**MADRAS INSTITUTE OF TECHNOLOGY** 

**ANNA UNIVERSITY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**IT7711- GRAPHICS AND MULTIMEDIA LABORATORY**

**RECORD**

**REGISTER NUMBER:** 2018506106

**NAME:** SANDHIYA R

**SEMESTER:** VII

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**ANNA UNIVERSITY, MIT CAMPUS**

CHROMEPET, CHENNAI – 600 044

**BONAFIDE CERTIFICATE**

Certified that the bonafide record of the practical work done by

Mr./Ms. **SANDHIYA R(2018506106)** of **Seventh** Semester **B.Tech Information Technology** during the academic period from **September 2021 to December 2021** in the **IT7711-Graphics and Multimedia Laboratory**

Date: Course Instructor

Examiner:

**TABLE OF CONTENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **DATE** | **TITLE** | **PAGE NO** | **MARKS** | **SIGNATURE** |
|  | 01-09-21 |  |  |  |  |
|  | 01-09-21 |  |  |  |  |
|  | 04-09-21 |  |  |  |  |
|  | 06-09-21 |  |  |  |  |
|  | 08-09-21 |  |  |  |  |
|  | 13-09-21 |  |  |  |  |
|  | 15-09-21 |  |  |  |  |
|  | 15-09-21 |  |  |  |  |
|  | 18-09-21 |  |  |  |  |
|  | 20-09-21 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| **Ex.No: 1** | **STUDY ON GRAPHICS FUNCTION** |
| **Date: 01-09-21** |

|  |  |
| --- | --- |
| **Ex.No: 2a** | **2D PRIMITIVES/OBJECTS**  **RAINBOW** |
| **Date: 01-09-21** |

**AIM :**

To create rainbow using graphics  in c

**PROGRAM**

#include<graphics.h>

int main()

{

int gd=DETECT,gm;

int x,y;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

//int x,y;

x=getmaxx()/2;

y=getmaxy()/2;

setcolor(5);

arc(x,y,0,180,40);

setcolor(13);

arc(x,y,0,180,50);

setcolor(1);

arc(x,y,0,180,60);

setcolor(2);

arc(x,y,0,180,70);

setcolor(14);

arc(x,y,0,180,80);

setcolor(12);

arc(x,y,0,180,90);

setcolor(4);

arc(x,y,0,180,100);

return 0;

}

**OUTPUT**



**RESULT**

Hence a rainbow is displayed using graphics functions in c

|  |  |
| --- | --- |
| **Ex.No: 2b** | **2D PRIMITIVES/OBJECTS**  **OLYMPIC SYMBOL** |
| **Date: 01-09-21** |

**AIM:**  
TO  animate Olympic Symbols using graphic functions in C

**PROGRAM**

#include<graphics.h>

#include<stdio.h>

#include<dos.h>

int main()

{

int gd = DETECT ,gm;

int i;

initgraph(&gd,&gm,"c:\\TURBOC3\\BGI");

setfillstyle(SOLID\_FILL, DARKGRAY);

floodfill(50,50,15);

for(i=0;i<1000;i++)

{

setcolor(BLUE);

circle(200,200+i,80);

setcolor(DARKGRAY);

circle(380,200+i,80);

setcolor(RED);

circle(560,200+i,80);

setcolor(YELLOW);

circle(300,270+i,80);

setcolor(GREEN);

circle(480,270+i,80);

delay(30);

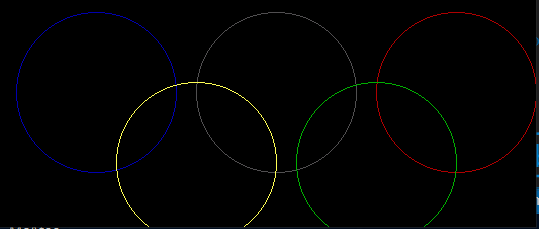
cleardevice();

}

return 0;

}

**OUTPUT**



**RESULT:**

Thus animated olympic symbol has been created using graphics functions in C

|  |  |
| --- | --- |
| **Ex.No: 2c** | **2D PRIMITIVES/OBJECTS**  **BOUNCING BALL** |
| **Date: 01-09-21** |

**AIM:**

To  simulate a bouncing ball using graphics function in C

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main() {

int gd = DETECT, gm = DETECT;

int x, y = 0, j, t = 400, c = 1;

initgraph(&gd, &gm, "");

setcolor(RED);

setfillstyle(SOLID\_FILL, RED);

for (x = 40; x < 602; x++) {

cleardevice();

circle(x, y, 30);

floodfill(x, y, RED);

delay(40);

if (y >= 400) {

c = 0;

t -= 20;

}

if (y <= (400 - t))

    c = 1;

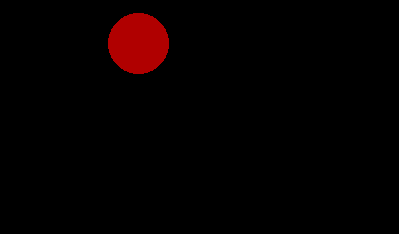
y = y + (c ? 15 : -15);

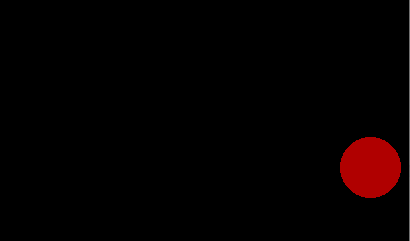
}

getch();

}

**OUTPUT:**





**RESULT:**

Thus a bouncing ball is simulated using graphics functions in C.

|  |  |
| --- | --- |
| **Ex.No: 2d** | **2D PRIMITIVES/OBJECTS**  **ADVERTISE VACCINATION** |
| **Date: 01-09-21** |

**AIM**

To advertise about COVID awareness using graphics function in C

**PROGRAM**

#include<graphics.h>

int main()

{

int gd=DETECT,gm;

int left=150,top=150;

int right=450,bottom=300;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

setbkcolor(BLUE);

setfillstyle(HATCH\_FILL,RED);

floodfill(151,151,RED);

rectangle(left,top,right,bottom);

outtextxy(200,160,"GET VACCINATED");

outtextxy(200,180,"PROTECT YOURSELF!!");

outtextxy(200,200," PROTECT YOUR SURROUNDING!!");

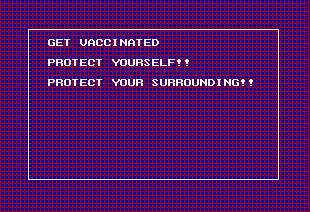
getch();

closegraph();

return 0;

}

**OUTPUT**



**RESULT**

Thus a program to advertise COVID is implemented using  graphics inbuilt function in C.

|  |  |
| --- | --- |
| **Ex.No: 3a** | **DDA ALGORITHM - SOLID LINES** |
| **Date: 04-09-21** |

AIM:

To Write a C Program to display your name in solid lines using DDA

algorithm.

CODE:

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

void dda(int x0, int y0,int x1, int y1)

{

int dx=x1-x0;

int dy=y1-y0;

int i,v1,v2;

int steps=abs(dx) > abs(dy) ? abs(dx) : abs(dy);

float xin=dx/(float)steps;

float yin=dy/(float)steps;

for(i=0;i<=steps;i++)

{

v1=(int)(x0 < 0 ?(x0-0.5):(x0+0.5));

v2=(int)(y0 < 0 ?(y0- 0.5):(y0 + 0.5));

putpixel(v1,v2,WHITE);

x0+=xin;

y0+=yin;

delay(5);

}

}

int main()

{

int gd=DETECT,gm;

int x0,y0,x1,y1;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

//setbkcolor(WHITE);

dda(50,150,100,200);

dda(150,150,100,200);

dda(200,200,250,150);

dda(300,200,250,150);

dda(220,175,270,175);

dda(320,200,320,150)

dda(450,150,400,150);

dda(400,150,400,175);

dda(400,175,450,175);

dda(450,175,450,200);

dda(450,200,400,200);

delay(2000);

return 0;

}

**OUTPUT**

****

RESULT

Thus the name is displayed using DDA Algorithm

|  |  |
| --- | --- |
| **Ex.No: 3b** | **DDA ALGORIHTM – DOTTED LINES** |
| **Date: 01-09-21** |

AIM:

To display the name in dotted lines using DDA Algorithm.

CODE:

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

void dda(int x0, int y0,int x1, int y1)

{

int dx=x1-x0;

int dy=y1-y0;

int i,v1,v2,m=0;

int steps=abs(dx) > abs(dy) ? abs(dx) : abs(dy);

float xin=dx/(float)steps;

float yin=dy/(float)steps;

for(i=0;i<=steps;i++)

{

v1=(int)(x0 < 0 ?(x0-0.5):(x0+0.5));

v2=(int)(y0 < 0 ?(y0- 0.5):(y0 + 0.5));

if(m==0){

putpixel(v1,v2,WHITE);

}

m=!m;

x0+=xin;

y0+=yin;

delay(5);

}

}

int main()

{

int gd=DETECT,gm;

int x0,y0,x1,y1;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

dda(50,150,100,200);

dda(150,150,100,200);

dda(200,200,250,150);

dda(300,200,250,150);

dda(220,175,270,175);

dda(320,200,320,150);

dda(450,150,400,150);

dda(400,150,400,175);

dda(400,175,450,175);

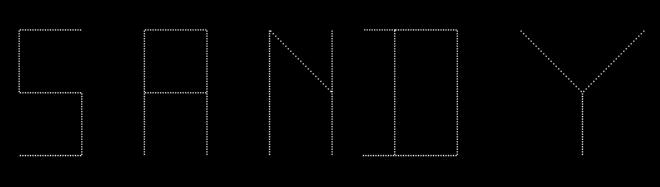
dda(450,175,450,200);

dda(450,200,400,200);

delay(2000);

return 0;}

OUTPUT:



RESULT

Thus the name is displayed in dotted lines using DDA Algorithm

|  |  |
| --- | --- |
| **Ex.No: 4a** | **BRESENHAM’S ALGORITHM – SPIDER WEB** |
| **Date: 06-09-21** |

**AIM :**

To build a spider web using Bresenham’s line drawing algorithm .

**PROGRAM :**

#include<graphics.h>

#include<conio.h>

void web(int x0, int y0, int x1, int y1)

{

int dx,dy,p,x,y;

dx= x1-x0;

dy= y1-y0;

x=x0;y=y0;

p= 2\*dy-dx;

while(x<x1)

{if(p>=0)

{putpixel(x,y,WHITE);

y++;

p= p+2\*dy-2\*dx;

}

else

{

putpixel(x,y,WHITE);

p= p+2\*dy;

}

x++;

}

}

void web1(int x0, int y0, int x1, int y1)

{int dx,dy,p,x,y;

dx= x0-x1;

dy= y1-y0;

x=x0;

y=y0;

p= 2\*dx-dy;

while(y<y1)

{if(p>=0)

{

putpixel(x,y,WHITE);

x--;

p= p+2\*dx-2\*dy;

}

else

{

putpixel(x,y,WHITE);

p= p+2\*dx;

}

y++;

}

}

void weby(int y0,int y1, int x)

{

int y;

for(y=y0;y<y1;y++)

{putpixel(x,y,WHITE);

}

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

web(50,200,350,200);

web(120,120,280,280);

web1(280,120,120,280);

//web(120,280,280,120);

weby(50,350,200);

web1(200,120,160,160);

web(200,120,240,240);

weby(160,240,240);

web1(240,240,200,280);

web(160,240,200,280);

weby(160,240,160);

web1(200,80,140,140);

web(200,80,260,260);

weby(140,260,260);

web1(260,260,200,320);

web(140,260,200,320);

weby(140,260,140);

web1(200,160,180,180);

web(200,160,220,180);

weby(180,220,220);

web1(220,220,200,240);

web(180,220,200,240);

weby(180,220,180);

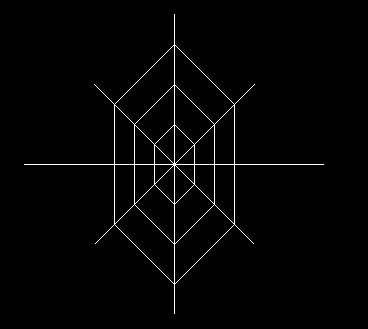
getch();

closegraph();

return 0;

}

**OUTPUT :**

****

**RESULT :**

Thus a spider web has been built using Bresenhman’s Line Algorithm.

|  |  |
| --- | --- |
| **Ex.No: 4b** | **BRESENHAM’S ALGORITHM – TABLE LAMP** |
| **Date: 06-09-21** |

**AIM**

To draw a table lamp using Bresenham’s algorithm and to animate it.

**PROGRAM:**#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void Bresenhams(int x0,int y0,int x1,int y1)

{int dx = x1 -x0;

int dy = y1 -y0;

int x = x0;

int y = y0;

if(x0<=x1 && y0<=y1)

{

if(abs(dx)>abs(dy))

{putpixel(x,y,WHITE);

int pk=(2\*abs(dy))-abs(dx);

for(int i=0;i<abs(dx);i++)

{

x=x+1;

if(pk<0)pk=pk+(2\*abs(dy));

else

{

y=y+1;

pk=pk+(2\*abs(dy))-(2\*abs(dx));

}

putpixel(x,y,WHITE);

}

}

else

{

putpixel(x,y,WHITE);

int pk=(2\*abs(dx))-abs(dy);

for(int i=0;i<abs(dy);i++)

{

y=y+1;

if(pk<0)pk=pk+(2\*abs(dx));

else{

x=x+1;

pk=pk+(2\*abs(dx))-(2\*abs(dy));

}

putpixel(x,y,WHITE);

}}}

else{

if(abs(dx)>abs(dy))

{

putpixel(x,y,WHITE);

int pk=(2\*abs(dy))-abs(dx);

for(int i=0;i<abs(dx);i++)

{

x=x+1;

if(pk<0)pk=pk+(2\*abs(dy));

else{

y=y-1;pk=pk+(2\*abs(dy))-(2 \* abs(dx));

}

putpixel(x,y,WHITE);

}

}

else{

putpixel(x,y,WHITE);

int pk=(2\*abs(dx))-abs(dy);

for(int i=0;i<abs(dy);i++)

{

y=y+1;

if(pk<0)pk=pk+(2\*abs(dx));

else

{

x=x-1;pk=pk+(2\*abs(dx))-(2\*abs(dy));

}

putpixel(x,y,WHITE);

}}} }

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

int trapezoid[10];

int check = 1;

trapezoid[0]=300;

trapezoid[1]=200;

trapezoid[2]=500;

trapezoid[3]=200;

trapezoid[4]=550;

trapezoid[5]=300;

trapezoid[6]=250;

trapezoid[7]=300;

trapezoid[8]=trapezoid[0];

trapezoid[9]=trapezoid[1];

Bresenhams(300,200,500,200);

Bresenhams(300,200,250,300);

Bresenhams(250,300,550,300);

Bresenhams(500,200,550,300);

Bresenhams(390,300,390,400);

Bresenhams(410,300,410,400);

Bresenhams(300,400,500,400);

Bresenhams(300,410,500,410);

Bresenhams(300,400,300,410);

Bresenhams(500,400,500,410);

for(int i=0;i<5;i++)

{

setcolor(getmaxcolor());

check ? setfillstyle(SOLID\_FILL,YELLOW):setfillstyle(SOLID\_FILL,BLACK);

fillpoly(5,trapezoid);

check = !check;

delay(1000);

}

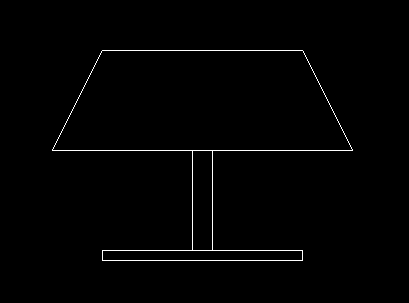
getch();

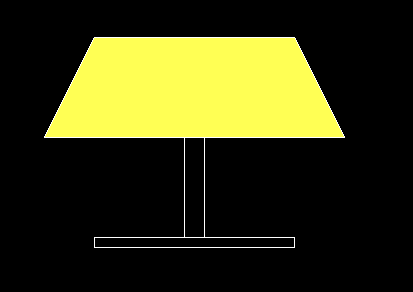
closegraph();

return 0;

}

**OUTPUT:**





**RESULT:**

Thus a table lamp has been created and animated using Bresenham’s algorithm.

|  |  |
| --- | --- |
| **Ex.No: 5a** | **MID POINT CIRCLE DRAWING ALGORIHTM**  **SMILEYS** |
| **Date: 08-09-21** |

**AIM :**

To create 5 different smileys using midpoint algorithm in C/C++. **PROGRAM :**

#include<graphics.h>

#include<conio.h>

void mcircle(int x\_centre, int y\_centre, int r,int c)

{  int x = r, y = 0;

   putpixel( x + x\_centre , y + y\_centre,c );

   if (r > 0)

    {

putpixel(x + x\_centre , -y + y\_centre,c);

putpixel(y + x\_centre ,x + y\_centre,c);

putpixel( -y + x\_centre , x + y\_centre,c );

    }

    int P = 1 - r;

    while (x > y)

    {

y++;

if (P <= 0)

    P = P + 2\*y + 1;

else

{

    x--;

    P = P + 2\*y - 2\*x + 1;

}

if (x < y)

    break;

putpixel(x + x\_centre , y + y\_centre ,c);

putpixel( -x + x\_centre ,  y + y\_centre,c);

putpixel(x + x\_centre ,  -y + y\_centre,c) ;

putpixel(-x + x\_centre , -y + y\_centre ,c);

if (x != y)

{

  putpixel( y + x\_centre ,  x + y\_centre,c );

  putpixel( -y + x\_centre , x + y\_centre ,c);

  putpixel(y + x\_centre ,  -x + y\_centre ,c);

  putpixel(-y + x\_centre ,  -x + y\_centre ,c);

}

    }

}

int main()

{       int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

//emoji1

mcircle(150, 150, 50,14);

       // setcolor(GREEN);

mcircle(132,132,9,11);

mcircle(170,132,9,11);

setcolor(RED);

line(128,164,174,164);

  //emoji2

mcircle(250,250,50,14);

mcircle(232,232,9,11);

mcircle(270,232,9,11);

setcolor(RED);

arc(250,275,0,180,20);

//emoji3

mcircle(350,350,50,14);

mcircle(332,332,9,11);

mcircle(370,332,9,11);

setcolor(RED);

arc(350,360,180,0,20);

//emoji4

mcircle(300,150,50,14);

mcircle(282,132,9,11);

mcircle(320,132,9,11);

mcircle(300,170,15,4);

//emoji5

mcircle(400,250,50,14);

setcolor(11);

line(360,240,384,240);

       line(412,240,435,240);

       setcolor(RED);

       line(380,270,423,270);

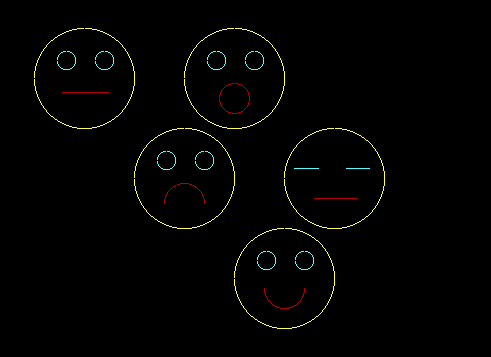
getch();

closegraph();

return 0;

}

**OUTPUT:**



**RESULT :**

Thus 5 different smileys are created using midpoint circle drawing algorithm in C/C++.

|  |  |
| --- | --- |
| **Ex.No: 5b** | **MID POINT CIRCLE DRAWING ALGORIHTM**  **BULLSEYE ARCHERY GAME** |
| **Date: 08-09-21** |

**AIM**

To animate Bullseye archery game using midpoint circle drawing algorithm in C/C++

**PROGRAM**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void mcircle(int x\_centre, int y\_centre, int r,int c)

{  int x = r, y = 0;

    // Printing the initial point on the axes

    // after translation

    putpixel( x + x\_centre , y + y\_centre,c );

    if (r > 0)

    {

putpixel(x + x\_centre , -y + y\_centre,c);

putpixel(y + x\_centre ,x + y\_centre,c);

putpixel( -y + x\_centre , x + y\_centre,c );

    }

    int P = 1 - r;

    while (x > y)

    {

y++;

if (P <= 0)

    P = P + 2\*y + 1;

else

{

    x--;

    P = P + 2\*y - 2\*x + 1;

}

if (x < y)

    break;

putpixel(x + x\_centre , y + y\_centre ,c);

putpixel( -x + x\_centre ,  y + y\_centre,c);

putpixel(x + x\_centre ,  -y + y\_centre,c) ;

putpixel(-x + x\_centre , -y + y\_centre ,c);

if (x != y)

{

  putpixel( y + x\_centre ,  x + y\_centre,c );

  putpixel( -y + x\_centre , x + y\_centre ,c);

  putpixel(y + x\_centre ,  -x + y\_centre ,c);

  putpixel(-y + x\_centre ,  -x + y\_centre ,c);

}

    }

}

int main()

{       int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

// setbkcolor(WHITE);

for(int x=10;x<300;x+=10)

{  cleardevice();

mcircle(101,200,100,4);

mcircle(101,200,97,4);

mcircle(101,200,80,1);

mcircle(101,200,77,1);

       // mcircle(101,200,80,4);

       // mcircle(101,200,77,4);

mcircle(101,200,70,1);

mcircle(101,200,67,1);

mcircle(101,200,60,4);

mcircle(101,200,57,4);

mcircle(101,200,50,1);

mcircle(101,200,47,1);

mcircle(101,200,40,4);

mcircle(101,200,37,4);

mcircle(101,200,30,1);

mcircle(101,200,27,1);

mcircle(101,200,20,4);

mcircle(101,200,17,4);

mcircle(101,200,10,4);

mcircle(101,200,7,4);

//       cleardevice();

setcolor(WHITE);

arc(600,200,90,270,40);

setcolor(RED);

line(500-x,200,580-x,200);

line(500-x,200,508-x,210);

// line(500,200,508,240);

line(500-x,200,508-x,190);

delay(100);

}

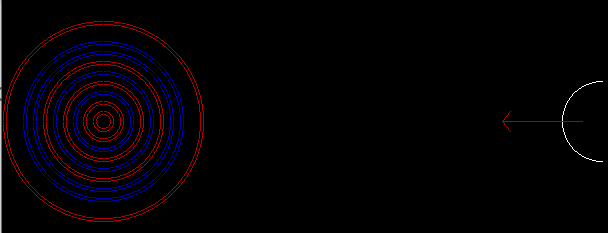
getch();

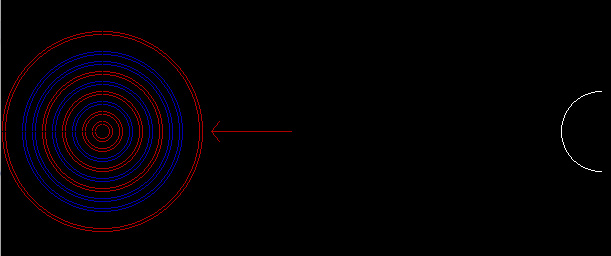
closegraph();

return 0;

}

**OUTPUT**

****

****

**RESULT**

Thus  Bullseye archery game is animated using midpoint circle drawing algorithm in C/C++

|  |  |
| --- | --- |
| **Ex.No: 6a** | **MIDPOINT ELLIPSE DRAWING ALGORIHTM**  **TOYOTA LOGO** |
| **Date: 13-09-21** |

**AIM**

To advertise Toyota by animating Toyota logo by using midpoint ellipse drawing algorithm

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

void mid(long x\_center, long y\_center,long a,long b,int c);

void main(){

      long x,y,x\_center,y\_center;

      long a\_sqr,b\_sqr, fx,fy, d,a,b,tmp1,tmp2;

      int gd=DETECT,gm;

     initgraph(&gd,&gm,"C:/TURBOC3/BGI");

    outtextxy(390,200,"Gear Up..");

    outtextxy(390,225,"Lets Speed Up..");

    setcolor(4);

    settextstyle(1,0,6);

    outtextxy(390,125,"TOYOTA");

  // outtextxy(390,135,"Let's Speed Up");

  mid(300,190,60,50,4);

    mid(300,167,45,25,4);

    mid(300,190,20,50,4);

    getch();

    closegraph();

    }

void mid(long x\_center,long y\_center,long a, long b,int c)

{

  long x=0;

   long y=b;

  long a\_sqr=a\*a;

  long b\_sqr=b\*b;

   long fx=2\*b\_sqr\*x;

   long fy=2\*a\_sqr\*y;

 long d=b\_sqr-(a\_sqr\*b)+(a\_sqr\*0.25);

  do

   {

  putpixel(x\_center+x,y\_center+y,c);

  putpixel(x\_center-x,y\_center-y,c);

  putpixel(x\_center+x,y\_center-y,c);

  putpixel(x\_center-x,y\_center+y,c);

   if(d<0)

    {

  d=d+fx+b\_sqr;

    }

   else

  {

  y=y-1;

  d=d+fx+-fy+b\_sqr;

  fy=fy-(2\*a\_sqr);

  }

  x=x+1;

  fx=fx+(2\*b\_sqr);

  delay(25);

   }

   while(fx<fy);

   long tmp1=(x+0.5)\*(x+0.5);

   long tmp2=(y-1)\*(y-1);

   d=b\_sqr\*tmp1+a\_sqr\*tmp2-(a\_sqr\*b\_sqr);

   do

   {

  putpixel(x\_center+x,y\_center+y,c);

  putpixel(x\_center-x,y\_center-y,c);

  putpixel(x\_center+x,y\_center-y,c);

  putpixel(x\_center-x,y\_center+y,c);

   if(d>=0)

  d=d-fy+a\_sqr;

   else

  {

  x=x+1;

  d=d+fx-fy+a\_sqr;

  fx=fx+(2\*b\_sqr);

  }

   y=y-1;

   fy=fy-(2\*a\_sqr);

   }

   while(y>0);

}

**OUTPUT:**



**RESULT :**

Thus the Toyota logo has been animated and advertised using midpoint ellipse drawing algorithm.

|  |  |
| --- | --- |
| **Ex.No: 6b** | **MIDPOINT ELLIPSE DRAWING ALGORIHTM**  **ELLIPTICAL POOL TABLE** |
| **Date: 13-09-21** |

**AIM**

To design an elliptical pool table by  midpoint ellipse drawing algorithm in C/C++

**PROGRAM**

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

void mid(long x\_center, long y\_center,long a,long b,int c);

void main(){

      long x,y,x\_center,y\_center;

      long a\_sqr,b\_sqr, fx,fy, d,a,b,tmp1,tmp2;

      int gd=DETECT,gm;

     initgraph(&gd,&gm,"C:/TURBOC3/BGI");

    mid(300,300,160,105,6);

    mid(300,300,155,100,6);

    setfillstyle(SOLID\_FILL,GREEN);

    fillellipse(300,300,155,100);

    mid(400,300,10,12,0);

    setfillstyle(SOLID\_FILL,BLACK);

    fillellipse(400,300,10,12);

    mid(200,300,10,10,4);

    mid(208,275,10,10,15);

    mid(208,325,10,10,1);

    setfillstyle(SOLID\_FILL,RED);

    fillellipse(200,300,10,10);

    setfillstyle(SOLID\_FILL,WHITE);

    fillellipse(208,275,10,10);

    setfillstyle(SOLID\_FILL,BLUE);

    fillellipse(208,325,10,10);

    getch();

    closegraph();

    }

void mid(long x\_center,long y\_center,long a, long b,int c)

{

  long x=0;

   long y=b;

  long a\_sqr=a\*a;

  long b\_sqr=b\*b;

   long fx=2\*b\_sqr\*x;

   long fy=2\*a\_sqr\*y;

 long d=b\_sqr-(a\_sqr\*b)+(a\_sqr\*0.25);

  do

   {

  putpixel(x\_center+x,y\_center+y,c);

  putpixel(x\_center-x,y\_center-y,c);

  putpixel(x\_center+x,y\_center-y,c);

  putpixel(x\_center-x,y\_center+y,c);

   if(d<0)

    {

  d=d+fx+b\_sqr;

    }

   else

  {

  y=y-1;

  d=d+fx+-fy+b\_sqr;

  fy=fy-(2\*a\_sqr);

  }

  x=x+1;

  fx=fx+(2\*b\_sqr);

  delay(25);

   }

   while(fx<fy);

   long tmp1=(x+0.5)\*(x+0.5);

   long tmp2=(y-1)\*(y-1);

   d=b\_sqr\*tmp1+a\_sqr\*tmp2-(a\_sqr\*b\_sqr);

   do

   {

  putpixel(x\_center+x,y\_center+y,c);

  putpixel(x\_center-x,y\_center-y,c);

  putpixel(x\_center+x,y\_center-y,c);

  putpixel(x\_center-x,y\_center+y,c);

   if(d>=0)

  d=d-fy+a\_sqr;

   else

  {

  x=x+1;

  d=d+fx-fy+a\_sqr;

  fx=fx+(2\*b\_sqr);

  }

   y=y-1;

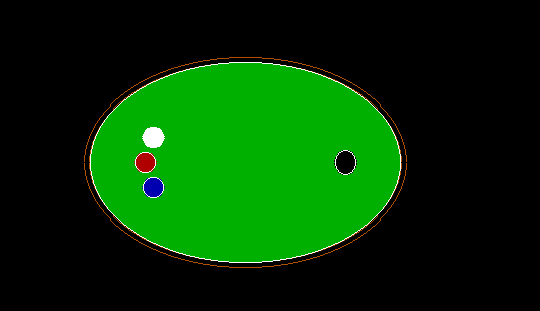
   fy=fy-(2\*a\_sqr);

   }

   while(y>0);

}

**OUTPUT :**



**RESULT:**

Thus an elliptical pool table has been designed using midpoint ellipse drawing algorithm.

|  |  |
| --- | --- |
| **Ex.No: 7** | **WINDOW TO VIEWPORT TRANSFORMATION** |
| **Date: 15-09-21** |

**AIM :**

To implement  window to viewport transformation of the given example.

**PROGRAM:**  
#include<graphics.h>

#include<conio.h>

#include<stdio.h>

int main ()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int W\_xmin,W\_xmax,W\_ymin,W\_ymax;

int V\_xmax,V\_ymax,V\_xmin,V\_ymin;

W\_xmin = 20;

W\_xmax = 340;

W\_ymin = 20;

W\_ymax = 400;

rectangle (W\_xmin, W\_ymin, W\_xmax, W\_ymax);

V\_xmin = 380;

V\_ymin = 100;

V\_xmax = 620;

V\_ymax = 320;

rectangle (V\_xmin, V\_ymin, V\_xmax, V\_ymax);

int x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22;

int y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15,y16,y17,y18,y19,y20,y21,y22;

ellipse(104,150,0,360,12,8);

ellipse(104,150,0,360,6,2);

x1=60;

x2=140;

x3=220;

x4=300;

x5=300;

x6=220;

x7=140;

x8=60;

x9=120;

x10=55;

x16=55;

x11=70;

x17=70;

x12=65;

x18=65;

x13=80;

x19=80;

x14=75;

x20=75;

x15=90;

x21=90;

x22=120;

y1=160;

y2=100;

y3=160;

y4=80;

y5=320;

y6=240;

y7=300;

y8=240;

y9=200;

y10=224;

y16=176;

y11=234;

y17=166;

y12=217;

y18=183;

y13=227;

y19=173;

y14=210;

y20=190;

y15=220;

y21=180;

y22=280;

line (x1, y1, x2, y2);

line (x2, y2, x3, y3);

line (x3, y3, x4, y4);

line (x4, y4, x5, y5);

line (x5, y5, x6, y6);

line (x6, y6, x7, y7);

line (x7, y7, x8, y8);

line (x8, y8, x9, y9);

line (x9, y9, x1, y1);

line (x1, y1, x16, y16);

line (x16, y16, x17, y17);

line (x17, y17, x18, y18);

line (x18, y18, x19, y19);

line (x19, y19, x20, y20);

line (x20, y20, x21, y21);

line (x8, y8, x10, y10);

line (x10, y10, x11, y11);

line (x11, y11, x12, y12);

line (x12, y12, x13, y13);

line (x13, y13, x14, y14);

line (x14, y14, x15, y15);

W\_xmin = 0;

W\_xmax = 120;

W\_ymin = 200;

W\_ymax = 280;

float sx, sy;

sx = (float) (V\_xmax - V\_xmin) / (W\_xmax - W\_xmin);

sy = (float) (V\_ymax - V\_ymin) / (W\_ymax - W\_ymin);

x8 = V\_xmin + (float) ((x8 - W\_xmin) \* sx);

x9 = V\_xmin + (float) ((x9 - W\_xmin) \* sx);

x10 = V\_xmin + (float) ((x10 - W\_xmin) \* sx);

x11 = V\_xmin + (float) ((x11 - W\_xmin) \* sx);

x12 = V\_xmin + (float) ((x12 - W\_xmin) \* sx);

x13 = V\_xmin + (float) ((x13 - W\_xmin) \* sx);

x14 = V\_xmin + (float) ((x14 - W\_xmin) \* sx);

x15 = V\_xmin + (float) ((x15 - W\_xmin) \* sx);

x22 = V\_xmin + (float) ((x22 - W\_xmin) \* sx);

y8 = V\_ymin + (float) ((y8 - W\_ymin) \* sy);

y9 = V\_ymin + (float) ((y9 - W\_ymin) \* sy);

y10 = V\_ymin + (float) ((y10 - W\_ymin) \* sy);

y11 = V\_ymin + (float) ((y11 - W\_ymin) \* sy);

y12 = V\_ymin + (float) ((y12 - W\_ymin) \* sy);

y13 = V\_ymin + (float) ((y13 - W\_ymin) \* sy);

y14 = V\_ymin + (float) ((y14 - W\_ymin) \* sy);

y15 = V\_ymin + (float) ((y15 - W\_ymin) \* sy);

y22 = V\_ymin + (float) ((y22 - W\_ymin) \* sy);

line (x8, y8, x9, y9);

line (x8, y8, x22, y22);

line (x8, y8, x10, y10);

line (x10, y10, x11, y11);

line (x11, y11, x12, y12);

line (x12, y12, x13, y13);

line (x13, y13, x14, y14);

line (x14, y14, x15, y15);

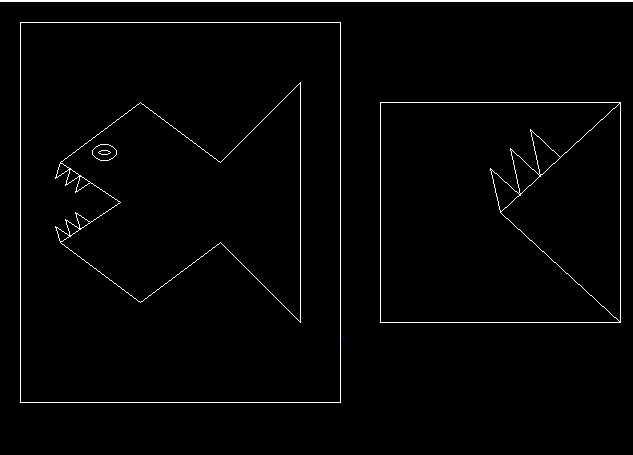
getch ();

closegraph ();

return 0;

}

**OUTPUT:**



**RESULT:**

Thus window to viewport transformation of the given example is obtained.

|  |  |
| --- | --- |
| **Ex.No: 8** | **COHEN SUTHERLAND CLIPPING ALGORITHM** |
| **Date: 15-09-21** |

**AIM:**

To write a C program to implement Cohen Sutherland Clipping algorithm for the given example and to clip the lines with respect to the clipping window.

**PROGRAM**:

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <dos.h>

 int x\_min = 100;

 int x\_max = 300;

 int y\_min = 100;

 int y\_max = 300;

 int INSIDE = 0;

 int LEFT = 1;

 int RIGHT = 2;

 int BOTTOM = 4;

 int ABOVE = 8;

int reg(double x, double y)

{

 int code = 0;

 if (x < x\_min)

 {

 code = code | LEFT;

 }

 else if (x > x\_max)

 {

 code = code | RIGHT;

 }

 if (y < y\_min)

 {

 code = code | BOTTOM;

 }

 else if (y > y\_max)

 {

 code = code | ABOVE;

 }

 return code;

}

void clip (double x1, double y1, double x2, double y2)

{

 int code1 = reg(x1, y1);

 int code2 = reg(x2, y2);

 int accept = 0;

 while (accept != 1)

 {

if (code1 == 0 && code2 == 0)

{

 setcolor(RED);

 line(x1,y1,x2,y2);

 accept = 1;

}

else if (code1 & code2)

{

 accept = 1;

}

else

{

 int line\_outside;

 double x, y;

 if (code1 != 0)

 {

line\_outside = code1;

 }

 else

 {

line\_outside = code2;

 }

 if (line\_outside & LEFT)

{

y = y1 + ((y2 - y1) \* (x\_min - x1))/(x2-x1);

x = x\_min;

 }

 else if (line\_outside & RIGHT)

 {

y = y1 + ((y2 - y1) \* (x\_max - x1))/(x2-x1);

x = x\_max;

 }

 else if (line\_outside & BOTTOM)

 {

x = x1 + ((x2 - x1) \* (y\_min - y1))/(y2-y1);

y = y\_min;

 }

 else if (line\_outside & ABOVE)

 {

x = x1 + ((x2 - x1) \* (y\_max - y1))/(y2-y1);

y = y\_max;

 }

 if (line\_outside == code1)

 {

x1 = x;

y1 = y;

code1 = reg(x1, y1);

 }

 else if (line\_outside == code2)

 {

x2 = x;

y2 = y;

code2 = reg(x2, y2);

 }

}

 }

}

int main ()

{

 int gd = DETECT, gm;

 initgraph (&gd, &gm, "C:/TURBOC3/BGI");

 setcolor(BLUE);

 rectangle(x\_min,y\_min,x\_max,y\_max);

// setbkcolor(WHITE);

 setcolor(BLUE);

 line(150,200,270,140);

 clip(150,200,270,140);

 setcolor(BLUE);

 line(30,200,70,100);

 clip(30,200,70,100);

 setcolor(BLUE);

 line(50,280,200,220);

 clip(50,280,200,220);

 setcolor(BLUE);

 line(220,320,350,220);

 clip(220,320,350,220);

 setcolor(BLUE);

 line(250,20,410,90);

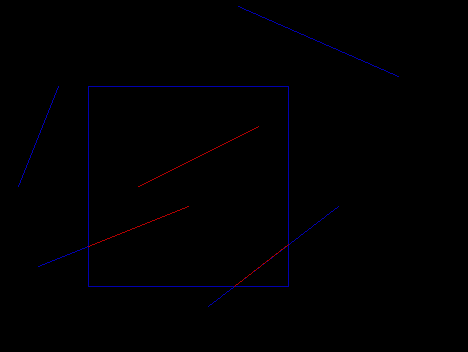
 clip(250,20,410,90);

 getch(); closegraph();

 return 0;

}

**OUTPUT:**



**RESULT :**  
Thus Cohen Sutherland Clipping algorithm is implemented for the given example  the lines are clipped  with respect to the clipping window.

|  |  |
| --- | --- |
| **Ex.No: 9** | **LIANG BARSKY CLIPPING ALGORITHM** |
| **Date: 18-09-21** |

**AIM**

To write a program to implement liangBarsky clipping algorithm to clip the line in the given example.

**PROGRAM:**

#include<conio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

void clip(int a1, int b1, int a2, int b2)

{

int x1,y1,x2,y2,i;

int xx1,xx2,yy1,yy2,dx,dy;

float t1,t2,p[4],q[4],temp;

int xmin,xmax,ymin,ymax;

xmin=60;

ymin=40;

xmax=140;

ymax=100;

x1=a1;

y1=b1;

x2=a2;

y2=b2;

dx=x2-x1;

dy=y2-y1;

p[0]=-dx;

p[1]=dx;

p[2]=-dy;

p[3]=dy;

q[0]=x1-xmin;

q[1]=xmax-x1;

q[2]=y1-ymin;

q[3]=ymax-y1;

for(i=0;i<4;i++)

{

if(p[i]==0)

{

if(q[i]>=0)

{

if(i<2)

{

if(y1<ymin)

{

y1=ymin;

}

if(y2>ymax)

{

y2=ymax;

}

line(x1,y1,x2,y2);

}

if(i>1)

{

if(x1<xmin)

{

x1=xmin;

}

if(x2>xmax)

{

x2=xmax;

}

line(x1,y1,x2,y2);

}

}

}

}

t1=0;

t2=1;

for(i=0;i<4;i++)

{

temp=q[i]/p[i];

if(p[i]<0)

{

if(t1<=temp)

t1=temp;

}

else

{

if(t2>temp)

t2=temp;

}

}

if(t1<t2)

{

xx1 = x1 + t1 \* p[1];

xx2 = x1 + t2 \* p[1];

yy1 = y1 + t1 \* p[3];

yy2 = y1 + t2 \* p[3];

setcolor(WHITE);

line(xx1,yy1,xx2,yy2);

}

delay(2000);

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

int xmin,xmax,ymin,ymax;

xmin=60;

ymin=40;

xmax=140;

ymax=100;

rectangle(xmin,ymin,xmax,ymax);

rectangle(5,5,190,150);

setcolor(RED);

line(70,80,80,60);

line(110,60,130,20);

line(10,65,40,10);

line(75,120,160,45);

line(120,140,170,80);

delay(2000);

clip(70,80,80,60);

clip(110,60,130,20);

clip(10,65,40,10);

clip(75,120,160,45);

clip(120,140,170,80);

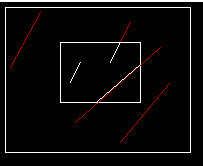
getch();

closegraph();

return 0;

}

**OUTPUT :**



**RESULT :**

Thus the liangBarsky clipping algorithm  is implemented to clip the lines in the given example.

|  |  |
| --- | --- |
| **Ex.No: 10 a** | **2D TRANSFORMATION OF OBJECTS** |
| **Date: 20-09-21** |

**AIM**

To implement 2D transformation for the given object

 i) translation

ii)Rotation(origin,pivot point)

iii)Scaling(origin, pivot point)

iv)Reflection(x,y,origin,x=y,x=-y);

v)Shearing(x,y)

***i)TRANSLATION***

**PROGRAM:**

#include<graphics.h>

#include<conio.h>

void translate ( int x1,int y1,int x2, int y2)

{

   int t1=40;

   int t2=20;

   setcolor(WHITE);

   line(x1,y1,x2,y2);

    setcolor(RED);

    line(x1+t1,y1+t2, x2+t1, y2+t2);

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

    int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

    translate(x1, y1, x2, y2);

    translate(x1, y1, x3, y3);

    translate(x3, y3, x4, y4);

    translate(x4,y4,x5,y5);

    translate(x5,y5,x6,y6);

    translate(x6,y6,x7,y7);

    translate(x7,y7,x8,y8);

    translate(x8,y8,x9,y9);

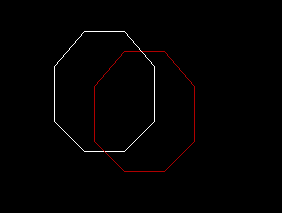
 getch();

 closegraph();

 return 0;

}

**OUTPUT**



***ii)ROTATION - with origin***

**PROGRAM :**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

void rotate (int &x1, int &y1)

{

    double s,c, angle;

     angle=20;

    c = cos(angle \*M\_PI/180);

    s = sin(angle \*M\_PI/180);

    x1 = floor(x1 \* c + y1 \* s);

    y1 = floor(-x1 \* s + y1 \* c);

}

int main()

{

int gd=DETECT,gm;

//float x1,y1,x2,y2,sx,sy;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

rotate(x1,y1);

rotate(x2,y2);

rotate(x3,y3);

rotate(x4,y4);

rotate(x5,y5);

rotate(x6,y6);

rotate(x7,y7);

rotate(x8,y8);

rotate(x9,y9);

setcolor(RED);

   line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

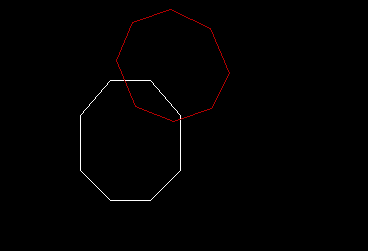
    line(x8,y8,x9,y9);

getch();

closegraph();

return 0;

}

**OUTPUT:**  


***ii)ROTATION - with pivot point :***

**PROGRAM :**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int fx=100;

int fy=100;

void rotate (int &x1, int &y1)

{

    double s,c, angle;

   angle=20;

    c = cos(angle \*M\_PI/180);

    s = sin(angle \*M\_PI/180);

 float sx=x1-fx;

 float sy=y1-fy;

  x1=fx+(sx \* c - sy \*s);

   y1=fy+(sx\*s + sy\*c);

}

int main()

{

int gd=DETECT,gm;

//float x1,y1,x2,y2,sx,sy;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

rotate(x1,y1);

rotate(x2,y2);

rotate(x3,y3);

rotate(x4,y4);

rotate(x5,y5);

rotate(x6,y6);

rotate(x7,y7);

rotate(x8,y8);

rotate(x9,y9);

setcolor(RED);

   line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

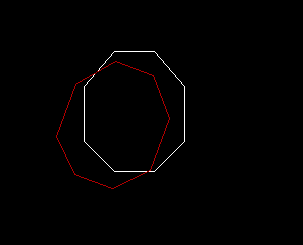
    line(x8,y8,x9,y9);

getch();

closegraph();

return 0;

}

**OUTPUT :**  


***iii)SCALING - with Origin***  
  
**PROGRAM**

#include <iostream.h>

#include <conio.h>

#include <graphics.h>

void scale(int &x1,int &y1)

{

int sx=2;

int sy=2;

x1=x1\*sx;

y1=y1\*sy;

}

void main()

{

int gd=DETECT,gm;

//float x1,y1,x2,y2,sx,sy;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

scale(x1,y1);

scale(x2,y2);

scale(x3,y3);

scale(x4,y4);

scale(x5,y5);

scale(x6,y6);

scale(x7,y7);

scale(x8,y8);

scale(x9,y9);

setcolor(RED);

    line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

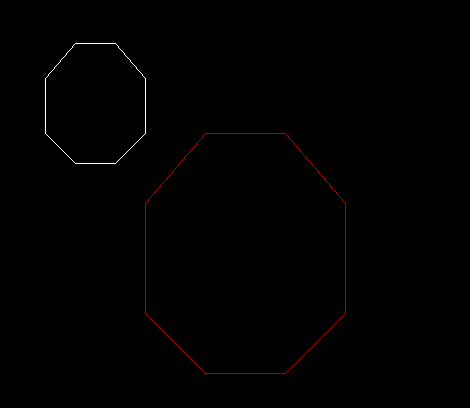
    line(x8,y8,x9,y9);

getch();

closegraph();

}

**OUTPUT**



***iii)SCALING - with fixed point***

**PROGRAM :**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

int fx=100;

int fy=160;

void xscale(int &x, int &y)

{

int xs=2;

int ys=2;

x= fx+ (x-fx)\*xs;

y= fy+(y-fy)\*ys;

}

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

xscale(x1,y1);

xscale(x2,y2);

xscale(x3,y3);

xscale(x4,y4);

xscale(x5,y5);

xscale(x6,y6);

xscale(x7,y7);

xscale(x8,y8);

xscale(x9,y9);

setcolor(RED);

    line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

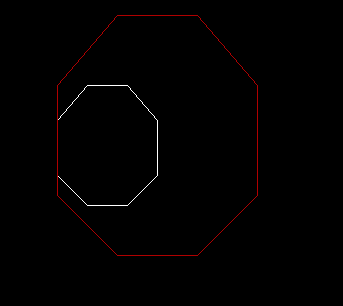
    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

getch();

closegraph();

}

**OUTPUT :**  


***iv)REFLECTION - about X axis, Y axis, Origin***

**PROGRAM**

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

// Driver Code

void main()

{

    int gm, gd = DETECT, ax, x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    cleardevice();

   // line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());

   // line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

    outtextxy(x3,y3-10,"ORIGINAL");

    setcolor(14);

    line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

//    getch();

    // Reflection along origin i.e.,

 setcolor(WHITE);

     outtextxy(getmaxx()-x8,getmaxx()-y8-10,"about origin");

    setcolor(4);

    line(getmaxx() - x1, getmaxy() - y1, getmaxx() - x2, getmaxy() - y2);

   line(getmaxx() - x1, getmaxy() - y1,getmaxx() - x3, getmaxy() - y3);

   line(getmaxx() - x3, getmaxy() - y3,getmaxx() - x4, getmaxy() - y4);

   line(getmaxx() - x4, getmaxy() - y4,getmaxx() - x5, getmaxy() - y5);

   line(getmaxx() - x5, getmaxy() - y5,getmaxx() - x6, getmaxy() - y6);

   line(getmaxx() - x6, getmaxy() - y6,getmaxx() - x7, getmaxy() - y7);

   line(getmaxx() - x7, getmaxy() - y7,getmaxx() - x8, getmaxy() - y8);

   line(getmaxx() - x8, getmaxy() - y8,getmaxx() - x9, getmaxy() - y9);

  // getch();

    // Reflection along x-axis i.e.,

 setcolor(WHITE);

    outtextxy(getmaxx()-x3,y3-10,"about x axis");

    setcolor(3);

    line(getmaxx() - x1, y1, getmaxx() - x2, y2);

    line(getmaxx() - x1, y1, getmaxx() - x3, y3);

    line(getmaxx() - x3, y3, getmaxx() - x4, y4);

    line(getmaxx() - x4, y4, getmaxx() - x5, y5);

    line(getmaxx() - x5, y5, getmaxx() - x6, y6);

    line(getmaxx() - x6, y6, getmaxx() - x7, y7);

    line(getmaxx() - x7, y7, getmaxx() - x8, y8);

    line(getmaxx() - x8, y8, getmaxx() - x9, y9);

    // getch();

    // Reflection along y-axis i.e.,

   setcolor(WHITE);

   outtextxy(x3,getmaxy()-y3+10,"about y axis");

    setcolor(2);

    line(x1, getmaxy() - y1, x2, getmaxy() - y2);

    line(x1, getmaxy() - y1, x3, getmaxy() - y3);

    line(x3, getmaxy() - y3, x4, getmaxy() - y4);

    line(x4, getmaxy() - y4, x5, getmaxy() - y5);

    line(x5, getmaxy() - y5, x6, getmaxy() - y6);

    line(x6, getmaxy() - y6, x7, getmaxy() - y7);

    line(x7, getmaxy() - y7, x8, getmaxy() - y8);

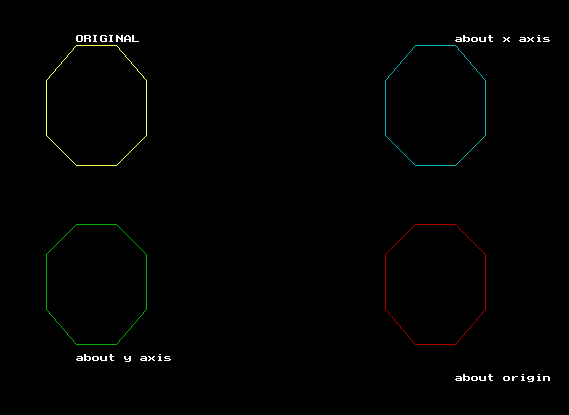
    line(x8, getmaxy() - y8, x9, getmaxy() - y9);

   getch();

    closegraph();

}

**OUTPUT**



***iv)REFLECTION - about X =Y ,X = -Y***

**PROGRAM :**

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

// Driver Code

void main()

{

    int gm, gd = DETECT, ax, x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    cleardevice();

   // line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());

   // line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

    outtextxy(x3,y3-10,"ORIGINAL");

    setcolor(WHITE);

    line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

  setcolor(RED);

  outtextxy(getmaxx()-y1-70,getmaxy()-x1+10,"X = - Y  line");

    line(getmaxx()-y1,getmaxy()-x1, getmaxx()-y2,getmaxy()- x2);

    line(getmaxx()-y1,getmaxy()- x1,getmaxx()-y3,getmaxy()- x3);

    line(getmaxx()-y3, getmaxy()-x3,getmaxx()- y4, getmaxy()-x4);

    line(getmaxx()-y4,getmaxy()-x4,getmaxx()-y5,getmaxy()-x5);

    line(getmaxx()-y5,getmaxy()-x5,getmaxx()-y6,getmaxy()-x6);

    line(getmaxx()-y6,getmaxy()-x6,getmaxx()-y7,getmaxy()-x7);

    line(getmaxx()-y7,getmaxy()-x7,getmaxx()-y8,getmaxy()-x8);

    line(getmaxx()-y8,getmaxy()-x8,getmaxx()-y9,getmaxy()-x9);

//setcolor(BLUE);

    outtextxy(y1-100,x1,"X = Y line");

    line(y1, x1, y2, x2);

    line(y1, x1, y3, x3);

    line(y3, x3, y4, x4);

    line(y4,x4,y5,x5);

    line(y5,x5,y6,x6);

    line(y6,x6,y7,x7);

    line(y7,x7,y8,x8);

    line(y8,x8,y9,x9);

getch();

     closegraph();

}

**OUTPUT:**



***v)SHEARING - X***

**PROGRAM**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

void xshear(int &x, int &y)

{

int xs=1;

x=x+ y\*xs;

}

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

xshear(x1,y1);

xshear(x2,y2);

xshear(x3,y3);

xshear(x4,y4);

xshear(x5,y5);

xshear(x6,y6);

xshear(x7,y7);

xshear(x8,y8);

xshear(x9,y9);

setcolor(RED);

    line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

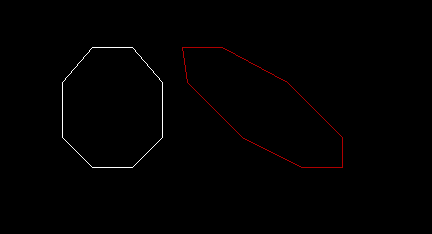
    line(x8,y8,x9,y9);

getch();

closegraph();

}

**OUTPUT:**



***v)SHEARING - Y***

**PROGRAM**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

void yshear(int &x, int &y)

{

int ys=1;

y=y+ x\*ys;

}

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1 = 100;

    int x2 = 100, x3=130, y1 = 125;

    int y2 = 180, y3=90;

    int x4=170,y4=90,x5=200,y5=125,x6=200,y6=180,x7=170,y7=210,x8=130,y8=210;

int x9=100,y9=180;

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

    line(x8,y8,x9,y9);

yshear(x1,y1);

yshear(x2,y2);

yshear(x3,y3);

yshear(x4,y4);

yshear(x5,y5);

yshear(x6,y6);

yshear(x7,y7);

yshear(x8,y8);

yshear(x9,y9);

setcolor(RED);

line(x1, y1, x2, y2);

    line(x1, y1, x3, y3);

    line(x3, y3, x4, y4);

    line(x4,y4,x5,y5);

    line(x5,y5,x6,y6);

    line(x6,y6,x7,y7);

    line(x7,y7,x8,y8);

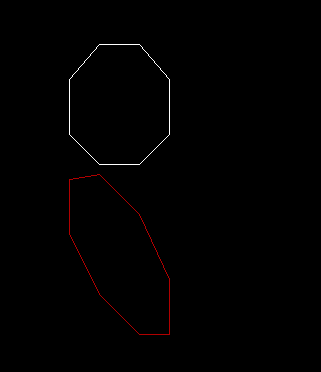
    line(x8,y8,x9,y9);

getch();

closegraph();

}

**OUTPUT**



**RESULT**

Thus 2D transformation for all the cases of  the given object has been implemented using graphics inbuilt functions in C++.

|  |  |
| --- | --- |
| **Ex.No: 10 b** | **2D COMPOSITE TRANSFORMATION OF OBJECTS** |
| **Date: 20-09-21** |

**AIM**

To implement Composite 2D transformation for the given object

i) Two successive Translation

ii)Rotation ( origin, pivot point)

iii) Scaling ( origin , fixed point)

***i)TRANSLATION***

**PROGRAM**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

void translate ( int &x1,int &y1)

{

   int t1=40;

   int t2=20;

   x1+=t1;

   y1+=t2;

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1,x2,x3,x4,x5,x6;

int y1,y2,y3,y4,y5,y6;

x1=100;

y1=200;

x2=200;

y2=200;

x3=150;

y3=150;

x4=150;

y4=300;

x5=140;

y5=310;

x6=160;

y6=310;

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

setcolor(RED);

//translate 1

translate(x1,y1);

translate(x2,y2);

translate(x3,y3);

translate(x4,y4);

translate(x5,y5);

translate(x6,y6);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

setcolor(YELLOW);

//translate 2

translate(x1,y1);

translate(x2,y2);

translate(x3,y3);

translate(x4,y4);

translate(x5,y5);

translate(x6,y6);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

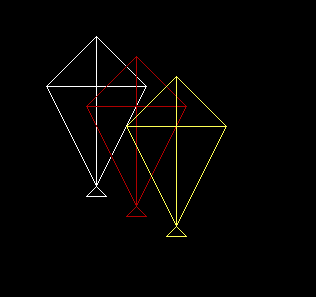
getch();

closegraph();

return 0;

}

**OUTPUT**



***ii)ROTATION - with origin***

**PROGRAM :**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

void rotate (int &x1, int &y1)

{

    double s,c, angle;

  angle=20;

    c = cos(angle \*M\_PI/180);

    s = sin(angle \*M\_PI/180);

    x1 = floor(x1 \* c + y1 \* s);

    y1 = floor(-x1 \* s + y1 \* c);

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1,x2,x3,x4,x5,x6;

int y1,y2,y3,y4,y5,y6;

x1=100;

y1=200;

x2=200;

y2=200;

x3=150;

y3=150;

x4=150;

y4=300;

x5=140;

y5=310;

x6=160;

y6=310;

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

rotate(x1,y1);

rotate(x2,y2);

rotate(x3,y3);

rotate(x4,y4);

rotate(x5,y5);

rotate(x6,y6);

setcolor(RED);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

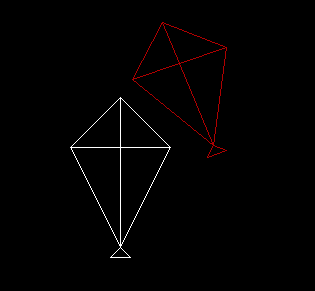
line(x5,y5,x6,y6);

getch();

closegraph();

return 0;

}

**OUTPUT :**  


***ii)ROTATION - with pivot point*** *:*

**PROGRAM :**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

int fx=150;

int fy=150;

void rotate (int &x1, int &y1)

{

    double s,c, angle;

  angle=20;

    c = cos(angle \*M\_PI/180);

    s = sin(angle \*M\_PI/180);

 float sx=x1-fx;

 float sy=y1-fy;

  x1=fx+(sx \* c - sy \*s);

   y1=fy+(sx\*s + sy\*c);

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1,x2,x3,x4,x5,x6;

int y1,y2,y3,y4,y5,y6;

x1=100;

y1=200;

x2=200;

y2=200;

x3=150;

y3=150;

x4=150;

y4=300;

x5=140;

y5=310;

x6=160;

y6=310;

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

rotate(x1,y1);

rotate(x2,y2);

rotate(x3,y3);

rotate(x4,y4);

rotate(x5,y5);

rotate(x6,y6);

setcolor(RED);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

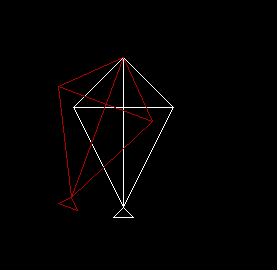
getch();

closegraph();

return 0;

}

**OUTPUT:**



***iii) SCALING - with origin***  
  
**PROGRAM**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

int fx=150;

int fy=150;

void scale(int &x1,int &y1)

{

int sx=2;

int sy=2;

x1=x1\*sx;

y1=y1\*sy;

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1,x2,x3,x4,x5,x6;

int y1,y2,y3,y4,y5,y6;

x1=100-20;

y1=200-80;

x2=200-20;

y2=200-80;

x3=150-20;

y3=150-80;

x4=150-20;

y4=300-80;

x5=140-20;

y5=310-80;

x6=160-20;

y6=310-80;

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

scale(x1,y1);

scale(x2,y2);

scale(x3,y3);

scale(x4,y4);

scale(x5,y5);

scale(x6,y6);

setcolor(RED);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

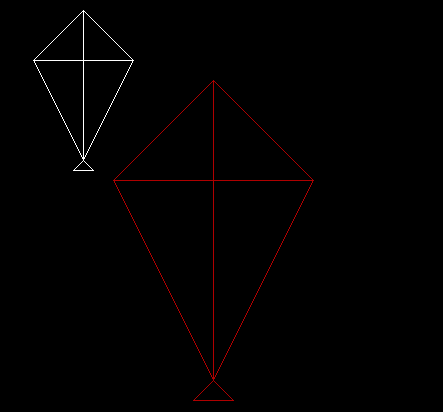
getch();

closegraph();

return 0;

}

**OUTPUT :**



***iii) SCALING - with fixed point***

**PROGRAM :**

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

int fx=150;

int fy=200;

void scale(int &x, int &y)

{

int xs=2;

int ys=2;

x= fx+ (x-fx)\*xs;

y= fy+(y-fy)\*ys;

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int x1,x2,x3,x4,x5,x6;

int y1,y2,y3,y4,y5,y6;

x1=100;

y1=200;

x2=200;

y2=200;

x3=150;

y3=150;

x4=150;

y4=300;

x5=140;

y5=310;

x6=160;

y6=310;

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

scale(x1,y1);

scale(x2,y2);

scale(x3,y3);

scale(x4,y4);

scale(x5,y5);

scale(x6,y6);

setcolor(RED);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

line(x3,y3,x4,y4);

line(x1,y1,x4,y4);

line(x2,y2,x4,y4);

line(x4,y4,x5,y5);

line(x4,y4,x6,y6);

line(x5,y5,x6,y6);

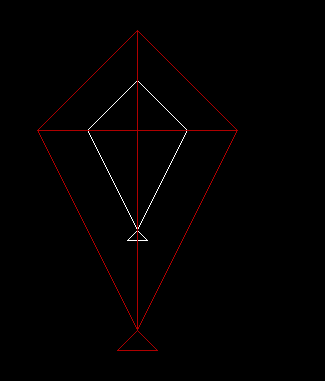
getch();

closegraph();

return 0;

}

**OUTPUT :**



**RESULT :**

Thus  Composite 2D transformation for all the cases of the given object has been implemented using graphics functions in C++.

|  |  |
| --- | --- |
| **Ex.No: 11a** | **STUDY ON OPENGL FUNCTIONS** |
| **Date: 22-09-21** |

|  |  |
| --- | --- |
| **Ex.No: 11 b** | **2D PRIMITIVES IN OPENGL** |
| **Date: 22-09-21** |

**AIM**

To write an OpenGL program to draw basic 2D objects with different colors.

**PROGRAM:**

#include "stdafx.h"

#include<stdlib.h>

#include <GL/glut.h>

#include<iostream>

#define pi 3.142857

const float DEG2RAD = 3.14159/180;

using namespace std;

void myInit (void)

{

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glColor3f(1.0, 0.0, 0.0);

    glPointSize(0.5);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-780, 420, -420, 220);

}

void display (void)

{

float x, y ,i;

    glClear(GL\_COLOR\_BUFFER\_BIT);

//TRIANGLE

glBegin(GL\_TRIANGLES);

glVertex2i(0, 0);

glVertex2i(200, 0);

glVertex2i(100,100);

glEnd();

    //CIRCLE

glColor3f(1.0,1.5,0.0);

glBegin(GL\_LINE\_LOOP);

   for ( i = 0; i < (2 \* pi); i+=0.001)

    {

        // let 200 is radius of circle and as,

        // circle is defined as x=r\*cos(i) and y=r\*sin(i)

        x = 100\* cos(i);

        y = 50\* sin(i);

        glVertex2i(-x-300, y+50);

    }

 glEnd();

//RECTANGLe

glColor3f(0.0, 1.0, 0.0);

glRecti(-100,-100,200,-200);

glFlush();

glEnd();

//QUADRILATERAL

glColor3f(0.0,0.0,1.0);

glBegin(GL\_POINTS);

for(x=0;x<200;x++)

glVertex2i(-200-x,-100);

for(y=0;y<100;y++)

glVertex2i(-200-x-y,-100-y);

for(x=0;x<100;x++)

glVertex2i(-200-x,-100-x);

for(i=0;i<200;i++)

glVertex2i(-200-x-i,-100-x);

glEnd();

glBegin(GL\_POLYGON);

glVertex2i(-200,-100);

glVertex2i(-400,-100);

glVertex2i(-200,-100);

glVertex2i(-300,-200);

glEnd();

//LINE

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINES);

glVertex2i(-500,-300);

glVertex2i(-300,-300);

glEnd();

glFlush();

}

int main (int argc, char\*\* argv)

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(0, 0);

    glutCreateWindow("GL Drawing");

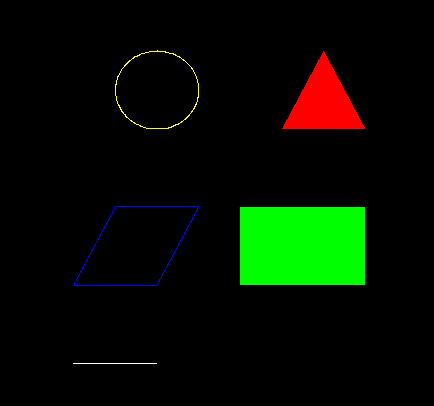
    myInit();

    glutDisplayFunc(display);

    glutMainLoop();

}

**OUTPUT:**



**RESULT :**

Thus 2D objects with different colors have been drawn using OpenGL.

|  |  |
| --- | --- |
| **Ex.No: 11c** | **2D OBJECTS IN OPENGL** |
| **Date: 22-09-21** |

**AIM**

To draw given 2D objects using OpenGL with necessary backgrounds and colors.

**PROGRAM :**

#include"stdafx.h"

#include<Windows.h>

#include<gl\GL.h>

#include<gl\GLU.h>

#include<gl\glut.h>

#include"stdio.h"

#include"stdlib.h"

#include"math.h"

const float DEG2RAD = 3.14159/180;

void init (void)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 500, 0, 500);

}

void display (void)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_LINE\_LOOP);

for(int i=0; i<360; i++)

{

float degInRad = i\*DEG2RAD;

glVertex2f(0.4+cos(degInRad)\*0.1,-0.7+sin(degInRad)\*0.1);

}

glEnd();

init();

glBegin(GL\_POLYGON);

glColor3f(0.0,0.0,0.0);

glVertex2i(345,100);

glVertex2i(355,90);

glVertex2i(350,80);

glVertex2i(340,80);

glVertex2i(335,90);

glEnd();

glBegin(GL\_LINES);

glPointSize(3.0);

glVertex2i(350, 80);

glVertex2i(355, 70);

glVertex2i(355, 70);

glVertex2i(350, 60);

glVertex2i(350, 60);

glVertex2i(340, 60);

glVertex2i(340, 60);

glVertex2i(335, 70);

glVertex2i(335, 70);

glVertex2i(340, 80);

glVertex2i(340, 80);

glVertex2i(350, 80);

glVertex2i(350, 80);

glVertex2i(355, 90);

glVertex2i(355, 90);

glVertex2i(365, 90);

glVertex2i(365, 90);

glVertex2i(370, 80);

glVertex2i(370, 80);

glVertex2i(365, 70);

glVertex2i(365, 70);

glVertex2i(355, 70);

glVertex2i(335, 90);

glVertex2i(330, 90);

glEnd();

glBegin(GL\_POLYGON);

glVertex2i(340, 60);

glVertex2i(335, 55);

glVertex2i(325, 70);

glVertex2i(335, 70);

glEnd();

glBegin(GL\_POLYGON);

glVertex2i(355, 70);

glVertex2i(365, 70);

glVertex2i(370, 60);

glVertex2i(360, 52);

glVertex2i(350, 60);

glEnd();

glBegin(GL\_POLYGON);

glVertex2i(370, 80);

glVertex2i(365, 90);

glVertex2i(368, 93);

glVertex2i(375, 80);

glEnd();

glRotatef(20.0, 0.0, 0.0, 1.0);

glBegin(GL\_POLYGON);

glColor3f(0.8,0.4,0.1);

glVertex2i(98,102);

glVertex2i(105,112);

glVertex2i(112,102);

glVertex2i(122,102);

glVertex2i(114,92);

glVertex2i(119,80);

glVertex2i(105,85);

glVertex2i(91,80);

glVertex2i(96,92);

glVertex2i(88,102);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1.0,0.69,0.19);

glVertex2i(100,100);

glVertex2i(105,110);

glVertex2i(110,100);

glVertex2i(120,100);

glVertex2i(112,92);

glVertex2i(117,82);

glVertex2i(105,87);

glVertex2i(93,82);

glVertex2i(98,92);

glVertex2i(90,100);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0.8,0.4,0.1);

glVertex2i(104,87);

glVertex2i(106,87);

glVertex2i(104,40);

glVertex2i(106,40);

glEnd();

glRotatef(-20.0, 0.0, 0.0, 1.0);

glColor3f(0.8,0.5,0.4);

glPointSize(3.0);

glRecti(150,40,180,200);

glColor3f(0.0,0.0,0.0);

glRecti(160,40,170,60);

glRecti(154,70,157,190);

glRecti(160,70,163,190);

glRecti(167,70,170,190);

glRecti(173,70,176,190);

glColor3f(0.3,0.6,0.6);

glPointSize(3.0);

glRecti(240,40,280,170);

glColor3f(0.0,0.0,0.0);

glRecti(255,40,270,80);

glRecti(257,90,267,100);

glRecti(257,110,267,120);

glRecti(257,130,267,140);

glRecti(257,150,267,160);

glColor3f(0.9,0.6,0.9);

glPointSize(3.0);

glRecti(180,40,250,130);

glColor3f(0.0,0.0,0.0);

glRecti(200,40,230,60);

glRecti(190,115,240,120);

glRecti(190,105,240,110);

glRecti(190,95,240,100);

glRecti(190,85,240,90);

glRecti(190,75,240,80);

glRecti(190,65,240,70);

glEnd();

glFlush();

}

int main (int argc, char \*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(10,10);

glutInitWindowSize(1000,1000);

glutCreateWindow("objects");

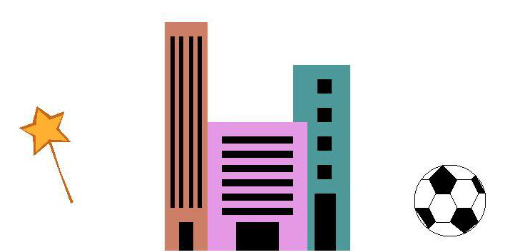
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT :**



**RESULT :**

Thus given 2D objects have been drawn using OpenGL with necessary background and colors.

|  |  |
| --- | --- |
| **Ex.No: 12** | **OPENGL KEYBOARD EVENTS** |
| **Date: 22-09-21** |

**AIM**

To implement basic transformations scaling, translation , rotations using openGL keyboard events in a simple tree.

**PROGRAM**

#include "stdafx.h"

#include<stdlib.h>

#include <GL/glut.h>

#include<iostream>

#define pi 3.142857

using namespace std;

float x=-60,y=-60,x2=-60,y2=40,x3=-10,y3=40,x4=-10,y4=-60;

float gx=40,gy=70;

float g1=-10,g2=-20,g3=-30,g4=-40,g5=-50,g6=-60,g7=-70,g8=-80,g9=-90,g10=-100,g11=-110;

float gg0=0,gg1=10,gg2=20,gg3=30,gg4=40;

void myInit (void)

{

    glClearColor(1.0, 1.0, 1.0, 0.0);

    glColor3f(0.0, 1.0, 0.0);

    glPointSize(0.5);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-780, 420, -420, 220);

}

void display (void)

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.5, 0.4, 0.2);

    glBegin(GL\_POLYGON);

    glVertex2f(x,y);

    glVertex2f(x2,y2);

    glVertex2f(x3,y3);

    glVertex2f(x4,y4);

    glEnd();

         glColor3f(0.0, 0.5, 0.0);

          glBegin(GL\_LINES);

glVertex2f(g3,gx);

glVertex2f(g8,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g2,gx);

glVertex2f(g7,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g4,gx);

glVertex2f(g9,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g5,gx);

glVertex2f(g10,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g1,gx);

glVertex2f(g6,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g6,gx);

glVertex2f(g11,gy);

glEnd();

//right

glBegin(GL\_LINES);

glVertex2f(g3,gx);

glVertex2f(gg2,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g2,gx);

glVertex2f(gg3,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g4,gx);

glVertex2f(gg1,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g5,gx);

glVertex2f(gg0,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g1,gx);

glVertex2f(gg4,gy);

glEnd();

glBegin(GL\_LINES);

glVertex2f(g6,gx);

glVertex2f(g1,gy);

glEnd();

    glFlush();

}

void glut\_Keys(unsigned char key, int x, int y)

{

switch(key)

{

case 'T' :{

x+=20,y+=20,x3+=20,y3+=20,x2+=20,y2+=20,x4+=20,y4+=20;

gx+=20,gy+=20;

g1+=20,g2+=20,g3+=20,g4+=20,g5+=20,g6+=20,g7+=20,g8+=20,g9+=20,g10+=20,g11+=20;

gg0+=20,gg1+=20,gg2+=20,gg3+=20,gg4+=20;

display();

break;

  }

case 't' :{

x-=20,y-=20,x3-=20,y3-=20,x2-=20,y2-=20,x4-=20,y4-=20;

gx-=20,gy-=20;

g1-=20,g2-=20,g3-=20,g4-=20,g5-=20,g6-=20,g7-=20,g8-=20,g9-=20,g10-=20,g11-=20;

gg0-=20,gg1-=20,gg2-=20,gg3-=20,gg4-=20;

display();

break;

  }

case 'S' :{

  y+=20,y2+=20,y3+=20,y4+=20;

  gx+=20;

  gy+=20;

  display();

  break;

  }

case 's' :{

  y-=20,y2-=20,y3-=20,y4-=20;

  gx-=20;

  gy-=20;

  display();

  break;

  }

case 'R':

glRotatef(-25.0,0.0,0.0,1.0);

display();

break;

case 'r':

glRotatef(25.0,0.0,0.0,1.0);

display();

break;

}

}

int main (int argc, char\*\* argv)

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(1366, 768);

    glutInitWindowPosition(0, 0);

    glutCreateWindow("TREE");

    myInit();

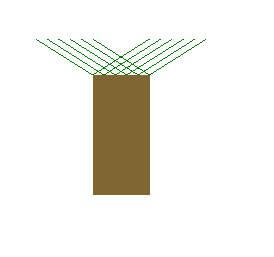
     glutKeyboardFunc(glut\_Keys);

     glutDisplayFunc(display);

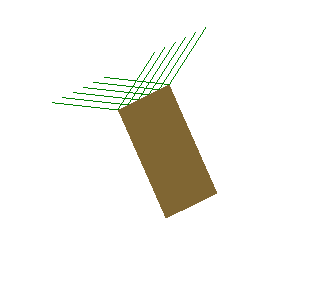
    glutMainLoop();

}

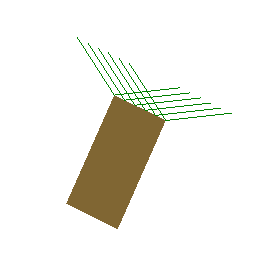
**OUTPUT :**



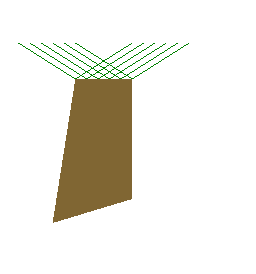
ROTATE  ‘R’



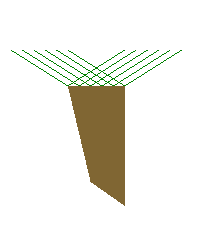
ROTATE  ‘r’



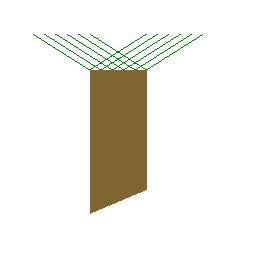
TRANSLATE ‘T’



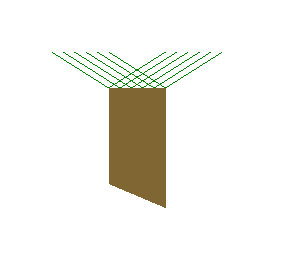
TRANSLATE ‘t’



SCALE UP ‘S’



SCALE DOWN ‘s’



**RESULT**

Thus with openGL keyboard events basic transformation has been implemented in a tree.

|  |  |
| --- | --- |
| **Ex.No: 13a** | **OPENGL MOUSE EVENTS** |
| **Date: 27-09-21** |

**AIM**

To simulate deepavali celebrations using 2D objects , keyboard and mouse events in openGL

**PROGRAM :**

#include<stdlib.h>

#include <GL/glut.h>

#include<iostream>

#define pi 3.142857

using namespace std;

void myInit (void)

{

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glColor3f(0.0, 1.0, 0.0);

    glPointSize(0.5);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0, 600, 0, 600);

}

void display (void)

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.5,0.25);

glTranslatef(0.0,0.5,0.0);

glColor3f(1.0,0.5,0.25);

glRotatef(0.0,0.0,1.0,0.0);

glutWireCone(0.5,0.5,20,30);

glColor3f(1.0, 0.0, 0.0);

    glBegin(GL\_POLYGON);

    glVertex2f(40,40);

    glVertex2f(40,160);

    glVertex2f(80,160);

    glVertex2f(80,40);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(140,40);

    glVertex2f(140,160);

    glVertex2f(180,160);

    glVertex2f(180,40);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(240,40);

    glVertex2f(240,160);

    glVertex2f(280,160);

    glVertex2f(280,40);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(340,40);

    glVertex2f(340,160);

    glVertex2f(380,160);

    glVertex2f(380,40);

    glEnd();

glFlush();

}

void glmouse(int button, int state , int x ,int y)

{

switch(button)

{

case GLUT\_RIGHT\_BUTTON :

if(state==GLUT\_DOWN)

{   glColor3f(0.8, 0.6, 0.0);

y=600-y;

            glBegin(GL\_LINES);

glVertex2i(x-10,y);

glVertex2i(x+10,y);

glVertex2i(x,y-10);

glVertex2i(x,y+10);

glVertex2i(x+8,y+8);

glVertex2i(x-8,y-8);

glVertex2i(x+8,y-8);

glVertex2i(x-8,y+8);

glEnd();

glFlush();

}

break;

case GLUT\_LEFT\_BUTTON :

if(state==GLUT\_DOWN)

{

  //lit

glColor3f(0.8, 0.6, 0.0);

    glBegin(GL\_POLYGON);

    glVertex2f(60,160);

    glVertex2f(50,170);

    glVertex2f(60,190);

    glVertex2f(70,170);

glVertex2f(60,160);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(160,160);

    glVertex2f(150,170);

    glVertex2f(160,190);

    glVertex2f(170,170);

glVertex2f(160,160);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(260,160);

    glVertex2f(250,170);

    glVertex2f(260,190);

    glVertex2f(270,170);

glVertex2f(260,160);

    glEnd();

    glBegin(GL\_POLYGON);

    glVertex2f(360,160);

    glVertex2f(350,170);

    glVertex2f(360,190);

    glVertex2f(370,170);

    glVertex2f(360,160);

    glEnd();

glFlush();

}

break;

default : break;

glFlush();

}

}

void glut\_Keys(unsigned char key, int x , int y)

{

switch(key)

{

case 'D' :

{

char buf[100];

glColor3f(0.0,0.6,0.8);

sprintf\_s(buf, "HAPPY HAPPY DEEPAVALI !!!");

char\* c;

glRasterPos2f(100,300);

for (c = buf; \*c != '\0'; c++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*c);

glFlush();

}

break;

}

}

int main (int argc, char\*\* argv)

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(0, 0);

    glutCreateWindow("session13");

    myInit();

    glutKeyboardFunc(glut\_Keys);

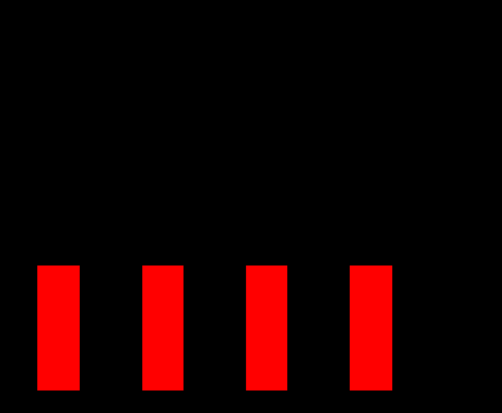
    glutDisplayFunc(display);

glutMouseFunc(glmouse);

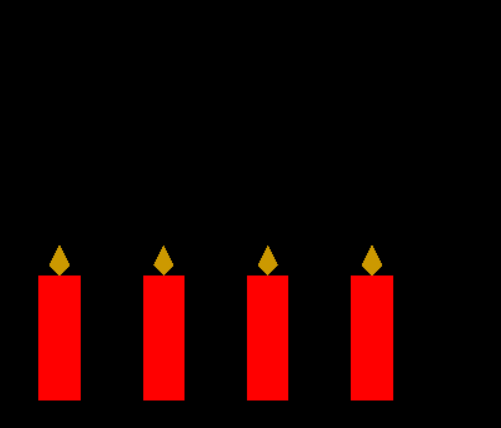
    glutMainLoop();

}

**OUTPUT :**

****

Left Click



Key ‘D’



Right Click





**RESULT :**

Thus deepavali celebration is simulated using 2D objects , keyboard and mouse events in openGL.

|  |  |
| --- | --- |
| **Ex.No: 13 b** | **OPENGL MENU EVENTS** |
| **Date: 27-09-21** |

**AIM**   
To create Menu card for a hotel using Menu Interaction function in OpenGL and to add necessary objects

**PROGRAM**

#include<stdlib.h>

#include<gl/glut.h>

#include<stdio.h>

void init()

{

glClearColor(0.0, 0.0, 0.0, 1.0);

    glColor3f(0.0, 1.0, 0.0);

    glPointSize(0.5);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0, 600, 0, 600);

}

void renderbitmap(float x, float y, void\* font, char\* string)

{

char\* c;

glRasterPos2f(x, y);

for (c = string; \*c != '\0'; c++)

glutBitmapCharacter(font, \*c);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

char buf[100];

sprintf\_s(buf, "HOTEL MENU");

renderbitmap(180, 500, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "click right to view the list");

renderbitmap(200, 400, GLUT\_BITMAP\_TIMES\_ROMAN\_10, buf);

glFlush();

}

void displayVeg()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

char buf[100];

sprintf\_s(buf, "HOTEL MENU");

renderbitmap(180, 500, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.9,1.0);

glRecti(120,40,500,450);

glColor3f(0.0,0.5,0.0);

sprintf\_s(buf, "Available Veg Items");

renderbitmap(160, 400, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.0,0.0);

sprintf\_s(buf, "Idly - Sambar");

renderbitmap(160, 360, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Veg Pulav");

renderbitmap(160, 320, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Nan - Paneer Butter Masala");

renderbitmap(160, 280, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Gobi Manchurian");

renderbitmap(160, 240, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Paneer Tikka");

renderbitmap(160, 200, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.5,0.0);

glBegin(GL\_LINES);

glVertex2i(360,100);

glVertex2i(360,200);

glEnd();

glColor3f(1.0,0.5,0.0);

glBegin(GL\_LINES);

glVertex2i(400,80);

glVertex2i(400,180);

glEnd();

glRecti(350,120,370,140);glRecti(350,145,370,155);glRecti(350,160,370,180);

glRecti(390,100,410,120);glRecti(390,125,410,135);glRecti(390,140,410,160);

glFlush();

}

void displayNonveg()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

char buf[100];

sprintf\_s(buf, "HOTEL MENU");

renderbitmap(180, 500, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.9,1.0);

glRecti(120,40,500,450);

glColor3f(0.0,0.5,0.0);

sprintf\_s(buf, "Available Non-Veg Items");

renderbitmap(160, 400, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.5,0.0);

sprintf\_s(buf, "Biriyani - chicken/mutton");

renderbitmap(160, 360, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Fried Rice - chicken/egg");

renderbitmap(160, 320, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Chicken Tandoori/Lolipop");

renderbitmap(160, 280, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Fish Curry/Fry");

renderbitmap(160, 240, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Chicken Noodles");

renderbitmap(160, 200, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.0,0.2);

glBegin(GL\_POLYGON);

glVertex2f(300,80);

glVertex2f(315,98);

glVertex2f(350,80);

glVertex2f(315,62);

glVertex2f(300,80);

glVertex2f(350,80);

glVertex2f(352,85);

glVertex2f(352,75);

glVertex2f(350,80);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(300+20,120);

glVertex2f(315+20,138);

glVertex2f(350+20,120);

glVertex2f(315+20,102);

glVertex2f(300+20,120);

glVertex2f(350+20,120);

glVertex2f(352+20,125);

glVertex2f(352+20,115);

glVertex2f(350+20,120);

glEnd();

glFlush();

}

void HotelMenuCard(GLint renderingOption)

{

switch(renderingOption)

{

case 1: displayVeg();break;

case 2: displayNonveg();break;

}

}

void submenu(GLint option)

{char buf[100];

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

sprintf\_s(buf, "HOTEL MENU");

renderbitmap(180, 500, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.9,1.0);

glRecti(120,40,500,450);

switch(option)

{

case 1:

{glColor3f(0.0,0.5,0.0);

sprintf\_s(buf, "SWEETS");

renderbitmap(160, 400, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.5,0.0);

sprintf\_s(buf, "Gulab Jamun");

renderbitmap(160, 360, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Kaju Katli");

renderbitmap(160, 320, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Laddu ");

renderbitmap(160, 280, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Payasam");

renderbitmap(160, 240, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Badhursha");

renderbitmap(160, 200, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(0.75,0.75,0.75);

glBegin(GL\_POLYGON);

glVertex2f(360,100);

glVertex2f(380,120);

glVertex2f(360,140);

glVertex2f(340,120);

glVertex2f(360,100);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(400,120);

glVertex2f(420,140);

glVertex2f(400,160);

glVertex2f(380,140);

glVertex2f(400,120);

glEnd();

glFlush();

break;

}

case 2:

{glColor3f(0.0,0.5,0.0);

sprintf\_s(buf, "ICE CREAMS");

renderbitmap(160, 400, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(1.0,0.5,0.0);

sprintf\_s(buf, "Chocolate");

renderbitmap(160, 360, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "StrawBerry");

renderbitmap(160, 320, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Vanilla");

renderbitmap(160, 280, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Kulfi");

renderbitmap(160, 240, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

sprintf\_s(buf, "Multi flavoured");

renderbitmap(160, 200, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glColor3f(0.6,0.9,0.6);

glBegin(GL\_POLYGON);

glVertex2f(350,120);

glVertex2f(360,180);

glVertex2f(390,180);

glVertex2f(400,120);

glEnd();

glColor3i(0,0,0);

glLineWidth(8);

glBegin(GL\_LINES);

glVertex2f(375,120);

glVertex2f(375,80);

glEnd();

glColor3i(0,0,0);

glLineWidth(8);

glBegin(GL\_LINES);

glVertex2f(445,120);

glVertex2f(445,80);

glEnd();

glColor3f(0.8,0.3,0.4);

glBegin(GL\_POLYGON);

glVertex2f(420,120);

glVertex2f(430,180);

glVertex2f(460,180);

glVertex2f(470,120);

glEnd();

break;

}

default : break;

}

glFlush();

}

int main(int argc, char\*\* argv)

{ GLint dessertMenu;

glutInit(&argc,argv);

 glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(0, 0);

    glutCreateWindow("Hotel menu");

    init();

glutDisplayFunc(display);

dessertMenu=glutCreateMenu(submenu);

glutAddMenuEntry("Sweets",1);

glutAddMenuEntry("IceCreams",2);

    glutCreateMenu(HotelMenuCard);

glutAddMenuEntry("Veg",1);

glutAddMenuEntry("Non-Veg",2);

glutAddSubMenu("Desserts",dessertMenu);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

return 0;

}

**OUTPUT**

****

****

****

****

****

**RESULT :**

Thus Menu card for a hotel using Menu Interaction function in OpenGL and necessary objects has been created

|  |  |
| --- | --- |
| **Ex.No: 13 c** | **OPENGL 2D/3D OBJECTS AND ANIMATION** |
| **Date: 27-09-21** |

**AIM :**

To advertise IPL by creating necessary 2D or 3D objects.

**PROGRAM :**

#include "StdAfx.h"

#include<gl/glut.h>

#include<math.h>

const float DEG2RAD = 3.14159/180;

void init(void)

{

glClearColor(0.0,0.0,0.0,1.0);

//glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, 1000, 0, 600, -1, 1);

}

void renderbitmap(float x, float y, void\* font, char\* string)

{

char\* c;

glRasterPos2f(x, y);

for (c = string; \*c != '\0'; c++)

glutBitmapCharacter(font, \*c);

}

void display(void)

{

glColor3f(1.0,0.0,0.0);

glBegin(GL\_POLYGON);

glBegin(GL\_LINE\_LOOP);

for (int i=0; i<360; i++)

{

float degInRad = i\*DEG2RAD;

glVertex2f(0.05+cos(degInRad)\*0.03,0.3+sin(degInRad)\*0.05);

}

glEnd();

glEnd();

init();

char buf[100];

glColor3f(0.0,0.6,0.8);

sprintf\_s(buf, "INDIAN PREMIER LEAGUE");

renderbitmap(350, 500, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buf);

glRotatef(20.0,0.0,0.0,1.0);

glColor3f(0.6,0.3,0.0);

glLineWidth(10.5);

glBegin(GL\_LINES);

glVertex2i(565,250);

glVertex2i(565,300);

glEnd();

glColor3f(1.0,0.9,0.8);

glRecti(550,150,580,250);

glEnd();

glRotatef(-20.0,0.0,0.0,1.0);

glColor3f(0.7,0.2,0.8);

sprintf\_s(buf, "Click to view the teams");

renderbitmap(400, 290, GLUT\_BITMAP\_HELVETICA\_18, buf);

glFlush();

}

void mouse(int button,int state,int x,int y)

{

switch(button)

{

case GLUT\_LEFT\_BUTTON:

if(state==GLUT\_DOWN)

{

char buf[100];

glColor3f(0.8,0.6,0.0);

sprintf\_s(buf, "Chennai Super Kings");

renderbitmap(300, 260, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(0.0,0.3,0.5);

sprintf\_s(buf, "Mumbai Indians");

renderbitmap(500, 260, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(1.0,0.3,0.0);

sprintf\_s(buf, "Sunrisers Hyderabad");

renderbitmap(300, 230, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(0.8,0.0,0.0);

sprintf\_s(buf, "Royal Challengers Bangalore");

renderbitmap(500, 230, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(0.0,0.0,1.0);

sprintf\_s(buf, "Delhi Capitals");

renderbitmap(300, 200, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(0.6,0.0,1.0);

sprintf\_s(buf, "Kolkata Knight Riders");

renderbitmap(500, 200, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(1.0,0.7,0.8);

sprintf\_s(buf, "Rajasthan Royals");

renderbitmap(300, 170, GLUT\_BITMAP\_HELVETICA\_18, buf);

glColor3f(1.0,0.07,0.5);

sprintf\_s(buf, "Punjab Kings");

renderbitmap(500, 170, GLUT\_BITMAP\_HELVETICA\_18, buf);

}

break;

default:

break;

}

glFlush();

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGBA);

glutInitWindowSize(1000,600);

glutInitWindowPosition(100,50);

glutCreateWindow("IPL");

glutDisplayFunc(display);

glutMouseFunc(mouse);

glutMainLoop();

return 0;

}

**OUTPUT**

****

****

**RESULT :**Thus with necessary objects in OpenGL IPL has been advertised.

|  |  |
| --- | --- |
| **Ex.No: 14** | **IMAGE EDITING AND MANIPULATION** |
| **Date: 29-09-21** |

**AIM**

To study an open source image editing tool/software to perform basic image editing  operations, to create gif images and to optimize the images.

**TOOL USED - OPENCV**

OpenCV is the huge open-source library for computer vision, machine learning, and  image processing and now it plays a major role in real-time operation which is very important in  today’s systems. The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence  it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and  supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main  focus was real-time applications for computational efficiency. All things are written in optimized  C/C++ to take advantage of multi-core processing.

In OpenCV, images are converted into multi-dimensional arrays, which greatly simplifies their manipulation. For instance, a grayscale image is interpreted as a 2D array with pixels varying from 0 to 255.

**Features of OpenCV Library**

* Read and write images
* Capture and save videos
* Process images (filter, transform)
* Perform feature detection
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

**BASIC IMAGE PROCESSING OPERATIONS USING OPENCV**

1. GrayScale Image

2. Edge Detection

3. Edge Smoothing

4. Image Resize

5. Gif Converter

**Original Image**



**1.Color Image to Grayscale image**

import cv2

image = cv2.imread('/tm2.jpg')

grayImage = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imwrite('/up22.png',grayImage)

**OUTPUT** Gray Scale Image



**2.Edge Detection**

import cv2

import numpy as np

img = cv2.imread('/tm2.jpg',0)

edge\_det = cv2.Canny(img,250,250)

cv2.imwrite('/up22.png',edge\_det)

**OUTPUT**

Edge Detector



**3.Edge Smoothing**

import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

img = cv2.imread('/tm2.jpg')

blur = cv2.blur(img,(5,5))

cv2\_imshow(blur)

**OUTPUT   -** Edge Smoothing



**4.Image Resizing**

img = cv2.imread('/tm2.jpg')

img=cv2.resize(img,(100,100))

cv2\_imshow(img)

**OUTPUT:**



**5.Video to Gif converter**

import imageio

cap = cv2.VideoCapture('/chipstop-tom-and-jerry.mp4')

image\_lst = []

while True:

 ret, frame = cap.read()

 frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

 image\_lst.append(frame\_rgb)

  imageio.mimsave('/content/video.gif', image\_lst, fps=60)

print("Saving GIF file")

with imageio.get\_writer("nature.gif", mode="I") as writer:

 for idx, frame in enumerate(image\_lst):

  writer.append\_data(frame)

**OUTPUT:**



**RESULT :**

Thus an open source image editing tool/software with  basic image editing  operations, has been implemented successfully.

|  |  |
| --- | --- |
| **Ex.No: 15** | **COMPRESSION ALGORITHM** |
| **Date: 04-10-21** |

**AIM :**

To Explore and make a study on any one Image and Text Compression Algorithm with its

implementation.

Algorithm Used : **Huffman Coding**

**HUFFMAN CODING**

Huffman coding is a lossless data compression algorithmHuffman coding is a method of data compression that is independent of the data type, that is, the data could represent an image, audio or spreadsheet. This compression scheme is used in JPEG and MPEG-2. Huffman coding works by looking at the data stream that makes up the file to be compressed.The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a Code Book which may be constructed for each image or a set of images.

Huffman coding approximates the population distribution with powers of two probability. If the true distribution does consist of powers of two probability (and the input symbols are completely uncorrelated), Huffman coding is optimal.

Huffman coding today is often used as a "back-end" to some other compression method. DEFLATE (PKZIP's algorithm) and multimedia codecs such as JPEG and MP3 have a front-end model and quantization followed by Huffman coding.

**TEXT COMPRESSION**

Text compression algorithm aim at statistical reductions in the volume of data.

The idea is to assign variable-length codes to input characters; lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

The variable-length codes assigned to input characters are Prefix Codes, meaning the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

**ALGORITHM :**

* Build a Huffman Tree from input characters.
* Traverse the Huffman Tree and assign codes to characters.

**IMPLEMENTATION**

 Ex :

character   Frequency

    a            5

    b           9

    c           12

    d           13

    e           16

    f            45

Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of the frequency field is used to compare two nodes in the min heap. Initially, the least frequent character is at root)

1. Extract two nodes with the minimum frequency from the min heap.
2. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.
3. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

**OUTPUT**

f: 0

c: 100

d: 101

a: 1100

b: 1101

e: 111

**IMAGE COMPRESSION**

The first step of Huffman coding technique is to reduce the input image to a ordered histogram, where the probability of occurrence of a certain pixel intensity value is as

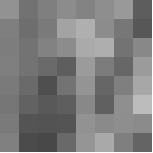
*prob\_pixel = numpix/totalnum*

where numpix is the number of occurrences of a pixel with a certain intensity value and totalnum is the total number of pixels in the input Image.

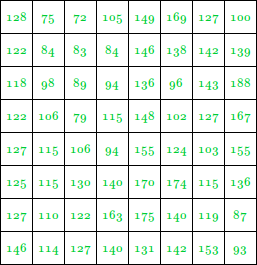
**ALGORITHM**

* Build a Huffman Tree :
* Combine the two lowest probability leaf nodes into a new node.
* Replace the two leaf nodes by the new node and sort the nodes according to the new probability values.
* Continue the steps (a) and (b) until we get a single node with probability value 1.0. We will call this node as root
* Backtrack from the root, assigning ‘0’ or ‘1’ to each intermediate node, till we reach the leaf nodes

IMPLEMENTATION :  
EX :



Pixel Intensity Values



This image contains 46 distinct pixel intensity values, hence we will have 46 unique Huffman code words.

It is evident that not all pixel intensity values may be present in the image and hence will not have non-zero probability of occurrence.

From here on, the pixel intensity values in the input Image will be addressed as leaf nodes.

Now, there are 2 essential steps to build a Huffman Tree :

1. Build a Huffman Tree :
   1. Combine the two lowest probability leaf nodes into a new node.
   2. Replace the two leaf nodes by the new node and sort the nodes according to the new probability values.
   3. Continue the steps (a) and (b) until we get a single node with probability value 1.0. We will call this node as root
2. Backtrack from the root, assigning ‘0’ or ‘1’ to each intermediate node, till we reach the leaf nodes

**ENCODED IMAGE VALUE :**  
0111010101000110011101101010001011010000000101111

00010001101000100100100100010010101011001101110111001

00000001100111101010010101100001111000110110111110010

10110001000000010110000001100001100001110011011110000

10011001101111111000100101111100010100011110000111000

01101001110101111100000111101100001110010010110101000

0111101001100101101001010111

This encoded Image is 342 bits in length, whereas the total number of bits in the original image is 512 bits. (64 pixels each of 8 bits).

**RESULT :**

Thus Image and Text Compression Algorithm with its implementation has been explored.

|  |  |
| --- | --- |
| **Ex.No: 16** | **ANIMATION – MINI PROJECT** |
| **Date: 06-10-21** |

**OBJECTIVE**

Game level design is one of the most important element of developing an enjoyable video game. Besides, game with difficult and dynamic level can make players more exciting. Space Shooter  games are a subgenre of [action video games](https://en.wikipedia.org/wiki/Action_video_game) where the focus is almost entirely on the defeat of the character's enemies using the weapons given to the player. Shooter games test the player's spatial awareness, reflexes, and speed in both isolated single player or networked [multiplayer](https://en.wikipedia.org/wiki/Multiplayer_video_game) environments. Shooter games encompass many subgenres that have the commonality of focusing on the actions of the [avatar](https://en.wikipedia.org/wiki/Avatar_(computing)) engaging in combat with a weapon against both code-driven enemies or other avatars controlled by other players.

The aim of the project is to build a space shooter game using OpenGL primitives, objects, Mouse and Keyboard events

**REQUIREMENTS**

**MINIMUM HARDWARE REQUIREMENTS**

* Processor : 1.6 GHZ

* RAM :1GB(32-bit),2GB(64-bit)

* Hard Disk : 3GB (approx) , 5400 RPM hard disk drive

* Display : DirectX 9 capable video card running at 1024 x768 or higher display

**MINIMUM SOFTWARE REQUIREMENTS**

* Operating System : Windows 10

* Language : OpenGL using C/C++

* Compiler : Microsoft Visual studio 2010 with OPENGL- GLUT packages.

**TOOLS USED**

**OpenGL**

OpenGL is the abbreviation for OpenGL Graphics Library.It is a software interface for graphics hardware.This interface consists of several hundred functions that allow you , a graphics programmer,to specify the objects and operations needed to produce high quality actually simple variations of each other,so in reality there are about 120 substantially different functions.The main purpose of OpenGL is to render two-dimensional and three- dimensional objects into frame buffer.These objects are defined as sequence of vertices or pixels.OpenGL performs several process on this data to convert it to pixel to form the final desired image in the frame buffer.

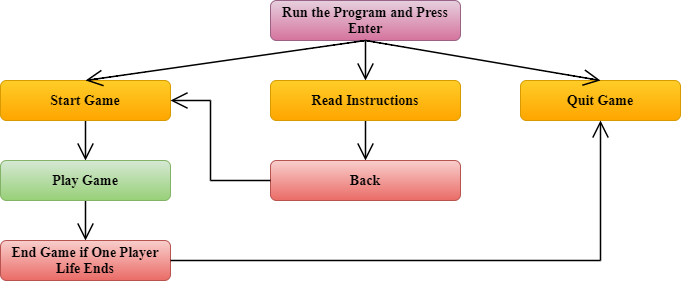
GLUT

* *GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL Programming.*

***GLU Library***

*GLU is the OpenGL Utility Library. This is a set of functions to create texture mipmaps from a base image, map coordinates between screen and object space, and draw quadric surfaces and NURBS. GLU 1.2 is the version of GLU that goes with OpenGL 1.1.  
GLU 1.3 is available and includes new capabilities corresponding to new OpenGL 1.2 features.*

**METHODOLOGY**



**WORKING:**

* User can start the game by pressing enter key
* User can read the game instructions before starting game
* Movement of players is controlled by keyboard
* On an instance only 2 players can compete
* The players use ‘c’ for player 1 and ‘m’ for player 2 to shoot the opponents
* Life of the players will decrease each time by 5 units when get shooted by the opponent
* When any one player life ends (life=0) the game will gets over and the opponent wins.

**CONTROLS:**

*Keyboard control for PLAYER 1*  :

W - UP  
S - DOWN  
A - LEFT  
D - RIGHT  
C - to shoot, Use 'w' and 's' to change direction.

*Keyboard control for PLAYER 2*  :

I - UP  
K - DOWN  
J - LEFT  
L - RIGHT  
M - to shoot, Use 'I' and 'K' to change direction.

Each time a player gets shot, LIFE decreases by 5 points.

**IMPLEMENTATION**

**Startgame,instructions,quit options:**

**void startScreenDisplay()**

{

glLineWidth(10);

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-750 ,-500);

glVertex2f(-750 ,550);

glVertex2f(750 ,550);

glVertex2f(750 ,-500);

glEnd();

glLineWidth(1);

glColor3f(1, 1, 0);

glBegin(GL\_POLYGON);

glVertex2f(-200 ,300);

glVertex2f(-200 ,400);

glVertex2f(200 ,400);

glVertex2f(200 ,300);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(-200, 50);

glVertex2f(-200 ,150);

glVertex2f(200 ,150);

glVertex2f(200 ,50);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(-200 ,-200);

glVertex2f(-200 ,-100);

glVertex2f(200, -100);

glVertex2f(200, -200);

glEnd();

if(mouseX>=-100 && mouseX<=100 && mouseY>=150 && mouseY<=200){

glColor3f(0 ,0 ,1) ;

if(mButtonPressed){

alienLife1 = alienLife2 = 300;

viewPage = GAME;

mButtonPressed = false;

}

} else

glColor3f(0 , 0, 0);

displayRasterText(-100 ,340 ,0.4 ,"Start Game");

if(mouseX>=-100 && mouseX<=100 && mouseY>=30 && mouseY<=80) {

glColor3f(0 ,0 ,1);

if(mButtonPressed){

viewPage = INSTRUCTIONS;

printf("button pressed bitch\n");

mButtonPressed = false;

}

} else

glColor3f(0 , 0, 0);

displayRasterText(-120 ,80 ,0.4 ,"Instructions");

if(mouseX>=-100 && mouseX<=100 && mouseY>=-90 && mouseY<=-40){

glColor3f(0 ,0 ,1);

if(mButtonPressed){

mButtonPressed = false;

exit(0);

}

}

else

glColor3f(0 , 0, 0);

displayRasterText(-100 ,-170 ,0.4 ,"    Quit");

glutPostRedisplay();

}

**InstructionsScreenDisplay:**

**void instructionsScreenDisplay()**

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(1, 0, 0);

displayRasterText(-900 ,550 ,0.4 ,"INSTRUCTIONS");

    glColor3f(1, 0, 0);

displayRasterText(-1000 ,400 ,0.4 ,"PLAYER 1");

displayRasterText(200 ,400 ,0.4 ,"PLAYER 2");

glColor3f(1, 1, 1);

displayRasterText(-1100 ,300 ,0.4 ,"Key 'w' to move up.");

displayRasterText(-1100 ,200 ,0.4 ,"Key 's' to move down.");

displayRasterText(-1100 ,100 ,0.4 ,"Key 'd' to move right.");

displayRasterText(-1100 ,0 ,0.4 ,"Key 'a' to move left.");

displayRasterText(100 ,300 ,0.4 ,"Key 'i' to move up.");

    displayRasterText(100 ,200 ,0.4 ,"Key 'k' to move down.");

    displayRasterText(100 ,100 ,0.4 ,"Key 'j' to move right.");

    displayRasterText(100 ,0 ,0.4 ,"Key 'l' to move left.");

displayRasterText(-1100 ,-100 ,0.4 ,"Key 'c' to shoot, Use 'w' and 's' to change direction.");

displayRasterText(100 ,-100 ,0.4 ,"Key 'm' to shoot, Use 'i' and 'k' to change direction.");

displayRasterText(-1100, -300,0.4,"The Objective is to kill your opponent.");

displayRasterText(-1100 ,-370 ,0.4 ,"Each time a player gets shot, LIFE decreases by 5 points.");

backButton();}

**AlienBody design:**

**void DrawAlienBody(bool isPlayer1)**

{

if(isPlayer1)

glColor3f(0,1,0);

else

glColor3f(1,1,0);

glBegin(GL\_POLYGON);

for(int i=0;i<=8;i++)

glVertex2fv(AlienBody[i]);

glEnd();

glColor3f(0,0,0);

glLineWidth(1);

glBegin(GL\_LINE\_STRIP);

for(int i=0;i<=8;i++)

glVertex2fv(AlienBody[i]);

glEnd();

glBegin(GL\_LINES);

glVertex2f(-13,11);

glVertex2f(-15,9);

glEnd();

}

**AlienFace Design:**

**void DrawAlienFace(bool isPlayer1)**

{

glColor3f(0,0,1);

glBegin(GL\_POLYGON);

for(int i=0;i<=42 ;i++)

glVertex2fv(ALienFace[i]);

glEnd();

glColor3f(0,0,0);

glBegin(GL\_LINE\_STRIP);

for(int i=0;i<=42 ;i++)

glVertex2fv(ALienFace[i]);

glEnd();

glBegin(GL\_LINE\_STRIP);

glVertex2f(3.3,22);

glVertex2f(4.4,23.5);

glVertex2f(6.3,26);

glEnd();

}

**AlienBeak design:**

**void DrawAlienBeak()**

{

glColor3f(1,1,0);

glBegin(GL\_POLYGON);

for(int i=0;i<=14 ;i++)

glVertex2fv(ALienBeak[i]);

glEnd();

glColor3f(0,0,0);

glBegin(GL\_LINE\_STRIP);

for(int i=0;i<=14 ;i++)

glVertex2fv(ALienBeak[i]);

glEnd();

}

**void DrawSpaceshipBody(bool isPlayer1)**

{

if(isPlayer1)

glColor3f(1, 0, 0);

else

glColor3f(0.5, 0, 0.5);

glPushMatrix();

glScalef(70,20,1);

glutSolidSphere(1,50,50);

glPopMatrix();

glPushMatrix();

glScalef(3,3,1);

glTranslated(-20,0,0);

glColor3fv(LightColor[(CI+0)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+1)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+2)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+0)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+1)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+2)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+0)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+1)%3]);

glutSolidSphere(1,1000,1000);

glTranslated(5,0,0);

glColor3fv(LightColor[(CI+2)%3]);

glutSolidSphere(1,1000,1000);

glPopMatrix();

}

**SteeringWheel design:**

**void DrawSteeringWheel()**

{

glPushMatrix();

glLineWidth(3);

glColor3f(0.20,0.,0.20);

glScalef(7,4,1);

glTranslated(-1.9,5.5,0);

glutWireSphere(1,8,8);

glPopMatrix();

}

**Spaceship design:**

**void SpaceshipCreate(int x, int y, bool isPlayer1){**

glPushMatrix();

glTranslated(x,y,0);

DrawSpaceshipDoom();

glPushMatrix();

glTranslated(4,19,0);

DrawAlien(isPlayer1);

glPopMatrix();

DrawSteeringWheel();

DrawSpaceshipBody(isPlayer1);

glEnd();

glPopMatrix();

}

**Laser design:**

**void DrawLaser(int x, int y, bool dir[])** {

int xend = -XMAX, yend = y;

if(dir[0])

yend = YMAX;

else if(dir[1])

yend = -YMAX;

glLineWidth(5);

glColor3f(1, 0, 0);

glBegin(GL\_LINES);

glVertex2f(x, y);

glVertex2f(xend, yend);

glEnd();

}

**LASER GENERATION**

**void checkLaserContact(int x, int y, bool dir[], int xp, int yp, bool player1)** {

int xend = -XMAX, yend = y;

xp += 8; yp += 8;

if(dir[0])

yend = YMAX;

else if(dir[1])

yend = -YMAX;

float m = (float)(yend - y) / (float)(xend - x);

float k = y - m \* x ;

int r = 50;

float b = 2 \* xp - 2 \* m \* (k - yp);

float a = 1 + m \* m;

float c = xp \* xp + (k - yp) \* (k - yp) - r \* r;

float d = (b \* b - 4 \* a \* c);

printf("\nDisc: %f x: %d, y: %d, xp: %d, yp: %d", d, x, y, xp, yp);

if(d >= 0) {

if(player1)

alienLife1 -= 5;

else

alienLife2 -= 5;

printf("%d %d\n", alienLife1, alienLife2);

}

}

**KEYBOARD AND MOUSE EVENTS**

**void keyOperations()** {

if(keyStates[13] == true && viewPage == INTRO) {

viewPage = MENU;

printf("view value changed to %d", viewPage);

printf("enter key pressed\n");

}

if(viewPage == GAME) {

laser1Dir[0] = laser1Dir[1] = false;

laser2Dir[0] = laser2Dir[1] = false;

if(keyStates['c'] == true) {

laser2 = true;

if(keyStates['w'] == true) laser2Dir[0] = true;

if(keyStates['s'] == true) laser2Dir[1] = true;

}

else {

laser2 = false;

if(keyStates['d'] == true) xTwo-=SPACESHIP\_SPEED;

if(keyStates['a'] == true) xTwo+=SPACESHIP\_SPEED;

if(keyStates['w'] == true) yTwo+=SPACESHIP\_SPEED;

if(keyStates['s'] == true) yTwo-=SPACESHIP\_SPEED;

}

if(keyStates['m'] == true) {

laser1 = true;

if(keyStates['i'] == true) laser1Dir[0] = true;

if(keyStates['k'] == true) laser1Dir[1] = true;

}

else {

laser1 = false;

if(keyStates['l'] == true) xOne+=SPACESHIP\_SPEED;

if(keyStates['j'] == true) xOne-=SPACESHIP\_SPEED;

if(keyStates['i'] == true) yOne+=SPACESHIP\_SPEED;

if(keyStates['k'] == true) yOne-=SPACESHIP\_SPEED;

}

}

}

**void passiveMotionFunc(int x,int y)** {

mouseX = float(x)/(m\_viewport[2]/1200.0)-600.0;

mouseY = -(float(y)/(m\_viewport[3]/700.0)-350.0);

glutPostRedisplay();

}

**void mouseClick(int buttonPressed ,int state ,int x, int y)** {

if(buttonPressed == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

mButtonPressed = true;

else

mButtonPressed = false;

glutPostRedisplay();

}

**void keyPressed(unsigned char key, int x, int y)**

{

keyStates[key] = true;

glutPostRedisplay();

}

**void refresh()** {

glutPostRedisplay();

}

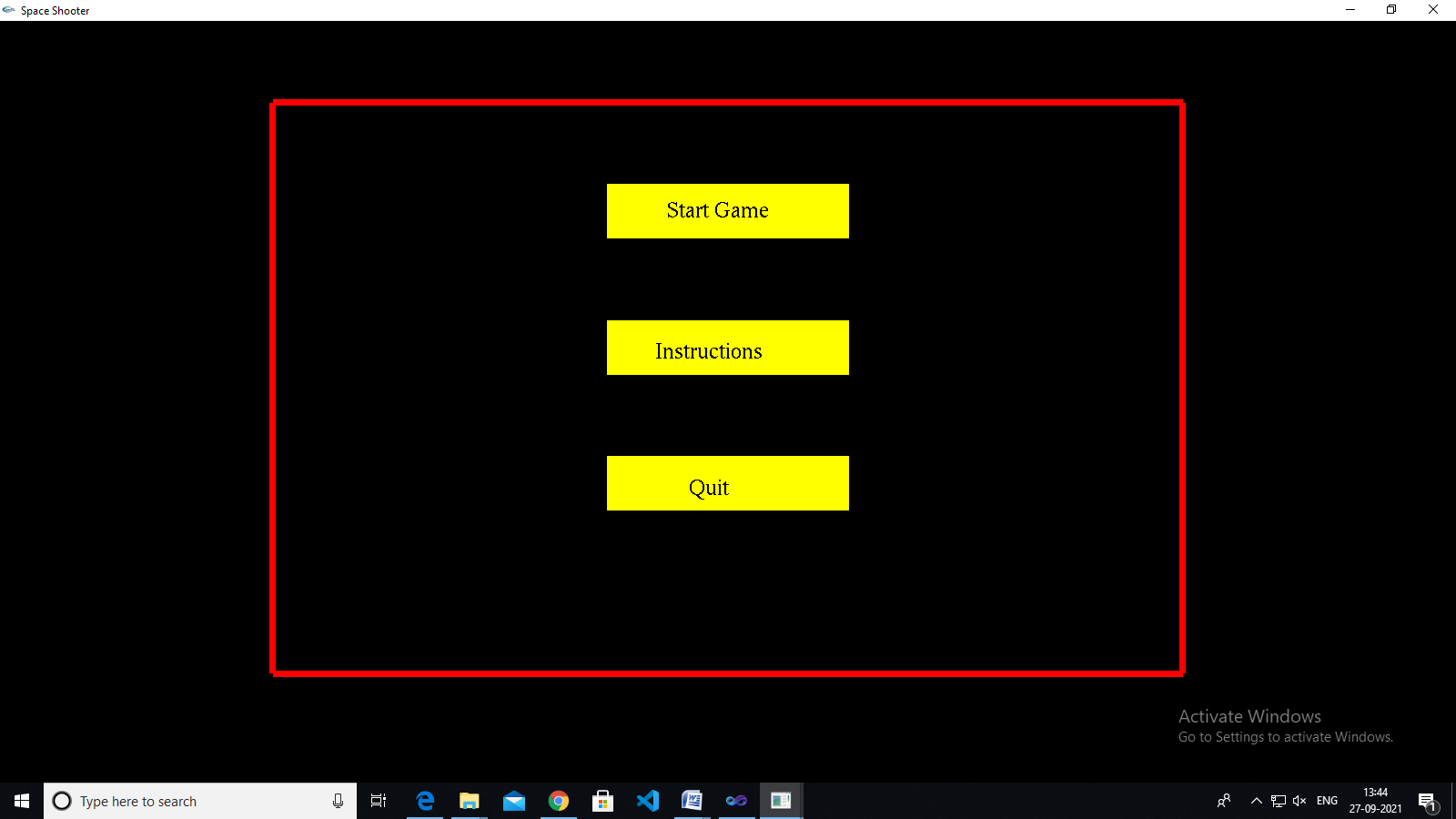
**void keyReleased(unsigned char key, int x, int y)** {

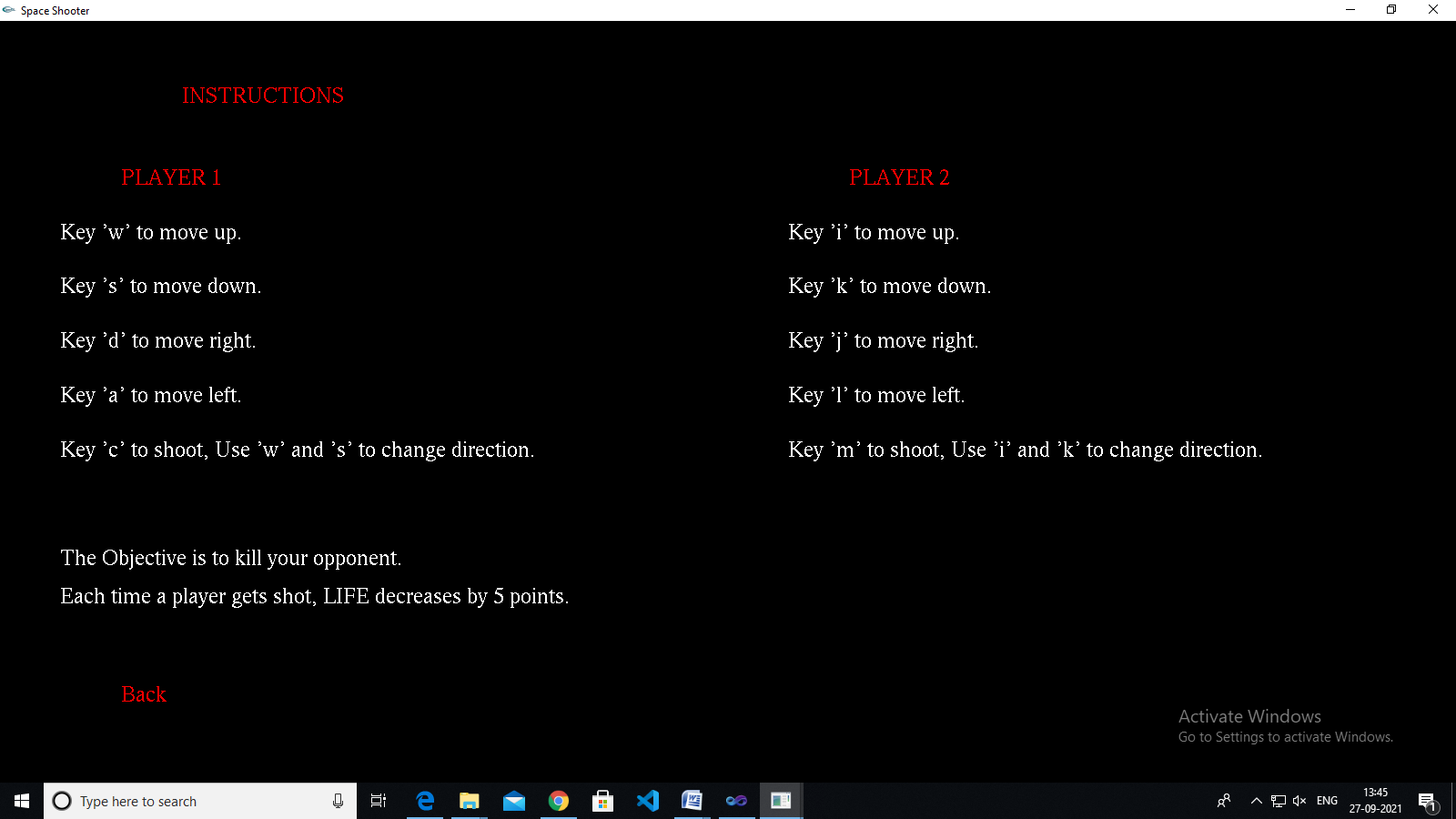
keyStates[key] = false;

}

**SCREENSHOTS :**

MAIN SCREEN



INSTRUCTIONS PAGE  


GAME SCREEN

