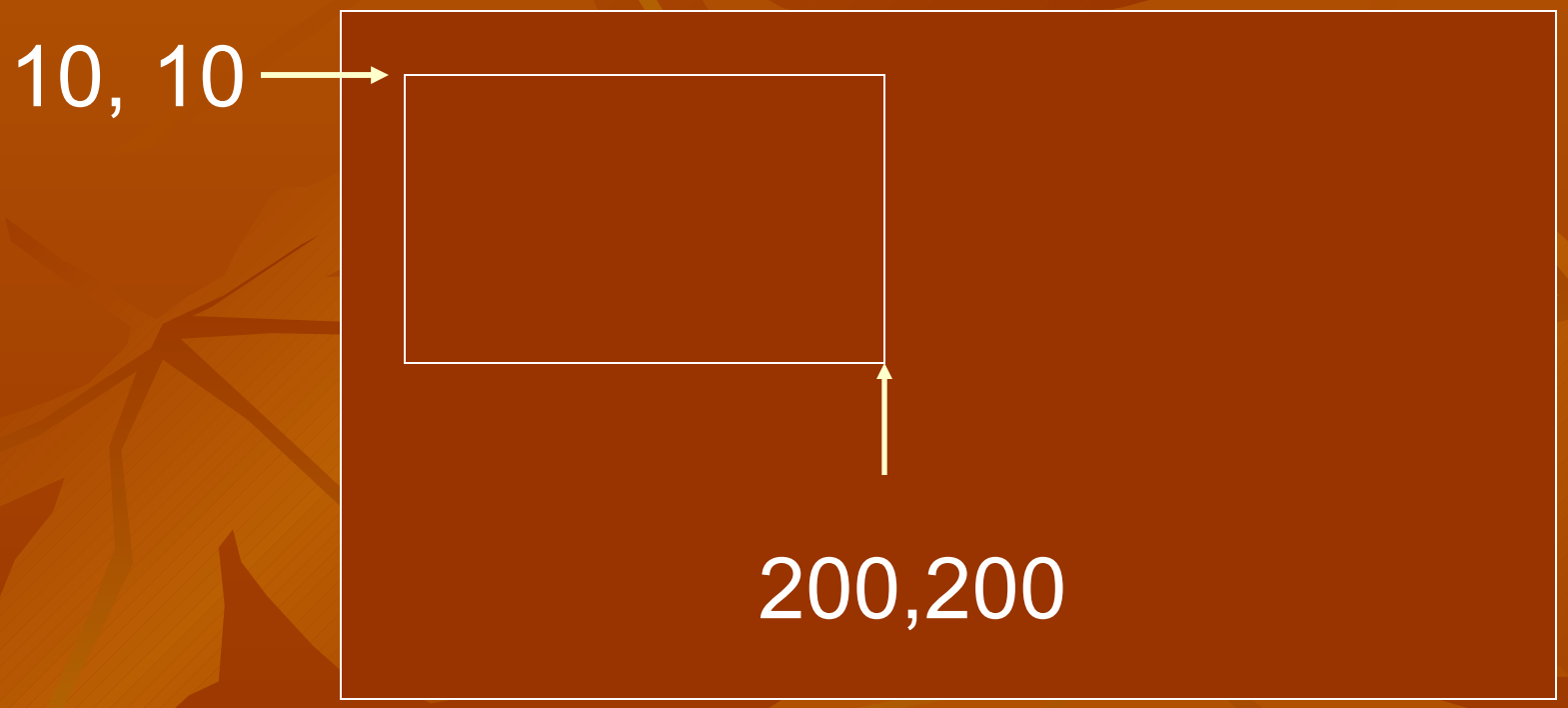
void rectangle(left, top, right, bottom);

Remarks:

- **rectangle draws a rectangle in the current line style, thickness, and drawing color.**
- **(left,top) is the upper left corner of the rectangle, and (right,bottom) is its lower right corner.**

e.g: rectangle() function

rectangle(10, 10, 200,200);



bar( ) function

bar(..) function Draws a bar

- **Syntax:**

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

- **Remarks:**
  - **bar draws a filled-in, rectangular, two-dimensional bar.**
  - **The bar is filled using the current fill pattern and fill color. bar does not outline the bar.**
  - **To draw an outlined two-dimensional bar, use bar3d with depth = 0.**

e.g: bar( ) function

Usage:

bar(10,10,200,200);

bar3d() function

Declaration:

void bar3d(left,top,right, bottom, depth,topflag);

Remarks:

bar3d draws a three-dimensional rectangular bar, then fills it using the current fill pattern and fill color. The three-dimensional outline of the bar is drawn in the current line style and color.

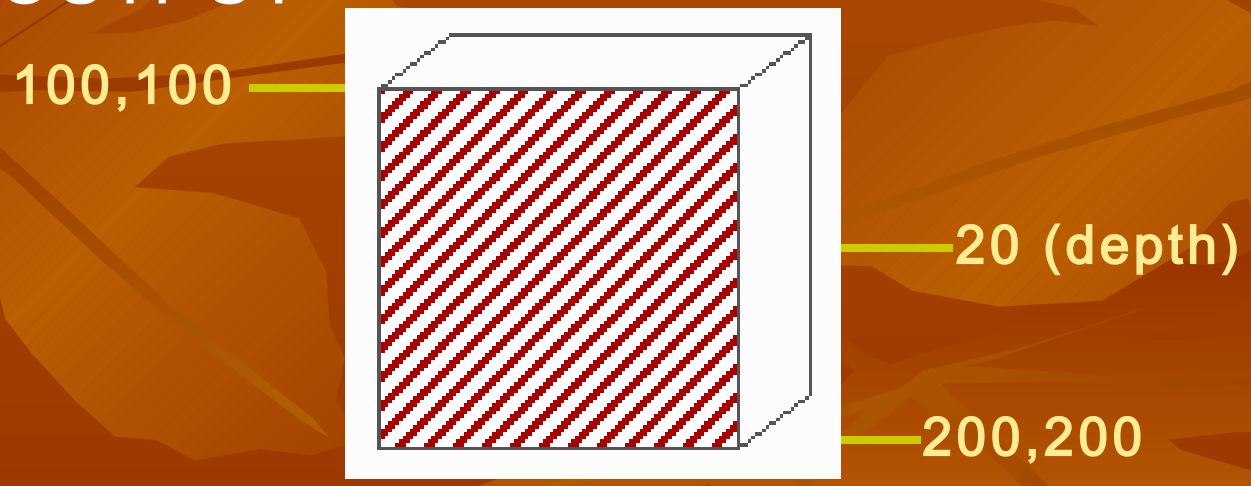
Parameter	What It Is/Does
depth	Bar's depth in pixels
topflag	Governs whether a three-dimensional top is put on the bar
(left, top)	Rectangle's upper left corner
(right, bottom)	Rectangle's lower right corner

eg: bar3d() function

Usage:

```
setfillstyle(4, 4);
bar3d(100, 100, 200, 200, 20, 1);
```

OUTPUT



circle() function

Declaration:

void circle(x,y,radius);

Remarks:

circle draws a circle in the current drawing color.

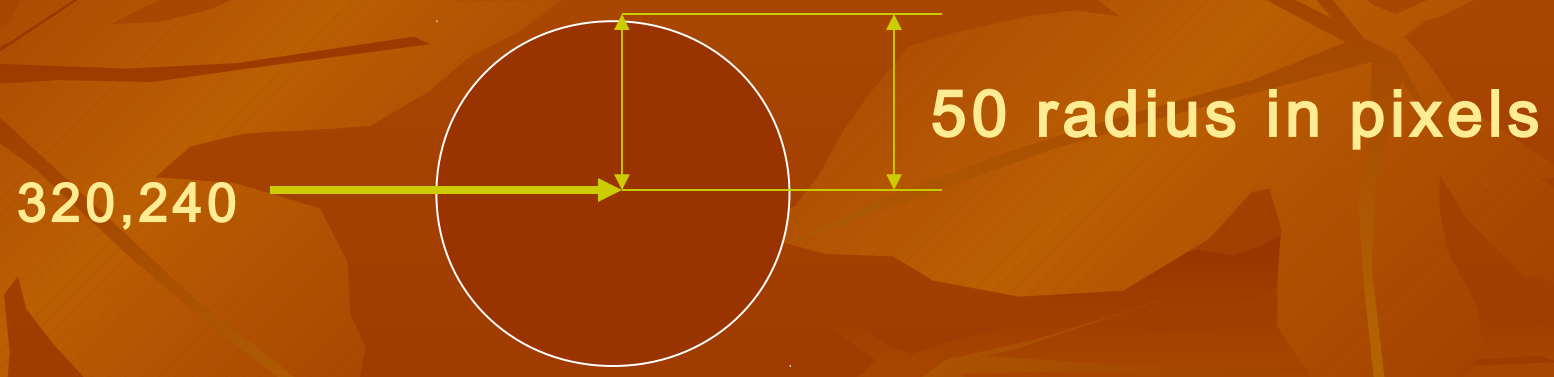
Argument	What It Is/Does
(x,y)	Center point of circle
radius	Radius of circle

eg: circle() function

Usage:

circle(320,240,50);

OUTPUT



arc() function

Declaration:

void arc(x,y,stangle,endangle radius);

Remarks:

arc draws a circular arc in the current drawing color.

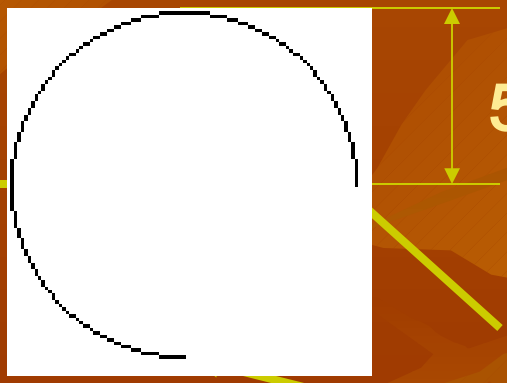
Argument	What It Is/Does
(x,y)	Center point of arc
stangle	Start angle in degrees
endangle	End angle in degrees
radius	Radius of circle

eg: arc() function

Usage:

arc(320, 240, 0, 270, 50);

OUTPUT



ellipese() function

Declaration:

void ellipse(x, y,stangle,endangle,xradius,yradius);

Remarks:

ellipse draws an elliptical arc in the current drawing color.

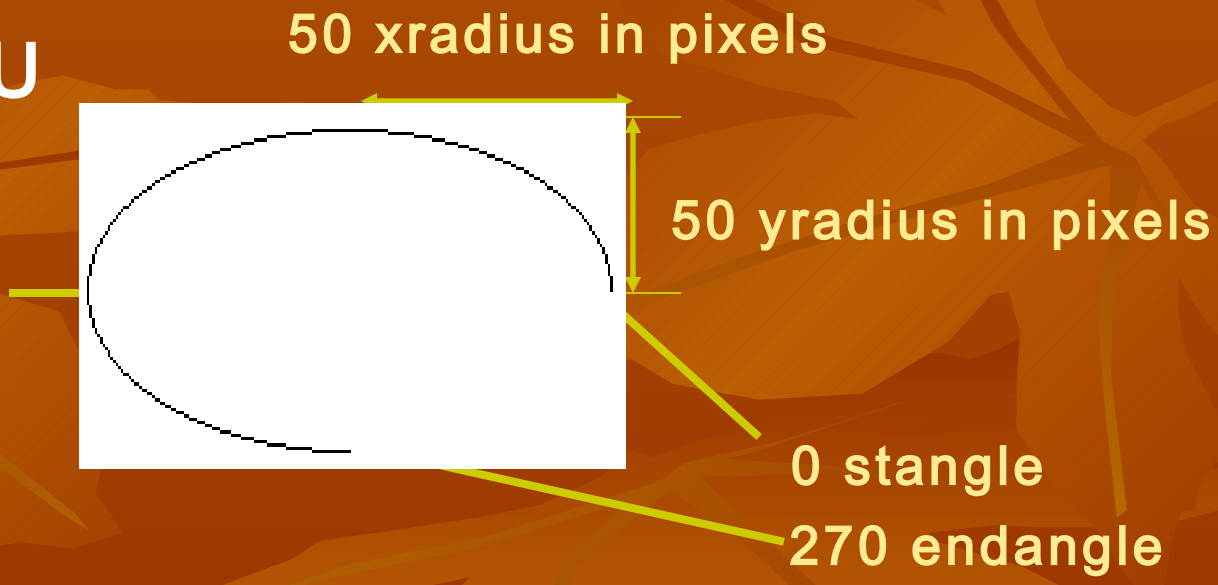
Argument	What It Is/Does
(x,y)	Center point of ellipse
stangle	Start angle in degrees
endangle	End angle in degrees
xradius	H

eg: ellipse() function

Usage:

ellipse(320, 240, 0, 270, 100,50);

OUTPUT



fillellipse() function

Declaration:

void far fillellipse(x, y,xradius, yradius);

Remarks:

fillellipse draws an ellipse, then fills the ellipse with the current fill color and fill pattern.

Argument	What It Is/Does
(x,y)	Center point of ellipse
xradius	Horizontal axis
yradius	Vertical axis

e.g: fillellipse() function

Declaration:

void far fillellipse(x, y, xradius, yradius);

Remarks:

fillellipse draws an ellipse, then fills the ellipse with the current fill color and fill pattern.

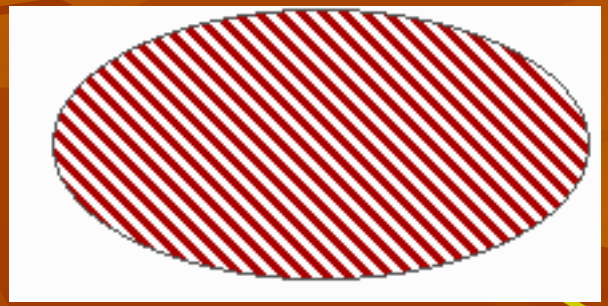
Argument	What It Is/Does
	Center point of ellipse
xradius	Horizontal axis

eg: fillellipse() function

Usage:

```
setfillstyle(5, 4);
fillellipse(320, 240, 0, 270, 100,50);
```

OUTPUT



current fill color and fill pattern

e.g: setfillstyle() function

Declaration:

void setfillstyle(pattern, color);

Remarks:

setfillstyle sets the current fill pattern and fill color.

e.g: setfillstyle() function

Fill patterns

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	
HATCH_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	User Defined

putpixel() function

Declaration:

void putpixel(x, y, color);

Remarks:

putpixel plots a point in the color defined by color at (x,y)

Viewports

- **Viewports** provide a way to restrict to an arbitrary size the area of the screen used for drawing. We can draw an image that would ordinary occupy the entire screen but if a view port is in use, only part of the image will be visible.
- The **View Ports** don't scale the image; that is, the image isn't compressed to fit the view port, Rather, the parts of the image that don't fit in the view port are simply not visible

setviewport() Funciton

Sets the current viewport for graphics output

- **Declaration:**

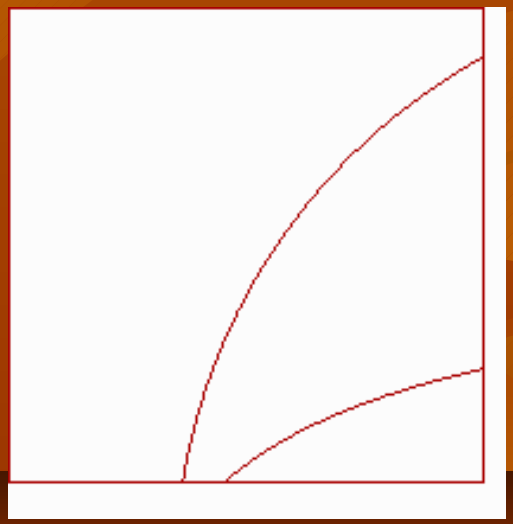
  void far setviewport(left,top,right,bottom,clip);

- **Remarks:**
  - **setviewport establishes a new viewport for graphics output.**
  - **The viewport's corners are given in absolute screen coordinates by (left,top) and (right,bottom).**
  - **The clip argument determines whether drawings are clipped (truncated) at the current viewport boundaries. If clip is non-zero, all drawings will be clipped to the current viewport.**

e.g: setviewport() Function

```
setviewport(0,0,200,200,1);
   rectangle(0,0,200,200);
   circle(320,240,250);
   ellipse(320,240,0,360,250,100);
```

OUTPUT:



clearviewport() Function

- Clear the current viewport.
- eg:

```
circle(320,240,260);
setviewport(0,0,200,200,1);
rectangle(0,0,200,200);
circle(320,240,250);
ellipse(320,240,0,360,250,100);
getch();
clearviewport();
```

Text with Graphics

- There are functions in C language that draw text characters in graphics mode. These functions can be used to mix text and graphics in the same image. These functions also make it possible to change text font and very the size of text.

outtext() function

- outtext displays a string in the viewport (graphics mode)

- **Declaration:**

  void outtext(far *textstring);

- **Remarks:**

  outtext display a text string, using the current justification settings and the current font, direction, and size. outtext outputs textstring at the current position (CP).

outtextxy() Function

- **outtextxy** displays a string at the specified location (graphics mode)
- **Declaration:**

  void outtextxy(x, y, far *textstring);

- **Remarks:**

  outtextxy() display a text string, using the current justification settings and the current font, direction, and size.(CP)

  outtextxy() displays textstring in the viewport at the position (x, y)

settextstyle( ) Function

Sets the current text characteristics

- **Declaration:**

  void settextstyle(font, direction, charsize);

- **Remarks:**
  - **settextstyle() sets the text font, the direction in which text is displayed, and the size of the characters.**
  - **A call to settextstyle() affects all text output by outtext and outtextxy.**

settextstyle( ) Function

- **direction**

  Font directions supported are horizontal text (left to right) and vertical text (rotated 90 degrees counterclockwise).
- The default direction is HORIZ_DIR.

Name	Value	Direction
HORIZ_DIR	0	Left to right
VERT_DIR	1	Bottom to top

settextstyle( ) Function

- Charsize
  - The size of each character can be magnified using the charsize factor.
  - If charsize is non-zero, it can affect bit-mapped or stroked characters.
  - A charsize value of 0 can be used only with stroked fonts.

settextstyle( ) Function

- **Fonts**

  There are currently five fonts available . But it is easy to add other to the systems. These are

Value	Constant	File	Comment
0	DEFAULT_FONT	Built in	Bit-mapped, 8x8
1	TIPLEX-FONT	TRIP.CHR	Stroked (Times Roman style)
2	SMALL_FONT	LITT.char	Stroked (for small lette4rs)
3	SANS_SERIF_FONT	SANS.CHR	Stroked(sans_serif style)
4	GOTHIC_FONT	GOTHIC.CHR	Stroked (gothic style)

The End