KG COLLEGE OF ARTS AND SCIENCE

GRAPHICS & MULTIMEDIA LAB MANUAL

PROGRAM : B.Sc. Information Technology

COURSE OBJECTIVE AND OUTCOME:

COURSE OBJECTIVE:

The main objective of this module is that to introduce the concepts of computer graphics to the students. It starts with an overview of interactive computer graphics, two dimensional systems and mapping. It presents the most important drawing algorithms, two-dimensional transformation; Clipping, filling.

- Students will able to understand the components of a graphics system and become familiar with building approach of graphics system components and algorithms related with them.
- To learn the basic principles of 2- dimensional computer graphics.
- Provide an understanding of mapping from a world coordinates to device coordinates, clipping, and projections.

COURSE OUTCOME:

- Have a basic understanding of the core concepts of computer graphics.
- Have a knowledge and understanding of the structure of an interactive computer graphics system, and the separation of system components.
- Have a knowledge and understanding of 2D geometrical transformations and 2D viewing.
- Have a knowledge and understanding of interaction techniques.

LIST OF PROGRAMS:

1. Write a program to rotate an image.
2. Write a program to drop each word of a sentence one by one from the top.
3. Write a program to drop a line using DDA Algorithm.
4. Write a program to move a car with sound effect.
5. Write a program to bounce a ball and move it with sound effect.

6. Write a program to test whether a given pixel is inside or outside or on a polygon.

LAB REQUIREMENTS:

Software requirements: Turbo C++ or Turbo C compiler

Operating System: Windows XP

PREREQUISITES:

Good programming skills in C.

Basic Graphics Function

First of all we have to call the initgraph function that will intialize the graphics mode on the computer. initigraph function have the following prototypes.

```
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
```

Initgraph initializes the graphics system by loading a 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 graph result to 0.

Graph Driver (gd) Integer that specifies the graphics driver to be used.

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

pathtodriver Specifies the directory path where initgraph looks for graphics drivers (*.BGI) first

1. If they're not there, initgraph looks in the current directory.

2. If pathtodriver is null, the driver files must be in the current directory. graphdriver and graphmode must be set to valid graphics_drivers and graphics_mode values or we will get unpredictable results

Basic Function :

- **Cleardevice()** Clears all previous graphical outputs generated by the previous programs. It's a good practice to include this method at the starting of each program.
- **gotoxy()** This will initialize the graphics cursor to the specified co-ordinate. In C, gotoxy() function is used very frequently to locate the cursor at different locations whenever as necessary.

Syntax : gotoxy(x,y)

- **putpixel()** It will colour the pixel specified by the co-ordinates.

Syntax: putpixel(x,y,WHITE)

- **outtextxy()** This method is used to display a text in any position on the screen. The numeric coordinates are substituted for x and y.

Syntax: outtextxy(x,y,"HELLO")

- **rectangle()** Draws a rectangle according to the given parameter x and y co-ordinates.

Syntax : rectangle(int left, int top, int right, int bottom)

- **circle()** Draws a circle with x,y as the center .

Syntax: circle(x,y,radius)

- **line()** Draws a line as per the given co-ordinates.

Syntax : line(int startx, int starty, int endx, int endy)

- **delay()** Cause a pause in execution of the program. 1000ms= 1 second

Syntax : delay(100)

- **closegraph()** Terminates all graphics operations and revert the hardware back to the normal mode.

MANUAL FOR PROGRAMS:

PROGRAM 1:

Write a program to rotate an image.

PROCEDURE:

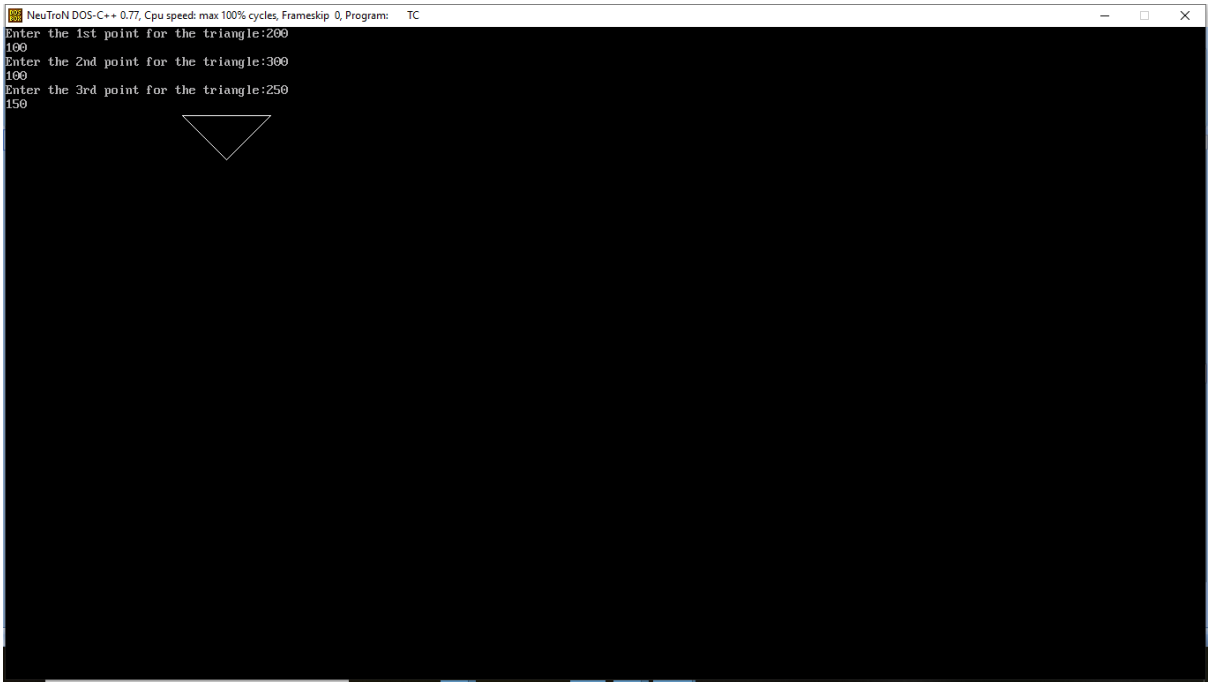
1. Start the process.
2. Declare variables x1,y1 ,x2, y2, x3, y3 and also declare the functions.
3. Declare gdriver=DETECT, mode and Initialize the graphic mode with the path location in TurboC3 folder.
4. Load the values of the variables x1,y1, x2 ,y2, x3, y3 into the frame buffer.
5. Using the function **triangle(x1,y1,x2,y2,x3,y3)** draw a triangle.
6. Load the value of the variable angle into the frame buffer.
7. Using Rotate function rotates the image to certain angle value.
8. Close the graph and stop the process.

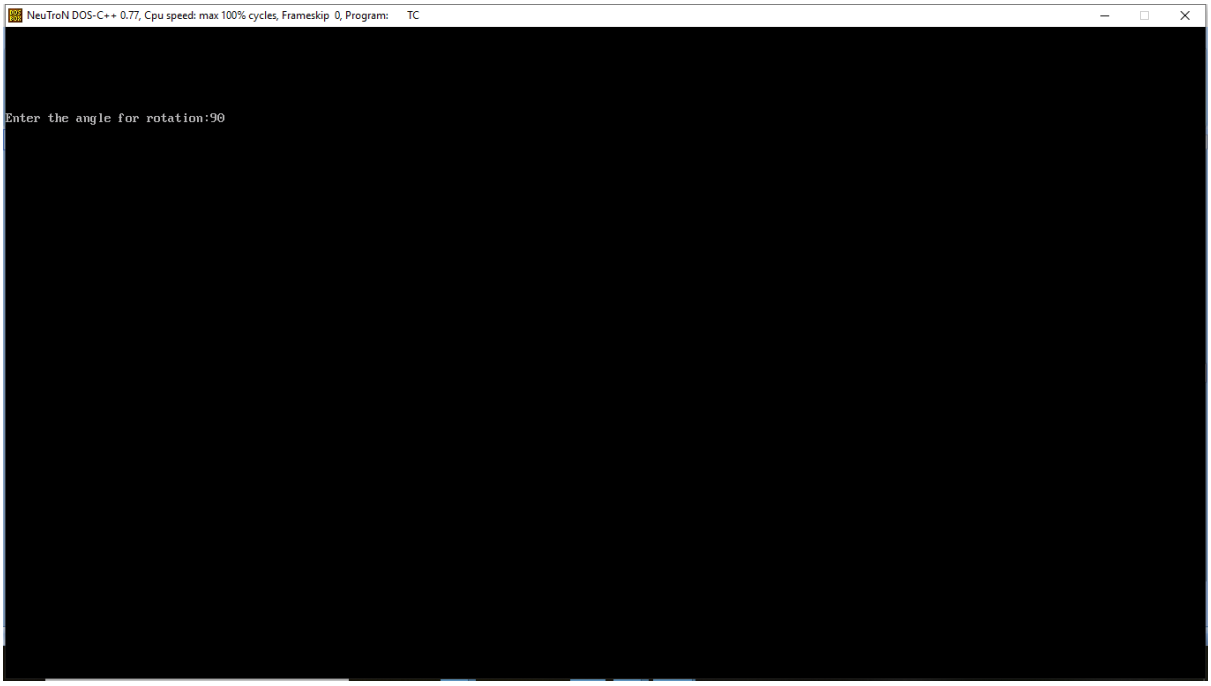
SOURCE CODE:

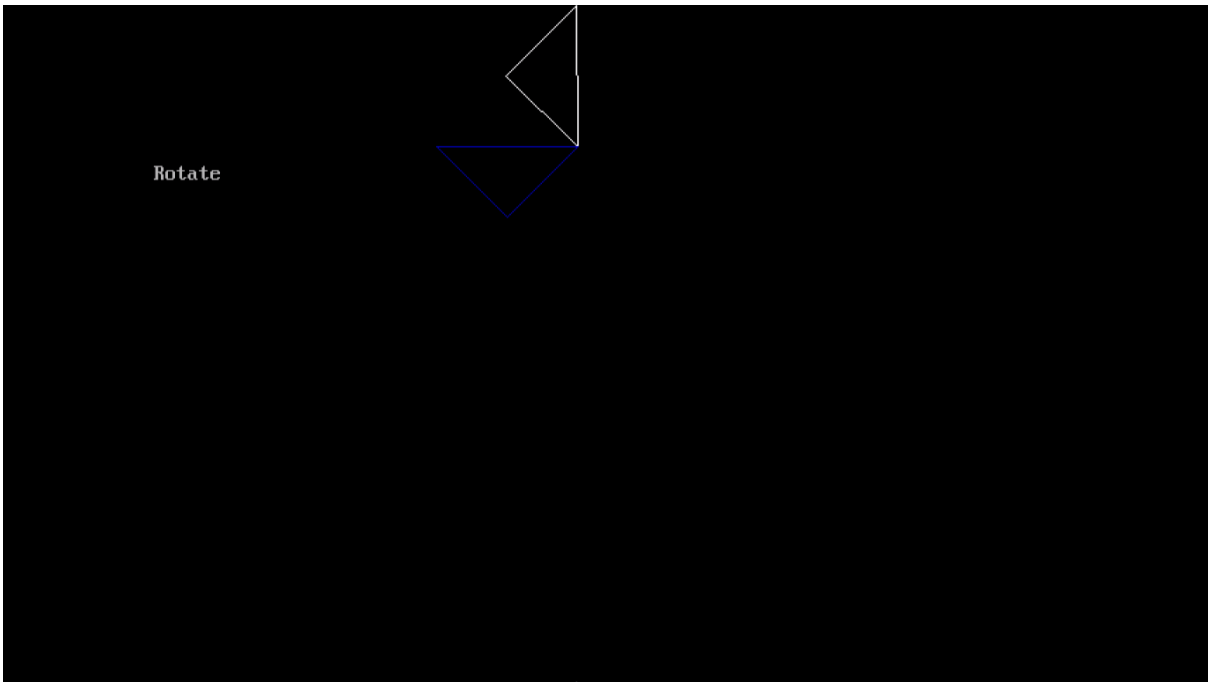
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>
void triangle(int x1,int y1,int x2,int y2,int x3,int y3);
void Rotate(int x1,int y1,int x2,int y2,int x3,int y3);
void main()
{
        int gd=DETECT,gm;
        int x1,y1,x2,y2,x3,y3;
        initgraph(&gd,&gm,"..\\bgi");
        printf("Enter the 1st point for the triangle:");
        scanf("%d%d",&x1,&y1);
        printf("Enter the 2nd point for the triangle:");
        scanf("%d%d",&x2,&y2);
        printf("Enter the 3rd point for the triangle:");
        scanf("%d%d",&x3,&y3);
        triangle(x1,y1,x2,y2,x3,y3);
        getch();
        cleardevice();
        Rotate(x1,y1,x2,y2,x3,y3);
        setcolor(1);
        triangle(x1,y1,x2,y2,x3,y3);
        getch();
}
void triangle(int x1,int y1,int x2,int y2,int x3,int y3)
{
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
}
```

```
void Rotate(int x1,int y1,int x2,int y2,int x3,int y3)
{
        int x,y,a1,b1,a2,b2,a3,b3,p=x2,q=y2;
        float Angle;
        printf("Enter the angle for rotation:");
        scanf("%f",&Angle);
        cleardevice();
        Angle=(Angle*3.14)/180;
        a1=p+(x1-p)*cos(Angle)-(y1-q)*sin(Angle);
        b1=q+(x1-p)*sin(Angle)+(y1-q)*cos(Angle);
        a2=p+(x2-p)*cos(Angle)-(y2-q)*sin(Angle);
        b2=q+(x2-p)*sin(Angle)+(y2-q)*cos(Angle);
        a3=p+(x3-p)*cos(Angle)-(y3-q)*sin(Angle);
        b3=q+(x3-p)*sin(Angle)+(y3-q)*cos(Angle);
        printf("Rotate");
        triangle(a1,b1,a2,b2,a3,b3);
}
```

OUTPUT:




NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the angle for rotation:90


Rotate

PROGRAM 2:

Write a program to drop each word of a sentence one by one from the top.

PROCEDURE:

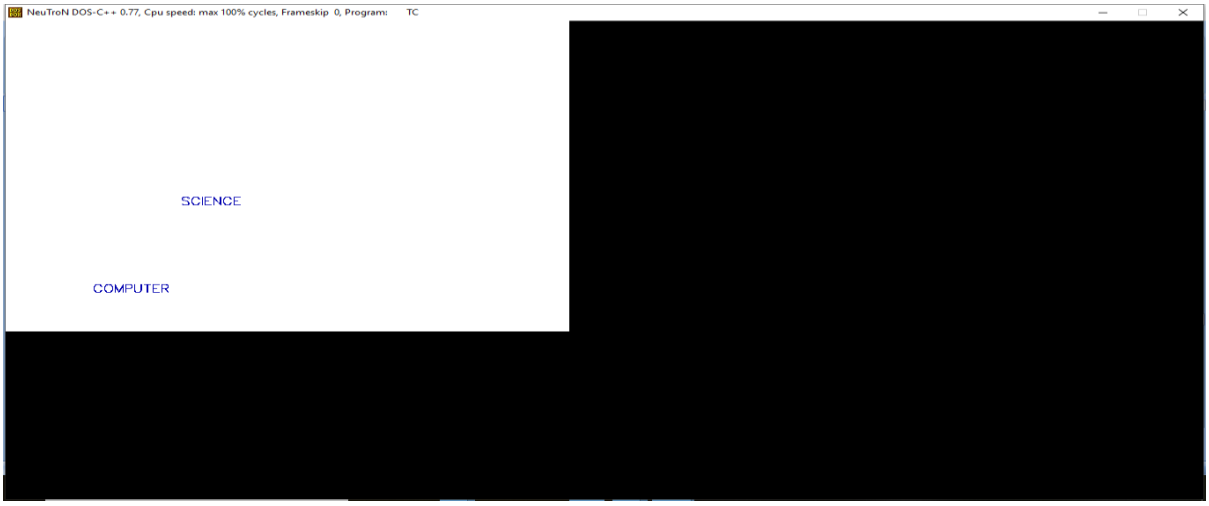
1. Start the process.
2. Declare variables i.
3. Set the background color of the screen to be white.
4. Using the function OUTTEXTXY() print the text on the screen.
5 Set the color of the text to be in BLUE color.
6. Drop the word one by one in the text by incrementing the Y coordinates value and using delay function.
7. Close Graph and stop.

SOURCE CODE:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"..\\bgi");
 int i;
 setbkcolor(WHITE); // Set the background color
 for(i=0;i<350;i++)
 {
        settextstyle(3,0,1); //used to change text appearance.
        outtextxy(200,40,"SCIENCE");
        setcolor(BLUE);
        outtextxy(100,40+i,"COMPUTER"); //Display text on screen with y coordinate.
        delay(25);
        cleardevice();
 }
 for(i=0;i<350;i++)
 {
        outtextxy(100,400,"COMPUTER");
```

```
        setcolor(BLUE);
        outtextxy(200,40+i,"SCIENCE"); //Display text on screen with x coordinate.
        delay(25);
        cleardevice();
  }
  outtextxy(100,400,"COMPUTER");
  outtextxy(200,400,"SCIENCE");
  getch();
  closegraph();}
```

OUTPUT:



PROGRAM 3:

Write a program to drop a line using DDA Algorithm.

PROCEDURE:

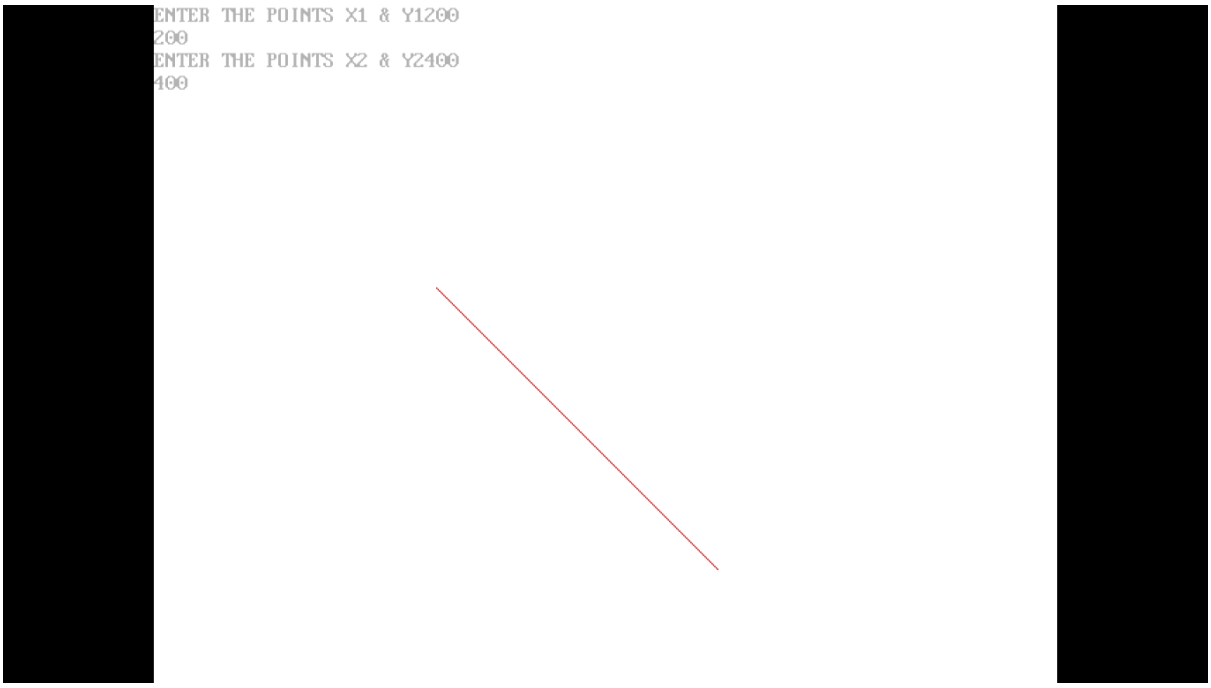
1. Start the process.
2. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also declare gdriver=DETECT, mode.
3. Initialize the graphic mode with the path location in TurboC3 folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1, y1) into the frame buffer; that is, plot the first point. put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs (dx) > abs (dy), do s=abs(dx).
8. Otherwise s= abs(dy).
9. Then xi=dx/s and yi=dy/s.
10. Start from k=0 and continuing till k
    i. x=x+xi.
    ii. Y=y+yi.
11. Plot pixels using putpixel at points (x,y) in specified colour.
12. Close Graph and stop.

SOURCE CODE:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int gd = DETECT ,gm, i;
    float x, y,dx,dy,steps;
    int x0, x1, y0, y1;
    initgraph(&gd, &gm, "..\\bgi");
    setbkcolor(WHITE);
    printf("Enter X0,Y0,X1,Y1:");
    scanf("%d %d %d %d",&x0,&y0,&x1,&y1);
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
```

```
        if(dx>=dy)
                steps = dx;
        else
                steps = dy;
        dx = dx/steps;
        dy = dy/steps;
        x = x0;
        y = y0;
        i = 1;
        while(i<= steps)
        {
                putpixel(x, y, RED);
                x += dx;
                y += dy;
                i=i+1;
        }
        getch();
        closegraph();
}
```

OUTPUT:



PROGRAM 4:

Write a program to move a car with sound effect.

PROCEDURE:

Step 1: Start the process.

Step 2: Create a function draw_car( ) to draw a car using line( ) and circle( ) commands.

Step 3: In the main function declare the variable gd and gm, auto detection of graphical driver is done using the variable gd.

Step 4: Initiate the graphical driver and graphic mode using init graph function.

Step 5: Invoke the draw_car function inside the main function.

Step 6: Using for loop move the car on the screen.

Step 7: Display the car with sound effects on the output screen using putimage( ) and sound( ) commands.

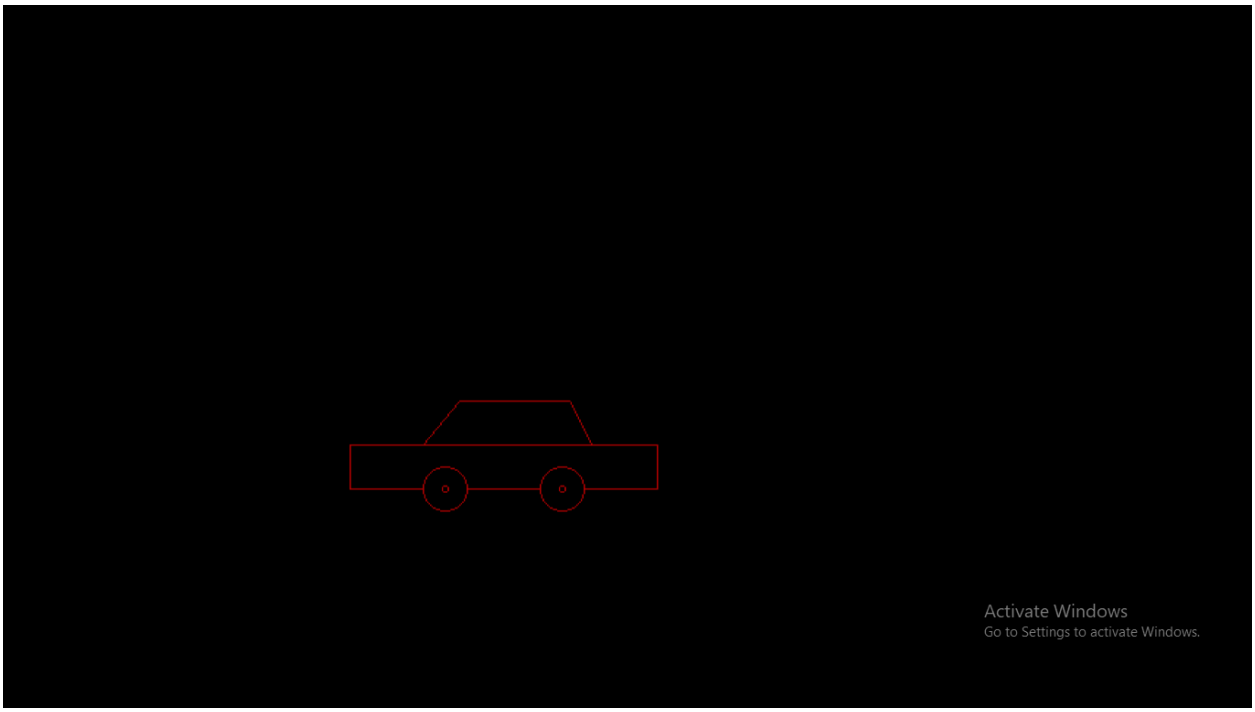
Step 8: Stop the process.

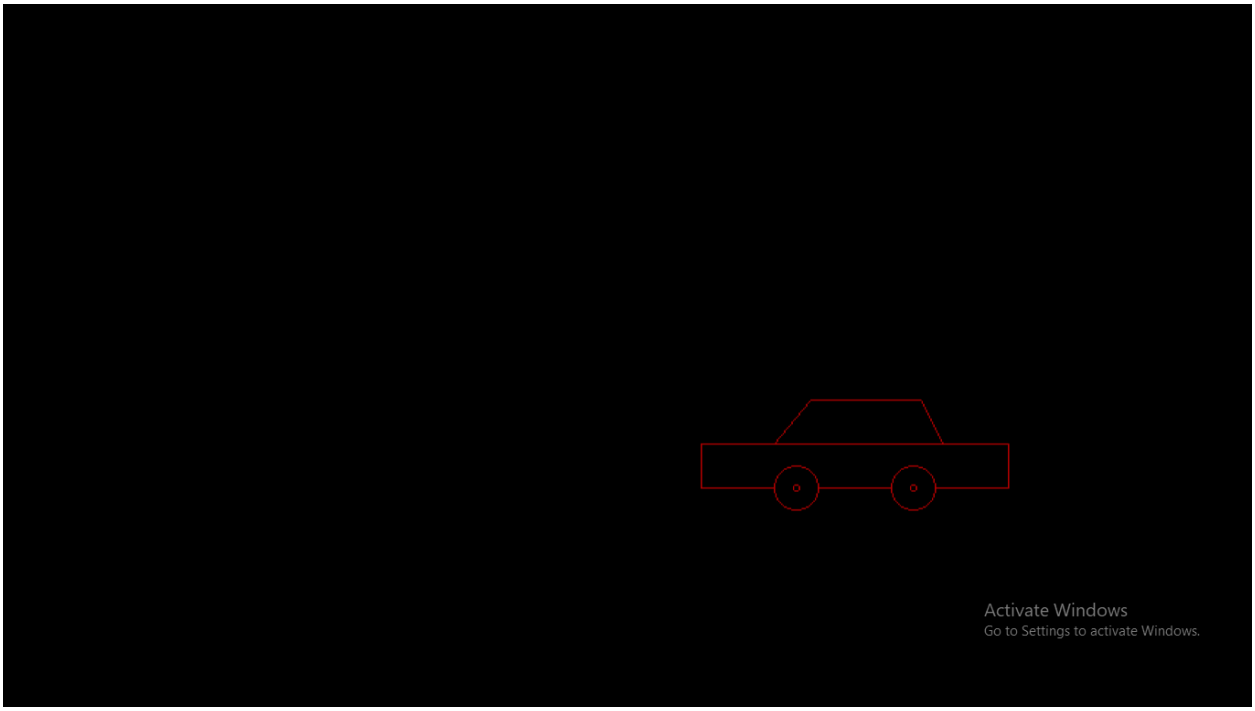
SOURCE CODE:

```
#include <graphics.h>
#include <stdio.h>
void draw_moving_car(void)
{
        int i, j = 0, gd = DETECT, gm;
        initgraph(&gd, &gm, "");
        for (i = 0; i <= 420; i = i + 10)
        {
                setcolor(RED);
                line(0 + i, 300, 210 + i, 300);
                line(50 + i, 300, 75 + i, 270);
                line(75 + i, 270, 150 + i, 270);
                line(150 + i, 270, 165 + i, 300);
                line(0 + i, 300, 0 + i, 330);
                line(210 + i, 300, 210 + i, 330);
                circle(65 + i, 330, 15);
```

```
            circle(65 + i, 330, 2);
            circle(145 + i, 330, 15);
            circle(145 + i, 330, 2);
            line(0 + i, 330, 50 + i, 330);
            line(80 + i, 330, 130 + i, 330);
            line(210 + i, 330, 160 + i, 330);
            delay(100);
            setcolor(BLACK);
            line(0 + i, 300, 210 + i, 300);
            line(50 + i, 300, 75 + i, 270);
            line(75 + i, 270, 150 + i, 270);
            line(150 + i, 270, 165 + i, 300);
            line(0 + i, 300, 0 + i, 330);
            line(210 + i, 300, 210 + i, 330);
            circle(65 + i, 330, 15);
            circle(65 + i, 330, 2);
            circle(145 + i, 330, 15);
            circle(145 + i, 330, 2);
            line(0 + i, 330, 50 + i, 330);
            line(80 + i, 330, 130 + i, 330);
            line(210 + i, 330, 160 + i, 330);
        }
        getch();
        closegraph();
}
int main()
{
        draw_moving_car();
        return 0;
}
```

OUTPUT:


Activate Windows
Go to Settings to activate Windows.


Activate Windows
Go to Settings to activate Windows.

PROGRAM 5:

Write a program to bounce a ball and move it with the sound effect.

PROCEDURE:

Step 1: Start the process.

Step 2: Declare the variable gd and gm, auto detection of graphical driver is done using the variable gd.

Step 3: Initiate the graphical driver and graphic mode using init graph function.

Step 4: Declare the pointer variable ball as a global variable.

Step 5: Declare the function image and draw the ball using fillellipse() and fill the color for the ball using setfillstyle().

Step 6: Declare the necessary variable in the main function.

Step 7: Move the ball using the variable mx and my with sound effects.

Step 8: Stop the process.

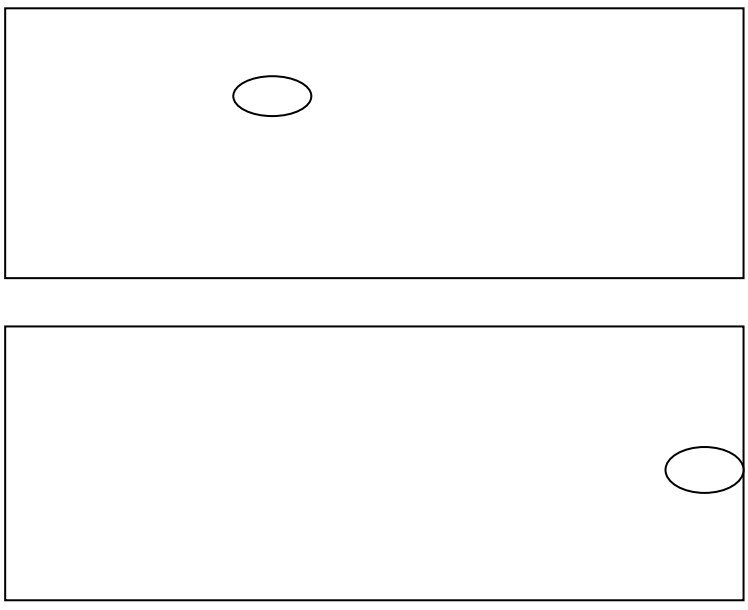
SOURCE CODE:

```
#include<graphics.h>
#include<conio.h>
#include<alloc.h>
#include<dos.h>
#include<stdlib.h>
void *ball;
void image()
{
    setcolor(RED);  //set the current drawing color
    setfillstyle(SOLID_FILL,GREEN);  //set current fill pattern and color
    fillellipse(10,10,10,10); //draws and fill the ellipse
    ball=malloc(imagesize(0,0,20,20)); //allocate a block of memory from the cache
    getimage(0,0,20,20,ball); //copies an image from screen to memory
    cleardevice(); //clears the graphics screen
 }
 void main()
 {
```

```
int gm,gd=DETECT;
 initgraph(&gd,&gm,"..\\bgi"); //initialize the graphics system
int mx=getmaxx()/2,my=0; //getmaxx() – returns the maximum  x coordinate
 int x=l,y=l,s=0,key=0,xstep=l,ystep=l;
image();
setbkcolor(WHITE); //returns the current background color
while(key!=27) //until user press Esc key
{
        while(!kbhit())
        {
                putimage(mx,t,ball,XOR_PUT); // outputs a bit image on to screen
                delay(5); // suspends execution for interval in millisecond
                putimage(mx,t,ball,XOR_PUT);
                if(mx>getmaxx()||mx<=0)
                {
                        x*=-1;
                        sound(1000);
                        // sound turns the PC speaker with specific frequency
                        s=0;
                         xstep=x*(random(4)+1);
                             // pick random number for x coordinator
                        ystep=y*(random(3)+1);
                             // pick random number for y coordinator
                         if(mx<=0)
                                 mx=0;
                        else
                                mx=getmaxx();
                }
                if(my>getmaxy()||my<=0)
                {
                        y*=-l;
                        sound(100);
                        s=0;
```

```
                    ystep=y*(random(4)+l);
                    xstep=x*(random(3)+l);
                     if(my<=0)
                            my=0;
                    else
                            my=getmaxy();
                }
                 mx+=x+xstep;
                 my+=y+ystep;
                s++;
                if(s==5)
                {
                        nosound(); // turns the PC speaker off
                }
            }
            key=getch();
            }
        closegraph(); // close the graphic system
    }
```

OUTPUT:



PROGRAM 6:

Write a program to test whether a given pixel is inside or outside or on a polygon.

PROCEDURE:

Step 1: Start the process.

Step 2: Declare the variable gd and gm, auto detection of graphical driver is done using the variable gd.

Step 3: Draw the polygon with given points.

Step 4: Set the values for a pixel co-ordinate.

Step 5: Using a "pointInpoly()" checking the pixel position.

Step 6: Based on the boundary values, print the result as "Inside or outside the boundary.

Step 7: close the graphical mode.

Step 8: Stop the process.

SOURCE CODE:

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
int polyx[6]={540,590,570,510,490,540};   // x co-ordinates for polygon
int polyy[6]={220,270,320,320,270,220};  // y co-ordinates for polygon
int polysides=5;
int x,y;
int pointInpoly();
void draw_polygon();
void main()
{
        int c,gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        draw_polygon(); // calling function to draw polygon
        printf("\nEnter the value of x:");
        scanf("%d",&x);
```

```
printf("\n Enter the value of y:");
scanf("%d",&y);
 putpixel(x,y,WHITE);   //plot a pixle in x and y co-ordinates
c=pointInpoly();
if(c==0)
        printf("\n point is outside the polygon");
else
        printf("\n point is inside the polygon");
getch();
}
void draw_polygon()
{
        int i,j;
        for(i=0;i<polysides;i++)
        {
                if(i==(polysodes-1))
                        line(polyX[i],polyY[i],polyX[0],polyY[0]);
                                // draw a lines for polygon
                else
                         line(polyX[i],polyY[i],polyX[i+1],polyY[i+1]);
        }
        int pointInpoly()
        {
                int i,j=polysides-1;
                int oddnodes=0;
                for(i=0;i<polysides;i++)
                {
                        if(polyy[i]<y&&polyy[j]>=y||polyy[j]<y&&polyy[i]>=y)
                        //checking the position of the pixel
                        {
                        if(polyX[i]+(y-polyY[i])/(polyY[j]-polyY[i]*(polyX[j]-polyX[i])<x))
                        {
                                oddnodes=1;
                        }
```

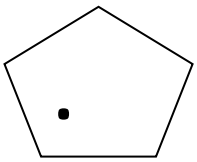
```
                    }
                    j=i;
                }
        return oddnodes;
}
```

OUTPUT:

ENTER THE VALUE FOR X: 550

ENTER THE VALUE FOR Y: 300

Inside



ENTER THE VALUE FOR X: 350

ENTER THE VALUE FOR Y: 400

Outside