MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R Patna, Mandya -571477.

COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT (15CSL68)

Academic Year: 2018-19 (Even Semester)

By:
Prof. E Santosh
Asst. Prof.
Dept. of CSE
MIT Mysore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Maharaja Institute of Technology Mysore

Vision

“ To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude”

Mission

1. To empower students with indispensable knowledge through dedicated teaching and collaborative learning.
2. To advance extensive research in science, engineering and management disciplines.
3. To facilitate entrepreneurial skills through effective institute-industry collaboration and interaction with alumni.
4. To instill the need to uphold ethics in every aspect.
5. To mould holistic individuals capable of contributing to the advancement of the society.

Department of Computer Science and Engineering

Vision

"To be a leading academic department offering computer science and engineering education, fulfilling industrial and societal needs effectively."

Mission

M1 : To enrich the technical knowledge of students in diversified areas of Computer Science and Engineering by adopting outcome based approaches.

M2 : To empower students to be competent professionals maintaining ethicality.

M3 : To facilitate the development of academia-industry collaboration.

M4 : To create awareness of entrepreneurship opportunities.

Program Educational Objectives Statements

PEO1	Be successful in solving engineering problems associated with computer science and engineering domains
PEO2	Work collaboratively on multidisciplinary projects and acquire high levels of professionalism backed by ethics
PEO3	Communicate effectively and exhibit leadership qualities, team spirit necessary for a successful career in either industry, research or entrepreneurship
PEO4	Continue to learn and advance their career through participation in the activities of professional bodies, obtaining professional certification, pursue of higher education

Program Specific Outcome (PSO)

PSO 1: Apply software engineering practices and strategies in diversified areas of computer science for solving problems using open source environment.

PSO 2: Develop suitable algorithms and codes for applications in areas of cognitive technology, computer networks with software engineering principles and practices.

INDEX PAGE

General Lab Guidelines:

- Maintain laboratory etiquettes during the laboratory sessions.
- Do not wander around or distract other students or interfere with the conduction of the experiments of other students.
- Keep the laboratory clean, do not eat, drink or chew gum in the laboratory.

DO’S

- Sign the log book when you enter/leave the laboratory.
- Read the hand out/procedure before starting the experiment. If you do not understand the procedure, clarify with the concerned staff.
- Report any problem in system (if any) to the person in-charge.
- After the lab session, shut down the computers.
- All students in the laboratory should follow the directions given by staff/lab technical staff.

DON’TS

- Do not insert metal objects such as pins, needle or clips into the computer casing. They may cause fire.
- Do not open any irrelevant websites in labs.
- Do not use flash drive on laboratory computers without the consent of lab instructor.
- Do not upload, delete or alter any software/ system files on laboratory computers.
- Students are not allowed to work in laboratory alone or without presence of the teaching staff/ instructor.
- Do not change the system settings and keyboard keys.
- Do not damage any hardware.

COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2016 -2017)
SEMESTER – VI

Subject Code	15CSL68	IA Marks	20
Number of Lecture Hours/Week	01I + 02P	Exam Marks	80
Total Number of Lecture Hours	40	Exam Hours	03

CREDITS – 02

Course objectives: This course will enable students to

- Demonstrate simple algorithms using OpenGL Graphics Primitives and attributes.
- Implementation of line drawing and clipping algorithms using OpenGL functions
- Design and implementation of algorithms Geometric transformations on both 2D and 3D objects.

Description (If any):

--

Lab Experiments:

PART A
Design, develop, and implement the following programs using OpenGL API

1. Implement Brenham’s line drawing algorithm for all types of slope.
   Refer:Text-1: Chapter 3.5
   Refer:Text-2: Chapter 8
2. Create and rotate a triangle about the origin and a fixed point.
   Refer:Text-1: Chapter 5-4
3. Draw a colour cube and spin it using OpenGL transformation matrices.
   Refer:Text-2: Modelling a Coloured Cube
4. Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.
   Refer:Text-2: Topic: Positioning of Camera
5. Clip a lines using Cohen-Sutherland algorithm
   Refer:Text-1: Chapter 6.7
   Refer:Text-2: Chapter 8
6. To draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of the light source along with the properties of the surfaces of the solid object used in the scene.
   Refer:Text-2: Topic: Lighting and Shading
7. Design, develop and implement recursively subdivide a tetrahedron to form 3D sierpinski gasket. The number of recursive steps is to be specified by the user.
   Refer: Text-2: Topic: sierpinski gasket.
8. Develop a menu driven program to animate a flag using Bezier Curve algorithm
   Refer: Text-1: Chapter 8-10
9. Develop a menu driven program to fill the polygon using scan line algorithm

Project:

PART –B (MINI-PROJECT) :

Student should develop mini project on the topics mentioned below or similar applications using Open GL API. Consider all types of attributes like color, thickness, styles, font, background, speed etc., while doing mini project.

(**During the practical exam: the students should demonstrate and answer Viva-Voce**)

Sample Topics:

Simulation of concepts of OS, Data structures, algorithms etc.

Course outcomes: The students should be able to:

- Apply the concepts of computer graphics
- Implement computer graphics applications using OpenGL
- Animate real world problems using OpenGL

Conduction of Practical Examination:

1. All laboratory experiments from part A are to be included for practical examination.
2. Mini project has to be evaluated for 30 Marks as per 6(b).
3. Report should be prepared in a standard format prescribed for project work.
4. Students are allowed to pick one experiment from the lot.
5. Strictly follow the instructions as printed on the cover page of answer script.
6. Marks distribution:
   a) Part A: Procedure + Conduction + Viva:10 + 35 +5 =50 Marks
   b) Part B: Demonstration + Report + Viva voce = 15+10+05 = 30 Marks
7. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

Reference books:

1. Donald Hearn & Pauline Baker: Computer Graphics-OpenGL Version,3rd Edition, Pearson Education,2011
2. Edward Angel: Interactive computer graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2011
3. M M Raikar, Computer Graphics using OpenGL, Fillip Learning / Elsevier, Bangalore / New Delhi (2013)

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
BELAWADI, SRIRANGAPATNA Taluk, MANDYA-571477
Department of Computer Science and Engineering

Academic Year: **2018-19** Date: **28/01/2019**

Course Outcome

Subject: **Computer Graphics Laboratory with Mini Project**

Course Code: **C368**

Subject Code**: 15CSL68**

CO's	DESCRIPTION OF THE OUTCOMES
C368.1	**Understand and apply the concepts of computer graphics using OpenGL.**
C368.2	**Apply the various techniques and API's of OpenGL to build computer graphic objects.**
C368.3	**Present 2D and 3D Computer Graphics using OpenGL and document it.**

CO No	PO No												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
C368.1	3	-	-	-	1	-	-	-	-	-	-	-	3	-
C368.2	3	-	-	-	1	-	-	-	-	-	-	-	3	-
C368.3	-	-	3	-	2	-	-	-	2	1	1	-	3	3
CO Average	**3.00**	**-**	**3.00**	**-**	**1.33**	**-**	**-**	**-**	**2.00**	**1.00**	**1.00**	**-**	**3.00**	**3.00**

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
BELAWADI, SRIRANGAPATNA Taluk, MANDYA-571477
Department of Computer Science and Engineering

Academic Year: **2018-19** Date: **28/01/2019**

Course Outcome

Subject: **Computer Graphics Laboratory with Mini Project**

Course Code: **C368**

Subject Code**: 15CSL68**

Lab Program	CO-1 (5) Comprehension	CO-2 (5) Implementation	CO-3 (10) Presentation & Documentation	Total (20 Marks)	Faculty Signature
Lab Program-1					
Lab Program-2					
Lab Program-3					
Lab Program-4					
Lab Program-5					
Lab Program-6					
Lab Program-7					
Lab Program-8					
Lab Program-9					
Mini Project					
Internal Test					
Average					

Max. Marks	Marks Obtained *(In Figure)*	Marks Obtained *(In Words)*
20		

(Faculty In-Charge) (H O D)

LAB PROGRAM – 1

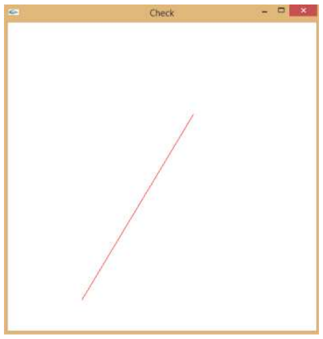
| Problem Statement |

Implement Brenham's line drawing algorithm for all types of slope.

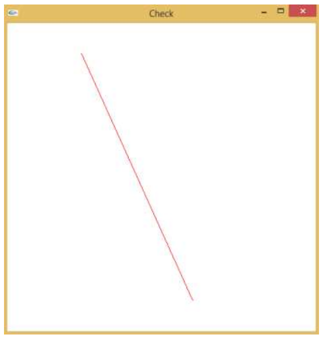
| Source Code |

```
#include<GL/glut.h>
#include<math.h>
void Draw()
{
    GLfloat x1=120,y1=50,x2=300,y2=350;
    GLfloat m,x,y,dx,dy,p,temp;
    glClear(GL_COLOR_BUFFER_BIT);
    m = (y2-y1) / (x2-x1);
    if(fabs(m)<1)
    {
        if(x1>x2)
        {
            temp = x1;
            x1 = x2;
            x2 = temp;

            temp = y1;
            y1 = y2;
            y2 = temp;
        }
        dx = fabs(x2 - x1);
        dy = fabs(y2 - y1);
        x=x1;
        y=y1;
        p = 2*dy-dx;
        while(x<=x2)
        {
        glBegin(GL_POINTS);
            glVertex2f(x,y);
        glEnd();
        x=x+1;
        if(p>=0)
        {
            if(m>=1)
                y=y+1;
            else
                y=y-1;
            p = p + 2*dy-2*dx;
        }
        else
        {
            y=y;
            p = p + 2*dy;
        }
        }
    }
    if(fabs(m)>=1)
    {
        if(y1>y2)
        {
            temp = x1;              x1 = x2;              x2 = temp;
            temp = y1;              y1 = y2;              y2 = temp;
        }

        dx = fabs(x2 - x1);
        dy = fabs(y2 - y1);
        x=x1;
        y=y1;
        p = 2*dx-dy;
        while(y<=y2)
        {
        glBegin(GL_POINTS);
            glVertex2f(x,y);
        glEnd();
        y=y+1;
        if(p>=0)
        {
            if(m>=1)
                x=x+1;
            else
                x=x-1;
            p = p + 2*dx-2*dy;
        }
        else
        {
            x=x;
            p = p + 2*dx;
        }

        }
    }
    glFlush();
}

void MyInit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glMatrixMode(GL_MODELVIEW);
    glClearColor(1,1,1,1);
    glColor3f(1,0,0);
}

int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Check");
    MyInit();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

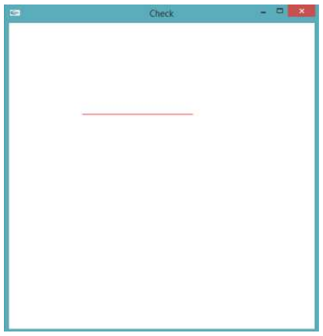
Output



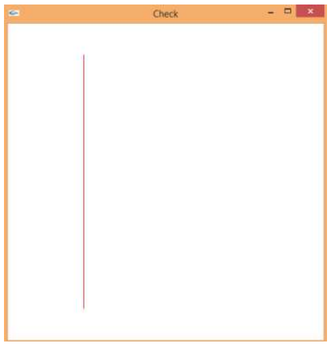
Positive Slope



Negative Slope



Zero Slope



Undefined Slope

LAB PROGRAM – 2

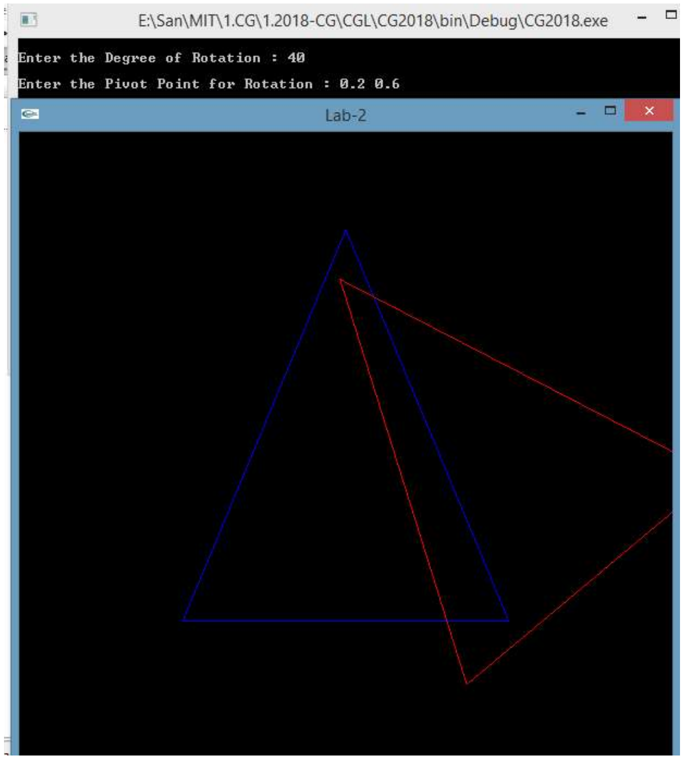
Problem Statement

Create and rotate a triangle about the origin and a fixed point.

Source Code

```
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
GLfloat d;
GLfloat rX=0,rY=0;
void Draw()
{
    GLfloat P[3][2] = {{-0.5,0},{0.5,0},{0,0.6}};
    GLfloat nP[3][2],r;
    int i;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
        glVertex2fv(P[0]);
        glVertex2fv(P[1]);
        glVertex2fv(P[2]);
    glEnd();
    r = d * 3.14/180;
    for(i=0;i<3;i++)
    {
        nP[i][0] = P[i][0]*cos(r)-P[i][1]*sin(r)-rX*cos(r)+rY*sin(r)+rX;
        nP[i][1] = P[i][0]*sin(r)+P[i][1]*cos(r)-rX*sin(r)-rY*cos(r)+rY;
    }
    glColor3f(0,1,0);
    glBegin(GL_LINE_LOOP);
        glVertex2fv(nP[0]);
        glVertex2fv(nP[1]);
        glVertex2fv(nP[2]);
    glEnd();
    glFlush();
}

int main(int argC,char *argV[])
{
    printf("\nEnter the Pivot Point for Rotation : ");
    scanf("%f%f",&rX,&rY);
    printf("\nEnter the Degree of Rotation : ");
    scanf("%f",&d);
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Lab-2");
    glutDisplayFunc(Draw);
    glutIdleFunc(Spin);
    glutMainLoop();
    return 0;
}
```

Output



```
Enter the Degree of Rotation : 40
Enter the Pivot Point for Rotation : 0.2 0.6
```

LAB PROGRAM – 3

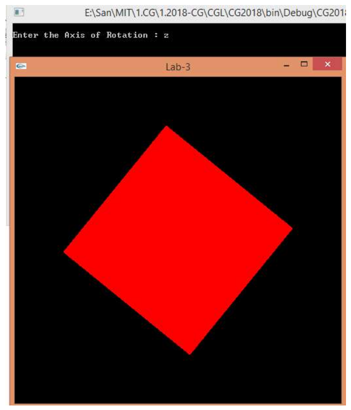
| Problem Statement |

Draw a color cube and spin it using OpenGL transformation matrices.

| Source Code |

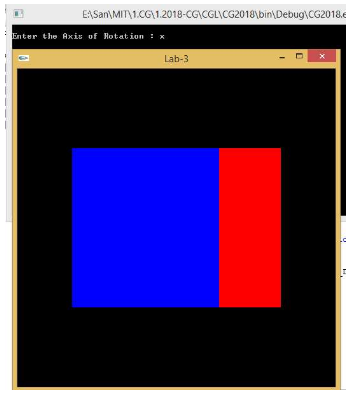
```
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
GLfloat d=0;
char a;
void Spin()
{
    d = d + 1;
    if(d > 360)
        d = 0;
    glutPostRedisplay();
}
void Face(GLfloat A[3],GLfloat B[3],GLfloat C[3],GLfloat D[3])
{
    glBegin(GL_QUADS);
        glVertex3fv(A);
        glVertex3fv(B);
        glVertex3fv(C);
        glVertex3fv(D);
    glEnd();
}
void   Cube(GLfloat   P1[3],GLfloat   P2[3],GLfloat   P3[3],GLfloat   P4[3],GLfloat
P5[3],GLfloat P6[3],GLfloat P7[3],GLfloat P8[3])
{
    glColor3f(1,0,0);
    Face(P1,P2,P3,P4);
    glColor3f(0,1,0);
    Face(P5,P6,P7,P8);
    glColor3f(0,0,1);
    Face(P1,P5,P8,P4);
    glColor3f(1,1,0);
    Face(P2,P6,P7,P3);
    glColor3f(1,0,1);
    Face(P1,P2,P6,P5);
    glColor3f(0,1,1);
    Face(P4,P3,P7,P8);
}
void Draw()
{
    GLfloat V[8][3] =   {
                            {-0.5,-0.5,-0.5},
                            { 0.5,-0.5,-0.5},
                            { 0.5, 0.5,-0.5},
                            {-0.5, 0.5,-0.5},
                            {-0.5,-0.5, 0.5},
                            { 0.5,-0.5, 0.5},
                            { 0.5, 0.5, 0.5},
                            {-0.5, 0.5, 0.5},
                        };
    GLfloat nV[8][3],r;
    int i;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    r = d*3.14/180;
    if(a=='z')
    {
        for(i=0;i<8;i++)
        {
            nV[i][0] = V[i][0]*cos(r)-V[i][1]*sin(r);
            nV[i][1] = V[i][0]*sin(r)+V[i][1]*cos(r);
            nV[i][2] = V[i][2];
        }
    }
    if(a=='x')
    {
    for(i=0;i<8;i++)
    {
        nV[i][0] = V[i][0];
        nV[i][1] = V[i][1]*cos(r)-V[i][2]*sin(r);
        nV[i][2] = V[i][1]*sin(r)+V[i][2]*cos(r);
    }
    }

    if(a=='y')
    {
    for(i=0;i<8;i++)
    {
        nV[i][0] = V[i][0]*cos(r) + V[i][2]*sin(r);
        nV[i][1] = V[i][1];
        nV[i][2] = -V[i][0]*sin(r) + V[i][2]*cos(r);
    }
    }

    Cube(nV[0],nV[1],nV[2],nV[3],nV[4],nV[5],nV[6],nV[7]);
    glutSwapBuffers();
}

int main(int argC,char *argV[])
{
    printf("\nEnter the Axis of Rotation : ");
    scanf("%c",&a);
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Lab-3");
    glutDisplayFunc(Draw);
    glutIdleFunc(Spin);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
    return 0;
}
```

Output



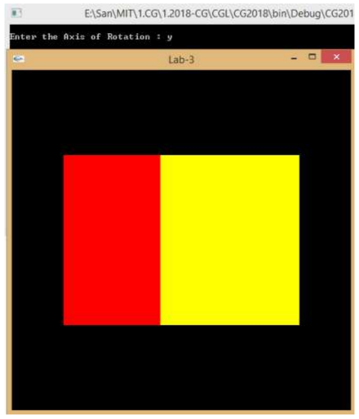
```
Enter the Axis of Rotation : z
```

Rotation with respect to z-axis



```
Enter the Axis of Rotation : x
```

Rotation with respect to x-axis



```
Enter the Axis of Rotation : y
```

Rotation with respect to y-axis

LAB PROGRAM – 4

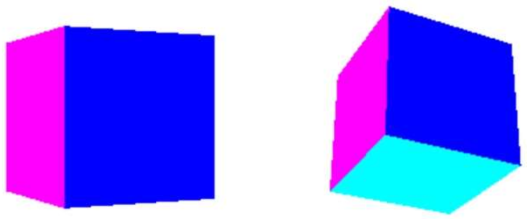
| Problem Statement |

Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.

| Source Code |

```
#include <GL/glut.h>

GLfloat CamX=0,CamY=0,CamZ=4;

void Face(GLfloat A[],GLfloat B[],GLfloat C[],GLfloat D[])
{
    glBegin(GL_POLYGON);
        glVertex3fv(A);
        glVertex3fv(B);
        glVertex3fv(C);
        glVertex3fv(D);
    glEnd();
}

void   Cube(GLfloat   P1[],GLfloat   P2[],GLfloat   P3[],GLfloat   P4[],GLfloat
P5[],GLfloat P6[],GLfloat P7[],GLfloat P8[])
{
    glColor3f(1,0,0);
    Face(P1,P2,P3,P4);

    glColor3f(0,1,0);
    Face(P2,P6,P7,P3);

    glColor3f(1,1,0);
    Face(P4,P3,P7,P8);

    glColor3f(1,0,1);
    Face(P1,P5,P8,P4);

    glColor3f(0,1,1);
    Face(P1,P2,P6,P5);

    glColor3f(0,0,1);
    Face(P5,P6,P7,P8);
}

void Display(void)
{
    GLfloat V[8][3] = {
                            {-0.5,-0.5,-0.5},
                            { 0.5,-0.5,-0.5},
                            { 0.5, 0.5,-0.5},
                            {-0.5, 0.5,-0.5},
                            {-0.5,-0.5, 0.5},
                            { 0.5,-0.5, 0.5},
                              { 0.5, 0.5, 0.5},
                              {-0.5, 0.5, 0.5}
                          };
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(CamX,CamY,CamZ,0,0,0,0,1,0);
    Cube(V[0],V[1],V[2],V[3],V[4],V[5],V[6],V[7]);
    glutSwapBuffers();
}

void Key(unsigned char Ch, int x, int y)
{
   switch(Ch)
   {
        case 'x' :   CamX = CamX - 0.5;     break;
        case 'X' :   CamX = CamX + 0.5;     break;
        case 'y' :   CamY = CamY - 0.5;     break;
        case 'Y' :   CamY = CamY + 0.5;     break;
        case 'z' :   CamZ = CamZ - 0.5;     break;
        case 'Z' :   CamZ = CamZ + 0.5;
    }
   glutPostRedisplay();
}
int  main(int c, char **v)
{
    glutInit(&c,v);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Lab-8: Perspective Viewing with Camera");
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1,1,-1,1,2,20);
    glMatrixMode(GL_MODELVIEW);
    glutDisplayFunc(Display);
    glutKeyboardFunc(Key);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
    return 0;
}
```

Output



Use key Z, Y or X for the movement of the camera

LAB PROGRAM – 5

| Problem Statement |

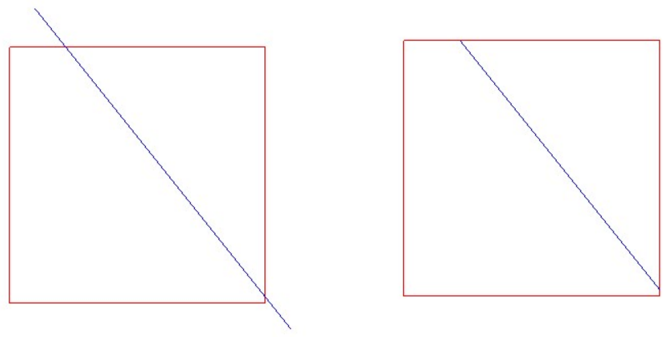
Clip a lines using Cohen-Sutherland algorithm.

| Source Code |

```
#include <GL/glut.h>

GLfloat Xmin=-0.5,Xmax=0.5,Ymin=-0.5,Ymax=0.5;
GLfloat X1=-0.4,Y1=0.65,X2=0.6,Y2=-0.6;

int Left=1,Right=2,Bottom=4,Top=8;
int C1,C2;
int Flag = 0;

int Get_Code(GLfloat x,GLfloat y)
{
    int C=0;

    if(x < Xmin)
        C = C | Left;

    if(x > Xmax)
        C = C | Right;

    if(y < Ymin)
        C = C | Bottom;

    if(y > Ymax)
        C = C | Top;

    return C;
}

void Clip()
{
    GLfloat X,Y;

    int C;

    if(C1)
        C = C1;
    else
        C = C2;

    if(C & Left)
    {
        X = Xmin;
        Y = Y1 + (Y2 - Y1) * ((Xmin - X1) / (X2 - X1));
    }

    if(C & Right)
    {
        X = Xmax;
        Y = Y1 + (Y2 - Y1) * ((Xmax - X1) / (X2 - X1));
    }

    if(C & Bottom)
    {
        Y = Ymin;
        X = X1 + (X2 - X1) * ((Ymin - Y1) / (Y2 - Y1));
    }

    if(C & Top)
    {
        Y = Ymax;
        X = X1 + (X2 - X1) * ((Ymax - Y1) / (Y2 - Y1));
    }

    if(C == C1)
    {
        X1 = X;
        Y1 = Y;
    }
    else
    {
        X2 = X;
        Y2 = Y;
    }
}

void Display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1,0,0);
    glRectf(Xmin,Ymin,Xmax,Ymax);

    glColor3f(0,0,1);
    glBegin(GL_LINES);
        glVertex2f(X1,Y1);
        glVertex2f(X2,Y2);
    glEnd();

    while(1 && Flag)
    {
        C1=Get_Code(X1,Y1);
        C2=Get_Code(X2,Y2);

        if((C1|C2) == 0)
            break;
        else if((C1&C2) != 0)
            break;
        else
            Clip();
    }
    glFlush();
}

void Key(unsigned char ch, int x, int y)
```

```
{
    Flag = 1;
    glutPostRedisplay();
}

int main(int argC, char *argV[])
{
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Lab5: Cohen-Sutherland Line Clipping");

    glClearColor(1,1,1,1);
    glutDisplayFunc(Display);
    glutKeyboardFunc(Key);

    glutMainLoop();
    return 0;
}
```

Output



Press any key from the key-board to Clip the Line.

LAB PROGRAM – 6

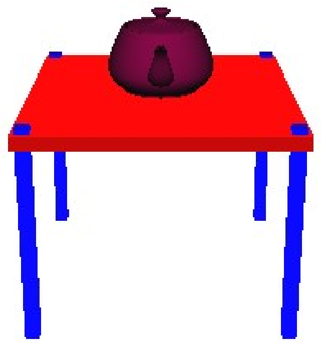
Problem Statement

To draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of the light source along with the properties of the surfaces of the solid object used in the scene.

Source Code

```
#include<GL/glut.h>

#include<windows.h>
#include <GL/glu.h>
#include <GL/glut.h>

void Display()
{
    GLfloat Pos[] = {-1,1,0,1};
    GLfloat S[] = {0.5,0,0.25,1};

    GLfloat D1[] = {1,0,0,1};
    GLfloat D2[] = {0,0,1,1};
    GLfloat D3[] = {0.5,0,0.25,1};

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLightfv(GL_LIGHT0,GL_POSITION,Pos);
    glLightfv(GL_LIGHT0,GL_SPECULAR,S);

    glLoadIdentity();

    gluLookAt(0,0.5,3,0,0,0,0,1,0);

        glPushAttrib(GL_ALL_ATTRIB_BITS);
            glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,D1);
            glPushMatrix();
                glScalef(1,0.05,1);
                glutSolidCube(1);           //Table Top
            glPopMatrix();
        glPopAttrib();

        glPushAttrib(GL_ALL_ATTRIB_BITS);
            glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,D2);
            glPushMatrix();
                glTranslatef(-0.5,-0.4,-0.5);
                glScalef(0.05, 0.8 ,0.05);
                glutSolidCube(1);                               //Leg-1
            glPopMatrix();

            glPushMatrix();
                glTranslatef( 0.5,-0.4,-0.5);
                glScalef(0.05, 0.8 ,0.05);
                glutSolidCube(1);                               //Leg-2
            glPopMatrix();

            glPushMatrix();
                glTranslatef( 0.5,-0.4, 0.5);
                glScalef(0.05, 0.8 ,0.05);
                glutSolidCube(1);                               //Leg-3
            glPopMatrix();

            glPushMatrix();
                glTranslatef(-0.5,-0.4, 0.5);
                glScalef(0.05, 0.8 ,0.05);
                glutSolidCube(1);                               //Leg-4
            glPopMatrix();
        glPopAttrib();

        glPushAttrib(GL_ALL_ATTRIB_BITS);
            glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,D3);
            glPushMatrix();
                glTranslatef(0,0.18,0);
                glutSolidTeapot(0.2);                               //TeaPot
            glPopMatrix();
        glPopAttrib();

    glutSwapBuffers();
}

int main(int argC, char *argV[])
{
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Lab-7: Scene Consisting of Tea Pot on Table");

    glutDisplayFunc(Display);

    glClearColor(1,1,1,1);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1,1,-1,1,2,20);
    glMatrixMode(GL_MODELVIEW);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glutMainLoop();
    return 0;
}
```

Output



LAB PROGRAM – 7

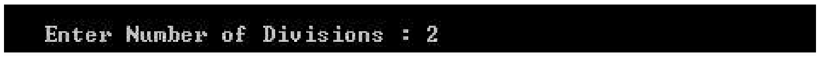
| Problem Statement |

Design, develop and implement recursively subdivide a tetrahedron to form 3D sierpinski gasket. The number of recursive steps is to be specified by the user.

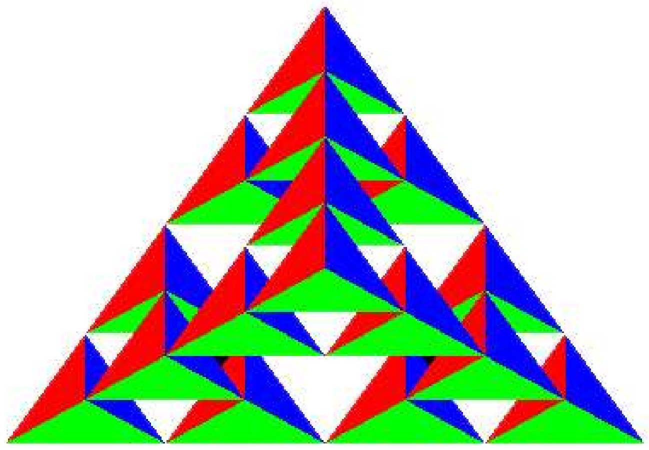
| Source Code |

```
#include<GL/glut.h>
#include<stdio.h>
int n;
void Triangle(GLfloat A[],GLfloat B[],GLfloat C[])
{
    glBegin(GL_TRIANGLES);
        glVertex3fv(A);
        glVertex3fv(B);
        glVertex3fv(C);
    glEnd();
}
void Tetra(GLfloat P1[],GLfloat P2[],GLfloat P3[],GLfloat P4[])
{
    glColor3f(1,1,1);
    Triangle(P1,P2,P3);

    glColor3f(0,1,0);
    Triangle(P1,P2,P4);

    glColor3f(0,0,1);
    Triangle(P2,P3,P4);

    glColor3f(1,0,0);
    Triangle(P1,P4,P3);
}


void Div(GLfloat P1[],GLfloat P2[],GLfloat P3[],GLfloat P4[],int n)
{
    GLfloat P12[3], P23[3], P31[3], P14[3], P24[3], P34[3];
    if(n>0)
    {
        P12[0] = (P1[0]+P2[0])/2;
        P12[1] = (P1[1]+P2[1])/2;
        P12[2] = (P1[2]+P2[2])/2;

        P23[0] = (P2[0]+P3[0])/2;
        P23[1] = (P2[1]+P3[1])/2;
        P23[2] = (P2[2]+P3[2])/2;

        P31[0] = (P3[0]+P1[0])/2;
        P31[1] = (P3[1]+P1[1])/2;
        P31[2] = (P3[2]+P1[2])/2;

        P14[0] = (P1[0]+P4[0])/2;
        P14[1] = (P1[1]+P4[1])/2;
        P14[2] = (P1[2]+P4[2])/2;

        P24[0] = (P2[0]+P4[0])/2;
        P24[1] = (P2[1]+P4[1])/2;
        P24[2] = (P2[2]+P4[2])/2;

        P34[0] = (P3[0]+P4[0])/2;
        P34[1] = (P3[1]+P4[1])/2;
        P34[2] = (P3[2]+P4[2])/2;


        Div(P1 ,P12,P31,P14,n-1);
        Div(P12,P2 ,P23,P24,n-1);
        Div(P31,P23,P3 ,P34,n-1);
        Div(P14,P24,P34,P4 ,n-1);
    }
    else
        Tetra(P1,P2,P3,P4);
}
void Display()
{
    GLfloat V[4][3] =   {
                           {-0.75 ,-0.5,-0.5},
                           { 0.75 ,-0.5,-0.5},
                           { 0    , 0.5,-0.5},
                           { 0    ,-0.1, 0.5}
                        };
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    Div(V[0],V[1],V[2],V[3],n);
    glutSwapBuffers();
}
int main(int c,char *v[])
{
    printf("\n\tEnter Number of Divisions : ");
    scanf("%d",&n);
    glutInit(&c,v);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Lab-1: 3D Sierpinski Gasket");
    glutDisplayFunc(Display);
    glutMainLoop();
    return 0;
}
```

Output



```
Enter Number of Divisions : 2
```



LAB PROGRAM – 8

Problem Statement

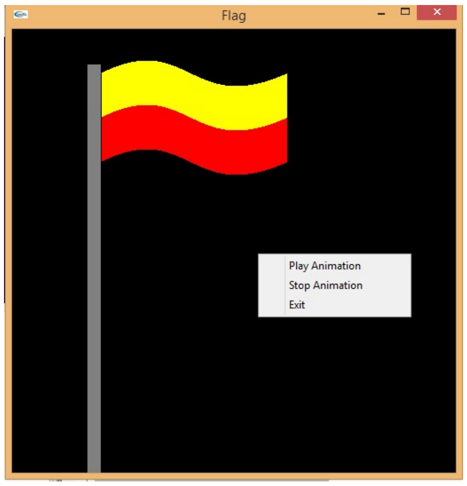
Develop a menu driven program to animate a flag using Bezier Curve algorithm.

Source Code

```
#include<windows.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include<math.h>

int AnFlag = 0;
int yFlag = 1,xFlag = 1;
float yC=-50,xC=-10;
float x[4],Y1[4],y2[4],y3[4];

void Menu(int Id)
{
    switch(Id)
    {
        case 1 : AnFlag = 1;    break;
        case 2 : AnFlag = 0;    break;
        case 3 : exit(0);
    }
}

void Idle()
{
    if(AnFlag == 1)
    {
        if(yC<50 && yFlag == 1)
            yC = yC + 0.2;

        if(yC>=50 && yFlag == 1)
            yFlag = 0;

        if(yC>-50 && yFlag == 0)
            yC = yC - 0.2;

        if(yC<=-50 && yFlag == 0)
            yFlag = 1;


        if(xC<20 && xFlag == 1)
            xC = xC + 0.2;

        if(xC>=20 && xFlag == 1)
            xFlag = 0;

        if(xC>-20 && xFlag == 0)
            xC = xC - 0.2;
        if(xC<=-20 && xFlag == 0)
            xFlag = 1;
    }
    glutPostRedisplay();
}


void Draw()
{
    int i;
    double t,xt[200],y1t[200],y2t[200],y3t[200],y4t[200];
    glClear(GL_COLOR_BUFFER_BIT);

    x[0] = 300-xC;    x[1] = 200;    x[2] = 200;    x[3] = 100;

    Y1[0] = 450;   Y1[1] = 450+yC;  Y1[2] = 450-yC;  Y1[3] = 450;
    y2[0] = 400;   y2[1] = 400+yC;  y2[2] = 400-yC;  y2[3] = 400;
    y3[0] = 350;   y3[1] = 350+yC;  y3[2] = 350-yC;  y3[3] = 350;

    i=0;
    for (t = 0.0; t < 1.0; t += 0.005)
    {
        xt[i]         =         pow(1-t,3)*x[0]+3*t*pow(1-t,2)*x[1]+3*pow(t,2)*(1-
t)*x[2]+pow(t,3)*x[3];
        y1t[i]       =       pow(1-t,3)*Y1[0]+3*t*pow(1-t,2)*Y1[1]+3*pow(t,2)*(1-
t)*Y1[2]+pow(t,3)*Y1[3];
        y2t[i]       =       pow(1-t,3)*y2[0]+3*t*pow(1-t,2)*y2[1]+3*pow(t,2)*(1-
t)*y2[2]+pow(t,3)*y2[3];
        y3t[i]       =       pow(1-t,3)*y3[0]+3*t*pow(1-t,2)*y3[1]+3*pow(t,2)*(1-
t)*y3[2]+pow(t,3)*y3[3];
        i++;
    }

    glColor3f(1,1,0);
    glBegin(GL_QUAD_STRIP);
    for (i=0;i<200;i++)
    {
        glVertex2d(xt[i],y1t[i]);
        glVertex2d(xt[i],y2t[i]);
    }
    glEnd();


    glColor3f(1,0,0);
    glBegin(GL_QUAD_STRIP);
    for (i=0;i<200;i++)
    {
        glVertex2d(xt[i],y2t[i]);
        glVertex2d(xt[i],y3t[i]);
    }
    glEnd();

    glColor3f(0.5,0.5,0.5);
    glRectf(85,460,100,0);

    glFlush();
```

```
}

void MyInit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glMatrixMode(GL_MODELVIEW);

    glutCreateMenu(Menu);
    glutAddMenuEntry("Play Animation",1);
    glutAddMenuEntry("Stop Animation",2);
    glutAddMenuEntry("Exit",3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}

int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Flag");
    MyInit();
    glutDisplayFunc(Draw);
    glutIdleFunc(Idle);
    glutMainLoop();
    return 0;
}
```

Output



Once you choose the 'Play Animation' Option from the menu, you will find a waving flag. The given figure is a stillimage **of the same.**

LAB PROGRAM – 9

Problem Statement

Develop a menu driven program to fill the polygon using scan line algorithm.

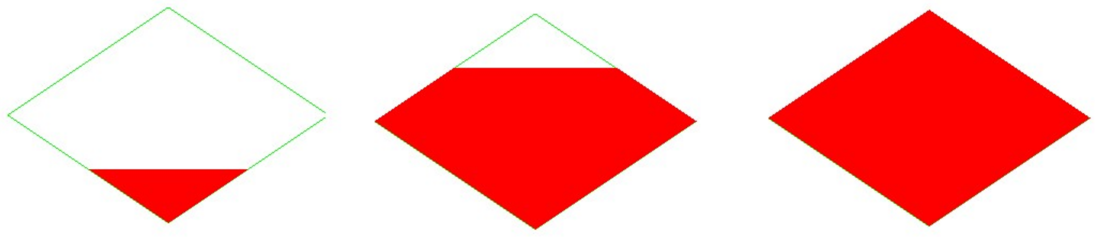
Source Code

```
#include <GL/glut.h>
#include<windows.h>
#include <GL/glu.h>
#include <GL/glut.h>

int LE[500],RE[500];
int Fill_Flag = 0,Line_Flag = 0;

void Intersection(GLfloat x1,GLfloat y1,GLfloat x2,GLfloat y2)
{
    GLfloat temp,M,x;
    int i;
    if(y1>y2)
    {
        temp = y1;
        y1 = y2;
        y2 = temp;

        temp = x1;
        x1 = x2;
        x2 = temp;
    }

    if(y2 - y1 == 0)
        M = x2 - x1;
    else
        M = ( x2 - x1 ) / (y2 - y1);

    x = x1;
    for(i=y1;i<=y2;i++)
    {
        if(x<LE[i])
        {
            LE[i] = x;
        }

        if(x>RE[i])
        {
            RE[i] = x;
        }
        x = x + M;
    }
}

void Display()
```

```
{
    GLfloat x1=250,y1=150;
    GLfloat x2=400,y2=250;
    GLfloat x3=250,y3=350;
    GLfloat x4=100,y4=250;
    int i;
    GLint x,y;

    glClear(GL_COLOR_BUFFER_BIT);

    for(i=0;i<500;i++)
    {
        LE[i] = 500;
        RE[i] = 0;
    }

    if(Line_Flag == 1)
    {
        glColor3f(0,1,0);
        glBegin(GL_LINE_LOOP);
            glVertex2f(x1,y1);
            glVertex2f(x2,y2);
            glVertex2f(x3,y3);
            glVertex2f(x4,y4);
        glEnd();
    }

    glColor3f(1,0,0);
    Intersection(x1,y1,x2,y2);
    Intersection(x2,y2,x3,y3);
    Intersection(x3,y3,x4,y4);
    Intersection(x4,y4,x1,y1);

    if(Fill_Flag == 1)
    {
      for(y=0;y<500;y++)
        {
            if(LE[y]<=RE[y])
            {
               for(x=LE[y];x<=RE[y];x++)
                {
                    glBegin(GL_POINTS);
                        glVertex2f(x,y);
                    glEnd();
                    glFlush();
                }
            }
        }
    }

}

void Line_Menu(int Id)
{
    if(Id == 1)
        Line_Flag = 1;
    if(Id == 2)
```

```
        Line_Flag = 2;
    glutPostRedisplay();
}

void Main_Menu(int Id)
{
    if(Id == 1)
        Fill_Flag = 1;
    if(Id == 2)
        exit(0);
    glutPostRedisplay();
}

int main(int argC, char *argV[])
{
    int Id;
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(50,50);
    glutCreateWindow("Lab9 : Scan-Line Fill Algorithm");

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glMatrixMode(GL_MODELVIEW);

    Id = glutCreateMenu(Line_Menu);
    glutAddMenuEntry("Yes",1);
    glutAddMenuEntry("No",2);
    glutCreateMenu(Main_Menu);
    glutAddSubMenu("Out Line",Id);
    glutAddMenuEntry("Start Fill",1);
    glutAddMenuEntry("Exit",2);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutDisplayFunc(Display);

    glutMainLoop();
    return 0;
}
```

Output



Viva Questions

• What is scan conversion?
A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is called scan conversion

• Write the properties of video display devices?
Properties of video display devices are persistence, resolution, and aspect ratio

• What is rasterization?
The process of determining the appropriate pixels for representing picture or graphics object is known as rasterization

• Define Computer graphics.
Computer graphics remains one of the most existing and rapidly growing computer fields. Computer graphics maybe defined as a pictorial representation or graphical representation of objects in a computer.

• Name any four input devices
Four input devices are keyboard, mouse, image scanners, and trackball.

• Write the two techniques for producing color displays with a CRT?
Beam penetration method, shadow mask method

• What is vertical retrace of the electron beam?
In raster scan display, at the end of one frame, the electron beam returns to the left top corner of the screen to start the next frame

• Short notes on video controller?
Video controller is used to control the operation of the display device. A fixed area of the system is reserved for the frame buffer, and the video controller is given direct access to the frame buffer memory

• What is bitmap?
Some system has only one bit per pixel; the frame buffer is often referred to as bitmap.

• Differentiate plasma panel display and thin film electroluminescent display?
In plasma panel display, the region between two glass plates is filled with neon gas. In thin film electroluminescent display, the region between two glasses plates are filled with phosphor, such as zinc sulphide doped with manganese.

• What is resolution?
The maximum number of points that can be displayed without overlap on a CRT is referred to as the resolution.

• What is horizontal retrace of the electron beam?
In raster scan display, the electron beam return to the left of the screen after refreshing each scan line, is called horizontal retrace of the electron beam.

• What is filament?
In the CRT, heat is applied to the cathode by directing a current through a coil of wire, is called filament

• What is pix map?
Some system has multiple bits per pixel, the frame buffer is often referred to as pix map.

• Write the types of clipping?
Point clipping, line clipping, area clipping, text clipping and curve clipping.

• What is meant by scan code?
When a key is pressed on the keyboard, the keyboard controller places a code carry to the key pressed into a part of the memory called as the keyboard buffer. This code is called as the scan code.

• List out the merits and demerits of Penetration techniques?
The merits and demerits of the Penetration techniques areas follows. It is an inexpensive technique. It has only four colors. The quality of the picture is not good when it is compared to other techniques. It can display color scans in monitors. Poor limitation etc.

• List out the merits and demerits of DVST?
The merits and demerits of direct view storage tubes[DVST] are as follows. It has a flat screen. Refreshing of screen is not required. Selective or part erasing of screen is not possible. It has poor contrast Performance is inferior to the refresh CRT.

• What do you mean by emissive and non-emissive displays?
The emissive display converts electrical energy into light energy. The plasma panels, thin film electro-luminescent displays are the examples. The Non-emissive are optical effects to convert the sun light or light from any other source to graphic form. Liquid crystal display is an example

• List out the merits and demerits of Plasma panel display?
Merits. Refreshing is not required. Produce a very steady image free of Flicker. Less bulky than a CRT. Demerits. Poor resolution of up to 60 d.p.i. It requires complex addressing and wiring. It is costlier than CRT.

• What is persistence?
The time it takes the emitted light from the screen to decay one tenth of its original intensity is called as persistence.

• What is Aspect ratio?
The ratio of vertical points to the horizontal points necessary to produce length of lines in both directions of the screen is called the Aspect ratio. Usually the aspect ratio is ¾.

• What is the difference between impact and non-impact printers?
Impact printer press formed character faces against an inked ribbon on to the paper. A line printer and dot-matrix printer are examples. Non-impact printer and plotters use Laser techniques, inkjet sprays, Xerographic process, electrostatic methods and electro thermal methods to get images onto the papers. Examples are: Inkjet/Laser printers.

- Define pixel?

Pixel is shortened forms of picture element. Each screen point is referred to as pixel or pel.

- What is frame buffer?

Picture definition is stored in a memory area called framebuffer or refresh buffer.

- Where the video controller is used?

A special purpose processor, which is used to control the operation of the display device, is known as video controller or display controller.

- What is run length encoding?

Run length encoding is a compression technique used to store the intensity values in the frame buffer, which stores each scan line as a set of integer pairs. One number each pair indicates an intensity value, and second number specifies the number of adjacent pixels on the scan line that are to have that intensity value.

- What is point in the computer graphics system?

The point is a most basic graphical element & is completely defined by a pair of user coordinates (x, y).

- Write short notes on lines?

A line is of infinite extent can be defined by an angle of slope q and one point on the line P=P(x,y). This can also be defined as y=mx+C where C is the Y intercept.

- Define Circle?

Circle is defined by its center xc, yc and its radius in user coordinate units. The equation of the circle is (x-xc) + (yyc)= r2.

- What are the various attributes of a line?

The line type, width and color are the attributes of the line. The line type include solid line, dashed lines, and dotted lines.

- What is antialiasing?

The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called antialiasing.

- What is Transformation?

Transformation is the process of introducing changes in the shape size and orientation of the object using scaling rotation reflection shearing & translation etc.

- What is translation?

Translation is the process of changing the position of an object in a straight-line path from one coordinate location to another. Every point (x , y) in the object must under go a displacement to (x|,y|). the transformation is:x| = x + tx ; y| = y+ty

- What is rotation?

A 2-D rotation is done by repositioning the coordinates along a circular path, in the x-y plane by making an angle with the axes. The transformation is given by: X| = r cos (q + f) and Y| = r sin (q + f).

- What is scaling?

A 2-D rotation is done by repositioning the coordinates along a circular path, in the x-y plane by making an angle with the axes. The transformation is given by: X| = r cos (q + f) and Y| = r sin (q + f).