COMPUTER GRAPHICS LAB

CSE-352

1.Draw a line Using Scan-Conversion line algorithm.

2. Scan-converting a circle using Midpoint Circle algorithm .

3.Write a C program to draw a car is moving on the road.

4.Write a c program to draw a man is walking on the road on a rainy day.

5.Write a C program which perform two dimensional transformations: translation ,scaling ,shearing ,reflection.

6.Write a C program to clip the lines fallen outside the window using Cohen Sutherland line clipping.

7.Write a C program to clip the polygon region fallen outside the window using Hodgeman Sutherland polygon clipping algorithm.

8.Write a program to implement 3D transformation. Translation, rotation, scaling of 3D object.

9.Write a program to create a 3D scene

10. write a c program to show a kite is flying in the sky.

11. Implementation of 3D projection.

12. Write a program to create a GIF animated images.

13.Implementation of 3D transformations, Translation, Scaling, reflection.

**BASIC**

**/\***

**Write a c++ code to build a 3D Image**

**\*/**

**#include<bits/stdc++.h>**

**#include<conio.h>**

**#include<graphics.h>**

**#include<dos.h>**

**using namespace std;**

**int main()**

**{**

**int gd=DETECT,gm;**

**/// the graphics hardware in the system**

**initgraph(&gd,&gm,"C:\\tc\\bgi");// initializes the graphics mode and clear the screen**

**///(int far \*driver,int far \*mode);**

**closegraph();**

**/// graphics mode to text mode**

**getpixel(int x,int y);**

**/// returns the colour of pixel present at location(x,y);**

**putpixel(x,y,GREEN);**

**///draw green colour a x,y point**

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

**/// DRAW a line from(x1,y1) to (x2,y2)**

**lineto(x,y);**

**///lineto function draw a line from currentPosition(CP)tp thr point(x,y)**

**/// you can get current position using getx and gety function;**

**circle(int x,int y,int radius);**

**/// draw a circle**

**ellipse(int x,int y,int stangle,int endangle,int xradius,int yradious);**

**/// Draw an elipse**

**drawpoly(int number,int \*polypoints);**

**int polypoints[]= {x1,y1,x2,y2...,x1,x2};**

**///Drawing a polygon**

**outtext(char \*string);**

**/// display text at current position**

**outtextxy(int x,int y,char \*string);**

**/// at specific position**

**rectangle();**

**///LEFT || TOP || RIGHT || BITTOM ANTICLOCK WISE**

**floodfill(int x,int y,int border);**

**/// fill and enclose aria**

**void drawpoly(int num,int \*polypoints);**

**///draw and fills a polygon**

**fillellipse(int x,inty,int xradius,int yradius);**

**///it fills ellipse**

**///x,y is the center xradius and y radius**

**}**

**1.Draw a line Using Scan-Conversion line algorithm.**

**Solution:**

|  |
| --- |
| **#include<stdio.h>** **#include<conio.h>** **#include<graphics.h>** **#include<bits/stdc++.h>** using namespace std; int maxx=500, maxy=500, midx=250, midy=250; int **main**() {  int gd, gm, z, o, x1, x2, y1, y2;  detectgraph(&gd, &gm);  initgraph(&gd, &gm, "e:\tc\bgi");  vector<int> X,Y;   cout <<"STARTING COORDINATE "<<endl;  cin >>x1;  cin >>y1;  cout <<"ENDING COORDINATE "<<endl;  cin >>x2;  cin >>y2;    *///USING BRESENHAM ALGORITHM*   int m\_new = 2 \* (y2 - y1);  int slope\_error\_new = m\_new - (x2 - x1);  for (int x = x1, y = y1; x <= x2; x++)  {   X.push\_back(x);  Y.push\_back(y);   *// Add slope to increment angle formed*  slope\_error\_new += m\_new;   *// Slope error reached limit, time to*  *// increment y and update slope error.*  if (slope\_error\_new >= 0)  {  y++;  slope\_error\_new -= 2 \* (x2 - x1);  }  }  for(int i=0; i<X.size(); i++)  {  putpixel(X[i],Y[i],10);  }  getch();  return 0; } |

**2. Scan-converting a circle using Midpoint Circle algorithm** .

**SOLUTION:**

|  |
| --- |
| **#include<stdio.h>** **#include<conio.h>** **#include<graphics.h>** **#include<bits/stdc++.h>** using namespace std; int maxx=500, maxy=500, midx=250, midy=250; int **main**() {  int gd, gm, z, o, x1, x2, y1, y2;  int x\_centre,y\_centre,r;  detectgraph(&gd, &gm);  initgraph(&gd, &gm, "e:\tc\bgi");  vector<int> X,Y;   cout <<"ENTER CENTER "<<endl;  cin >>x\_centre;  cin >>y\_centre;  cout <<"ENTER RADIUS "<<endl;  cin >>r;    *///Using Mid point Circle Drawing Algorithm*   int x = r, y = 0;  X.push\_back(x + x\_centre);  Y.push\_back(y + y\_centre);    *// When radius is zero only a single*  *// point will be printed*  if (r > 0)  {   X.push\_back(x + x\_centre),Y.push\_back(-y + y\_centre);  X.push\_back(y + x\_centre),Y.push\_back(x + y\_centre);  X.push\_back(-y + x\_centre),Y.push\_back(x + y\_centre);  }   *// Initialising the value of P*  int P = 1 - r;  while (x > y)  {  y++;   *// Mid-point is inside or on the perimeter*  if (P <= 0)  P = P + 2\*y + 1;   *// Mid-point is outside the perimeter*  else  {  x--;  P = P + 2\*y - 2\*x + 1;  }   *// All the perimeter points have already been printed*  if (x < y)  break;   *// Printing the generated point and its reflection*  *// in the other octants after translation*  X.push\_back(x + x\_centre),Y.push\_back(y + y\_centre);   X.push\_back(-x + x\_centre),Y.push\_back(y + y\_centre);    X.push\_back( x + x\_centre),Y.push\_back(-y + y\_centre);   X.push\_back(-x + x\_centre),Y.push\_back(-y + y\_centre);   *// If the generated point is on the line x = y then*  *// the perimeter points have already been printed*  if (x != y)  {   X.push\_back( y + x\_centre),Y.push\_back(x + y\_centre);   X.push\_back( -y + x\_centre),Y.push\_back( x + y\_centre);   X.push\_back( y + x\_centre),Y.push\_back(-x + y\_centre);   X.push\_back(-y + x\_centre),Y.push\_back(-x + y\_centre);  }  }    for(int i=0; i<X.size(); i++)  {  putpixel(X[i],Y[i],10);  }  getch();  return 0; } |

**3.Write a C program to draw a car is moving on the road.**

**Solution**

|  |
| --- |
| **#include<bits/stdc++.h>** **#include<conio.h>** **#include<graphics.h>** **#include<dos.h>** using namespace std;  int **main**() {   int gd=DETECT,gm;  initgraph(&gd,&gm,"C:\\tc\\bgi");   for (int i=0; i<=700; i++) {   rectangle(0+i,50,150+i,150 );  line(150+75+i,150 ,150+75+i,100);  line(150+i+40,80 ,150+75+i,100);  line(150+i,150,150+75+i,150);  line(150+i,80,150+40+i,80);    rectangle(170+i,130-10,190+i,90);  circle(180+i,105,5);  circle(173+i,105,2);  circle(187+i,105,2);  if(i%2)  {   circle(40+i,160,5);  circle(40+i,160,10);  }  else {   circle(110+i+70,160,10);  circle(110+i+70,160,5);  *//setcolor(10);*  }   *// print E LOGO*  line(50+i,70,20+30+i,130); *// |*  line(50+i,70,50+i+30,70); *//--*  line(50+i,100,35+i+30,100); *//--*  line(50+i,130,50+i+30,130); *//-*   delay(1);  cleardevice();   line(0,170,800,170);   }  getch(); } |

**4.Write a c program to draw a man is walking on the road on a rainy day.**

**Solution:**

|  |
| --- |
| **#include<stdio.h>** **#include<graphics.h>** **#define ScreenWidth getmaxx()** **#define ScreenHeight getmaxy()** **#define GroundY ScreenHeight\*0.75** int a=50; int ldisp=0; int ppp=0; void **drawCloud**() {  ppp++;  int y=40;  int z=ppp;  int r = 40;  arc(z, y, 45, 135, r);   arc(z + 50, y, 45, 135, r);  arc(z + 100, y, 45, 135, r);  arc(z, y, 135, 225, r);  arc(z+50,y,135+90,225+90,r);  arc(z,y,135+90,225+90,r);  arc(z+100,y,135+90,225+90,r);  arc(z+100,y,315,45,r); } void **DrawManAndUmbrella**(int x,int ldisp) {  circle(x,GroundY-90,10);  line(x,GroundY-80,x,GroundY-30);   rectangle(x,GroundY-80,x+5,GroundY-30);  for(int i=-5;i<=5;i++)  {   rectangle(x,GroundY-80,x+i,GroundY-30);  }   rectangle(x,GroundY-80,x-5,GroundY-30);    line(x,GroundY-70,x+10,GroundY-60);  line(x,GroundY-65,x+10,GroundY-55);  line(x+10,GroundY-60,x+20,GroundY-70);  line(x+10,GroundY-55,x+20,GroundY-70);   line(x,GroundY-30,x+ldisp,GroundY);  line(x,GroundY-30,x-ldisp,GroundY);   setfillstyle(SOLID\_FILL,BLUE);  pieslice(x+20,GroundY-120,0,180,40);  line(x+20,GroundY-120,x+20,GroundY-70);    *//second manush*    circle(a+x,GroundY-90,10);  line(a+x,GroundY-80,a+x,GroundY-30);   rectangle(a+x,GroundY-80,a+x+1,GroundY-30);  rectangle(a+x,GroundY-80,a+x-1,GroundY-30);   line(a+x,GroundY-70,a+x+10-25,GroundY-60);  line(a+x,GroundY-65,a+x+10-25,GroundY-55);  line(a+x+10-25,GroundY-60,x+20,GroundY-70); *//thik ase*  line(a+x+10-25,GroundY-55,x+20,GroundY-70); *//thik ase*   line(a+x,GroundY-30,a+x+ldisp,GroundY);  line(a+x,GroundY-30,a+x-ldisp,GroundY); } void **Rain2**(int x) {  int t=1000;  while(t--)  {  putpixel( rand()% getmaxx() , rand()%750,5 );  } } void **SUN**() {  setfillstyle(SOLID\_FILL,YELLOW);  pieslice(300,70,0,360,30); } void **Rain**(int x) {  int i,rx,ry;  for(i=0; i<600; i++)  {  rx=rand() % ScreenWidth;  ry=rand() % ScreenHeight;  if(ry<GroundY-4)  {  if(ry<GroundY-120 || (ry>GroundY-120 && (rx<x-20 || rx>x+60)))  putpixel(rx,ry,10),putpixel(rx+.5,ry+4,10);  }  } } int **main**() {  int gd=DETECT,gm,x=0;  initgraph(&gd,&gm,"C:\\TurboC3\\BGI");  while(true)  {  line(0,GroundY,ScreenWidth,GroundY);  Rain(x);SUN();  drawCloud();  ldisp=(ldisp+2)%20;i  DrawManAndUmbrella(x,ldisp);  delay(75);  cleardevice();  x=(x+2)%ScreenWidth;  }  getch(); } |

**5.Write a C program which perform two dimensional transformations: translation ,scaling ,shearing ,reflection.**

**TRANSLATION**

|  |
| --- |
| *// C++ program for translation* *// of a rectangle* **#include<bits/stdc++.h>** **#include<graphics.h>** using namespace std;  *// function to translate rectangle* void **translateRectangle** ( int P[][2], int T[]) {  */\* init graph and rectangle() are used for  representing rectangle through graphical functions \*/*  int gd = DETECT, gm, errorcode;  initgraph (&gd, &gm, "c:\\tc\\bgi");  setcolor (2);  *// rectangle (Xmin, Ymin, Xmax, Ymax)*  *// original rectangle*  rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);   *// calculating translated coordinates*  P[0][0] = P[0][0] + T[0];  P[0][1] = P[0][1] + T[1];  P[1][0] = P[1][0] + T[0];  P[1][1] = P[1][1] + T[1];   *// translated rectangle (Xmin, Ymin, Xmax, Ymax)*  *// setcolor(3);*  rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);  getch();  closegraph(); }  *// driver program* int **main**() {  *// Xmin, Ymin, Xmax, Ymax as rectangle*  *// coordinates of top left and bottom right points*  int P[2][2] = {50, 80, 120, 180};  int T[] = {40, 40}; *// translation factor*  translateRectangle (P, T);  return 0; } |

**SCALLING**

|  |
| --- |
| *// C program to demonstrate scaling of abjects* **#include<stdio.h>** **#include<graphics.h>** **#include<bits/stdc++.h>** using namespace std;   *// Matrix Multiplication to find new Coordinates.* *// s[][] is scaling matrix. p[][] is to store* *// points that needs to be scaled.* *// p[0][0] is x coordinate of point.* *// p[1][0] is y coordinate of given point.* void **findNewCoordinate**(int s[][2], int p[][1]) {  int temp[2][1] = { 0 };   for (int i = 0; i < 2; i++)  for (int j = 0; j < 1; j++)  for (int k = 0; k < 2; k++)  temp[i][j] += (s[i][k] \* p[k][j]);   p[0][0] = temp[0][0];  p[1][0] = temp[1][0]; }  *// Scaling the Polygon* void **scale**(int x[], int y[], int sx, int sy) {  *// Triangle before Scaling*  line(x[0], y[0], x[1], y[1]);  line(x[1], y[1], x[2], y[2]);  line(x[2], y[2], x[0], y[0]);   *// Initializing the Scaling Matrix.*  int s[2][2] = { sx, 0, 0, sy };  int p[2][1];   *// Scaling the triangle*  for (int i = 0; i < 3; i++)  {  p[0][0] = x[i];  p[1][0] = y[i];   findNewCoordinate(s, p);   x[i] = p[0][0];  y[i] = p[1][0];  }   *// Triangle after Scaling*  line(x[0], y[0], x[1], y[1]);  line(x[1], y[1], x[2], y[2]);  line(x[2], y[2], x[0], y[0]); }  *// Driven Program* int **main**() {  int x[] = { 100, 200, 300 };  int y[] = { 200, 100, 200 };  cout <<"Insert Scalling Factor"<<endl;  int sx = 2, sy = 2;  cin >>sx>>sy;   int gd, gm;  detectgraph(&gd, &gm);  initgraph(&gd, &gm," ");   scale(x, y, sx,sy);  getch();   return 0; } |

**Shearing**

|  |
| --- |
| **#include<stdio.h>** **#include<graphics.h>** **#include<math.h>** int graDriver=DETECT,graMode; int n,xs[100],ys[100],i; float shearXfactor,shearYfactor;  void **DrawFn**() {  for(i=0; i<n; i++)  line(xs[i],ys[i],xs[(i+1)%n],ys[(i+1)%n]); }  void **shearAlongX**() {  for(i=0; i<n; i++)  xs[i]=xs[i]+shearXfactor\*ys[i]; }  void **shearAlongY**() {  for(i=0; i<n; i++)  ys[i]=ys[i]+shearYfactor\*xs[i]; }  int **main**() {  printf("Enter number of sides: ");  scanf("%d",&n);  printf("Enter co-rdinates: x,y for each point ");  for(i=0; i<n; i++)  scanf("%d%d",&xs[i],&ys[i]);  printf("Enter x shear factor:");  scanf("%f",&shearXfactor);  printf("Enter y shear factor:");  scanf("%f",&shearYfactor);   initgraph(&graDriver,&graMode,"C:\\TURBOC3\\BGI\\");  setcolor(RED);  DrawFn();*//original*  shearAlongX();  setcolor(BLUE);  DrawFn();*//Xshear*  shearAlongY();  setcolor(GREEN);  DrawFn();*//Yshear*  getch();  } |

**Reflection**

|  |
| --- |
| **#include<stdio.h>** **#include<graphics.h>** int graDriver=DETECT,graMode; int n,xs[100],ys[100],i; int tempYaxis,tempXaxis;  void **DrawFn**() {  for(i=0; i<n; i++)  line(xs[i],ys[i],xs[(i+1)%n],ys[(i+1)%n]); }  void **FlipV**() {  tempXaxis=getmaxy()/2;  for(i=0; i<n; i++)  ys[i]=tempXaxis+(tempXaxis-ys[i]); *//drawing horizontal axis to flip about*   for(i=0; i<getmaxx(); i++)  putpixel(i,tempXaxis,WHITE); }  void **FlipH**() {  tempYaxis=getmaxx()/2;  for(i=0; i<n; i++)  xs[i]=tempYaxis+(tempYaxis-xs[i]);  setcolor(WHITE); *//drawing vertical axis*  for(i=0; i<getmaxy(); i++)  putpixel(tempYaxis,i,WHITE);  }  int **main**() {  printf("Enter number of sides: ");  scanf("%d",&n);  printf("Enter co-rdinates: x,y for each point ");  for(i=0; i<n; i++)  scanf("%d%d",&xs[i],&ys[i]);   initgraph(&graDriver,&graMode,"C:\\TURBOC3\\BGI\\");  setcolor(RED);  DrawFn();*//original*  FlipV();  setcolor(BLUE);  DrawFn();*//vertical flip*  FlipH();  setcolor(GREEN);  DrawFn();*//Horizontal flip*  getch();  } |

**6.Write a C program to clip the lines fallen outside the window using Cohen Sutherland line clipping.**

|  |
| --- |
| *// C++ program to implement Cohen Sutherland algorithm* *// for line clipping.* **#include<bits/stdc++.h>** **#include<conio.h>** **#include<graphics.h>** **#include<dos.h>** using namespace std;   int dp[10]; int a,b,c,d; *// Defining region codes* const int INSIDE = 0; *// 0000* const int LEFT = 1; *// 0001* const int RIGHT = 2; *// 0010* const int BOTTOM = 4; *// 0100* const int TOP = 8; *// 1000*  *// Defining x\_max, y\_max and x\_min, y\_min for* *// clipping rectangle. Since diagonal points are* *// enough to define a rectangle*  int x\_max = 400;  int y\_max = 400;  int x\_min = 50;  int y\_min = 50;  *// Function to compute region code for a point(x, y)* int **computeCode**(double x, double y) {  *// initialized as being inside*  int code = INSIDE;   if (x < x\_min) *// to the left of rectangle*  code |= LEFT;  else if (x > x\_max) *// to the right of rectangle*  code |= RIGHT;  if (y < y\_min) *// below the rectangle*  code |= BOTTOM;  else if (y > y\_max) *// above the rectangle*  code |= TOP;   return code; }  *// Implementing Cohen-Sutherland algorithm* *// Clipping a line from P1 = (x2, y2) to P2 = (x2, y2)* void **cohenSutherlandClip**(double x1, double y1,  double x2, double y2) {  *// Compute region codes for P1, P2*  int code1 = computeCode(x1, y1);  int code2 = computeCode(x2, y2);   *// Initialize line as outside the rectangular window*  bool accept = false;   while (true)  {  if ((code1 == 0) && (code2 == 0))  {  *// If both endpoints lie within rectangle*  accept = true;  break;  }  else if (code1 & code2)  {  *// If both endpoints are outside rectangle,*  *// in same region*  break;  }  else  {  *// Some segment of line lies within the*  *// rectangle*  int code\_out;  double x, y;   *// At least one endpoint is outside the*  *// rectangle, pick it.*  if (code1 != 0)  code\_out = code1;  else  code\_out = code2;   *// Find intersection point;*  *// using formulas y = y1 + slope \* (x - x1),*  *// x = x1 + (1 / slope) \* (y - y1)*  if (code\_out & TOP)  {  *// point is above the clip rectangle*  x = x1 + (x2 - x1) \* (y\_max - y1) / (y2 - y1);  y = y\_max;  }  else if (code\_out & BOTTOM)  {  *// point is below the rectangle*  x = x1 + (x2 - x1) \* (y\_min - y1) / (y2 - y1);  y = y\_min;  }  else if (code\_out & RIGHT)  {  *// point is to the right of rectangle*  y = y1 + (y2 - y1) \* (x\_max - x1) / (x2 - x1);  x = x\_max;  }  else if (code\_out & LEFT)  {  *// point is to the left of rectangle*  y = y1 + (y2 - y1) \* (x\_min - x1) / (x2 - x1);  x = x\_min;  }   *// Now intersection point x,y is found*  *// We replace point outside rectangle*  *// by intersection point*  if (code\_out == code1)  {  x1 = x;  y1 = y;  code1 = computeCode(x1, y1);  }  else  {  x2 = x;  y2 = y;  code2 = computeCode(x2, y2);  }  }  }  if (accept)  {  a=x1;  b=y1;  c=x2;  d=y2;  dp[0]= x1;  dp[1]=y1;  dp[2]=x2;  dp[3]=y2;  *// Here the user can add code to display the rectangle*  *// along with the accepted (portion of) lines*  }  else  cout << "Line rejected" << endl; }  *// Driver code* int **main**() {  int px1,px2,py1,py2;   cout <<endl;  cout <<"ENTER LINES X1"<<endl,cin >>px1;  cout <<"ENTER LINES Y1"<<endl,cin >>py1;  cout <<"ENTER LINES X2"<<endl,cin >>px2;  cout <<"ENTER LINES Y2"<<endl,cin >>py2;    cohenSutherlandClip(px1,py1,px2,py2);     int gd=DETECT,gm;  initgraph(&gd,&gm,"C:\\tc\\bgi");   cleardevice();  getchar();  rectangle(x\_max,y\_max,x\_min,y\_min);  line(px1,py1,px2,py2);   getchar();  getchar();  cleardevice();  rectangle(x\_max,y\_max,x\_min,y\_min);  line(a,b,c,d);   getchar();   getch();   return 0; } |

**7.Write a C program to clip the polygon region fallen outside the window using Hodgeman Sutherland polygon clipping algorithm.**

|  |
| --- |
| **#include<stdio.h>** **#include<conio.h>** **#include<graphics.h>** **#include<math.h>** void **clip**(float,float,float); int i,j=0,n; int rx1,rx2,ry1,ry2; float x1[8],y1[8]; int **main**() {  int gd=DETECT,gm;  int i,n;  float x[8],y[8],m;   initgraph(&gd,&gm,"");  printf("coordinates for rectangle : ");  scanf("%d%d%d%d",&rx1,&ry1,&rx2,&ry2);  printf("no. of sides for polygon : ");  scanf("%d",&n);  printf("coordinates : ");     for(i=0; i<n; i++)  {  scanf("%f%f",&x[i],&y[i]);  }  cleardevice();  outtextxy(10,10,"Before clipping");  outtextxy(10,470,"Press any key....");  rectangle(rx1,ry1,rx2,ry2);  for(i=0; i<n-1; i++)  line(x[i],y[i],x[i+1],y[i+1]);  line(x[i],y[i],x[0],y[0]);  getch();  cleardevice();  for(i=0; i<n-1; i++)  {  m=(y[i+1]-y[i])/(x[i+1]-x[i]);  clip(x[i],y[i],m);  }  clip(x[0],y[0],m);  outtextxy(10,10,"After clipping");  outtextxy(10,470,"Press any key....");  rectangle(rx1,ry1,rx2,ry2);  for(i=0; i<j-1; i++)  line(x1[i],y1[i],x1[i+1],y1[i+1]);  getch(); }  void **clip**(float e,float f,float m) {  while(e<rx1 && e>rx2 && f<ry1 && f>ry2)  {  if(e<rx1)  {  f+=m\*(rx1-e);  e=rx1;  }  else if(e>rx2)  {  f+=m\*(rx2-e);  e=rx1;  }  if(f<ry1)  {  e+=(ry1-f)/m;  f=ry1;  }  else if(f>ry2)  {  e+=(ry2-f)/m;  f=ry2;  }  x1[j]=e;  y1[j]=f;  j++;  } } |

**8.Write a program to implement 3D transformation. Translation, rotation, scaling of 3D object.**

**SOLUTIONS:**

**See 13**

**9.Write a program to create a 3D scene**

|  |
| --- |
| */\* Write a c++ code to build a 3D Image \*/* **#include<bits/stdc++.h>** **#include<conio.h>** **#include<graphics.h>** **#include<dos.h>** using namespace std; int **main**() {  int gd=DETECT,gm;  initgraph(&gd,&gm,"C:\\tc\\bgi");  bar3d(150, 150, 200, 250 , 20, 5);  getch(); } |

1**0 Write a C program to show a kite is flying in the sky**

|  |
| --- |
| **#include<bits/stdc++.h>** **#include<stdio.h>** **#include<time.h>** **#include<conio.h>** **#include<graphics.h>** **#include<stdlib.h>** **#include<dos.h>** using namespace std;  int **main**() {  int gd=DETECT,gm;  int x=10,y=480;  initgraph(&gd,&gm,"..\\bgi");  while(!kbhit())  {  cleardevice();  if(y==0)  {  y=rand()% 480;  x=rand()%640;  }  else  {  y=y-1;  x=x+1;  line(x-50,y,x,y-70);  line(x,y-70,x+50,y);  line(x+50,y,x,y+70);  line(x,y+70,x-50,y);  line(x,y-70,x,y+70);  line(x,y+70,x+10,y+140);  line(x,y+70,x-10,y+140);  line(x-50,y,x+50,y);  line(x,y,x+130,y+640);  }  delay(20);  }  closegraph();  restorecrtmode(); } |

|  |
| --- |
| ALTERNATIVE SOLUTION |

|  |
| --- |
| **#include<stdio.h>** **#include<conio.h>** **#include<graphics.h>** **#include<stdlib.h>** int **main**() {  int gm,gd=DETECT;  int i= 0,j=0,rnd\_x=0,rnd\_y,stop\_me=0;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  *// srand(time());*  while(stop\_me<=1000)  {   if(i>=180 &&j>=100 ) *// controlling kite, so that it wouldn't disappear from screen*  {  rnd\_x=rand()%4 -3;  rnd\_y=rand()%5 -4;  }  else  {  rnd\_x=rand()%3;  rnd\_y=rand()%3;  }  line(200+i,200-j,250+i,250-j);  line(200+i,200-j,150+i,250-j);  line(150+i,250-j,200+i,350-j);  line(200+i,350-j,250+i,250-j);  line(200+i,200-j,200+i,350-j);   line(200+i,350-j,200+i ,350-j+30 );   arc(200+i,275-j,25,155,50);  line(0,500,200+i,225-j);  i=i+rnd\_x;  j=j+rnd\_y;  stop\_me=5+stop\_me;  delay(100);  clearviewport(); *// clearing image which would make illusion of flying kite*   }  getch();  closegraph();  } |

**11 ) Implementation of 3D projection**

|  |
| --- |
| **#include<stdio.h>** **#include<math.h>** **#include<graphics.h>**  int **main**() {   int x1,y1,x2,y2,gd,gm;  int ymax,a[4][8];  float par[4][4],b[4][8];  int i,j,k,m,n,p;  int xp, yp, zp, x, y, z;    a[0][0] = 100;  a[1][0] = 100;  a[2][0] = -100;  a[0][1] = 200;  a[1][1] = 100;  a[2][1] = -100;   a[0][2] = 200;  a[1][2] = 200;  a[2][2] = -100;  a[0][3] = 100;  a[1][3] = 200;  a[2][3] = -100;   a[0][4] = 100;  a[1][4] = 100;  a[2][4] = -200;  a[0][5] = 200;  a[1][5] = 100;  a[2][5] = -200;   a[0][6] = 200;  a[1][6] = 200;  a[2][6] = -200;  a[0][7] = 100;  a[1][7] = 200;  a[2][7] = -200;    detectgraph(&gd,&gm);  initgraph(&gd,&gm, "c:\\tc\\bgi");   ymax = getmaxy();  xp = 300;  yp = 320;  zp = 100;   for(j=0; j<8; j++)  {  x = a[0][j];  y = a[1][j];  z = a[2][j];   b[0][j] = xp - ( (float)( x - xp )/(z - zp)) \* (zp);  b[1][j] = yp - ( (float)( y - yp )/(z - zp)) \* (zp);  }   */\*- front plane display -\*/*   for(j=0; j<3; j++)  {  x1=(int) b[0][j];  y1=(int) b[1][j];  x2=(int) b[0][j+1];  y2=(int) b[1][j+1];  line( x1,ymax-y1,x2,ymax-y2);   }  x1=(int) b[0][3];  y1=(int) b[1][3];  x2=(int) b[0][0];  y2=(int) b[1][0];  line( x1, ymax-y1, x2, ymax-y2);   */\*- back plane display -\*/*  setcolor(11);  for(j=4; j<7; j++)  {  x1=(int) b[0][j];  y1=(int) b[1][j];  x2=(int) b[0][j+1];  y2=(int) b[1][j+1];  line( x1, ymax-y1, x2, ymax-y2);   }  x1=(int) b[0][7];  y1=(int) b[1][7];  x2=(int) b[0][4];  y2=(int) b[1][4];  line( x1, ymax-y1, x2, ymax-y2);   setcolor(7);  for(i=0; i<4; i++)  {  x1=(int) b[0][i];  y1=(int) b[1][i];  x2=(int) b[0][4+i];  y2=(int) b[1][4+i];  line( x1, ymax-y1, x2, ymax-y2);  }    getch();  getch();  } |

**Alternative solution**

|  |
| --- |
| **#include <stdio.h>** **#include <stdlib.h>** **#include<graphics.h>** **#include<conio.h>** void **draw3d**(int fs,int x[20],int y[20],int tx,int ty,int d); void **draw3d**(int fs,int x[20],int y[20],int tx,int ty,int d) {  int i,j,k=0;  for(j=0; j<2; j++)  {  for(i=0; i<fs; i++)  {  if(i!=fs-1)  line(x[i]+tx+k,y[i]+ty-k,x[i+1]+tx+k,y[i+1]+ty-k);  else  line(x[i]+tx+k,y[i]+ty-k,x[0]+tx+k,y[0]+ty-k);  }  k=d;  }  for(i=0; i<fs; i++)  {  line(x[i]+tx,y[i]+ty,x[i]+tx+d,y[i]+ty-d);  } }  int **main**() {  int gd=DETECT,gm;  int x[20],y[20],tx=0,ty=0,i,fs,d;  initgraph(&gd,&gm,"");  printf("No of sides (front view only) : ");  scanf("%d",&fs);  printf("Co-ordinates : ");  for(i=0; i<fs; i++)  {  printf("(x%d,y%d)",i,i);  scanf("%d%d",&x[i],&y[i]);  }  printf("Depth :");  scanf("%d",&d);  draw3d(fs,x,y,tx,ty,d);  getch();*//front view*  setcolor(14);  for(i=0; i<fs; i++)  {  if(i!=fs-1)  line(x[i]+200,y[i],x[i+1]+200,y[i+1]);  else  line(x[i]+200,y[i],x[0]+200,y[0]);  }  getch();*//top view*  for(i=0; i<fs-1; i++)  {  line(x[i],300,x[i+1],300);  line(x[i],300+d\*2,x[i+1],300+d\*2);  line(x[i],300,x[i],300+d\*2);  line(x[i+1],300,x[i+1],300+d\*2);  }  getch();*//side view*  for(i=0; i<fs-1; i++)  {  line(10,y[i],10,y[i+1]);  line(10+d\*2,y[i],10+d\*2,y[i+1]);  line(10,y[i],10+d\*2,y[i]);  line(10,y[i+1],10+d\*2,y[i+1]);  }  getch();  closegraph(); } |

12 ) gif implementation

|  |
| --- |
| **#include <stdio.h>**  **#include <conio.h>**  **#include <graphics.h>**  **#include <dos.h>**   int **main**() {   int gdriver = DETECT, gmode, err;  int i, x, y;    initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");  err = graphresult();   if (err != grOk) {   printf("Graphics Error: %s",  grapherrormsg(err));  return 0;  }   x = 50;  y = getmaxy() - 100;   for (i = 0; i < 900; i++) {  */\* drawing the balloon first \*/*  setcolor(LIGHTRED);  setfillstyle(SOLID\_FILL, LIGHTRED);  arc(x, y, 0, 180, 40);   */\* arcs in the baloon \*/*  arc(x - 10, y, 0, 180, 10);  arc(x - 30, y, 0, 180, 10);  arc(x + 10, y, 0, 180, 10);  arc(x + 30, y, 0, 180, 10);  floodfill(x, y - 35, LIGHTRED);   */\* threads connecting basket and balloon \*/*  setcolor(DARKGRAY);  setfillstyle(SOLID\_FILL, DARKGRAY);  *// sector(x, y + 40, 0, 360, 30, 5);*  *// line(x - 40, y, x - 30, y + 40);*  *//line(x + 40, y, x + 30, y + 40);*  line(x, y, x, y + 40);  *//line(x - 20, y, x - 10, y + 40);*  *//line(x + 20, y, x + 10, y + 40);*    *// line(0, getmaxy(), x - 30, y + 40);*  if (i % 3 == 0) {  x = x - 1;  y = y + 1;  } else {  x = x + 1;  y = y - 1;  }  delay(100);  cleardevice();   }   getch();   */\* deallocate memory allocated for graphic screen \*/*  closegraph();  return 0;  } |

13.Implementation of 3D transformations, Translation, Scaling, reflection.

**>> TRANSLATION**

|  |
| --- |
| **\*/**  **TRANSLATION**  **/\***  **#include<stdio.h>**  **#include<conio.h>**  **#include<graphics.h>** **#include<bits/stdc++.h>** using namespace std;  int maxx=500, maxy=500, midx=250, midy=250;   int **main**()  {  int gd, gm, z, o, x1, x2, y1, y2;  detectgraph(&gd, &gm);  initgraph(&gd, &gm, "e:\tc\bgi");   bar3d(midx + 50, midy - 150, midx + 60, midy - 100, 10, 1);  line(midx, 0, midx, maxy);  line(0, midy, maxx, midy);   getch();    int x;  int y;  x=50;  y=50;   cout <<"ENTER TRANSFORMATION FACTOR"<<endl;  cin >>x>>y;     bar3d(midx + 50+x, midy - 150+y, midx + 60+x, midy - 100+y, 10, 1);  line(midx, 0, midx, maxy);  line(0, midy, maxx, midy);  getch();  } |

**>>SCALLING**

|  |
| --- |
| **#include<stdio.h>** **#include<conio.h>** **#include<graphics.h>** **#include<bits/stdc++.h>** using namespace std; int maxx=500, maxy=500, midx=250, midy=250; int **main**()  {  int gd, gm, o, x1, x2, y1, y2;  detectgraph(&gd, &gm);  initgraph(&gd, &gm, "e:\tc\bgi");   bar3d(midx + 50, midy - 150, midx + 60, midy - 100, 10, 1);  line(midx, 0, midx, maxy);  line(0, midy, maxx, midy);   getch();   cout <<"ENTER SCALLING FACTOR X Y Z"<<endl;   int x;  int y;  int z;  cin >>x>>y>>z;  cleardevice();     bar3d(midx + (x\*50), midy - (y\*150), midx + (x\* 60 ) , midy - (y\*100), 10\*z, 1);  line(midx, 0, midx, maxy);  line(0, midy, maxx, midy);  getch(); } |

**>>ROTATION**

|  |
| --- |
| **#include<bits/stdc++.h>** **#include<conio.h>** **#include<graphics.h>** **#include<dos.h>** using namespace std;  int **main**() {  int gd = DETECT, gm;  initgraph(&gd, &gm, "C:\\tc\\bgi");  int midx = 300;  int midy = 300;  line(300, 0, 300, 600);  line(0, 300, 600, 300);   int ang;  ang = 90;   int x1 = 100 \* cos(ang \* 3.14 / 180) - 20 \* sin(ang \* 3.14 / 180);  int y1 = 100 \* sin(ang \* 3.14 / 180) + 20 \* sin(ang \* 3.14 / 180);  int x2 = 60 \* cos(ang \* 3.14 / 180) - 90 \* sin(ang \* 3.14 / 180);  int y2 = 60 \* sin(ang \* 3.14 / 180) + 90 \* sin(ang \* 3.14 / 180);     bool X = 0, Y = false, Z = true;  if(Z)  {  printf("\n After rotating about z-axis\n");  bar3d(midx + 100, midy - 20, midx + 60, midy - 90, 20, 5);  bar3d(midx + x1, midy - y1, midx + x2, midy - y2, 20, 5);  }   if(X)  {  printf("\n After rotating about x-axis\n");  bar3d(midx + 100, midy - 20, midx + 60, midy - 90, 20, 5);  bar3d(midx + 100, midy - x1, midx + 60, midy - x2, 20, 5);   }  if(Y)  {  printf("\n After rotating about y-axis\n");  bar3d(midx + 100, midy - 20, midx + 60, midy - 90, 20, 5);  bar3d(midx + x1, midy - 20, midx + x2, midy - 90, 20, 5);  }    getch(); } |