Ref No:

SRI KRISHNA INSTITUTE OF TECHNOLOGY,BANGALORE

LABORATORY PLAN

Academic Year 2019-20

Program:	B E – Computer Science & Engineering
Semester :	6
Course Code:	17CSL68
Course Title:	Computer graphics Laboratory with mini project
Credit / L-T-P:	2 / 0-0-2
Total Contact Hours:	40
Course Plan Author:	Priyanka H V

Academic Evaluation and Monitoring Cell

No. 29, Chimny hills, Hesarghtta Road, Chikkabanavara
Bangalore – 560090, Karnataka, India
Phone/Fax: +91-08023721315/ 23721477
www.skit.org.in

INSTRUCTIONS TO TEACHERS

- Classroom / Lab activity shall be started after taking attendance.
- Attendance shall only be signed in the classroom by students.
- Three hours attendance should be given to each Lab.
- Use only Blue or Black Pen to fill the attendance.
- Attendance shall be updated on-line & status discussed in DUGC.
- No attendance should be added to late comers.
- Modification of any attendance, over writings, etc is strictly prohibited.
- Updated register is to be brought to every academic review meeting as per the COE.

Table of Contents

A. LABORATORY INFORMATION....4
1. Laboratory Overview....4
2. Laboratory Content....4
3. Laboratory Material....5
4. Laboratory Prerequisites:....5
5. Content for Placement, Profession, HE and GATE....5
B. Laboratory Instructions....6
1. General Instructions....6
2. Laboratory Specific Instructions....6
C. OBE PARAMETERS....6
1. Laboratory Outcomes....6
2. Laboratory Applications....7
3. Mapping And Justification....8
4. Articulation Matrix....8
5. Curricular Gap and Experiments....9
6. Experiments Beyond Syllabus....9
D. COURSE ASSESSMENT....10
1. Laboratory Coverage....10
2. Continuous Internal Assessment (CIA)....10
E. EXPERIMENTS....11
Experiment 01 : Structure of C program....11
Experiment 02 : Keywords and identifiers....12
Experiment 03 : 13
Experiment 04 : 13
F. Content to Experiment Outcomes....14
1. TLPA Parameters....14
2. Concepts and Outcomes:....15

Note : Remove “Table of Content” before including in CP Book
Each Laboratory Plan shall be printed and made into a book with cover page
Blooms Level in all sections match with A.2, only if you plan to teach / learn at higher levels

A. LABORATORY INFORMATION

1. Laboratory Overview

Degree:	B E	*Program:*	CS
Year / Semester :	3/ 6	*Academic Year:*	2019-20
Course Title:	Computer graphics Laboratory with mini project	*Course Code:*	17CSL68
Credit / L-T-P:	3/ 3-0-0	*SEE Duration:*	180 Minutes
Total Contact Hours:	40 Hrs	*SEE Marks:*	60 Marks
CIA Marks:	40	*Assignment*	-
Lab. Plan Author:	Priyanka H V	*Sign*	Dt :
Checked By:		*Sign*	Dt :

2. Laboratory Content

Unit	Title of the Experiments	Lab Hours	Concept	Blooms Level
1.	Implement Brenham's line drawing algorithm for all types of slope	4	Open GL API	L3 Apply
2	Create and rotate a triangle about the origin and a fixed point.	4	Graphics Model	L4 Analyze
3	Draw a colour cube and spin it using OpenGL transformation matrices.	3	Graphics Models	L4 Analyze
4	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing	4	Viewport Transformation	L4 Analyze
5	Clip a lines using Cohen-Sutherland algorithm	3	Clipping	L3 Apply
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.	3	Viewport Transformation	L4 Analyze
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.	3	Input Interactive Graphics	L4 Analyze
8	Develop a menu driven program to animate a flag using Bezier Curve algorithm	3	Input Interactive Graphics	L4 Analyze
9	Develop a menu driven program to fill the polygon using scan line algorithm	3	Open GL API	L3 Apply
10	Mini-Project with applications using Open GL API	10	Open GL API	L3 Apply

3. Laboratory Material

Books & other material as recommended by university (A, B) and additional resources used by Laboratory teacher (C).

Expt.	Details	Expt. in book	Availability
A	**Text books (Title, Authors, Edition, Publisher, Year.)**	**-**	**-**
1, 2, 3, 4, 5	Computer Graphics with OpenGL, Donald Hearn & Pauline Baker, Version 3 rd / 4 th Edition, Pearson Education,2011	1,2,3, 4,5,6	In Lib / In Dept
1	Interactive Computer Graphics- A Top Down approach with OpenGL,Edward Angel, 5 th edition. Pearson Education, 2008	7,8,9	In Lib/ In dept
B	**Reference books (Title, Authors, Edition, Publisher, Year.)**	**-**	**-**
1, 2,3,4,5	james D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL pearson education	-	In Lib/In dept

1, 2	Xiang, Plastock : Computer Graphics,sham's outline series, 2nd edition, TMG.	-	In Lib/In dept
C	**Concept Videos or Simulation for Understanding**	**-**	**-**
c1	https://youtu.be/N_QgVSZXk9w		
c2	https://youtu.be/xErzZgnrLFs		
D	**Software Tools for Design**	**-**	**-**
	OpenGL tool		
	Visual c++ 6.0		
E	**Recent Developments for Research**	**-**	**-**
		?	In lib
F	**Others (Web, Video, Simulation, Notes etc.)**	**-**	**-**
1			
?			

4. Laboratory Prerequisites:

Refer to GL01. If prerequisites are not taught earlier, GAP in curriculum needs to be addressed. Include in Remarks and implement in B.5.
Students must have learnt the following Courses / Topics with described Content . . .

Expt.	Lab. Code	Lab. Name	Topic / Description	Sem	Remarks	Blooms Level
1	17cpl26	Computer Programming	1. Knowledge on C & C ++	2	Videos of C++	L3
2						

5. Content for Placement, Profession, HE and GATE

The content is not included in this course, but required to meet industry & profession requirements and help students for Placement, GATE, Higher Education, Entrepreneurship, etc. Identifying Area / Content requires experts consultation in the area.
Topics included are like, a. Advanced Topics, b. Recent Developments, c. Certificate Courses, d. Course Projects, e. New Software Tools, f. GATE Topics, g. NPTEL Videos, h. Swayam videos etc.

Expt.	Topic / Description	Area	Remarks	Blooms Level
1				
3				
3				
5				
-				

B. Laboratory Instructions

1. General Instructions

SNo	Instructions	Remarks
1	Observation book and Lab record are compulsory.	
2	Students should report to the concerned lab as per the time table.	
3	After completion of the program, certification of the concerned staff in-charge in the observation book is necessary.	
4	Student should bring a notebook of 100 pages and should enter the readings /observations into the notebook while performing the experiment.	
5	The record of observations along with the detailed experimental procedure of the experiment in the Immediate last session should be submitted and certified staff member in-charge.	
6	Should attempt all problems / assignments given in the list session wise.	

7	It is responsibility to create a separate directory to store all the programs, so that nobody else can read or copy.	
8	When the experiment is completed, student should save the experiment with relevant filenames and exit from the Turbo C IDE compiler.	
9	Any damage of the equipment of the computer system will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year	
10	Completed lab assignments should be submitted in the form of a Lab Record in which you have to write the algorithm, Flowchart, program code along with comments and output for various inputs given	

2. Laboratory Specific Instructions

SNo	Specific Instructions	Remarks
1	Start computer	
2	Open visual basic c++ 6.0	
3	Go to file click new	
4	Give the file name and save with .cpp file extension	
5	Completion of program complie the code	
6	Rebuild the code	
7	Run	

C. OBE PARAMETERS

1. Laboratory Outcomes

Expt.	Lab Code #	COs / Experiment Outcome	Teach. Hours	Concept	Instr Method	Assessment Method	Blooms' Level
-	-	**At the end of the experiment, the student should be able to . . .**	-	-	-	-	-
1	17csl68	Explain effectively and creatively solve a wide range of graphic design application using Open GL	7	Open GL API	Case studies	Test	L3 Apply
2	17csl68	Apply the concepts of Computer Graphics for creating Applications.	7	Graphics Model	Case Studies	Seminar	L4 Analyze
3	17csl68	Demonstrate interactive graphics applications in 2D and 3D	7	Viewport Transformation	Case Studies	Assignment	L4 Analyze
4	17csl68	Implement graphics primitives and demonstrate geometrical transformations.	3	Clipping	procedure	Small group discussions	L3 Apply
5	17csl68	Implement algorithms for different geometric shapes line, circle, ellipse.	16	Input Interactive Graphics	Graphic Organizers	Seminar	L4 Analyze
-		**Total**	**40**	**-**	**-**	**-**	**-**

2. Laboratory Applications

Expt.	Application Area	CO	Level
1	Multiprocessor computers	CO1	L3
2	Text editors,web browsers	CO1	L4
3	Image processing	CO2	L4
4	Optimisation problem	CO3	L3
5	Huffman trees	CO4	L4

6	Mind games, puzzles.	CO5	L4

Note: Write 1 or 2 applications per CO.

4. Articulation Matrix

CO – PO Mapping with mapping level for each CO-PO pair, with course average attainment.

-	-	Experiment Outcomes	Program Outcomes														-
Expt.	CO.#	**At the end of the experiment student should be able to . . .**	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3	Level
1	co1	Explain effectively and creatively solve a wide range of graphic design application using Open GL	2	2	2	2	3	-	-	-	-	-	2	1	2	3	2	L3
2	co2	Apply the concepts of Computer Graphics for creating Applications.	2	2	2	2	3	-	-	-	-	-	2	1	2	3	2	L4
3	co3	Demonstrate interactive graphics applications in 2D and 3D	2	2	2	2	3	-	-	-	-	-	2	1	2	3	2	L4
4	co4	Implement graphics primitives and demonstrate geometrical transformations.	2	2	2	2	3	-	-	-	-	-	2	1	2	3	2	L3
5	co5	Implement algorithms for different geometric shapes line, circle, ellipse.	2	2	2	2	3	-	-	-	-	-	2	1	2	3	2	L4
-	**17CSL68**	**Average attainment (1, 2, or 3)**																**-**
-	*PO, PSO*	*1.Engineering Knowledge; 2.Problem Analysis; 3.Design / Development of Solutions; 4.Conduct Investigations of Complex Problems; 5.Modern Tool Usage; 6.The Engineer and Society; 7.Environment and Sustainability; 8.Ethics; 9.Individual and Teamwork; 10.Communication; 11.Project Management and Finance; 12.Life-long Learning; S1.Software Engineering; S2.Data Base Management; S3.Web Design*																

5. Curricular Gap and Experiments

Topics & contents not covered (from A.4), but essential for the course to address POs and PSOs.

Expt	Gap Topic	Actions Planned	Schedule Planned	Resources Person	PO Mapping
1					
2					
3					
4					
5					

Note: Write Gap topics from A.4 and add others also.

6. Experiments Beyond Syllabus

Topics & contents required (from A.5) not addressed, but help students for Placement, GATE, Higher Education, Entrepreneurship, etc.

Expt	Gap Topic	Actions Planned	Schedule Planned	Resources Person	PO Mapping
1					
2					
3					
4					
5					
6					
7					

8					
9					
10					
11					
12					
13					
14					
15					

D. COURSE ASSESSMENT

1. Laboratory Coverage

Assessment of learning outcomes for Internal and end semester evaluation. Distinct assignment for each student. 1 Assignment per chapter per student. 1 seminar per test per student.

Unit	Title	Teaching Hours	No. of question in Exam							CO	Levels
			CIA-1	CIA-2	CIA-3	Asg-1	Asg-2	Asg-3	SEE		
1.	Implement Brenham's line drawing algorithm for all types of slope	3	1	-	1	-	-	-	1	co1	L3
2	Create and rotate a triangle about the origin and a fixed point.	3	1		1					co2	L4
3	Draw a colour cube and spin it using OpenGL transformation matrices.	3	1	-	1	-	-	-	1	co2	L4
4	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing	3	1	-	1	-	-	-	1	co3	L3
5	Clip a lines using Cohen-Sutherland algorithm	3	1	-	1	-	-	-	1	co3	L3
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.	3	1	-	1	-	-	-	1	co4	L4
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.	3	1	-	1	-	-	-	1	co4	L4
8	Develop a menu driven program to animate a flag using Bezier Curve algorithm	3	1	-	1	-	-	-	1	co5	L4
9	Develop a menu driven program to fill the polygon using scan line algorithm	3	1	-	1					co1	L3
10	Mini-Project with applications using Open GL API	3	1	-	1	-	-	-	1	co1	L3
-	**Total**	**40**	**8**		**12**	**-**	**-**	**-**	**12**	**-**	**-**

2. Continuous Internal Assessment (CIA)

Assessment of learning outcomes for Internal exams. Blooms Level in last column shall match with A.2.

Evaluation	Weightage in Marks	CO	Levels

CIA Exam – 1	40	CO1,CO2,CO3,CO4	L3,L4
CIA Exam – 2	40	CO3,CO4,CO5	L3
CIA Exam – 3	40	CO1,CO2,CO3,CO4,CO5	L3,L4
Assignment - 1	-	-	-
Assignment - 2	-	-	-
Assignment - 3	-	-	-
	-	-	-

Seminar - 1	-	-	-
Seminar - 2	-	-	-
Seminar - 3	-	-	-
	-	-	-
Other Activities – define – Slip test	-	-	-
Final CIA Marks	**40**	**-**	**-**

SNo	Description	Marks
1	Observation and Weekly Laboratory Activities	05 Marks
2	Record Writing	15 Marks for each Expt
3	Internal Exam Assessment	20Marks
4	Internal Assessment	40 Marks
5	SEE	60Marks
-	**Total**	**100 Marks**

E. EXPERIMENTS

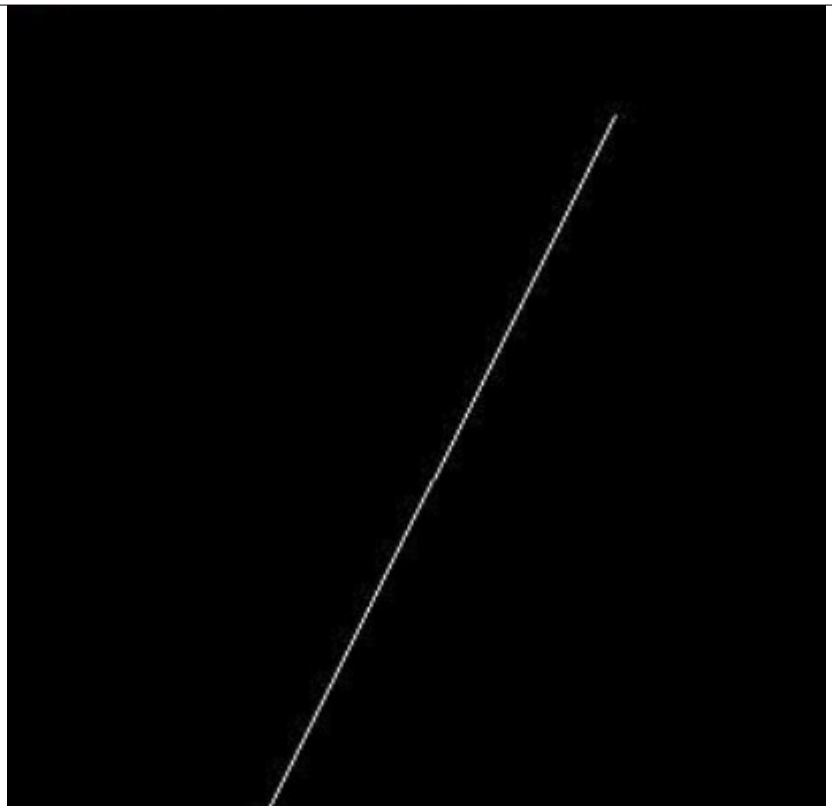
Experiment 01 : OpenGL API

-	**Experiment No.:**	1	**Marks**		**Date Planned**		**Date Conducted**	
1	Title	Design, develop, and implement the following programs in C/C++ using OpenGL API.						
2	Course Outcomes							
3	Aim							
4	Material / Equipment Required	Lab Manual						
5	Theory, Formula, Principle, Concept	1. Open the terminal. 2. Create your own directory using "mkdir 1ATXXCSXXX". THIS IS ONE TIME INSTRUCTION . 3. "cd 1ATXXCSXXX" 4. "gedit 1.cpp"						

6	Procedure, Program, Activity, Algorithm, Pseudo Code	#include <GL/glut.h>
		• #include <stdio.h>
		• int x1, y1, x2, y2;
		• void myInit() {
		• glClear(GL_COLOR_BUFFER_BIT);
		• glClearColor(0.0, 0.0, 0.0, 1.0);
		• glMatrixMode(GL_PROJECTION);
		• gluOrtho2D(0, 500, 0, 500);
		• }
		• void draw_pixel(int x, int y) {
		• glBegin(GL_POINTS);
		• glVertex2i(x, y);
		• glEnd();
		• }
		• void draw_line(int x1, int x2, int y1, int y2) { int dx, dy, i, e;
		• int incx, incy, inc1, inc2; int x,y;
		• dx = x2-x1; dy
		• = y2-y1;						

		• if (dx <0) dx = -dx; if (dy <
		• 0) dy = -dy; incx = 1;
		• if (x2 <x1) incx = -1;
		• incy = 1;
		• if (y2 <y1) incy = -1; x = x1; y
		• = y1;
		• if (dx >dy) { draw_pixel(x,
		• y); e = 2 * dy-dx; inc1
		• = 2*(dy-dx); inc2 =
		• 2*dy;
		• for (i=0; i<dx; i++) { if (e >=
		• 0) {
		• y += incy; e
		• += inc1;
		• }
		• else
		• e += inc2; x
		• += incx;
		• draw_pixel(x, y);
		• }
		• } else { draw_pixel(x, y); e
		• = 2*dx-dy; inc1 =
		• 2*(dx-dy); inc2 =
		• 2*dx;
		• for (i=0; i<dy; i++) { if (e >=
		• 0) {
		• x += incx; e
		• += inc1;
		• }
		• else
		• e += inc2; y
		• += incy;
		• draw_pixel(x, y);
		• }
		• }
		• }
		• void myDisplay() { draw_line(x1, x2,
		• y1, y2); glFlush();
		• }
		• int main(int argc, char **argv) {
		• printf("Enter (x1, y1, x2, y2)\n"); scanf("%d %d %d
		• %d", &x1, &y1, &x2, &y2);
		• glutInit(&argc, argv);
		• glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		• glutInitWindowSize(500, 500); glutInitWindowPosition(0,
		• 0); glutCreateWindow("Bresenham's Line Drawing");
		• myInit();
		• glutDisplayFunc(myDisplay);
		• glutMainLoop();
		• return 0;
		• }

7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	
9	Sample Calculations	•

10	Graphs, Outputs	
11	Results &Analysis	•
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 02 : Triangle about the origin and a _xed point

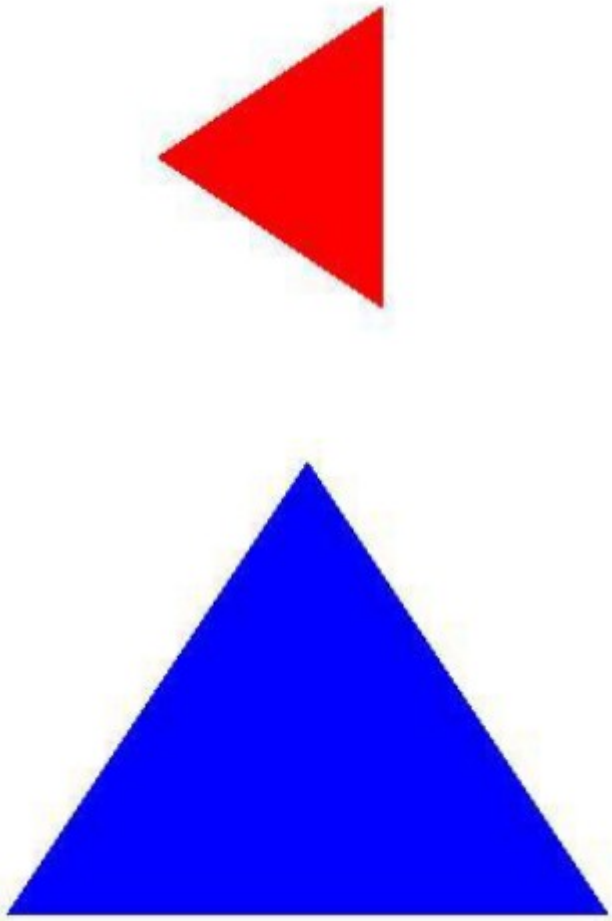
-	**Experiment No.:**	2	**Marks**		**Date Planned**		**Date Conducted**	
1	Title	Create and rotate a triangle about the origin and a _xed point.						
2	Course Outcomes							
3	Aim							
4	Material / Equipment Required	Lab Manual						
5	Theory, Formula, Principle, Concept							
6	Procedure, Program, Activity, Algorithm, Pseudo Code	• #include • <GL/glut.h> • #include • <stdlib.h> • #include <math.h> • /* Set initial display-window size. */ GLsizei • winWidth = 600, winHeight = 600; /* Set range for • world coordinates. */ GLfloat xwcMin = 0.0, xwcMax • = 225.0; GLfloat ywcMin = 0.0, ywcMax = 225.0; • class wcPt2D { • public: GLfloat • x, y; }; • typedef GLfloat Matrix3x3 [3][3]; Matrix3x3 • matComposite;						

```
const GLdouble pi = 3.14159; void
init (void)
{
/* Set color of display window to white. */ glClearColor
(1.0, 1.0, 1.0, 0.0);
}
/* Construct the 3 x 3 identity matrix. */
void matrix3x3SetIdentity (Matrix3x3 matIdent3x3)
{
GLint row, col;
for (row = 0; row <3; row++) for (col
= 0; col <3; col++)
matIdent3x3 [row][col] = (row == col);
}
void matrix3x3PreMultiply (Matrix3x3 m1, Matrix3x3 m2)
{
GLint row, col;
Matrix3x3 matTemp;
for (row = 0; row <3; row++) for (col =
0; col <3 ; col++)
matTemp [row][col] = m1 [row][0] * m2 [0][col] + m1 [row][1] * m2 [1][col] + m1
[row][2] * m2 [2][col];
for (row = 0; row <3; row++)
for (col = 0; col <3; col++)
m2 [row][col] =
matTemp [row][col];
}
void translate2D (GLfloat tx, GLfloat ty)
{
Matrix3x3 matTransl;
/*
Initialize translation matrix to identity. */
matrix3x3SetIdentity (matTransl);
matTransl [0][2] = tx;
matTransl [1][2] = ty;
/*
Concatenate matTransl with the composite matrix.
*/
matrix3x3PreMultiply (matTransl, matComposite);
}
void rotate2D (wcPt2D pivotPt, GLfloat theta)
{
Matrix3x3 matRot;
/*
Initialize rotation matrix to identity.
*/
matrix3x3SetIdentity (matRot);
matRot [0][0] =
cos (theta);
matRot [0][1] =
-sin (theta);
matRot [0][2] =
pivotPt.x * (1 - cos (theta)) +
pivotPt.y * sin
(theta);
matRot [1][0] =
sin (theta);
matRot [1][1] =
cos (theta);
matRot [1][2] =
pivotPt.y * (1 - cos (theta)) -
```

```
pivotPt.x * sin
(theta);
/*
Concatenate matRot with the composite matrix. */
matrix3x3PreMultiply (matRot, matComposite);
}
void scale2D (GLfloat sx, GLfloat sy, wcPt2D fixedPt)
{
Matrix3x3 matScale;
/*
Initialize scaling matrix to identity.
*/
matrix3x3SetIdentity (matScale);
matScale [0][0] = sx;
matScale [0][2] = (1 - sx) * fixedPt.x;
matScale [1][1] = sy;
matScale [1][2] = (1 - sy) * fixedPt.y;
/*
Concatenate matScale with the composite matrix.
*/
matrix3x3PreMultiply (matScale, matComposite);
}
/* Using the composite matrix, calculate transformed coordinates. */ void
transformVerts2D (GLint nVerts, wcPt2D * verts)
{
GLint k; GLfloat
temp;
for (k = 0; k <nVerts; k++) {
temp = matComposite [0][0] * verts [k].x + matComposite [0][1] * verts [k].y +
matComposite [0][2];
verts [k].y = matComposite [1][0] * verts [k].x + matComposite [1][1] *
verts [k].y + matComposite [1][2];
verts [k].x = temp;
}
}
void triangle (wcPt2D *verts)
{
GLint k;
glBegin (GL_TRIANGLES);
for (k = 0; k <3; k++)
glVertex2f (verts [k].x, verts [k].y);
glEnd ( );
}
void displayFcn (void)
{
/*
Define initial position for triangle. */
GLint nVerts = 3;
wcPt2D verts [3] = { {50.0, 25.0}, {150.0, 25.0}, {100.0, 100.0} };
/*
Calculate position of triangle centroid.
*/
wcPt2D centroidPt;
GLint k, xSum = 0, ySum = 0; for (k = 0;
k <nVerts; k++) { xSum += verts [k].x;
ySum += verts [k].y;
}
centroidPt.x = GLfloat (xSum) / GLfloat (nVerts); centroidPt.y =
GLfloat (ySum) / GLfloat (nVerts);
/*
Set geometric transformation parameters.
```

		• */
		• wcPt2D pivPt,fixedPt;
		• pivPt = centroidPt;
		• fixedPt = centroidPt;
		• GLfloat tx = 0.0, ty = 100.0;
		• GLfloat sx = 0.5, sy = 0.5;
		• GLdouble theta = pi/2.0;
		• glClear (GL_COLOR_BUFFER_BIT);
		• //
		• Clear display window.
		• glColor3f (0.0, 0.0, 1.0);
		• // Set initial fill color to blue.
		• triangle (verts);
		• //
		• Display blue triangle.
		• /*
		• Initialize composite matrix to identity.
		• */
		• matrix3x3SetIdentity (matComposite);
		• /*
		• Construct composite matrix for transformation sequence. */
		• scale2D (sx, sy, fixedPt);
		• // First transformation: Scale.
		• rotate2D (pivPt, theta);
		• // Second transformation: Rotate
		• translate2D (tx, ty);
		• // Final transformation: Translate.
		• /*
		• Apply composite matrix to triangle vertices. */
		• transformVerts2D (nVerts, verts);
		• glColor3f (1.0, 0.0, 0.0); // Set color for transformed triangle. triangle (verts);
		• glFlush ( );
		• }
		• void winReshapeFcn (GLint newWidth, GLint newHeight)
		• {
		• glMatrixMode (GL_PROJECTION);
		• glLoadIdentity ( );
		• gluOrtho2D (xwcMin, xwcMax, ywcMin, ywcMax);
		• glClear (GL_COLOR_BUFFER_BIT);
		• }
		• int main (int argc, char ** argv)
		• {
		• glutInit (&argc, argv);
		• glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
		• glutInitWindowPosition (50, 50); glutInitWindowSize
		• (winWidth, winHeight);
		• glutCreateWindow ("Geometric Transformation Sequence"); init ( );
		• glutDisplayFunc (displayFcn);
		• glutReshapeFunc (winReshapeFcn);
		• glutMainLoop ( );
		• return 0;
		• }

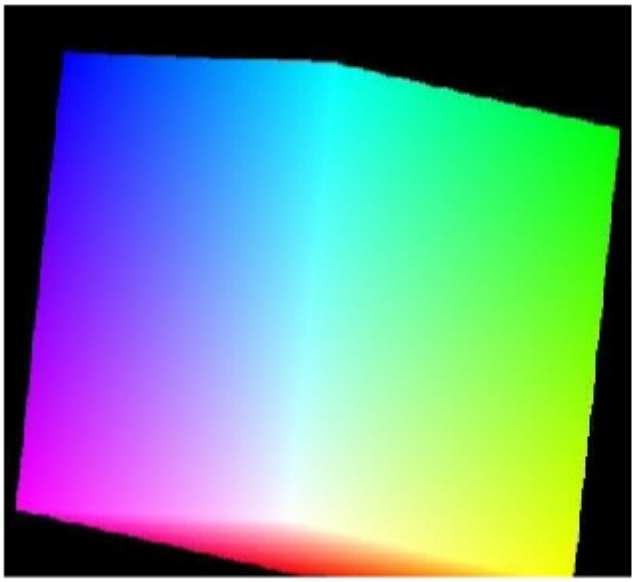
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	•
9	Sample Calculations	

10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 03 : OpenGL transformation matrices

-	**Experiment No.:**	3	**Marks**		**Date Planned**		**Date Conducted**	
1	Title	Draw a color cube and spin it using OpenGL transformation matrices.						
2	Course Outcomes							
3	Aim							
4	Material / Equipment Required	Lab Manual						
5	Theory, Formula, Principle, Concept							
6	Procedure, Program, Activity, Algorithm, Pseudo Code	• #include <stdio.h> • #include <stdlib.h> • GLfloat vertices[ ]={ -1.0,-1.0,-1.0, 1.0,-1.0,-1.0, • 1.0, 1.0,-1.0, • - 1.0, 1.0,-1.0, • - 1.0,-1.0, 1.0, 1.0,- • 1.0, 1.0, • 1.0, 1.0, 1.0, -1.0,						

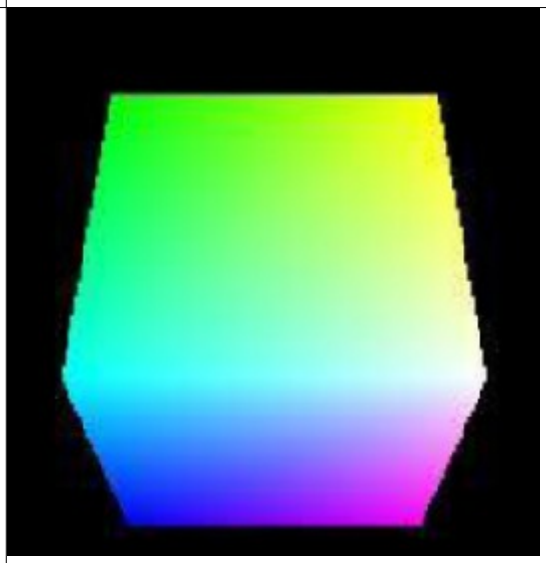
```
    1.0, 1.0 };
    GLfloat normals[ ]={ -1.0,-1.0,-1.0, 1.0,-1.0,-
    1.0, 1.0, 1.0,-1.0,
    -1.0, 1.0,-1.0, -1.0,-
    1.0, 1.0,
    1.0,-1.0, 1.0, 1.0,
    1.0, 1.0, -1.0, 1.0, 1.0
    };
    GLfloat colors[ ]={0.0,0.0,0.0,
    1.0,0.0,0.0,
    1.0,1.0,0.0,
    0.0,1.0,0.0,
    0.0,0.0,1.0,
    1.0,0.0,1.0,
    1.0,1.0,1.0,
    0.0,1.0,1.0};
    GLubyte cubeIndices[]={0,3,2,1, 2,3,7,6,
    0,4,7,3,
    1,2,6,5,
    4,5,6,7,
    0,1,5,4 };
    static GLfloat theta[]={0.0,0.0,0.0}; static GLint
    axis=2;
    void display(void)
    {
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity(); glRotatef(theta[0],1.0,0.0,0.0);
    glRotatef(theta[1],0.0,1.0,0.0); glRotatef(theta[2],0.0,0.0,1.0);
    glDrawElements(GL_QUADS,24,GL_UNSIGNED_BYTE,cubeIndices);
    glFlush();
    glutSwapBuffers();
    }
    void mouse(int btn,int state,int x,int y)
    {
    if(btn==GLUT_LEFT_BUTTON &&state==GLUT_DOWN)axis=0;
    if(btn==GLUT_RIGHT_BUTTON &&state==GLUT_DOWN) axis=1;
    if(btn==GLUT_MIDDLE_BUTTON &&state==GLUT_DOWN) axis=2;
    }
    void spincube()
    {
    theta[axis]+=2.0;
    if(theta[axis]>360.0)
    theta[axis]-=360.0;
    glutPostRedisplay();
    }
    void myReshape(int w,int h)
    {
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if(w<=h)
glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,10.0,10.0);
    else
    glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-
10.0,10.0);
    glMatrixMode(GL_MODELVIEW);
    }
    int main(int argc,char **argv)
    {
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
```

		• glutInitWindowSize(500,500); • glutCreateWindow("color cuce"); • glutReshapeFunc(myReshape); • glutDisplayFunc(display); glutMouseFunc(mouse); • glutIdleFunc(spincube); • glEnable(GL_DEPTH_TEST); • glEnableClientState(GL_COLOR_ARRAY); • glEnableClientState(GL_VERTEX_ARRAY); • glEnableClientState(GL_NORMAL_ARRAY); • glVertexPointer(3,GL_FLOAT,0,vertices); • glColorPointer(3,GL_FLOAT,0,colors); • glNormalPointer(GL_FLOAT,0,normals); • glColor3f(1.0,1.0,1.0); • glutMainLoop(); • } •
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	•
9	Sample Calculations	
10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 04 : color cube

-	**Experiment No.:**	4	**Marks**		**Date Planned**		**Date Conducted**	
1	Title							
2	Course Outcomes	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.						
3	Aim							
4	Material / Equipment Required							
5	Theory, Formula, Principle, Concept							
6	Procedure, Program, Activity, Algorithm, Pseudo Code	• #include <GL/glut.h> • #include <stdio.h> • #include <stdlib.h> • GLfloat vertices[ ]={ -1.0,-1.0,-1.0, • 1.0,-1.0,-1.0, • 1.0, 1.0,-1.0, • - 1.0, 1.0,-1.0, • - 1.0,-1.0, 1.0, • 1.0,-1.0, 1.0, • 1.0, 1.0, 1.0, • -1.0, 1.0, 1.0 }; • GLfloat normals[ ] ={ -1.0,-1.0,-1.0, • 1.0,-1.0,-1.0, • 1.0, 1.0,-1.0, • -1.0, 1.0,-1.0, • -1.0,-1.0, 1.0, • 1.0,-1.0, 1.0, • 1.0, 1.0, • 1.0, • -1.0, 1.0, 1.0 }; • GLfloat • 1.0, • 0.0, • 0.0, • 1.0, • 1.0, • 0.0, • 1.0, • 1.0, • 0.0, • 0.0, • 1.0, • 1.0, • colors[ ]={ • 1.0, • 0.0, • 0.0, • 0.0, • 1.0, • 1.0, • 1.0, • 1.0}; • 0.0,0.0,0.0, • 0.0, • GLubyte cubeIndices[]={0,3,2,1,						

```
    2,3,7,6, 0,4,7,3, 1,2,6,5, 4,5,6,7,
    0, 1, 5, 4 };
    static GLfloat theta[]={0.0,0.0,0.0}; static GLint
    axis=2;
    static GLdouble viewer[]={0.0,0.0,5.0};
    void display(void)
    {
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT); glLoadIdentity();
    gluLookAt(viewer[0],viewer[1],viewer[2],0.0,0.0,0.0,0.0,1.0,0.0);
    glRotatef(theta[0],1.0,0.0,0.0); glRotatef(theta[1],0.0,1.0,0.0);
    glRotatef(theta[2],0.0,0.0,1.0);
    glDrawElements(GL_QUADS,24,GL_UNSIGNED_BYTE,cubeIndices);
    glFlush();
    glutSwapBuffers();
    }
    void mouse(int btn, int state, int x, int y)
    {
    if(btn==GLUT_LEFT_BUTTON &&state==GLUT_DOWN)axis=0;
    if(btn==GLUT_RIGHT_BUTTON &&state==GLUT_DOWN) axis=1;
    if(btn==GLUT_MIDDLE_BUTTON &&state==GLUT_DOWN) axis=2;
    theta[axis]+=2.0;
    if(theta[axis]>360.0)
    theta[axis]-=360.0;
    glutPostRedisplay();
    }
    void keys(unsigned char key, int x, int y)
    {
    if(key=='x') viewer[0]-=1.0; if(key=='X')
    viewer[0]+=1.0;
    if(key=='y') viewer[1]-=1.0; if(key=='Y')
    viewer[1]+=1.0; if(key=='z') viewer[2]-
    =1.0;
    if(key=='Z')
    viewer[2]+=1.0;
    glutPostRedisplay();
    }
    void myReshape(int w, int h)
    {
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if(w<=h)
glFrustum(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,2.0,20.0);
else
    glFrustum(-2.0,2.0,-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/
(GLfloat)h,2.0,20.0);
    glMatrixMode(GL_MODELVIEW);
    }
    int main(int argc, char **argv)
    {
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(500,500);
    glutCreateWindow("color cuce");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutKeyboardFunc(keys); glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glEnableClientState(GL_COLOR_ARRAY);
    glEnable(GL_DEPTH_TEST);
    glEnableClientState(GL_COLOR_ARRAY);
```

		• glEnableClientState(GL_VERTEX_ARRAY); • glEnableClientState(GL_NORMAL_ARRAY); • glVertexPointer(3,GL_FLOAT,0,vertices); • glColorPointer(3,GL_FLOAT,0,colors); • glNormalPointer(GL_FLOAT,0,normals); • glColor3f(1.0,1.0,1.0); glutMainLoop(); • }
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	•
9	Sample Calculations	
10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

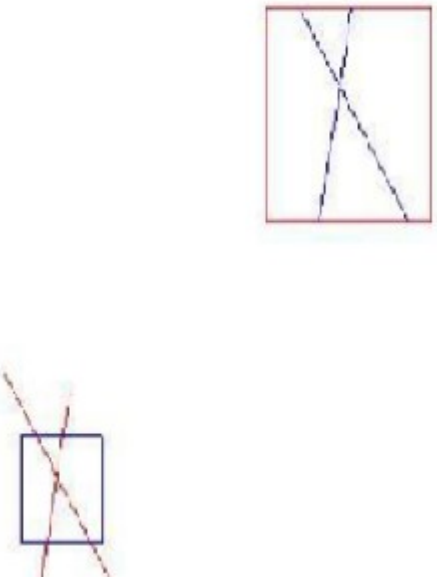
Experiment 05 : Clip a line using Cohen-Sutherland algorithm.

-	**Experiment No.:**	5	**Marks**		**Date Planned**		**Date Conducted**	
1	Title	Clip a line using Cohen-Sutherland algorithm.						
2	Course Outcomes	Clip a line using Cohen-Sutherland algorithm.						
3	Aim							
4	Material / Equipment Required							
5	Theory, Formula, Principle, Concept							
6	Procedure, Program, Activity, Algorithm, Pseudo Code	• #include<stdio.h> • #include<stdbool.h> • #include<GL/glut.h> • #define outcode int • #define true 1 #define • false 0						

```
• double xmin=50,ymin=50, xmax=100,ymax=100; // Window boundaries double
• xvmin=200,yvmin=200,xvmax=300,yvmax=300; // Viewport boundaries //int x1, x2, y1, y2;
• //bit codes for the right, left, top, &bottom const int RIGHT
• = 8;
• const int LEFT = 2; const
• int TOP = 4; const int
• BOTTOM = 1;
• //used to compute bit codes of a point outcode
• ComputeOutCode (double x, double y);
• //Cohen-Sutherland clipping algorithm clips a line from //P0 = (x0, y0) to
• P1 = (x1, y1) against a rectangle with //diagonal from (xmin, ymin) to
• (xmax, ymax).
• void CohenSutherlandLineClipAndDraw (double x0, double y0,double x1, double y1)
• {
• //Outcodes for P0, P1, and whatever point lies outside the clip rectangle outcode outcode0,
• outcode1, outcodeOut;
• bool accept = false, done = false;
• //compute outcodes
• outcode0 = ComputeOutCode (x0, y0);
• outcode1 = ComputeOutCode (x1, y1);
• do
• {
• if (!(outcode0 | outcode1))
• //logical or is 0 Trivially accept &exit
• {
• accept = true;
• done = true;
• }
• else if (outcode0 &outcode1) //logical and is not 0. Trivially reject and exit done = true;
• else
• {
• //failed both tests, so calculate the line segment to clip //from an outside
• point to an intersection with clip edge double x, y;
• //At least one endpoint is outside the clip rectangle; pick it. outcodeOut =
• outcode0? outcode0: outcode1;
• //Now find the intersection point;
• //use formulas y = y0 + slope * (x - x0), x = x0 + (1/slope)* (y - y0)
• if(outcodeOut &TOP)
• //point is above the clip rectangle
• {
• x = x0 + (x1 - x0) * (ymax - y0)/(y1 - y0); y = ymax;
• }
• else if(outcodeOut &BOTTOM) //point is below the clip rectangle
• {
• x = x0 + (x1 - x0) * (ymin - y0)/(y1 - y0); y = ymin;
• }
• else if(outcodeOut &RIGHT) //point is to the right of clip rectangle
• {
• y = y0 + (y1 - y0) * (xmax - x0)/(x1 - x0); x = xmax;
• }
• else
• //point is to the left of clip rectangle
• {
• y = y0 + (y1 - y0) * (xmin - x0)/(x1 - x0); x = xmin;
• }
• //Now we move outside point to intersection point to clip //and get ready
```

```
for next pass.
if (outcodeOut == outcode0)
{
x0 = x;
y0 = y;
outcode0 = ComputeOutCode (x0, y0);
}
else
{
x1 = x;
y1 = y;
outcode1 = ComputeOutCode (x1, y1);
}
}
}while (!done);
if (accept)
{
// Window to viewport mappings
double sx=(xvmax-xvmin)/(xmax-xmin); // Scale parameters double
sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
//draw a red colored viewport
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xvmin, yvmin);
glVertex2f(xvmax, yvmin);
glVertex2f(xvmax, yvmax);
glVertex2f(xvmin, yvmax);
glEnd();
glColor3f(0.0,0.0,1.0); // draw blue colored clipped line
glBegin(GL_LINES);
glVertex2d (vx0, vy0);
glVertex2d (vx1, vy1); glEnd();
}
}
//Compute the bit code for a point (x, y) using the clip rectangle
//bounded diagonally
by (xmin, ymin), and (xmax, ymax)
outcode ComputeOutCode (double x, double y)
{
outcode code = 0;
if (y >ymax)
//above the clip window
code |= TOP;
else if (y <ymin)
//below the clip window
code |= BOTTOM;
if (x >xmax)
//to the right of clip window
code |= RIGHT;
else if (x <xmin)
code |= LEFT;
return code;
VI
15CSL68
//to the left of clip window
}
void display()
```

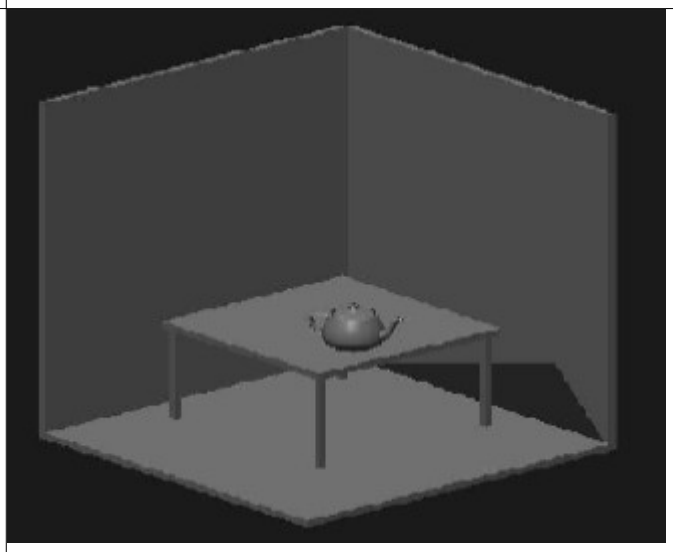
		• { • double x0=120,y0=10,x1=40,y1=130; • glClear(GL_COLOR_BUFFER_BIT); • //draw the line with red color • glColor3f(1.0,0.0,0.0); • //bres(120,20,340,250); • glBegin(GL_LINES); • glVertex2d • (x0, • y0); • glVertex2d • (x1, • y1); • glVertex2d • (60,20); • glVertex2d (80,120); • glEnd(); • //draw a blue colored window • glColor3f(0.0, 0.0, 1.0); • glBegin(GL_LINE_LOOP); • glVertex2f(xmin, ymin); • glVertex2f(xmax, ymin); • glVertex2f(xmax, ymax); • glVertex2f(xmin, ymax); • glEnd(); • CohenSutherlandLineClipAndDraw(x0,y0,x1,y1); • CohenSutherlandLineClipAndDraw(60,20,80,120); • glFlush(); • } • void myinit() • { • glClearColor(1.0,1.0,1.0,1.0); • glColor3f(1.0,0.0,0.0); • glPointSize(1.0); • glMatrixMode(GL_PROJECTION); • glLoadIdentity(); • gluOrtho2D(0.0,499.0,0.0,499.0); • } • int main(int argc, char** argv) • { • //printf("Enter End points:"); • //scanf("%d%d%d%d", &x1,&x2,&y1,&y2); • glutInit(&argc,argv); • glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); • glutInitWindowSize(500,500); • glutInitWindowPosition(0,0); • glutCreateWindow("Cohen Suderland Line Clipping Algorithm"); • glutDisplayFunc(display); • myinit(); • glutMainLoop(); • }
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	•
9	Sample Calculations	

10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 06 : simple shaded scene

-	**Experiment No.:**	6	**Marks**		**Date Planned**		**Date Conducted**	
1	Title							
2	Course Outcomes	To draw a simple shaded scene consisting of a tea pot on a table. De_ne suitably the position and properties of the light source along with the properties of the surfaces of the solid object used in the scene.						
3	Aim							
4	Material / Equipment Required							
5	Theory, Formula, Principle, Concept							
6	Procedure, Program, Activity, Algorithm, Pseudo Code	• #include <GL/glut.h> • #include <stdio.h> • #include <stdlib.h> • void wall(double thickness) • { • glPushMatrix(); • glTranslated(0.5,0.5*thickness,0.5); • glScaled(1.0,thickness,1.0); • glutSolidCube(1.0); • glPopMatrix(); • } • void tableleg(double thick,double len) • { • glPushMatrix(); • glTranslated(0,len/2,0);						

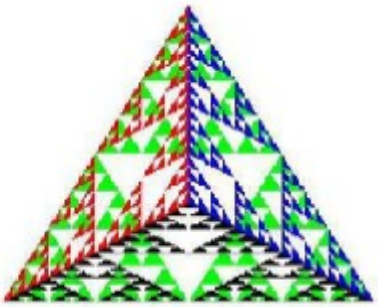
```
glScaled(thick,len,thick);
glutSolidCube(1.0);
glPopMatrix();
}
void table(double topw,double topt,double legt,double legl)
{
glPushMatrix();
glTranslated(0,legl,0);
glScaled(topw,topt,topw);
glutSolidCube(1.0);
glPopMatrix();
double dist=0.95*topw/2.0-legt/2.0;
glPushMatrix();
glTranslated(dist,0,dist);
tableleg(legt,legl);
glTranslated(0,0,-2*dist);
tableleg(legt,legl);
glTranslated(-2*dist,0,2*dist);
tableleg(legt,legl);
glTranslated(0,0,-2*dist);
tableleg(legt,legl); glPopMatrix();
}
void displaysolid(void)
{
GLfloat mat_ambient[]={0.7f,0.7f,0.7f,1.0f}; GLfloat
mat_diffuse[]={0.5f,0.5f,0.5f,1.0f};
GLfloat mat_specular[]={1.0f,1.0f,1.0f,1.0f}; GLfloat
mat_shininess[]={50.0f};
glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);
GLfloat lightint[]={0.7f,0.7f,0.7f,1.0f};
GLfloat lightpos[]={2.0f,6.0f,3.0f,0.0f};
glLightfv(GL_LIGHT0,GL_POSITION,lightpos);
glLightfv(GL_LIGHT0,GL_DIFFUSE,lightint);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
double winht=1.0; glOrtho(-winht*64/48.0,winht*64/48.0,-
winht,winht,0.1,100.0); glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(2.3,1.3,2.0,0.0,0.25,0.0,0.0,1.0,0.0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glRotated(90.0,0.0,0.0,1.0);
wall(0.02);
glPopMatrix();
wall(0.02);
glPushMatrix(); glRotated(-
90.0,1.0,0.0,0.0); wall(0.02);
glPopMatrix();
glPushMatrix();
glTranslated(0.4,0,0.4);
table(0.6,0.02,0.02,0.3);
glPopMatrix();
glPushMatrix();
glTranslated(0.6,0.38,0.5);
glRotated(30,0,1,0);
glutSolidTeapot(0.08);
glPopMatrix();
glFlush();
```

		• } • int main(int argc,char**argv) • { • glutInit(&argc,argv); • glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH); • glutInitWindowSize(500,500); glutInitWindowPosition(0,0); • glutCreateWindow("teapot"); glutDisplayFunc(displaysolid); • glEnable(GL_LIGHTING); • glEnable(GL_LIGHT0); • glShadeModel(GL_SMOOTH); • glEnable(GL_DEPTH_TEST); • glEnable(GL_NORMALIZE); • glClearColor(0.1,0.1,0.1,0.0); • glViewport(0,0,640,480); • glutMainLoop(); • } •
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	•
8	Observation Table, Look-up Table, Output	•
9	Sample Calculations	
10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 07 : Recursively subdivide

-	**Experiment No.:**	7	**Marks**		**Date Planned**		**Date Conducted**	
1	Title							
2	Course Outcomes	Design, develop and implement recursively subdivide a tetrahedron to form 3D sierpinski gasket. The number of recursive steps is to be speci_ed by the user.						
3	Aim							

4	Material / Equipment Required	
5	Theory, Formula, Principle, Concept	
6	Procedure, Program, Activity, Algorithm, Pseudo Code	#include <stdlib.h>#include • <stdio.h>#include <GL/glut.h> • typedef float point[3]; /* initial • tetrahedron */ • point v[]={{0.0, 0.0, 1.0}, {0.0, 0.942809, -0.33333}, • {-0.816497, -0.471405, -0.333333}, {0.816497, -0.471405, -0.333333}}; static GLfloat • theta[] = {0.0,0.0,0.0}; • int n; • void triangle(point a, point b, point c) • /* display one triangle using a line loop for wire frame, a single • normal for constant shading, or three normals for interpolative shading */ • { • glBegin(GL_POLYGON); • glNormal3fv(a); • glVertex3fv(a); • glVertex3fv(b); • glVertex3fv(c); • glEnd(); • } • void divide_triangle(point a, point b, point c, int m) • { /* triangle subdivision using vertex numbers • Righth and rule applied to create outward pointing faces */ point v1, v2, v3; • int j; • if(m>0) • { • for(j=0; j<3; j++) v1[j]=(a[j]+b[j])/2; for(j=0; j<3; j++) • v2[j]=(a[j]+c[j])/2; for(j=0; j<3; j++) v3[j]=(b[j]+c[j])/2; • divide_triangle(a, v1, v2, m-1); divide_triangle(c, • v2, v3, m-1); divide_triangle(b, v3, v1, m-1); • } • else(triangle(a,b,c)); /* draw triangle at end of recursion */ • } • void tetrahedron(int m) • { /* Apply triangle subdivision to faces of tetrahedron */ • glColor3f(1.0,0.0,0.0); • divide_triangle(v[0], v[1], v[2], m); • glColor3f(0.0,1.0,0.0); • divide_triangle(v[3], v[2], v[1], m); • glColor3f(0.0,0.0,1.0); • divide_triangle(v[0], v[3], v[1], m); • glColor3f(0.0,0.0,0.0); • divide_triangle(v[0], v[2], v[3], m); • } • void display(void) • { • glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); • glLoadIdentity(); • tetrahedron(n); • glFlush(); • } • void myReshape(int w, int h) • { • glViewport(0, 0, w, h); • glMatrixMode(GL_PROJECTION); • glLoadIdentity();

		• if (w <= h) • glOrtho(-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w, 2.0 * (GLfloat) h / • (GLfloat) w, -10.0, 10.0); • else • glOrtho(-2.0 * (GLfloat) w / (GLfloat) h, • 2.0 * (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0); • glMatrixMode(GL_MODELVIEW); • glutPostRedisplay(); • } • void main(int argc, char **argv) • { • printf("No. of Divisions ? "); • scanf("%d",&n); • glutInit(&argc, argv); • glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH); • glutInitWindowSize(500, 500); • glutCreateWindow("3D Gasket"); • glutReshapeFunc(myReshape); • glutDisplayFunc(display); • glEnable(GL_DEPTH_TEST); • glClearColor (1.0, 1.0, 1.0, 1.0); • glutMainLoop(); • }
7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	
8	Observation Table, Look-up Table, Output	
9	Sample Calculations	
10	Graphs, Outputs	 
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

Experiment 08 :Bezier Curve algorithm

-	**Experiment No.:**	1	**Marks**		**Date Planned**		**Date Conducted**	

1	Title	
2	Course Outcomes	Develop a menu driven program to animate a _ag using Bezier Curve algorithm.
3	Aim	
4	Material / Equipment Required	
5	Theory, Formula, Principle, Concept	

6	Procedure, Program, Activity, Algorithm, Pseudo Code	#include<GL/glut.h>
		#include<math.h>
		#include<stdio.h>
		void bezierCoefficients(int n,int *c)
		
		int k,i;
		for(k=0;k<=n;k++)
		
		c[k]=1; for(i=n;i>=k+1;i--
		) c[k]*=i; for(i=n-k;i>=2;i-
		
		c[k]/=i;
		
		void display()
		
		int cp[4][2]={{10,10},{100,200},{200,50},{300,300}}; int c[4],k,n=3;
		float x,y,u,blend;
		bezierCoefficients(n,c);
		glClear(GL_COLOR_BUFFER_BIT);
		glColor3f(1.0,0.0,0.0);
		glLineWidth(5.0);
		glBegin(GL_LINE_STRIP);
		for(u=0;u<1.0;u+=0.01)
		{x=0;y=0;
		for(k=0;k<4;k++)
		
		blend=c[k]*pow(u,k)*pow(1-u,n-k);
		x+=cp[k][0]*blend; y+=cp[k][1]*blend;
		
		glVertex2f(x,y);
		
		glEnd();
		glFlush();
		
		void myinit()
		
		glClearColor(1.0,1.0,1.0,1.0);
		glColor3f(1.0,0.0,0.0);
		glPointSize(5.0);
		gluOrtho2D(0.0,600,0.0,600.0);
		
		int main(int argc, char ** argv)
		
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(600,600);
		glutCreateWindow("Bezier Curve");
		glutDisplayFunc(display); myinit();
		glutMainLoop();
		return 0;

7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	
8	Observation Table, Look-up Table, Output	
9	Sample Calculations	
10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

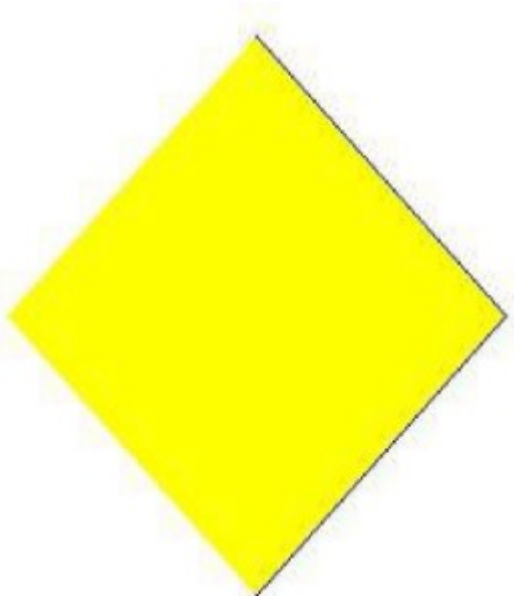
Experiment 09 : Polygon using scan line algorithm

-	**Experiment No.:**	9	**Marks**		**Date Planned**		**Date Conducted**	
1	Title							
2	Course Outcomes	Develop a menu driven program to _ll the polygon using scan line algorithm.						
3	Aim							
4	Material / Equipment Required							
5	Theory, Formula, Principle, Concept							

6	Procedure, Program, Algorithm, Pseudo Code	#include <stdlib.h>
		#include <stdio.h>
		#include <GL/glut.h>
		float x1,x2,x3,x4,y1,y2,y3,y4;
		void edgedetect(float x1,float y1,float x2,float y2,int *le,int *re)
		
		float mx,x,temp; int i;
		if((y2-y1)<0)
		
		temp=y1;y1=y2;y2=temp;						

```
temp=x1;x1=x2;x2=temp;

if((y2-y1)!=0) mx=(x2-x1)/(y2-

else
mx=x2-x1; x=x1;
for(i=y1;i<=y2;i++)

if(x<(float)le[i])
le[i]=(int)x;
if(x>(float)re[i])
re[i]=(int)x;
x+=mx;


void draw_pixel(int x,int y)

glColor3f(1.0,1.0,0.0);
glBegin(GL_POINTS);
glVertex2i(x,y); glEnd();

void scanfill(float x1,float y1,float x2,
float y2,float x3,float y3,float x4,float y4)

int le[500],re[500]; int i,y;
for(i=0;i<500;i++)

le[i]=500;
re[i]=0;

edgedetect(x1,y1,x2,y2,le,re);
edgedetect(x2,y2,x3,y3,le,re);
edgedetect(x3,y3,x4,y4,le,re);
edgedetect(x4,y4,x1,y1,le,re);
for(y=0;y<500;y++)

if(le[y]<=re[y])
for(i=(int)le[y];i<(int)re[y];i++)
draw_pixel(i,y);


void display()

x1=200.0;y1=200.0;x2=100.0;y2=300.0;x3=200.0;y3=400.0;x4=300.0;y4=300.0;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glVertex2f(x3,y3);
glVertex2f(x4,y4);
glEnd();
scanfill(x1,y1,x2,y2,x3,y3,x4,y4);
glFlush();

void myinit()

glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
```

		glLoadIdentity();
		gluOrtho2D(0.0,499.0,0.0,499.0);
		
		int main(int argc, char** argv)
		
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(0,0);
		glutCreateWindow("Filling a Polygon using Scan-line Algorithm");
		glutDisplayFunc(display);
		myinit();
		glutMainLoop();

7	Block, Circuit, Model Diagram, Reaction Equation, Expected Graph	
8	Observation Table, Look-up Table, Output	
9	Sample Calculations	
10	Graphs, Outputs	
11	Results &Analysis	
12	Application Areas	
13	Remarks	
14	Faculty Signature with Date	

F. Content to Experiment Outcomes

1. TLPA Parameters

Table 1: TLPA – Example Course

Module-	Course Content or Syllabus (Split module content into 2 parts which have similar concepts)	Content Teaching	Blooms' Learning	Final Blooms'	Ide Actio

#		Hours	Levels for Content	Level	for L
A	B	C	D	E	
1.	Implement Brenham's line drawing algorithm for all types of slope	4	- L2 -L3	L3	-Unde -Apply
2	Create and rotate a triangle about the origin and a fixed point.	4	- L2 - L4	L4	-unde -Analy
3	Draw a colour cube and spin it using OpenGL transformation matrices.	3	- L2 - L4	L4	-unde -Analy
4	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing	4	- L2 - L4	L4	-Unde -Distin
5	Clip a lines using Cohen-Sutherland algorithm	3	- L2 -L3	L3	-Unde -Cons
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.	3	- L2 - L4	L4	-Unde -Exam
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.	3	- L2 - L4	L4	-Unde -Comp
8	Develop a menu driven program to animate a flag using Bezier Curve algorithm	3	- L2 - L4	L4	-Unde -Distin
9	Develop a menu driven program to fill the polygon using scan line algorithm	3	- L2 - L3	L3	-Unde -Distin
10	Mini-Project with applications using Open GL API	10	- L2 -L3	L3	-Unde -Expe

2. Concepts and Outcomes:

Table 2: Concept to Outcome – Example Course

Module-#	Learning from study of the Content or Syllabus	Identified Concepts from Content	Final Concept	Concept Justification (What all Learning Happened from the study of Content / Syllabus. A short word for learning or outcome)	CO Compo (1.Action V 2.Knowle 3.Conditi Methodol 4.Bench m
A	I	J	K	L	M
1.	Implement Brenham's line drawing algorithm for all types of slope	- Brenham's line drawing algorithm	-Open GL API	Learning GL,GLUT Library Function	-Impliment -OpenGL API
2	Create and rotate a triangle about the origin and a fixed point.	-Transformation -Graphics Model	Graphics Model	Analyze how rotation,scaling,Transformation happens on 2D-0bjects	-Apply - Transformati Graphics Prin
3	Draw a colour cube and spin it using OpenGL transformation matrices.	-Vertex Array -Graphics Model	Graphics Models	Examine how to use vertex Array & Transformation.	-Apply - Transformati Graphics Prin
4	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing	-perspective viewing -Viewport Transformation	Viewport Transformation	Examine how perspective viewing works.	-Illustrates -Viewport Transformati Graphics Prin
5	Clip a lines using Cohen-Sutherland algorithm	--Clipping	Clipping	Use Cohen-Sutherland algorithm for clipping.	-Impliments -Clipping Alg
6	To draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of the light	- properties of the light source -Viewport Transformation	Viewport Transformation	Analyze properties of the light source,Viewport Transformation	Illustrates -Viewport Transformati Graphics Prin

	source along with the properties of the surfaces of the solid object used in the scene.				
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.	- 3D sierpinski gasket -Input Interactive Graphics	Input Interactive Graphics	Examine Input Interactive Graphics	-Interprete -various grap premitives
8	Develop a menu driven program to animate a flag using Bezier Curve algorithm	- menu driven program -Input Interactive Graphics	Input Interactive Graphics	Examine Input Interactive Graphics	-Interprete -various grap premitives
9	Develop a menu driven program to fill the polygon using scan line algorithm	-scan line algorithm -Open GL API	Open GL API	Apply open-gl Functions	-Impliment -OpenGL API
10	Mini-Project with applications using Open GL API	-Open GL API	Open GL API	Apply open-gl Functions	--Develope a -Mini-Project OpenGL API