

Computer Graphics Laboratory

Assignment No.1

Title:- Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm.

Program:-

```
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;
class point
{
public:
int x,y;
};
class poly
{
private:
point p[20];
int inter[20],x,y;
int v,xmin,ymin,xmax,ymax;
public:
int c;
void read();
void calcs();
void display();
void ints(float);
void sort(int);
};
void poly::read()
{
int i;
cout<<"\n Scan Fill Algorithm ";
cout<<"\n Enter Number Of Vertices Of Polygon: ";
cin>>v;
if(v>2)
{
for(i=0;i<v; i++)
{
cout<<"\nEnter co-ordinate no. "<<i+1<<" : ";
cout<<"\n\tx"<<(i+1)<<"=";
cin>>p[i].x;
cout<<"\n\ty"<<(i+1)<<"=";
cin>>p[i].y;
```

```

    }
    p[i].x=p[0].x;
    p[i].y=p[0].y;
    xmin=xmax=p[0].x;
    ymin=ymax=p[0].y;
    }
    else
    cout<<"\n Enter valid no. of vertices.";
    }
    void poly::calcs()
    {
    for(int i=0;i<v;i++)
    {
    if(xmin>p[i].x)
    xmin=p[i].x;
    if(xmax<p[i].x)
    xmax=p[i].x;
    if(ymin>p[i].y)
    ymin=p[i].y;
    if(ymax<p[i].y)
    ymax=p[i].y;
    }
    }
    void poly::display()
    {
    int ch1;
    char ch='y';
    float s,s2;
    do
    {
    cout<<"\n\nMENU:";
    cout<<"\n\n\t1 . Scan line Fill ";
    cout<<"\n\n\t2 . Exit ";
    cout<<"\n\nEnter your choice:";
    cin>>ch1;
    switch(ch1)
    {
    case 1:
    s=ymin+0.01;
    delay(100);
    cleardevice();
    while(s<=ymax)
    {
    ints(s);
    sort(s);
    s++;
    }
    break;
    case 2:
    exit(0);

```

```

}
cout<<"Do you want to continue?: ";
cin>>ch;
}while(ch=='y' || ch=='Y');
}
void poly::ints(float z)
{
int x1,x2,y1,y2,temp;
c=0;
for(int i=0;i<v;i++)
{
x1=p[i].x;
y1=p[i].y;
x2=p[i+1].x;
y2=p[i+1].y;
if(y2<y1)
{
temp=x1;
x1=x2;
x2=temp;
temp=y1;
y1=y2;
y2=temp;
}
if(z<=y2&& z>=y1)
{
if((y1-y2)==0)
x=x1;
else
{
x=((x2-x1)*(z-y1))/(y2-y1);
x=x+x1;
}
if(x<=xmax && x>=xmin)
inter[c++]=x;
}
}
}
void poly::sort(int z)
{
int temp,j,i;
for(i=0;i<v;i++)
{
line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);
}
delay(100);
for(i=0; i<c;i+=2)
{
delay(100);
line(inter[i],z,inter[i+1],z);
}
}

```

```

    }
}
int main() //main
{
    int cl;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,NULL);
    cleardevice();
    poly x;
    x.read();
    x.calcs();
    cleardevice();
    cout<<"\n\tEnter The Color You Want :(In Range 0 To 15 )->";
    cin>>cl;
    setcolor(cl);
    x.display();
    closegraph();
    getch();
    return 0;
}

```

Output:-

```
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ gedit cg3.cpp
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ g++ cg3.cpp -o cg3 -lgraph
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ ./cg3

Scan Fill Algorithm
Enter Number Of Vertices Of Polygon: [xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
cg3: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
4

Enter co-ordinate no. 1 :
    x1=200

    y1=200

Enter co-ordinate no. 2 :
    x2=200

    y2=400

Enter co-ordinate no. 3 :
    x3=400

    y3=200

Enter co-ordinate no. 4 :
    x4=400

    y4=400

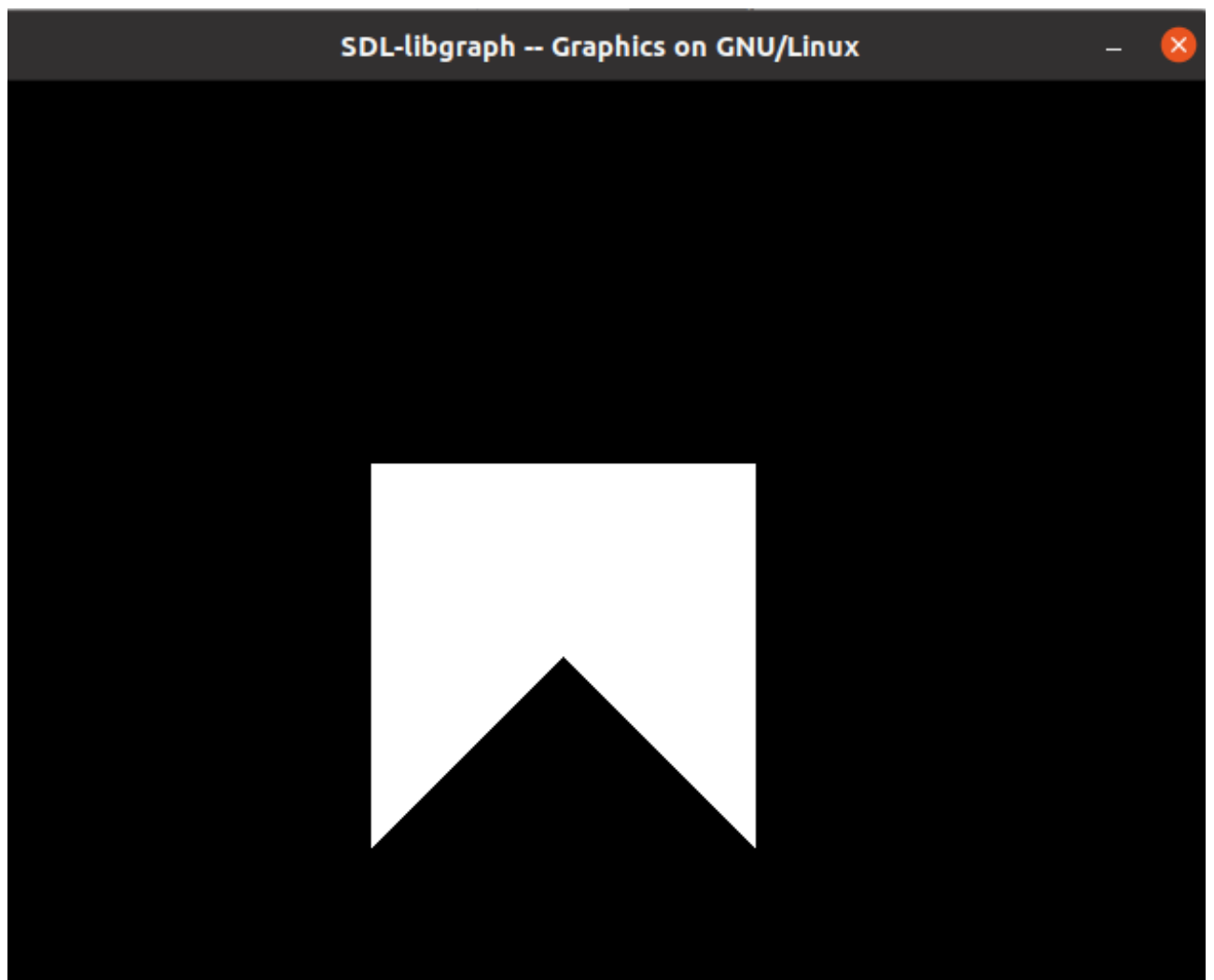
Enter The Color You Want :(In Range 0 To 15 )->15

MENU:

    1 . Scan line Fill

    2 . Exit

Enter your choice:1
Do you want to continue?: █
```



Assignment no.2

Title:- Write C++ program to implement Cohen Southerland line clipping algorithm.

Program:-

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
using namespace std;
class Coordinate
{
public:
int x,y;
char code[4];
};
class Lineclip
{
public:
Coordinate PT;
void drawwindow();
void drawline(Coordinate p1,Coordinate p2);
Coordinate setcode(Coordinate p);
int visibility(Coordinate p1,Coordinate p2);
Coordinate resetendpt(Coordinate p1,Coordinate p2);
};
int main()
{
Lineclip lc;
int gd = DETECT,v,gm;
Coordinate p1,p2,p3,p4,ptemp;

cout<<"\n Enter x1 and y1\n";
cin>>p1.x>>p1.y;
cout<<"\n Enter x2 and y2\n";
cin>>p2.x>>p2.y;

initgraph(&gd,&gm,NULL);
lc.drawwindow();
delay(2000);

lc.drawline (p1,p2);
delay(2000);
cleardevice();

delay(2000);
p1=lc.setcode(p1);
```

```

p2=lc.setcode(p2);
v=lc.visibility(p1,p2);
delay(2000);

switch(v)
{
case 0: lc.drawwindow();
delay(2000);
lc.drawline(p1,p2);
break;
case 1:lc.drawwindow();
delay(2000);
break;
case 2:p3=lc.resetendpt(p1,p2);
p4=lc.resetendpt(p2,p1);
lc.drawwindow();
delay(2000);
lc.drawline(p3,p4);
break;
}
delay(2000);
closegraph();
}
void Lineclip::drawwindow()
{
line(150,100,450,100);
line(450,100,450,350);
line(450,350,150,350);
line(150,350,150,100);
}
void Lineclip::drawline(Coordinate p1,Coordinate p2)
{
line(p1.x,p1.y,p2.x,p2.y);
}
Coordinate Lineclip::setcode(Coordinate p)
{
Coordinate ptemp;
if(p.y<100)
ptemp.code[0]='1';
else
ptemp.code[0]='0';
if(p.y>350)
ptemp.code[1]='1';
else
ptemp.code[1]='0';
if(p.x>450)
ptemp.code[2]='1';
else
ptemp.code[2]='0';
if(p.x<150)

```



```

ptemp.code[3]='1';
else
ptemp.code[3]='0';
ptemp.x=p.x;
ptemp.y=p.y;
return(ptemp);
};
int Lineclip:: visibility(Coordinate p1,Coordinate p2)
{
int i,flag=0;
for(i=0;i<4;i++)
{
if(p1.code[i]!='0' || (p2.code[i]=='1'))
flag='0';
}
if(flag==0)
return(0);
for(i=0;i<4;i++)
{
if(p1.code[i]==p2.code[i] && (p2.code[i]=='1'))
flag='0';
}
if(flag==0)
return(1);
return(2);
}
Coordinate Lineclip::resetendpt(Coordinate p1,Coordinate p2)
{
Coordinate temp;
int x,y,i;
float m,k;
if(p1.code[3]=='1')
x=150;
if(p1.code[2]=='1')
x=450;
if((p1.code[3]=='1') || (p1.code[2]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(p1.y+(m*(x-p1.x)));
temp.y=k;
temp.x=x;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
if(temp.y<=350 && temp.y>=100)
return (temp);
}
if(p1.code[0]=='1')
y=100;
if(p1.code[1]=='1')
y=350;

```

```

if((p1.code[1]=='1') || (p1.code[1]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(float)p1.x+(float)(y-p1.y)/m;
temp.x=k;
temp.y=y;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
return(temp);
}
else
return(p1);
}

```

Output:-

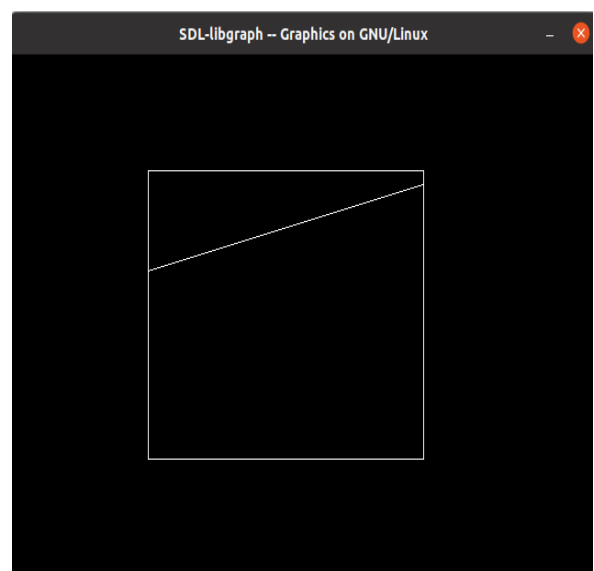
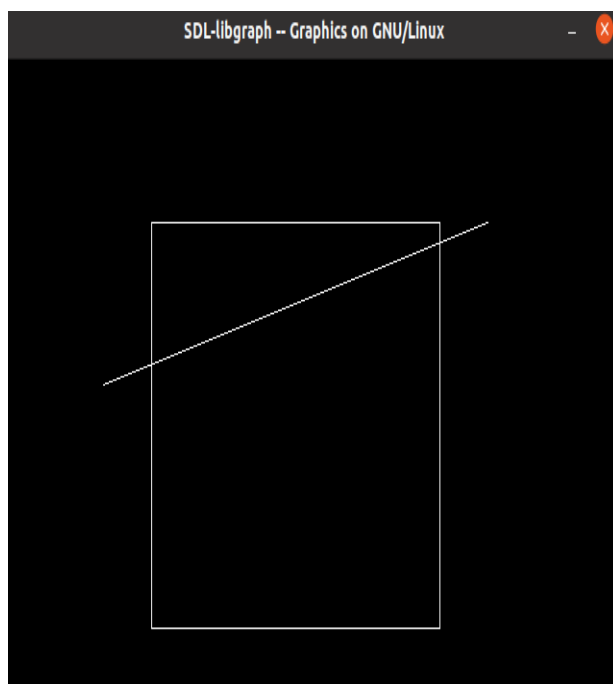
```

d-comp-pl-ii-18@dcompplii18-OptiPlex-3070: ~
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ gedit cg4.cpp
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ g++ cg4.cpp -o hs.cpp -lgraph
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ ./hs.cpp

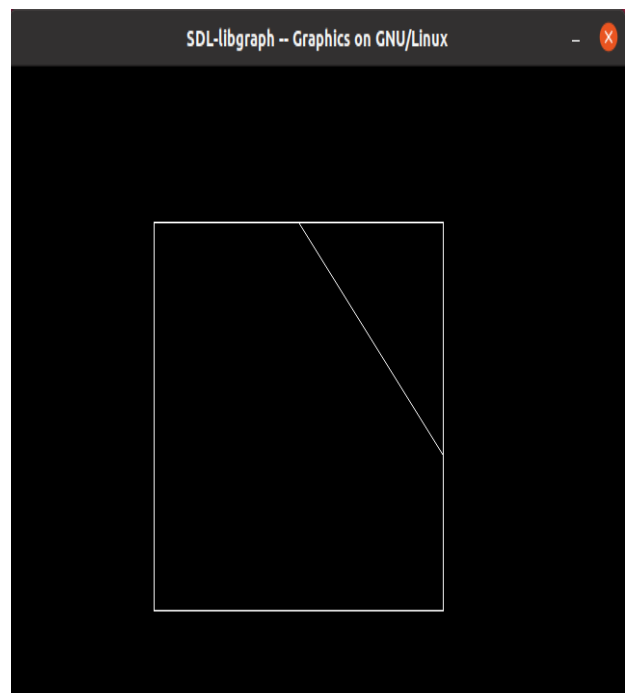
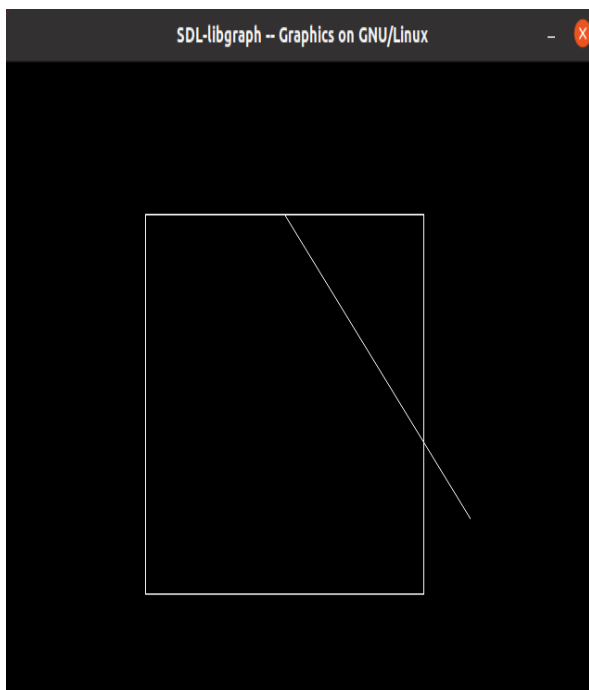
Enter x1 and y1
100
200

Enter x2 and y2
500
100
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been
called
[xcb] Aborting, sorry about that.
hs.cpp: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_seq
uence_lost' failed.
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$

```

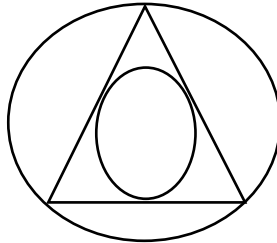


```
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070: ~  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ gedit cg4.cpp  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ g++ cg4.cpp -o hs.cpp -lgraph  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ ./hs.cpp  
  
Enter x1 and y1  
500  
300  
  
Enter x2 and y2  
300  
100  
[xcb] Unknown sequence number while processing queue  
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been  
called  
[xcb] Aborting, sorry about that.  
hs.cpp: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_seq  
uence_lost' failed.  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$
```



Assignment no.3

Title:- Write C++ program to draw the following pattern. Use DDA and Bresenham's Line drawing algorithm.



Program:-

```
#include <iostream>
# include <graphics.h>
# include <stdlib.h>
using namespace std;
class dcircle
{
private:
int x0, y0;
public:
dcircle()
{
x0=0;
y0=0;
}
void setoff(int xx, int yy)
{
x0=xx;
y0=yy;
}
void drawc(int x1, int y1, int r)
{
float d;
int x,y;
x=0;
y=r;
d=3-2*r;
do
{
putpixel(x1+x0+x, y0+y-y1, 15);
putpixel(x1+x0+y, y0+x-y1,15);
putpixel(x1+x0+y, y0-x-y1,15);
putpixel(x1+x0+x,y0-y-y1,15);
putpixel(x1+x0-x,y0-y-y1,15);
putpixel(x1+x0-y, y0-x-y1,15);
putpixel(x1+x0-y, y0+x-y1,15);
```

```

    putpixel(x1+x0-x, y0+y-y1,15);
if (d<=0)
{
    d = d+4*x+6;
}
else
{
    d=d+4*(x-y)+10;
    y=y-1;
}
x=x+1;
}
while(x<y);
};
class pt
{
protected:
    int xco, yco,color;
public:
    pt()
    {
        xco=0,yco=0,color=15;
    }
    void setco(int x, int y)
    {
        xco=x;
        yco=y;
    }
    void setcolor(int c)
    {
        color=c;
    }
    void draw()
    {
        putpixel(xco,yco,color);
    }
};
class dline:public pt
{
private: int x2, y2;
public:
    dline():pt()
    {
        x2=0;
        y2=0;
    }
    void setline(int x, int y, int xx, int yy)
    {
        pt::setco(x,y);
    }
};

```

```

x2=xx;
y2=yy;
}
void drawl( int colour)
{
float x,y,dx,dy,length;
int i;
pt::setcolor(colour);
dx= abs(x2-xco);
dy=abs(y2-yco);
if(dx>=dy)
{
length= dx;
}
else
{
length= dy;
}
dx=(x2-xco)/length;
dy=(y2-yco)/length;
x=xco+0.5;
y=yco+0.5;
i=1;
while(i<=length)
{
pt::setco(x,y);
pt::draw();
x=x+dx;
y=y+dy;
i=i+1;
}
pt::setco(x,y);
pt::draw();
};
int main()
{
int gd=DETECT, gm;
initgraph(&gd, &gm, NULL);
int x,y,r, x1, x2, y1, y2, xmax, ymax, xmid, ymid, n, i;
dcircle c;
cout<<"\nEnter coordinates of centre of circle : ";
cout<<"\nEnter the value of x : ";
cin>>x;
cout<<"\nEnter the value of y : ";
cin>>y;
cout<<"\nEnter the value of radius : ";
cin>>r;
xmax= getmaxx();
ymax= getmaxy();

```

```

xmid=xmax/2;
ymid=ymax/2;
setcolor(1);
c.setoff(xmid,ymid);
line(xmid, 0, xmid, ymax);
line(0,ymid,xmax,ymid);
setcolor(15);
c.drawc(x,y,r);
pt p1;
p1.setco(100,100);
p1.setcolor(14);
dline l;
l.setline(x1+xmid, ymid-y1, x2+xmid, ymid-y2);
cout<<"Enter Total Number of lines : ";
cin>>n;
for(i=0;i<n;i++)
{
cout<<"Enter co-ordinates of point x1 : ";
cin>>x1;
cout<<"enter coordinates of point y1 : ";
cin>>y1;
cout<<"Enter co-ordinates of point x2 : ";
cin>>x2;
cout<<"enter coordinates of point y2 : ";
cin>>y2;
l.setline(x1+xmid, ymid-y1, x2+xmid, ymid-y2);
l.drawl(15);
}
cout<<"\nEnter coordinates of centre of circle : ";
cout<<"\nEnter the value of x : ";
cin>>x;
cout<<"\nEnter the value of y : ";
cin>>y;
cout<<"\nEnter the value of radius : ";
cin>>r;
setcolor(5);
c.drawc(x,y,r);
getch();
delay(2000);
closegraph();
return 0;
}

```

Output:-

```
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ gedit harshada.cpp
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ g++ harshada.cpp -o pixel.cpp -lgraph
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ ./pixel.cpp

Enter coordinates of centre of circle :
Enter the value of x : [xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
pixel.cpp: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
102

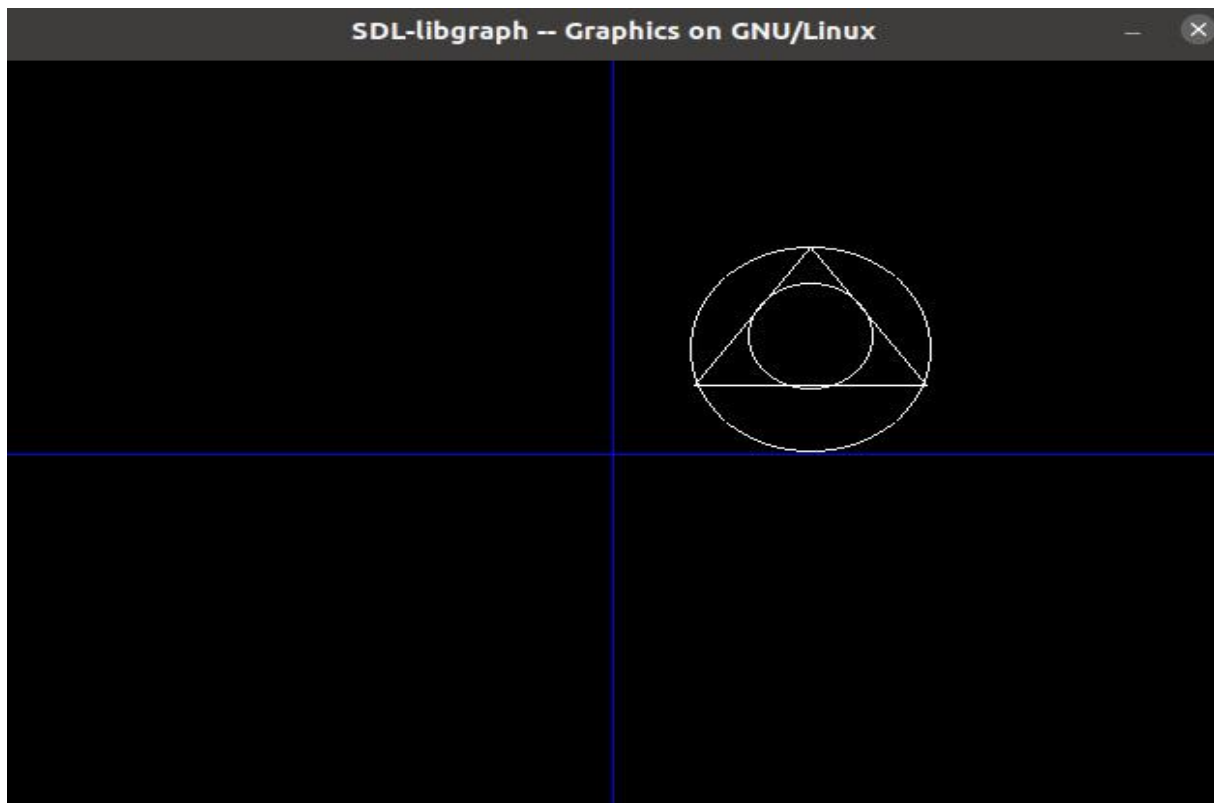
Enter the value of y : 72

Enter the value of radius : 32
Enter Total Number of lines : 3
Enter co-ordinates of point x1 : 42
enter coordinates of point y1 : 42
Enter co-ordinates of point x2 : 102
enter coordinates of point y2 : 126
Enter co-ordinates of point x1 : 42
enter coordinates of point y1 : 42
Enter co-ordinates of point x2 : 162
enter coordinates of point y2 : 42
Enter co-ordinates of point x1 : 162
enter coordinates of point y1 : 42
Enter co-ordinates of point x2 : 102
enter coordinates of point y2 : 126

Enter coordinates of centre of circle :
Enter the value of x : 102

Enter the value of y : 64

Enter the value of radius : 62
█
```



Assignment No.4

Title:- Write C++ program to draw 2-D object and perform following basic transformations,

a) Scaling b) Translation c) Rotation. Use operator overloading

Program:-

```
#include<iostream>
#include<graphics.h>
#include<math.h>
using namespace std;
class transform
{
public:
int m,a[20][20],c[20][20];
int i,j,k;
public:
void object();
void accept();
void operator *(float b[20][20])
{
for(int i=0;i<m;i++)
{
for(int j=0;j<m;j++)
{
c[i][j]=0;
for(int k=0;k<m;k++)
{
c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
}
}
}
}
};

void transform::object()
{
int gd, gm;
gd=DETECT;
initgraph(&gd,&gm,NULL);
line(300,0,300,600);
line(0,300,600,300);
for( i=0;i<m-1;i++)
{
line(300+a[i][0],300-a[i][1],300+a[i+1][0],300-a[i+1][1]);
}
line(300+a[0][0],300-a[0][1],300+a[i][0],300-a[i][1]);
for( i=0;i<m-1;i++)
```

```

{
line(300+c[i][0],300-c[i][1],300+c[i+1][0],300-c[i+1][1]);
}
line(300+c[0][0],300-c[0][1],300+c[i][0],300-c[i][1]);
int temp;
cout << "Press 1 to continue";
cin >> temp;
closegraph();
}
void transform::accept()
{
cout<<"\n";
cout<<"Enter the Number Of Edges:";
cin>>m;
cout<<"\nEnter The Coordinates :";
for(int i=0;i<m;i++)
{
for(int j=0;j<3;j++)
{
if(j>=2)
a[i][j]=1;
else
cin>>a[i][j];
}
}
}
int main()
{
int ch,tx,ty,sx,sy;
float deg,theta,b[20][20];
transform t;
t.accept();
cout<<"\nEnter your choice";
cout<<"\n1.Translation"
"\n2.Scaling"
"\n3.Rotation";
cin>>ch;
switch(ch)
{
case 1: cout<<"\nTRANSLATION OPERATION\n";
cout<<"Enter value for tx and ty:";
cin>>tx>>ty;
b[0][0]=b[2][2]=b[1][1]=1;
b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;
b[2][0]=tx;
b[2][1]=ty;
t * b;
t.object();
break;
case 2: cout<<"\nSCALING OPERATION\n";

```

```

cout<<"Enter value for sx,sy:";
cin>>sx>>sy;
b[0][0]=sx;
b[1][1]=sy;
b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;
b[2][0]=b[2][1]=0;
b[2][2] = 1;
t * b;
t.object();
break;
case 3: cout<<"\nROTATION OPERATION\n";
cout<<"Enter value for angle:";
cin>>deg;
theta=deg*(3.14/100);
b[0][0]=b[1][1]=cos(theta);
b[0][1]=sin(theta);
b[1][0]=sin(-theta);
b[0][2]=b[1][2]=b[2][0]=b[2][1]=0;
b[2][2]=1;
t * b;
t.object();
break;
default:
cout<<"\nInvalid choice";
}
getch();
return 0;
}

```

Output:-

```

d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ gedit cg4.cpp
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ g++ cg4.cpp -o vb21 -lgraph
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ ./vb21

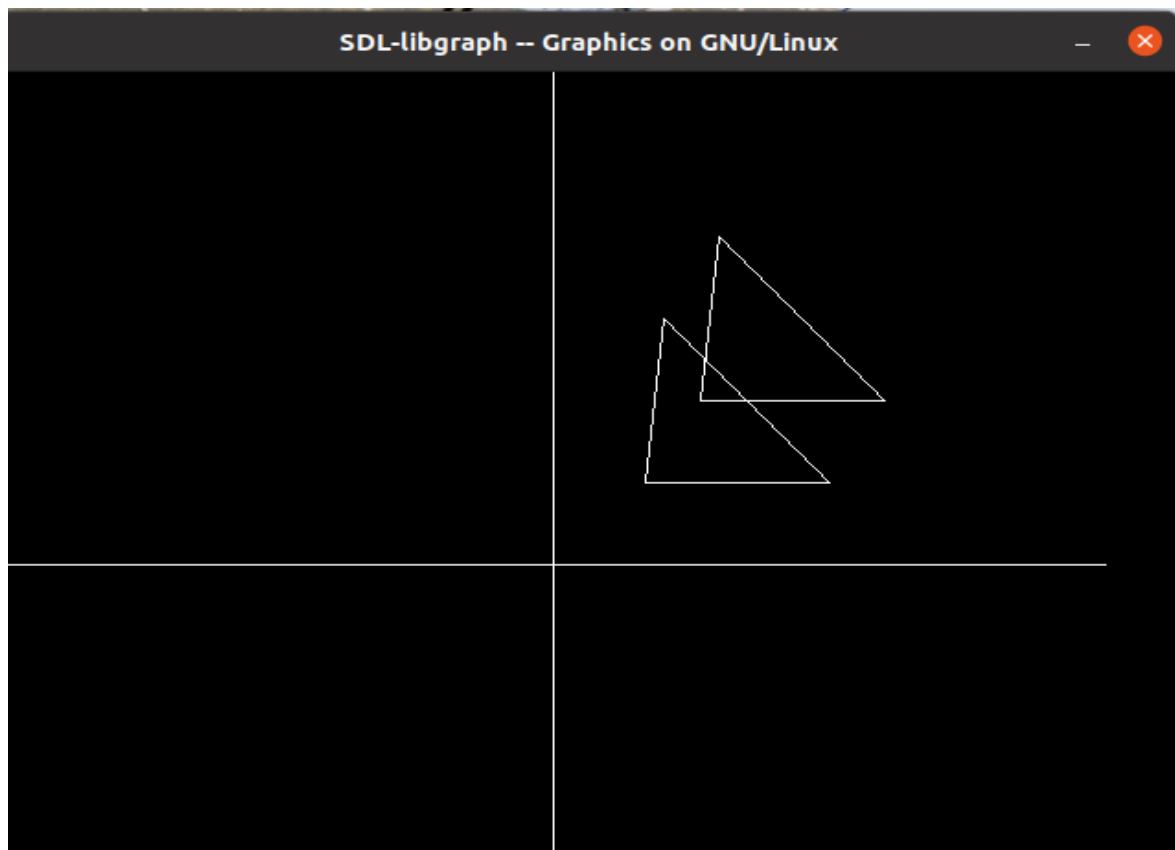
Enter the Number Of Edges:3

Enter The Coordinates :50
50
150
50
60
150

Enter your choice
1.Translation
2.Scaling
3.Rotation1

TRANSLATION OPERATION
Enter value for tx and ty:30
50
Press 1 to continue[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
vb21: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.

```



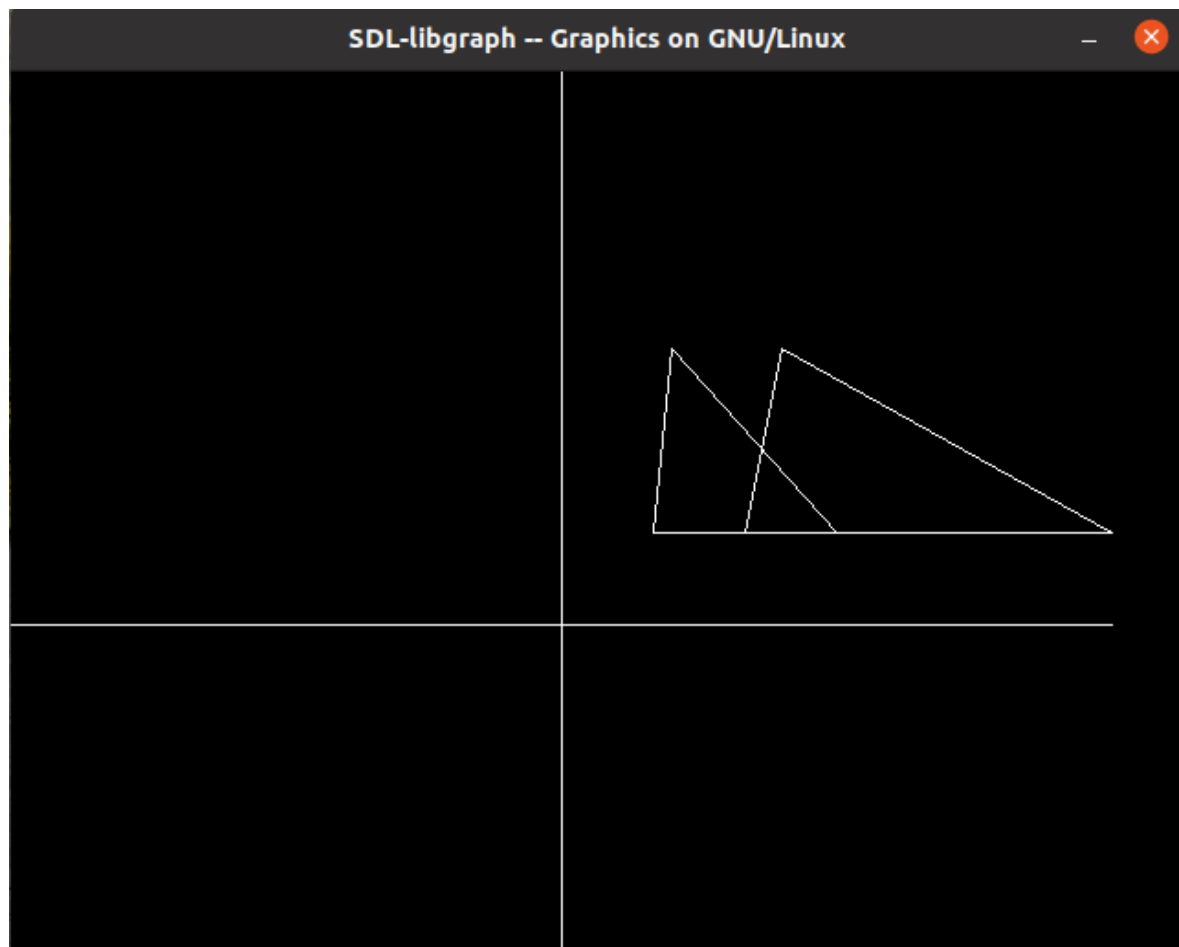
```
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ gedit cg4.cpp
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ g++ cg4.cpp -o vb21 -lgraph
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ ./vb21

Enter the Number Of Edges:3

Enter The Coordinates :50
50
150
50
60
150

Enter your choice
1.Translation
2.Scaling
3.Rotation2

SCALING OPERATION
Enter value for sx,sy:2
1
Press 1 to continue[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
vb21: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
█
```



```

d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ gedit cg4.cpp
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ g++ cg4.cpp -o vb21 -lgraph
d-comp-rsl-08@dcomprsl08-OptiPlex-3070:~$ ./vb21

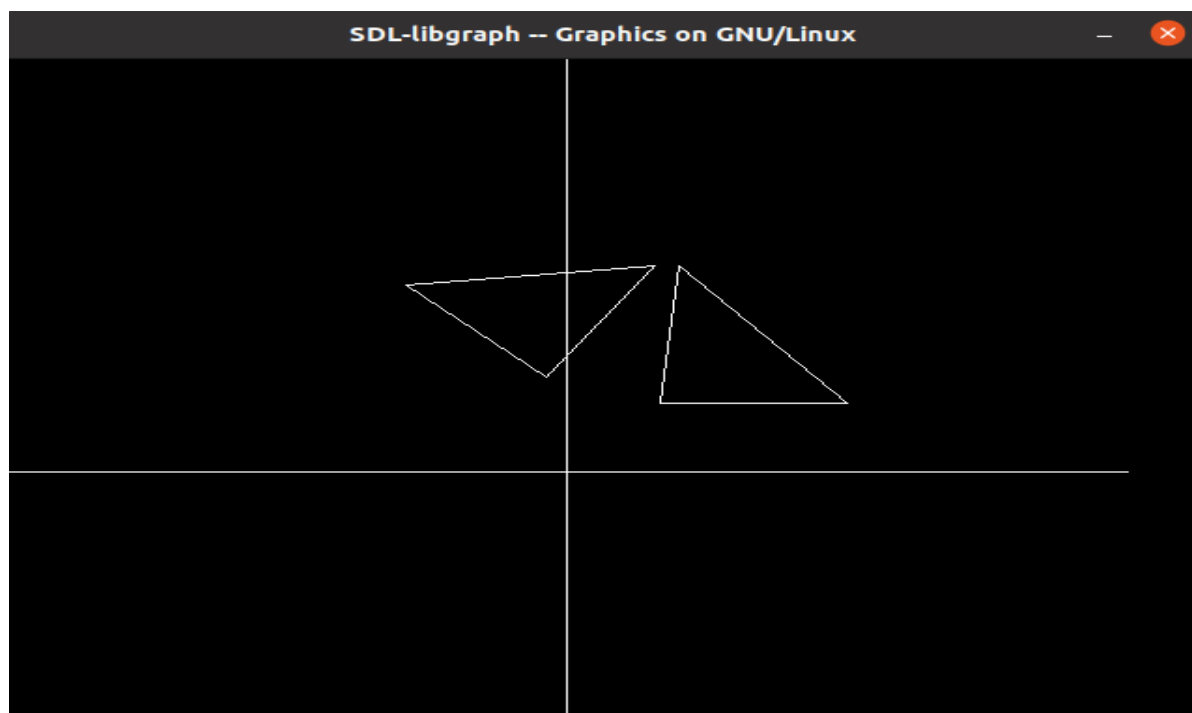
Enter the Number Of Edges:3

Enter The Coordinates :50
50
150
50
60
150

Enter your choice
1.Translation
2.Scaling
3.Rotation3

ROTATION OPERATION
Enter value for angle:30
Press 1 to continue[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
vb21: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_number' failed

```



Assignment no.5

Title:- Write C++ program to generate Hilbert curve using concept of fractals (use constructor).

Program:-

```
#include<iostream>
#include<stdlib.h>
#include<graphics.h>
#include<math.h>
using namespace std;
void move(int j, int h, int &x, int &y)
{
    if(j==1)
        y=y-h;
    else if(j==2)
        x+=h;
    else if(j==3)
        y+=h;
    else if(j==4)
        x-=h;
    lineto(x,y);
}
void hilbert(int r, int d, int l, int u, int i, int h, int &x, int &y)
{
    if(i>0)
    {
        i--;
        hilbert(d,r,u,l,i,h,x,y);
        move(r,h,x,y);
        hilbert(r,d,l,u,i,h,x,y);
        move(d,h,x,y);
        hilbert(r,d,l,u,i,h,x,y);
        move(l,h,x,y);
        hilbert(u,l,d,r,i,h,x,y);
    }
}
int main()
{
    int n,x1,y1;
    int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;

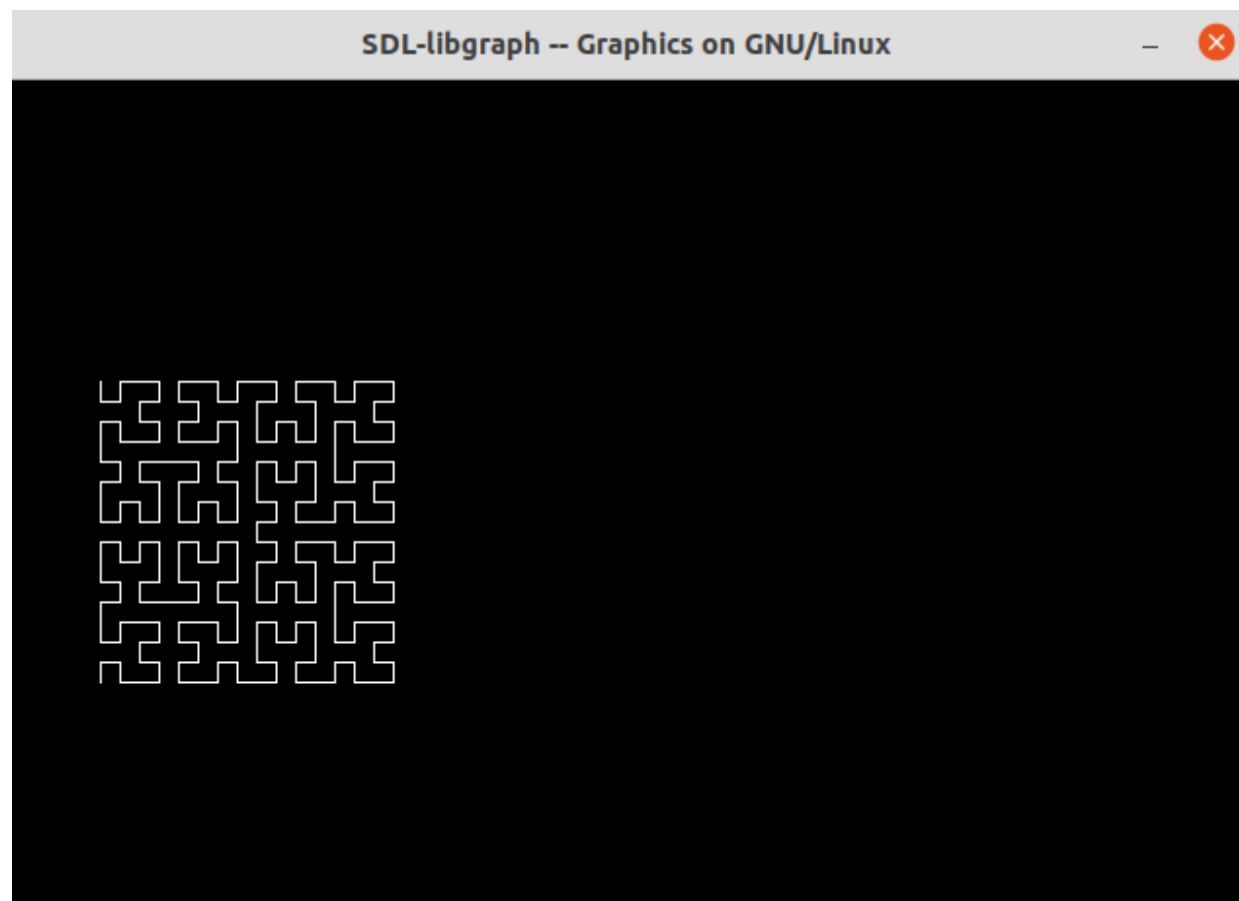
    cout<<"\n Give value of n:";
    cin>>n;
    x=x0;
    y=y0;
    int gm, gd=DETECT;
    initgraph(&gd,&gm,NULL);
    moveto(x,y);
```

```
hilbert(r,d,l,u,n,h,x,y);
delay(20000);
closegraph();
return 0;
}
```

Output:-

```
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ gedit fractal.cpp
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ g++ fractal.cpp -o hs.cpp -lgraph
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ ./hs.cpp

Give value of n:4
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
hs.cpp: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$
```



Assignment no.6

Title:- Write OpenGL program to draw Sun Rise and Sunset.

Program:-

```
#include<iostream>
#include<stdlib.h>
#ifdef __APPLE__
#include<OpenGL/OpenGL.h>
#include<GLUT/glut.h>
#else
#include<GL/glut.h>
#endif
using namespace std;
float ballX = -0.8f;
float ballY = -0.3f;
float ballZ = -1.2f;
float colR=3.0;
float colG=1.5;
float colB=1.0;
float bgColR=0.0;
float bgColG=0.0;
float bgColB=0.0;
static int flag=1;

void drawBall(void)
{
    glColor3f(colR,colG,colB);
    glTranslatef(ballX,ballY,ballZ);
    glutSolidSphere (0.05, 30, 30);
}

void drawAv(void)
{
    glBegin(GL_POLYGON);
    glColor3f(1.0,1.0,1.0);
    glVertex3f(-0.9,-0.7,-1.0);
    glVertex3f(-0.5,-0.1,-1.0);
    glVertex3f(-0.2,-1.0,-1.0);
    glVertex3f(0.5,0.0,-1.0);
    glVertex3f(0.6,-0.2,-1.0);
    glVertex3f(0.9,-0.7,-1.0);
    glEnd();
}

void drawClouds(){}

void keyPress(int key, int x, int y)
```

```

{
    if(key==GLUT_KEY_RIGHT)
        ballX -= 0.05f;
    if(key==GLUT_KEY_LEFT)
        ballX += 0.05f;
    glutPostRedisplay();
}

void initRendering()
{
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    glEnable(GL_NORMALIZE);
}

void handleResize(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0,(double)w / (double)h, 1.0, 200.0);
}

void drawScene()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glClearColor(bgColR,bgColG,bgColB,0.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f};
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);
    GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f};
    GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);
    GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f};
    GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
    glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);
    glPushMatrix();
        drawBall();
    glPopMatrix();
    glPushMatrix();
        drawAv();
    glPopMatrix();
    glPushMatrix();

```

```

        drawClouds();
        glPopMatrix();
        glutSwapBuffers();
    }

void update(int value)
{
    if(ballX>0.9f)
    {
        ballX = -0.8f;
        ballY = -0.3f;
        flag=1;
        colR=2.0;
        colG=1.50;
        colB=1.0;
        bgColB=0.0;
    }
    if(flag)
    {
        ballX += 0.001f;
        ballY +=0.0007f;
        colR-=0.001;
        colB+=0.005;
        bgColB+=0.001;
        if(ballX>0.01)
        {
            flag=0;
        }
    }
    if(!flag)
    {
        ballX += 0.001f;
        ballY -=0.0007f;
        colR+=0.001;
        colB-=0.01;
        bgColB-=0.001;
        if(ballX<-0.3)
        {
            flag=1;
        }
    }
    glutPostRedisplay();
    glutTimerFunc(25, update, 0);
}

int main(int argc,char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(400,400);

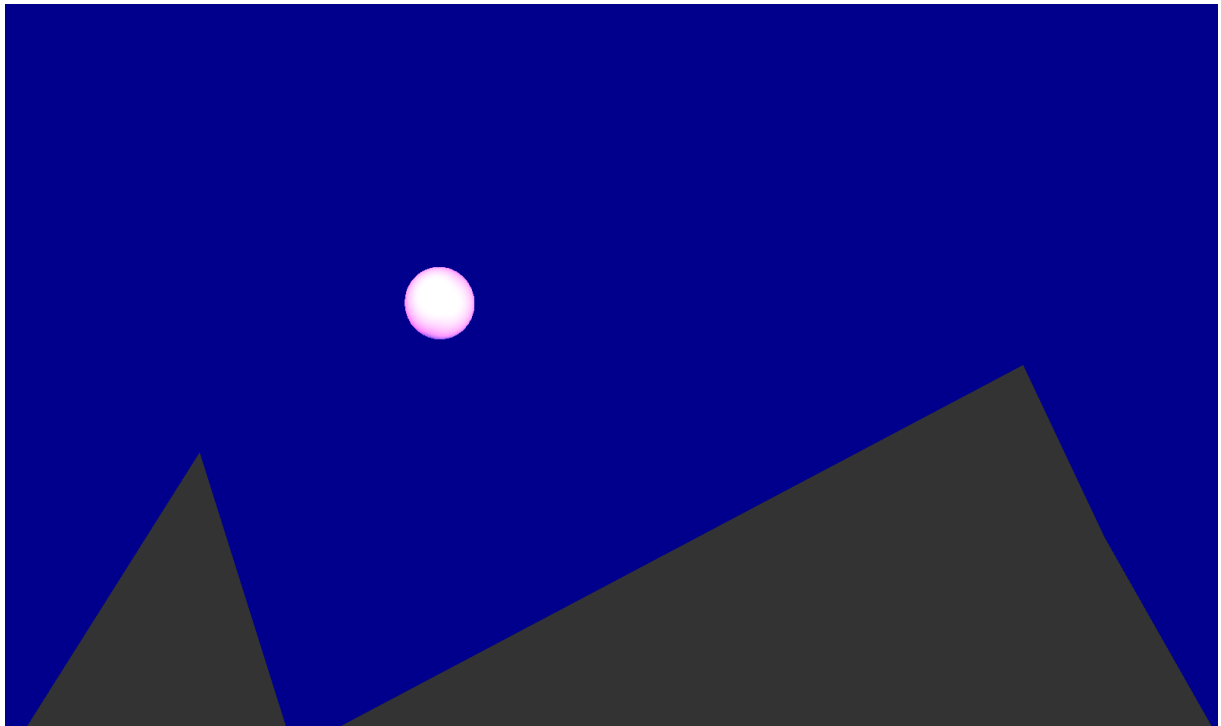
```

```
glutCreateWindow("Sun");
initRendering();
glutDisplayFunc(drawScene);
glutFullScreen();
glutSpecialFunc(keyPress);
glutReshapeFunc(handleResize);
glutTimerFunc(25, update, 0);
glutMainLoop();
return(0);
}
```

Output:-

```
d-comp-rsl-07@dcomprsl07-OptiPlex-3070:
d-comp-rsl-07@dcomprsl07-OptiPlex-3070:~$ gedit opengl.cpp
d-comp-rsl-07@dcomprsl07-OptiPlex-3070:~$ g++ opengl.cpp -o hs.cpp -lGL -lGLU -lglut
d-comp-rsl-07@dcomprsl07-OptiPlex-3070:~$ ./hs.cpp
```





Assignment no.7

Title:- Write a C++ program to implement bouncing ball using Sine wave form. Apply the concept of Polymorphism.

Program:-

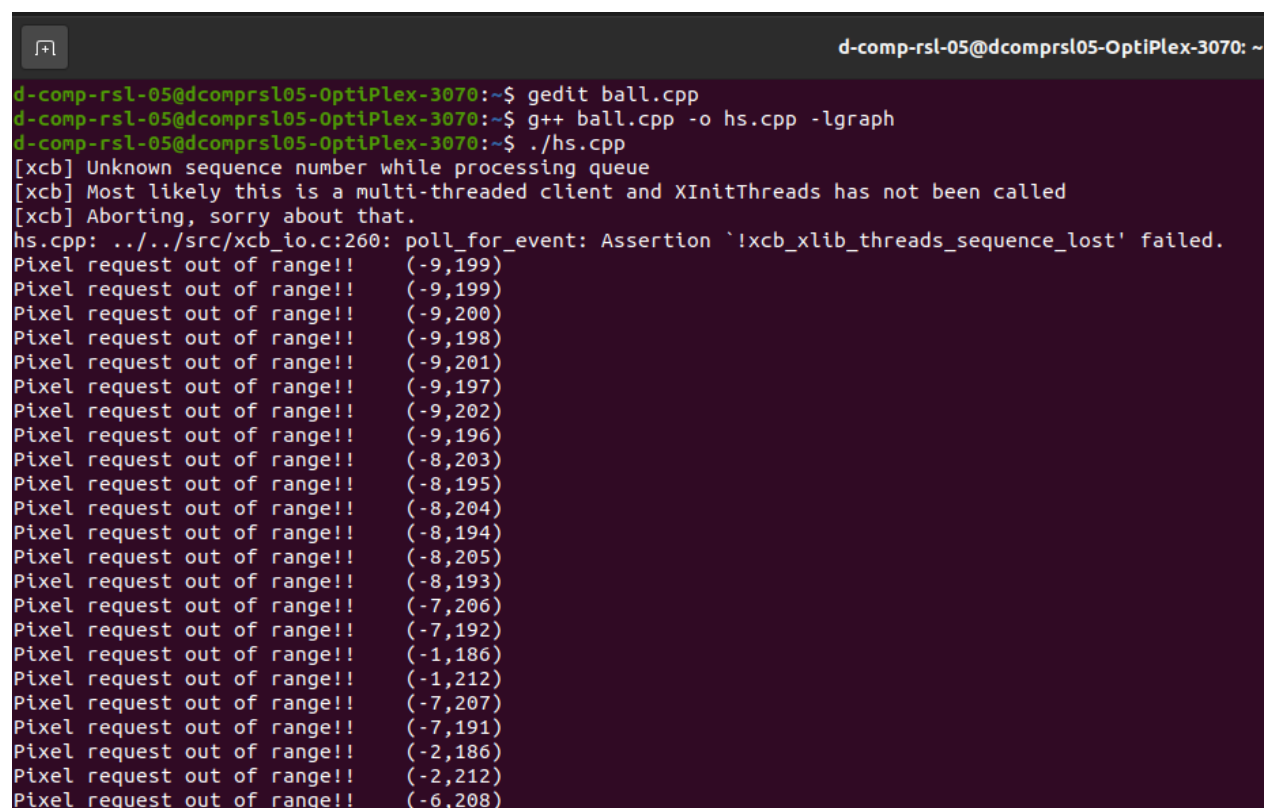
```
#include<iostream>
#include<graphics.h>
#include<math.h>
using namespace std;
class Base
{
public:
void connect()
{
float x=1,y=0.00000,j=0.5,count=0.1;
float r=15;
setcolor(9);
line(0,215,650,215);
sleep(1);
for(int k=0;k<=7;k++)
{
for(float i=90;i<270;i+=10)
{
y=cos(((i*22/7)/180))/j;
if(y>0)
y=-y;
x+=5;
setcolor(9);
circle(x,y*100+200,r);
floodfill(x,y*100+200,14);
delay(200);
setcolor(0);
circle(x,y*100+200,r);
floodfill(x,y*100+200,0);
}
}
}
};
class Derived:public Base
{
public:
void connect()
{
cout<<"You connect the wrong function ... \n"<<endl;
cout<<"Go back to Right function... \n"<<endl;
}
```

```

};
int main()
{ int gd=DETECT,gm;
initgraph(&gd,&gm,NULL);
Base *ptrToBase;
Derived objToDerived;
ptrToBase=&objToDerived;
ptrToBase->connect();
getch();
return 0;
}

```

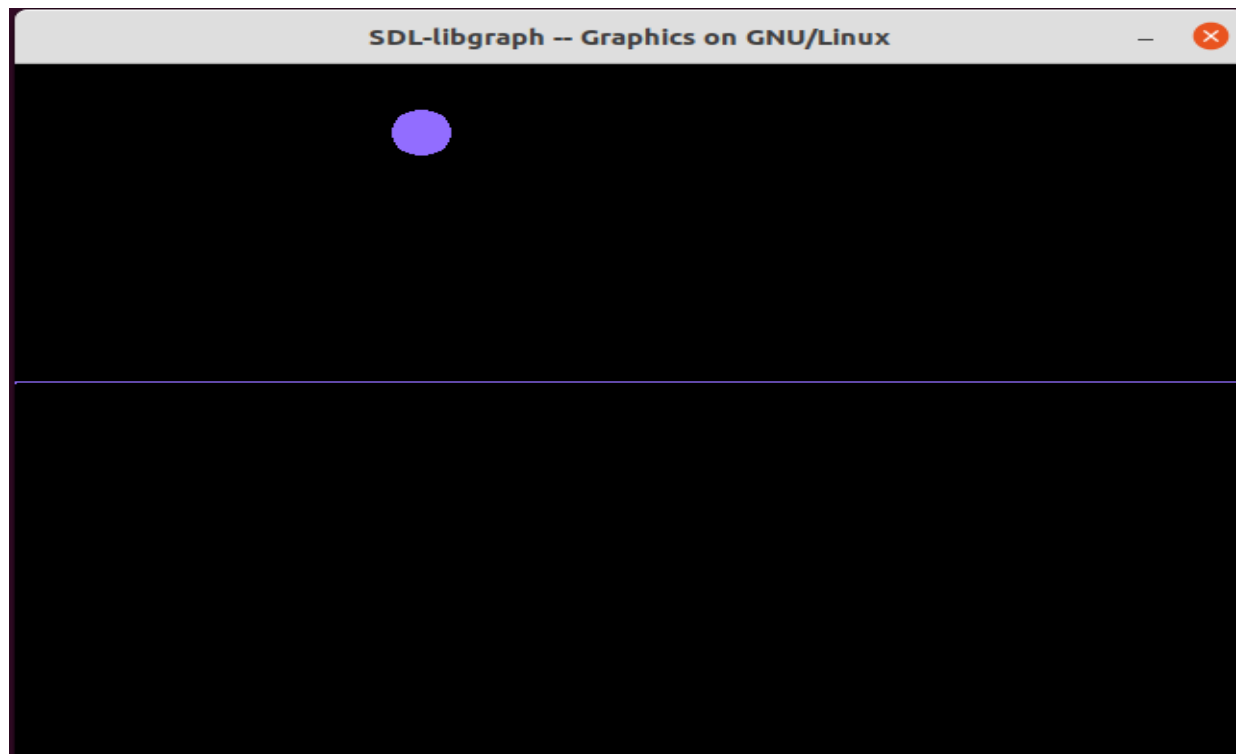
Output:-

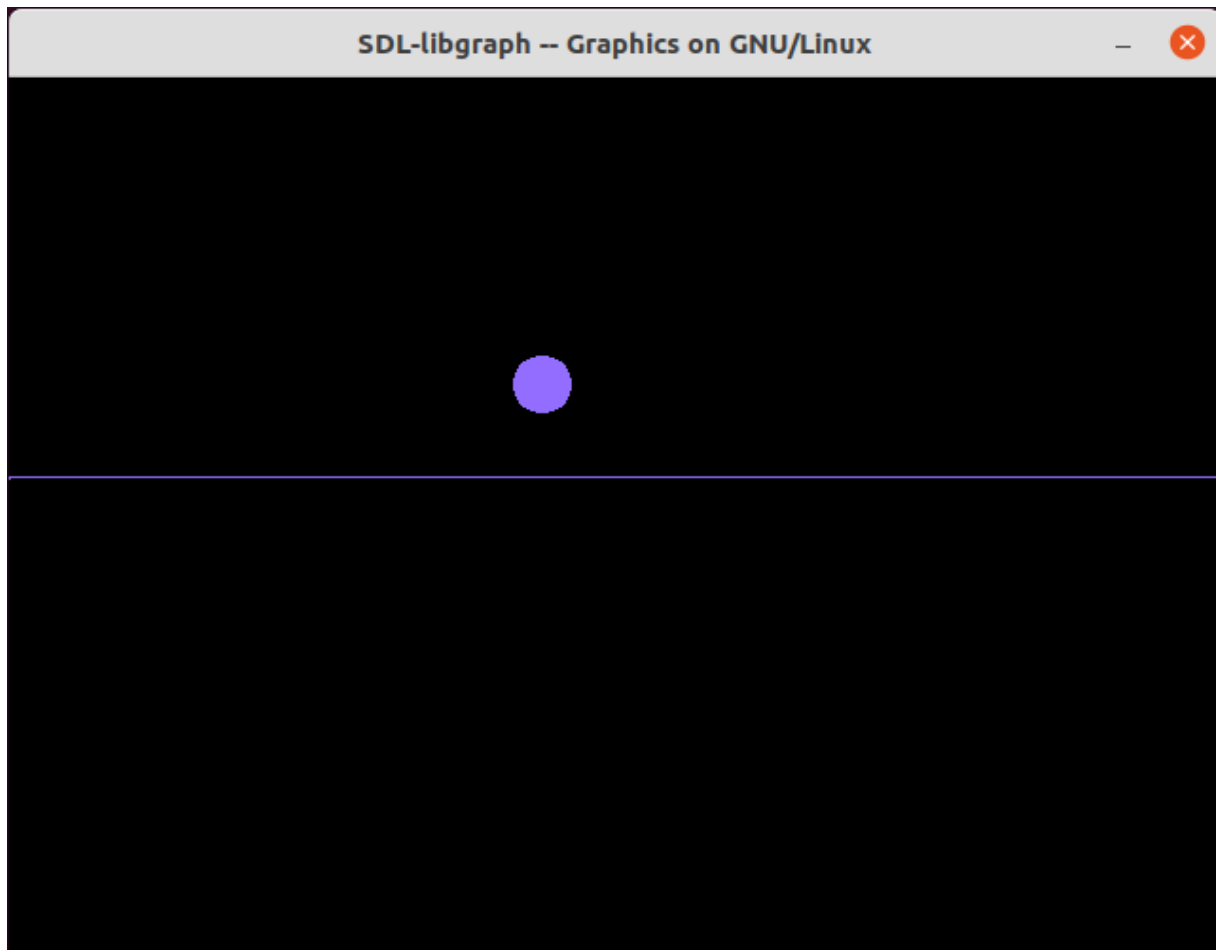


```

d-comp-rsl-05@dcomprsl05-OptiPlex-3070: ~
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ gedit ball.cpp
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ g++ ball.cpp -o hs.cpp -lgraph
d-comp-rsl-05@dcomprsl05-OptiPlex-3070:~$ ./hs.cpp
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that.
hs.cpp: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Pixel request out of range!! (-9,199)
Pixel request out of range!! (-9,199)
Pixel request out of range!! (-9,200)
Pixel request out of range!! (-9,198)
Pixel request out of range!! (-9,201)
Pixel request out of range!! (-9,197)
Pixel request out of range!! (-9,202)
Pixel request out of range!! (-9,196)
Pixel request out of range!! (-8,203)
Pixel request out of range!! (-8,195)
Pixel request out of range!! (-8,204)
Pixel request out of range!! (-8,194)
Pixel request out of range!! (-8,205)
Pixel request out of range!! (-8,193)
Pixel request out of range!! (-7,206)
Pixel request out of range!! (-7,192)
Pixel request out of range!! (-1,186)
Pixel request out of range!! (-1,212)
Pixel request out of range!! (-7,207)
Pixel request out of range!! (-7,191)
Pixel request out of range!! (-2,186)
Pixel request out of range!! (-2,212)
Pixel request out of range!! (-6,208)

```





MINI PROJECT REPORT

On

“Simple Village Using OpenGL”

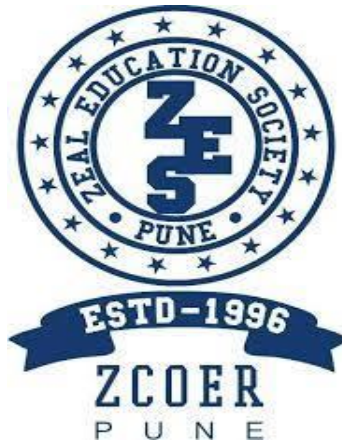
A Report Submitted for A mini project for computer graphics lab in 3rd Semester of Second Year Computer Engineering.

Second Year (COMPUTER ENGINEERING)

Academic Year 2022-23

Submitted by-

No	Name of the student	Roll No
1.	Vaishnavi Ganesh Bhokare	S211021
2.	Harshada Satish Daundkar	S211037



Zeal Education Society's

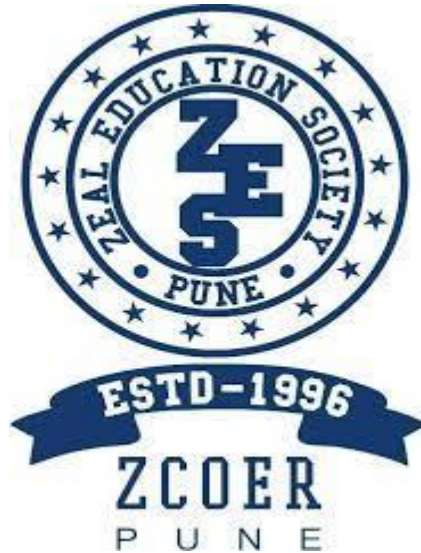
Zeal College of Engineering & Research Narhe,

Pune – 411041

Department of Computer Engineering

Zeal Education Society's

**Zeal College of Engineering & Research Department of Computer
Engineering**



CERTIFICATE

Certified that the project entitled “**Simple Village Using OpenGL**” is a bona fide work carried out by **Vaishnavi Ganesh Bhokare(S211021)**, **Harshada Satish Daundkar(S211037)**. It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

Prof. Rupali Jadhav

Project Guide

Prof. A. V. Mote

H. O. D

ACKNOWLEDGEMENT

We take this opportunity to thank our project guide **Prof. Rupali Jadhav** mam and Head of the department **Prof. A. V. Mote** mam for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of Computer Engineering Department for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, Internet access and important books.

INDEX

Sr. No	CONTENT	Page No.
1.	Abstract	5
2.	Software Requirement	6
3.	Introduction	7
4.	Problem Statement	8
5.	Objective And Outcome	8
6.	Implementation Code	8-17
7.	Output	18-20
8.	Conclusion	21
9.	References	21

ABSTRACT

This project is about the creation of Simple Village We are implementing it using different primitives available in OpenGL library and combining them together in a required manner.

A simple village is a mini project in computer graphics which is simple, good looking and useful. We have mainly created some artifacts in this mini project, like a home, sun, clouds, trees. In the project a house is present in a small village and in the sky couple of clouds will be passing in the sky and there are few trees and in the house we can turn off the light and turn on the light whenever it is needed and it has got two modes in it day mode and the night mode.

It highlights the key features of the data structures and its high quality efficiency that is obtained on its usage in the application program. This project consists of Simple Village Day-Night Scenario which is constructed by using different primitives available in OpenGL library. It illustrates the role of different callback functions that provides easier way to accomplish our project in an effective manner. The project has been implemented by efficiently using the data structures to obtain the optimized results and also various functions and features that are made available by the OpenGL software package have been utilized effectively.

SOFTWARE AND HARDWARE REQUIREMENT

❖ **Ubuntu (Linux OS)**

❖ **Programming languages:**

- **C++**

❖ **Software: GNU G++**

❖ **System: Ubuntu 20.04 LTS**

INTRODUCTION



Computers have become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer graphics so widespread. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, we find computer graphics used routinely in such diverse areas as science, engineering, medicine, business, industry, government, art, entertainment, advertising, education, and training.

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer. The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

A major use of computer graphics is in design processes. particularly for engineering and architectural systems, but almost all products are now computer designed. Generally referred to as CAD, computer-aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, and many other products. Here we have used “OpenGL” as the graphics software system to implement our mini project, “Simple Village”. Now let us have a quick look at OpenGL. It is a library for doing computer graphics. By using it, we can create interactive applications which render high-quality colour images composed of 3D geometric objects and images. OpenGL is a window and operating system independent. As such, the part of our application which does rendering is platform independent. However, in order for OpenGL to be able to render, it needs a window to draw into. Generally, this is controlled by the windowing system on whatever platform we are working on. As OpenGL is platform independent, we need some way to integrate OpenGL into each windowing system. Every windowing system where OpenGL is supported has additional API calls for managing OpenGL windows, colour maps and other features. These additional APIs are platform dependent. For the sake of simplicity we are using an additional freeware library for simplifying interacting with windowing systems GLUT. GLUT the OpenGL Utility Toolkit is a library to make writing OpenGL programs regardless of windowing systems much easier.

Problem Statement : Design and implement Simple Village using open source graphics library.

Objective : To understand the concept of OpenGL and designing the animation based mini project.

Outcome : Implementing **Simple Village** using OpenGL API.

Implementation Code:-

```
#include<iostream>
#include<GL/glut.h>
#include <GL/gl.h>
#define SPEED 30.0
using namespace std;
float i=0.0,m=0.0,n=0.0,o=0.0,c=0.0,b=0.0;
float p=0.75,q=0.47,r=0.14;
float e=0.90,f=0.91,g=0.98;
int count=0;
int light=1,day=1,plane=0,xm=900;
char ch;
void declare(char *string)
{
    while(*string)
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *string++);
}
void draw_pixel(GLint cx, GLint cy)
{
    glBegin(GL_POINTS);
    glVertex2i(cx,cy);
    glEnd();
}
void plotpixels(GLint h,GLint k, GLint x,GLint y)
{
    draw_pixel(x+h,y+k);
    draw_pixel(-x+h,y+k);
    draw_pixel(x+h,-y+k);
    draw_pixel(-x+h,-y+k);
    draw_pixel(y+h,x+k);
    draw_pixel(-y+h,x+k);
    draw_pixel(y+h,-x+k);
    draw_pixel(-y+h,-x+k);
}
void draw_circle(GLint h, GLint k, GLint r)
```

```

{
    GLint d=1-r, x=0, y=r;
    while(y>x)
    {
        plotpixels(h,k,x,y);
        if(d<0) d+=2*x+3;
        else
        {
            d+=2*(x-y)+5;
            --y;
        }
        ++x;
    }
    plotpixels(h,k,x,y);
}
void draw_object()
{
    int l;
    if(day==1)
    {
        //sky
        glColor3f(0.0,0.9,0.9);
        glBegin(GL_POLYGON);
        glVertex2f(0,380);
        glVertex2f(0,700);
        glVertex2f(1100,700);
        glVertex2f(1100,380);
        glEnd();
        //sun
        for(l=0;l<=35;l++)
        {
            glColor3f(1.0,0.9,0.0);
            draw_circle(100,625,l);
        }
        //cloud1
        for(l=0;l<=20;l++)
        {
            glColor3f(1.0,1.0,1.0);
            draw_circle(160+m,625,l);
        }
        for(l=0;l<=35;l++)
        {
            glColor3f(1.0,1.0,1.0);

```

```

    draw_circle(200+m,625,1);
    draw_circle(225+m,625,1);
}
for(l=0;l<=20;l++)
{
    glColor3f(1.0,1.0,1.0);
    draw_circle(265+m,625,1);
}
//cloud2
for(l=0;l<=20;l++)
{
    glColor3f(1.0,1.0,1.0);
    draw_circle(370+m,615,1);
}
for(l=0;l<=35;l++)
{
    glColor3f(1.0,1.0,1.0);
    draw_circle(410+m,615,1);
    draw_circle(435+m,615,1);
    draw_circle(470+m,615,1);
}
for(l=0;l<=20;l++)
{
    glColor3f(1.0,1.0,1.0);
    draw_circle(500+m,615,1);
}
//grass
glColor3f(0.6,0.8,0.196078);
glBegin(GL_POLYGON);
glVertex2f(0,160);
glVertex2f(0,380);
glVertex2f(1100,380);
glVertex2f(1100,160);
glEnd();
}
else
{
    //sky
    glColor3f(0.0,0.0,0.0);
    glBegin(GL_POLYGON);
    glVertex2f(0,380);
    glVertex2f(0,700);
    glVertex2f(1100,700);

```

```

glVertex2f(1100,380);
glEnd();
//moon
int l;
for(l=0;l<=35;l++)
{
    glColor3f(1.0,1.0,1.0);
    draw_circle(100,625,l);
}
//grass
glColor3f(0.0,0.3,0.0);
glBegin(GL_POLYGON);
glVertex2f(0,160);
glVertex2f(0,380);
glVertex2f(1100,380);
glVertex2f(1100,160);
glEnd();
}
//Ground
glColor3f(0.0,0.3,0.0);
glBegin(GL_POLYGON);
glVertex2f(-600,0);
glVertex2f(-600,185);
glVertex2f(1100,185);
glVertex2f(1100,0);
glEnd();
//tree
glColor3f(0.9,0.2,0.0);
glBegin(GL_POLYGON);
glVertex2f(280,185);
glVertex2f(280,255);
glVertex2f(295,255);
glVertex2f(295,185);
glEnd();
for(l=0;l<=30;l++)
{
    glColor3f(0.0,0.5,0.0);
    draw_circle(270,250,l);
    draw_circle(310,250,l);
}
for(l=0;l<=25;l++)
{
    glColor3f(0.0,0.5,0.0);

```

```

        draw_circle(280,290,1);
        draw_circle(300,290,1);
    }
    for(l=0;l<=20;l++)
    {
        glColor3f(0.0,0.5,0.0);
        draw_circle(290,315,1);
    }
    //tree 1
    glColor3f(0.9,0.2,0.0);
    glBegin(GL_POLYGON);
    glVertex2f(100,135);
    glVertex2f(100,285);
    glVertex2f(140,285);
    glVertex2f(140,135);
    glEnd();
    for(l=0;l<=40;l++)
    {
        glColor3f(0.0,0.5,0.0);
        draw_circle(40,280,1);
        draw_circle(90,280,1);
        draw_circle(150,280,1);
        draw_circle(210,280,1);
        draw_circle(65,340,1);
        draw_circle(115,340,1);
        draw_circle(175,340,1);
    }
    for(l=0;l<=55;l++)
    {
        glColor3f(0.0,0.5,0.0);
        draw_circle(115,360,1);
    }
    //chim
    glColor3f(0.35,0.0,0.0);
    glBegin(GL_POLYGON);
    glVertex2f(540,330);
    glVertex2f(540,430);
    glVertex2f(960,430);
    glVertex2f(960,330);
    glEnd();
    //home
    glColor3f(p,q,r);
    glBegin(GL_POLYGON);

```

```

    glVertex2f(550,100);
    glVertex2f(550,330);
    glVertex2f(950,330);
    glVertex2f(950,100);
    glVertex2f(850,100);
    glVertex2f(850,250);
    glVertex2f(650,250);
    glVertex2f(650,100);
    glEnd();
//window border
    glColor3f(0.35,0.0,0.0);
    glBegin(GL_POLYGON);
    glVertex2f(595,205);
    glVertex2f(595,285);
    glVertex2f(675,285);
    glVertex2f(675,205);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(825,205);
    glVertex2f(825,285);
    glVertex2f(905,285);
    glVertex2f(905,205);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(845,205);
    glVertex2f(845,285);
    glVertex2f(850,285);
    glVertex2f(850,205);
    glEnd();
//door
    glColor3f(e,f,g);
    glBegin(GL_POLYGON);
    glVertex2f(800,100);
    glVertex2f(800,220);
    glVertex2f(700,220);
    glVertex2f(700,100);
    glEnd();
    glColor3f(0.35,0.0,0.0);
    glBegin(GL_POLYGON);
    glVertex2f(760,120);
    glVertex2f(760,200);
    glVertex2f(700,220);
    glVertex2f(700,100);

```

```

        glEnd();
//window
        glColor3f(e,f,g);
        glBegin(GL_POLYGON);
            glVertex2f(600,210);
            glVertex2f(600,280);
            glVertex2f(670,280);
            glVertex2f(670,210);
        glEnd();
        glBegin(GL_POLYGON);
            glVertex2f(830,210);
            glVertex2f(830,280);
            glVertex2f(900,280);
            glVertex2f(900,210);
        glEnd();
        glColor3f(0.35,0.0,0.0);
        glBegin(GL_POLYGON);
            glVertex2f(620,210);
            glVertex2f(620,280);
            glVertex2f(625,280);
            glVertex2f(625,210);
        glEnd();
        glBegin(GL_POLYGON);
            glVertex2f(650,210);
            glVertex2f(650,280);
            glVertex2f(655,280);
            glVertex2f(655,210);
        glEnd();
        glColor3f(0.35,0.0,0.0);
        glBegin(GL_POLYGON);
            glVertex2f(850,205);
            glVertex2f(850,285);
            glVertex2f(855,285);
            glVertex2f(855,205);
        glEnd();
        glBegin(GL_POLYGON);
            glVertex2f(880,205);
            glVertex2f(880,285);
            glVertex2f(885,285);
            glVertex2f(885,205);
        glEnd();
    glFlush();
}

```

```

void idle()
{
if(light==0 && (i>=0 && i<=1150))
{
    i+=SPEED/10;
    m+=SPEED/150;
    n-=2;
    o+=0.2;
    c+=2;
}
if(light==0 && (i>=2600 && i<=3000))
{
    i+=SPEED/10;
    m+=SPEED/150;
    n-=2;
    o+=0.2;
    c+=2;
}
}
if(light==0)
{
    i=i;
    m+=SPEED/150;
    n-=2;
    o+=0.2;
    c+=2;
}
}
if(count<=3)
{
glClearColor(1.0,1.0,1.0,1.0);
    i+=SPEED/10;
    b+=SPEED/10;
    m+=SPEED/150;
    n-=2;
    o+=0.2;
    c+=2;
}
}
if(i>1900)
    i=800.0;
if(m>1100)
    m=0.0;
glutPostRedisplay();
}
void mouse(int btn,int state,int x,int y)

```



```

{
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_UP)
        exit(0);
}
void keyboardFunc( unsigned char key, int x, int y )
{
    switch( key )
    {
        case 'd':
            case 'D':
                day=1;
                p=0.75;
                q=0.47;
                r=0.14;
                break;

            case 'n':
        case 'N':
                day=0;
                p=0.52;
                q=0.37;
                r=0.26;
                break;

            case 'l':
        case 'L':
                e=0.90;
                f=0.91;
                g=0.98;
                break;

            case 'f':
        case 'F':
                e=0.0;
                f=0.0;
                g=0.0;
                break;

    };
}
void myinit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(0.0,0.0,1.0);
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);

```

```

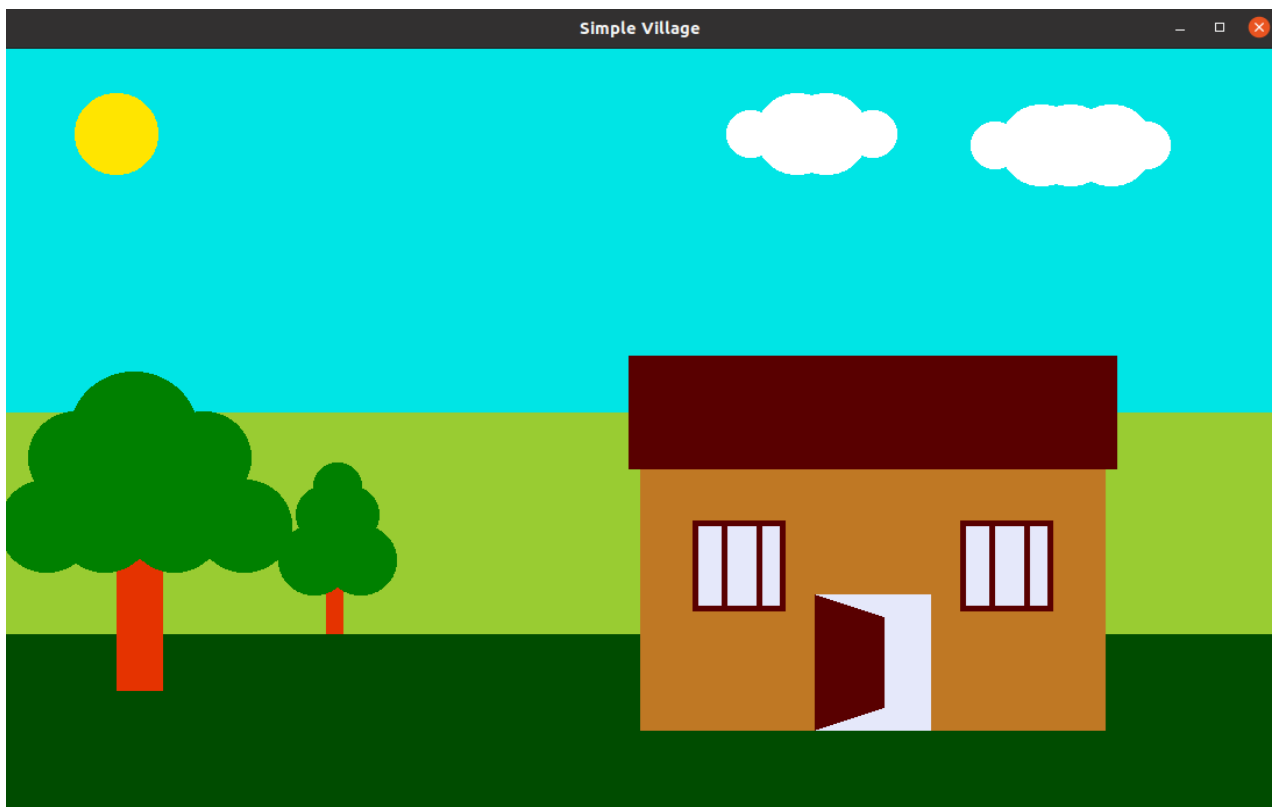
glLoadIdentity();
gluOrtho2D(0.0,1100.0,0.0,700.0);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
draw_object();
glFlush();
}
int main(int argc,char** argv)
{
int c_menu;
    cout<<"Project by CSEMiniProjects.com\n";
    cout<<"-----";
    cout<<"          Simple Village          "<<<endl;
    cout<<"-----\n\n";
    cout<<"Press 'd' or 'D' to make it day. \n\n";
    cout<<"Press 'n' or 'N' to make it night. \n\n";
    cout<<"Press 'l' or 'L' to turn On the lights. \n\n";
    cout<<"Press 'f' or 'F' to turn Off the lights. \n\n";
    cout<<"Press RIGHT MOUSE BUTTON to display menu. \n\n";
    cout<<"Press LEFT MOUSE BUTTON to quit the program. \n\n\n";
    cout<<"Press any key and Hit ENTER.\n";
    cin>>ch;
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(1100.0,700.0);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Simple Village");
    glutDisplayFunc(display);
    glutIdleFunc(idle);
    glutKeyboardFunc(keyboardFunc);
    glutMouseFunc(mouse);
    myinit();
    glutAddMenuEntry("Aeroplane",1);
    glutAddMenuEntry("Comet",2);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutMainLoop();
    return 0;
}

```

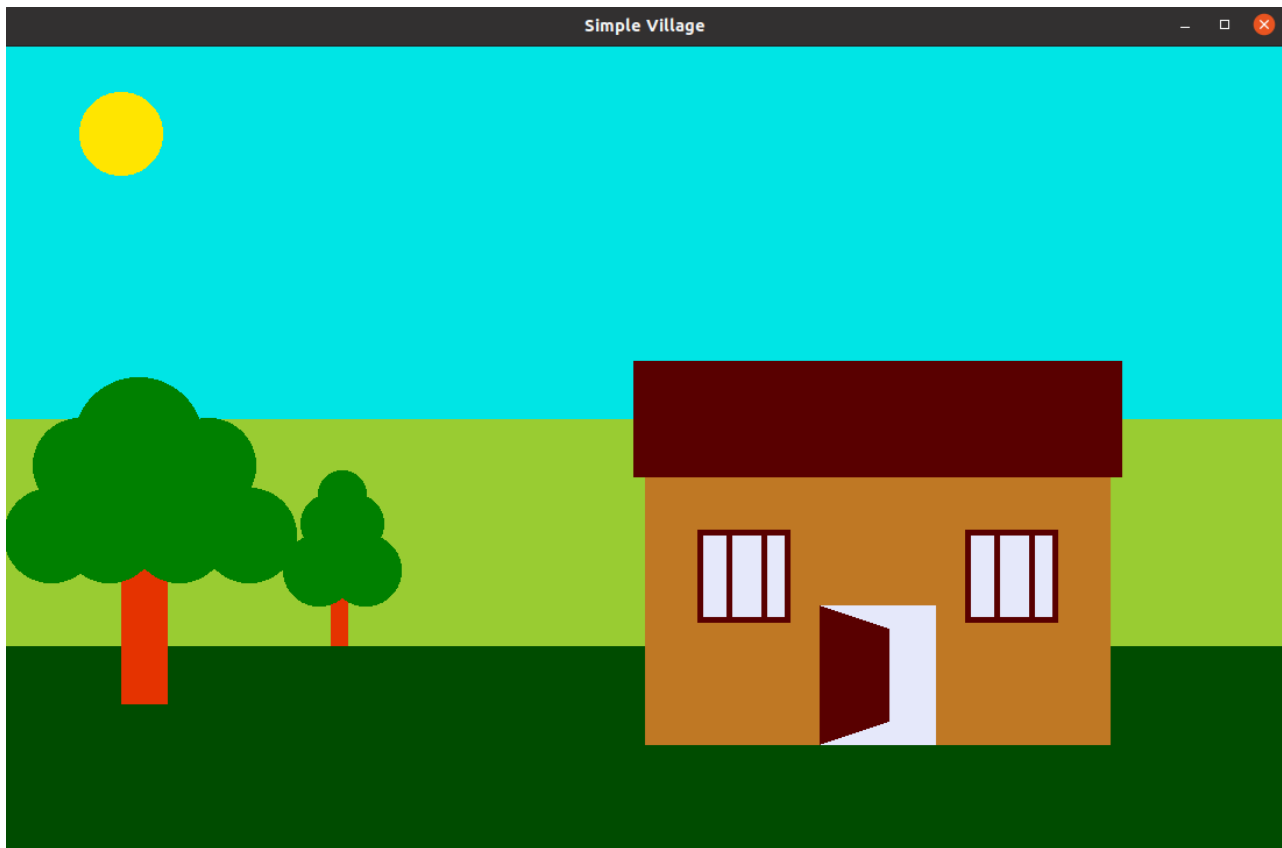
Output:-

```
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070: ~  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ g++ village.cpp -o hv.cpp -lGL -lGLU -lglut  
d-comp-pl-ii-18@dcompplii18-OptiPlex-3070:~$ ./hv.cpp  
Project by CSEMiniProjects.com  
-----  
Simple Village  
-----  
Press 'd' or 'D' to make it day.  
Press 'n' or 'N' to make it night.  
Press 'l' or 'L' to turn On the lights.  
Press 'f' or 'F' to turn Off the lights.  
Press RIGHT MOUSE BUTTON to display menu.  
Press LEFT MOUSE BUTTON to quit the program.  
Press any key and Hit ENTER.  
d
```

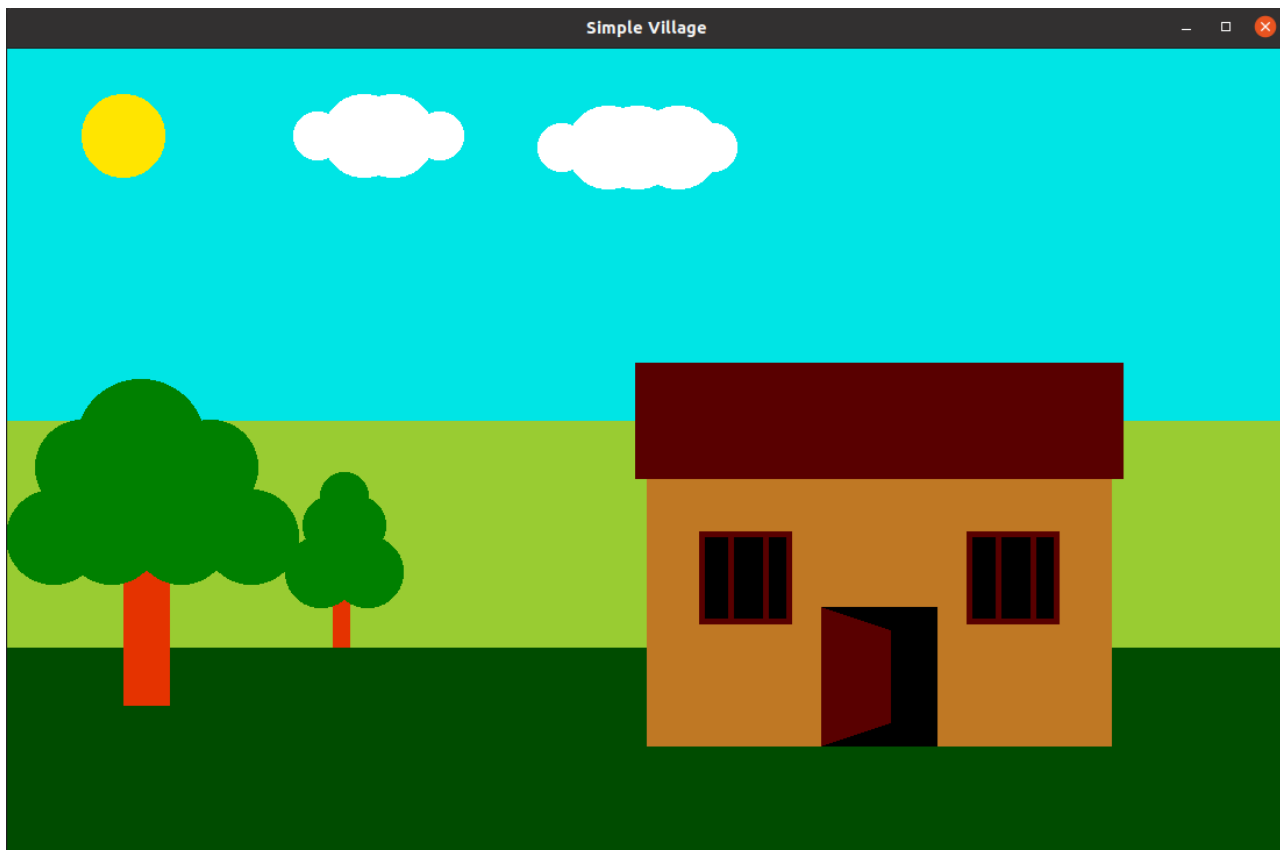
1)Day Mode with clouds in the sky.



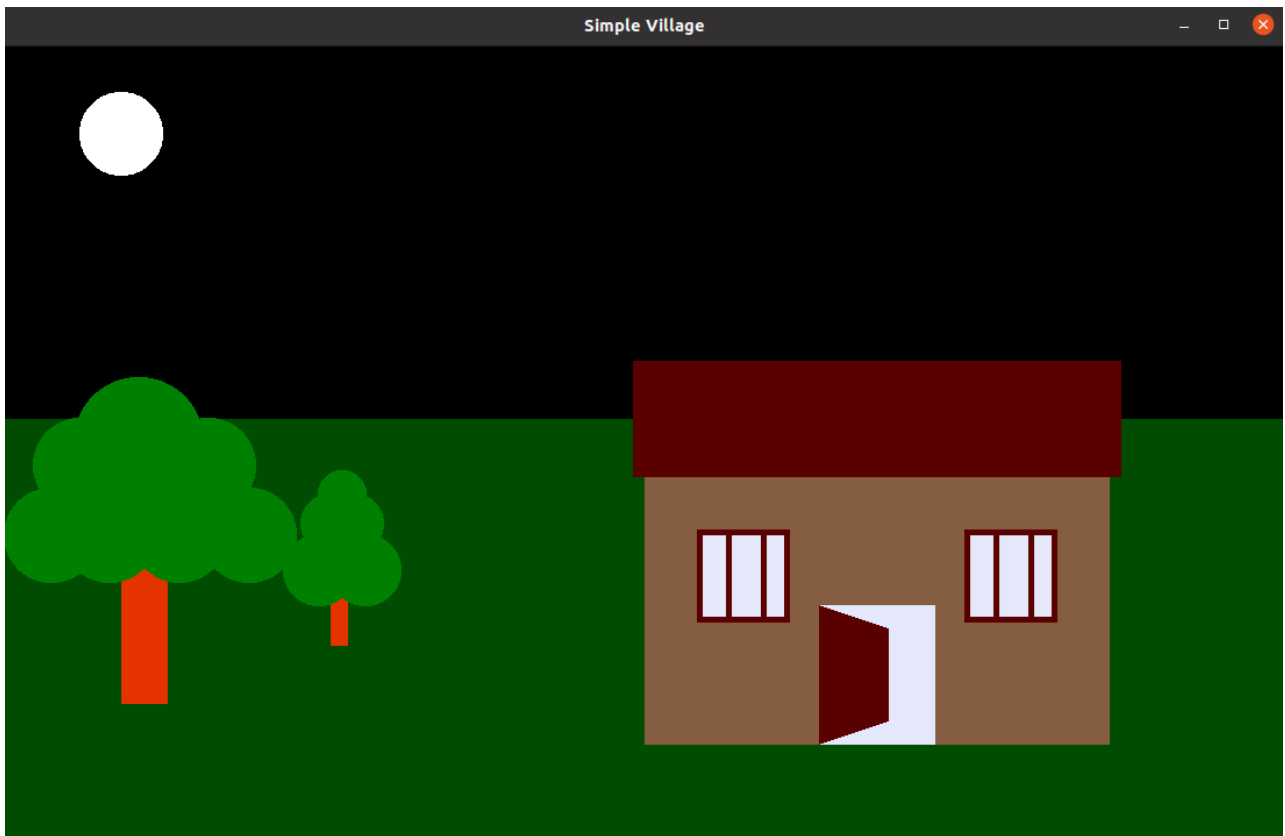
2)Day mode without clouds in the sky.



3)Day mode with Light OFF in the house.



4)Night mode with light ON in the house.



5)Night mode with light OFF in the house.



CONCLUSION

While developing a system a conscious effort has been made to create and develop a software package, making use of available tools, techniques and resources – that will generate the proper system cases.

While making the system an eye has been kept on making it as user friendly as such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. As in case of any system development process where are number of short comings, there have been some short comings in the development of this system also.

References:-

Source Code:- <https://drive.google.com/file/d/1emd5>

OpenGL : 1) <https://www.opengl.org/>
2) <https://en.wikipedia.org/wiki/OpenGL>
3) https://www.khronos.org/opengl/wiki/Getting_Starte

