

Computer Graphics Practical

Write C++ program to draw a concave polygon and fill it with desired color using scan fill algorithm. Apply the concept of inheritance.

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

Write C++ program to draw a given pattern. Use DDA line and Bresenham's circle drawing algorithm. Apply the concept of encapsulation.

Write C++ program to draw 2-D object and perform following basic transformations: Scaling. Apply the concept of operator overloading.

Write C++ program to draw 2-D object and perform following basic transformations: Translation. Apply the concept of operator overloading.

Write C++ program to draw 2-D object and perform following basic transformations: Rotation. Apply the concept of operator overloading.

Write C++ program to generate Hilbert curve using concept of fractals.

Write a C++ program to implement bouncing ball using sine wave form. Apply the concept of polymorphism.

Translation Program:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

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,"C:\\\\TURBOC3\\\\BGI");
    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;
    float b[20][20];
    transform t;
    t.accept();

    cout<<"\nEnter your choice";
    cout<<"\n1.Translation";
        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;

        default:
            cout<<"\nInvalid choice";
    }

    getch();
    return 0;
}

```

Scaling Program:

```

#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

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, "C:\\TURBOC3\\BGI");
    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,sx,sy;
    float b[20][20];
    transform t;
    t.accept();

    cout<<"\nEnter your choice";
    cout<<"\n1. Scaling";
    cin>>ch;
    switch(ch)
    {

```

```

        case 1: 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;
        default:
                cout<<"\nInvalid choice";

    }

    getch();

    return 0;
}

```

Rotation Program:

```

#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

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,"C:\\\\TURBOC3\\\\BGI");
    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;
    float deg,theta,b[20][20];
    transform t;
    t.accept();

    cout<<"\nEnter your choice";
    cout<<"\n1.Rotation";
        cin>>ch;
        switch(ch)
        {

            case 1: 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;
}
```