Rashtreeya Shikshana Samithi Trust

RV UNIVERSITY

School of Computer Science and Engineering

Bengaluru – 560059

COMPUTER GRAPHICS

COURSE CODE: CS3102

VI SEMESTER B.Tech. (Hons.)

LABORATORY RECORD

2024-2025

RV UNIVERSITY

School of Computer Science and Engineering

Bengaluru – 560 059

LABORATORY CERTIFICATE

This is to certify that Mr./Ms. _____ has satisfactorily completed the course of experiments in Practical ***Computer Graphics*****(CS)** prescribed by the **School of Computer Science and Engineering** during the year **2024-25.**

Name of the Candidate: _____

USN: _____ Semester: _____

Marks	
Maximum	**Obtained**
25	

Signature of Faculty in-charge **Program Director**

Date:

Vision and Mission of the School of Computer Science and Engineering

Vision

To be a pioneering school of Computer Science and Engineering com mitted to fostering liberal education and empowering the next generation of technologists to make a positive global socio-economic impact.

Mission

- To be a pioneer in computer science education and benchmark ourselves with the world's top computer science and engineering institutions.
- To provide state-of-the-art facilities that enable exemplary pedagogy, advanced research, innovation and entrepreneurship in emerging technologies of computer science.
- To promote a culture of cooperation and inclusiveness among students and faculty from diverse communities enabling them to take part in interdisciplinary and multidisciplinary research, contributing to institution-building.
- To foster excellence through national and international academic, industry collaborations, bringing in diverse perspectives to drive innovation.
- To nurture a talented pool of ethical, self-driven and empathetic problem solvers to achieve sustainable development goals.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will be able to demonstrate excellence in the field of Computer Science and Engineering through advanced research and entrepreneurship.

PEO2: Graduates will be able to achieve excellence in innovation through national and international academic and industry collaborations.

PEO3: Graduates will be able to actively engage in cutting-edge interdisciplinary and multidisciplinary research.

PEO4: Graduates will be able to function effectively as talented pool of ethical, sustainable, self-driven and life-long empathetic problem solvers.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1: **Engineering Knowledge.** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: **Problem Analysis.** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: **Design/Development of Solutions.** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: **Conduct Investigations of Complex Problems.** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: **Modern Tool Usage.** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: **The Engineer and Society.** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.

PO7: **Environment and Sustainability.** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: **Individual and Teamwork.** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO9: **Ethics.** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO10: **Communication.** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: **Project Management and Finance.** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: **Life-Long Learning.** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1:

The student will apply the knowledge of data structures, database systems, system programming, networking web development, and AI & ML techniques in engineering the software.

PSO2:

The student will exhibit solid foundations and advancements in developing software /hardware systems for solving contemporary problems.

Course Outcomes: After completing the course, the students will be able to:	
CO1	Design and implement algorithms for 2D graphics primitives and attributes.
CO2	Apply 2D and 3D Transformations to create and manipulate graphical objects using OpenGL.
CO3	Apply concepts of clipping and visible surface detection in 2D and 3D viewing, color model and Illumination Models.
CO4	Develop interactive graphics applications and create smooth curves and animations using appropriate tools and techniques.

LIST OF PROGRAMS

Part-A

Sl. No.	Program	Week
1.	Implement Bresenham's line drawing algorithm for all types of slope.	**Week 2**
2.	Create and rotate a triangle about the origin and a fixed point.	**Week 3**
3.	Draw a colour cube and spin it using OpenGL transformation matrices.	**Week 4**
4.	Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.	**Week 6**
5.	Apply Cohen-Sutherland algorithm for line clipping.	**Week 6**
6.	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.	**Week 7**
7.	Develop a menu driven program to animate a flag using Bezier Curve algorithm	**Week 8**
8.	Develop a menu driven program to fill the polygon using scan line algorithm	**Week 10**

Part-B

Sl. No.	Course Mini Project
1.	Students should develop a project by applying the computer graphics skills to solve a real-world problem using Open GL API. Consider all types of attributes like color, thickness, styles, font, background, speed etc, while doing mini project.

Marks distribution

S.No.	Component	Marks
1	Lab Program execution	6
	UNIT1-4 Viva	4
2	**Course Mini Project**	
	1. Project mid review	5
	2. Final submission	10

Lab Program Execution rubrics (Max: 6 marks)					
Sl. No	**Criteria**	**Measuring Methods**	**Excellent**	**Good**	**Poor**
1	**Understanding of problem statement. (2 Marks)**	Observations	Student exhibits thorough understanding of requirements and applies suitable algorithm for the problem **(2 M)**	Student has sufficient understanding of requirements and applies suitable algorithm for the problem. **(1 M)**	Student does not have a clear understanding of requirements and is unable to apply suitable algorithm for the problem. **(0 M)**
2	**Execution (2 Marks)**	Observations	Student demonstrates the execution of the program with required output **(2 M)**	Student demonstrates the execution of the program with partial output. **(1 M)**	Student has not executed the program. **(0 M)**
3	**Results and Documentation (2 Marks)**	Observations	Documentation with appropriate comments and output is covered in data sheets. **(2 M)**	Documentation with only few comments and only few output cases is covered in data sheets. **(1 M)**	Documentation with no comments and no output cases is covered in data sheets. **(0 M)**
Viva Voce Rubrics (Max: 4 marks)					
1	**Conceptual Understanding (2 Marks)**	Viva Voce	Explains thoroughly the algorithm or related concepts. **(2 M)**	Adequately explains the algorithm and related concepts. **(1 M)**	Unable to explain algorithm and related concepts. **(0 M)**

2	**Use of appropriate Problem-Solving Techniques (2 Marks)**	Viva Voce	Insightful explanation of appropriate method to be used for the given problem to get desired solution. **(2 M)**	Sufficiently explains the use of appropriate method for the given problem to get desired solution. **(1 M)**	Unable to explain the method to be used for the given problem. **(0 M)**

Course Mini Project Assessment Plan

Mid-review schedule: Week-10

Details to be presented for Mid-review

S.No.	Item	Marks distribution
1	Project: Title, Input, and Expected Outputs	1
2	Method/s used to solve the problem	1
3	Understanding of the problem and possible methods to be used.	1
4	Expected Results	1
5	Further Plan	1

Final Project Submission

Final review schedule: Week-15

S.No.	Item	Marks distribution
1	Report submission	3
2	Code Demo & Presentation	4
3	Viva	3

Note: For both reviews, the team has to prepare a presentation (slides) to present their work to the audience.

Grading Rubrics for final Project Demo and Submission

Project Components **[10 Marks]**	**Excellent**	**Very Good**	**Good**	**Not yet completed**
Presentation **[2 Marks]**	Engaging, clear, and well-structured presentation that effectively communicates key points. **2 Marks**	Presentation is mostly clear and organized, with minor areas for improvement. **1.5 Marks**	Presentation is somewhat unclear or disorganized, with noticeable room for improvement. **1 Marks**	Presentation is missing or significantly incomplete. **0 Marks**
Implementation Demo [**2 Marks]**	Implementation with proper understanding of the project's technical aspects. **2 Marks**	Implementation is mostly successful, with minor areas for improvement. **1.5 Marks**	Implementation has noticeable shortcomings or errors that require attention. **1 Marks**	No implementation demo provided. **0 Marks**
Subject Knowledge [VIVA] **[3 Marks]**	Demonstrates a profound understanding of the subject matter, answering questions with clarity and depth. **3 Marks**	Answers questions accurately and confidently, with minor gaps in understanding. **2 Marks**	Provides satisfactory answers but lacks depth or clarity in some areas. **1 Marks**	Unable to assess subject knowledge due to no answer. **0 Marks**
Report Writing **[3 Marks]**	Report is well-written, organized, and thoroughly covers all necessary aspects with clarity. **3Marks**	Report is mostly clear and well-structured, with minor areas for improvement. **2 Marks**	Report has noticeable deficiencies in clarity, organization, or content. **1 Marks**	No report provided. **0 Marks**

INDEX

Part-A

Sl. no.	Program Name.	Date	Record Marks (max 6)	Viva Voice (max 4)	Total Marks (max 10)	Sign
1.						
2.						
3.						
4.						
5.						
6.						
7						
8						
	Total (80)					
	Total (10)					

RECORD	**Max – 10**	
MINI PROJECT	**Max – 15**	
TOTAL	**Max - 25**	
Signature of the faculty		

Part-B

Project Title	Date	Mid review-1 (max 5)	Final Submission (max 10)			
			Report (max 3)	Code demo (max 2)	Presentation (max 2)	Viva Voice (max 3)
Total						
Total (15)						

OPENGL

OpenGL, or the Open Graphics Library, is a 3D graphics language developed by Silicon Graphics. Before OpenGL was available, software developers had to write unique 3D graphics code for each operating system platform as well as different graphics hardware. However, with OpenGL, developers can create graphics and special effects that will appear nearly identical on any operating system and any hardware that supports OpenGL. This makes it much easier for developers of 3D games and programs to port their software to multiple platforms.

When programmers write OpenGL code, they specify a set of commands. Each command executes a drawing action or creates a special effect. Using hundreds or even thousands of these OpenGL commands, programmers can create 3D worlds which can include special effects such as texture mapping, transparency (alpha blending), hidden surface removal, antialiasing, fog, and lighting effects. An unlimited amount of viewing and modeling transformations can be applied to the OpenGL objects, giving developers an infinite amount of possibilities.

GLUT gives you the ability to create a window, handle input and render to the screen without being Operating System dependent.

The first things you will need are the OpenGL and GLUT header files and libraries for your current Operating System.

Once you have them setup on your system correctly, open your first c++ file and include them at the start of your file like so:

```
#include <GL/gl.h> //include the gl header file
#include <GL/glut.h> //include the GLUT header file
```

Now, just double check that your system is setup correctly, and try compiling your current file.
If you get no errors, you can proceed. If you have any errors, try your best to fix them.

Once you are ready to move onto the next step, create a main() method in your current file.

Inside this is where all of your main GLUT calls will go.

The first call we are going to make will initialize GLUT and is done like so:

```
glutInit (&argc, argv); //initialize the program.
```

Keep in mind for this, that argc and argv are passed as parameters to your main method. You can see how to do this below.

Once we have GLUT initialized, we need to tell GLUT how we want to draw. There are several parameters we can pass here, but we are going to stick the with most

basic GLUT_SINGLE, which will give use a single buffered window.
```
glutInitDisplayMode
(GLUT_SINGLE);//set up a basic display buffer (only singular for now)
```

The next two methods we are going to use, simply set the size and position of the GLUT window on our screen:
```
glutInitWindowSize (500, 500); //set whe width and height of the window
glutInitWindowPosition (100, 100); //set the position of the window
```

And then we give our window a caption/title, and create it.
```
glutCreateWindow ("A basic OpenGL Window"); //set the caption for the window
```

We now have a window of the size and position that we want. But we need to be able to draw to it. We do this, by telling GLUT which method will be our main drawing method. In this case, it is a void method called display()
```
glutDisplayFunc (display); //call the display function to draw our world
```

Finally, we tell GLUT to start our program. It does this by executing a loop that will continue until the program ends.
```
glutMainLoop (); //initialize the OpenGL loop cycle
```

So thus far, we have a window. But the display method that I mentioned is needed. Lets take a look at this and dissect it.

```
void display (void) {
   glClearColor (0.0,0.0,0.0,1.0); //clear the color of the window
   glClear (GL_COLOR_BUFFER_BIT); //Clear teh Color Buffer (more buffers later on)
   glLoadIdentity();  //load the Identity Matrix
   gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //set the view
   glFlush(); //flush it all to the screen
}
```

The first method, glClearColor will set the background color of our window. In this example, we are setting the background to black (RGB 0, 0, 0). The 1.0 value on the end is an alpha value and makes no difference at this stage as we don't have alpha enabled.

The next thing we want to do, is erase everything currently stored by OpenGL. We need to do this at the start of the method, as the method keeps looping over itself and if we draw something, we need to erase it before we draw our next frame. If we don't do this, we can end up with a big mess inside our buffer where frames have been drawn over each other.

The third method, glLoadIdentity resets our model view matrix to the identity matrix, so that our drawing transformation matrix is reset.

From then, we will set the 'camera' for our scene. I am placing it 5 units back into the user so that anything we draw at 0,0,0 will be seen infront of us.

The final thing we need to do is then flush the current buffer to the screen so we can see it. This can be done with glFlush() as we are only using a single buffer.

PART - A

PROGRAM – 1

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

- An **accurate**, **efficient** raster line drawing algorithm developed by Bresenham, scan converts lines using only *incremental integer* calculations that can be adapted to display circles and other curves.
- Efficient compared to DDA algorithm because it avoid round functions.

  |m| < 1
- Samples a line by incrementing by one either x or y depending on the slope of the line.

  Starting from the left end point (*x*k,*y*k) of a given line.

  Assuming we have determined that the pixel at (*x*k,*y*k) is to be displayed, we next need to decide which pixel to plot in column *x*k+1.

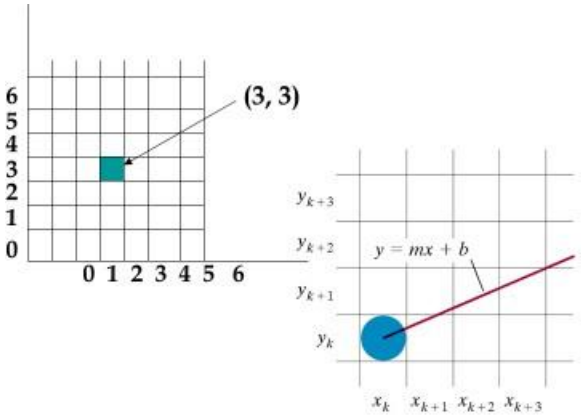

Figure 3-10

A section of the screen showing a pixel in column x_k on scan line y_k that is to be plotted along the path of a line segment with slope $0 < m < 1$.

Choices are$(x_k + 1, y_k)$ and (x_k+1, y_K+1)

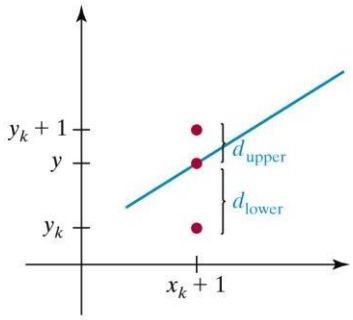

Figure 3-11

Vertical distances between pixel positions and the line y coordinate at sampling position $x_k + 1$.

- Difference between the vertical distances is computed and **the sign of the difference is used to select the pixel whose distance from C is smaller as the best approximation to the line.**
- At sampling position x_k+1,we Label the vertical pixel separation from mathematical line path d_1 *and* d_2.

y coordinate on the math line at pixel column position x_{k+1} is calculated as

$$y=m(x_k+1)+b$$

$$d1=y- y_k$$

$$=m(x_k+1)+b-y_k$$

$$d2= (y_k+1)-y$$

$$= y_k+1-m(x_k+1)-b$$

$$d_1 = y – y_k = m(x_k + 1) + b – y_k$$

$$d_2 = (y_k + 1) – y = y_k + 1- m(x_k + 1) – b$$

The difference between these 2 separations is

$$d1-d2 = 2m(xk + 1) – 2\ yk + 2b – 1$$

Substitute $m= \Delta y/ \Delta x$ *in the above equation*

- You will get $\Delta x\ (d_1-d_2) =2\Delta y x_k-2\ \Delta x y_k + c$

We call this Δx (d_1-d_2) as decision parametr p_k

Define

$$P_k = \Delta x (d_1-d_2) = 2\Delta y x_k - 2 \Delta x y_k + c$$

The sign of P_k is the same as the sign of d_1-d_2, since $\Delta x > 0$. *Parameter c* is a constant and has the value $2\Delta y + \Delta x(2b-1)$ (independent of pixel position)

If *pixel at* y_k is closer to line-path than pixel at $y_k + 1$ *(i.e, if* $d_1 < d_2$*)* then p_k is negative. We plot lower pixel in such a case. Otherwise , upper pixel will be plotted.

If $p_k<0$ then $d_1<d_2$ hence choose x_k+1,y_k

If $p_k>0$ then $d_1>d_2$ hence choose x_k+1,y_k+1

At step $k + 1$, the decision parameter can be evaluated as,

$$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c$$

Taking the difference of p_{k+1} and p_k we get the following.

$$p_{k+1} - p_k = 2\Delta y(x_{k+1}- x_k)-2\Delta x(y_{k+1} - y_k)$$

But, $x_{k+1} = x_k + 1$, so that

$$p_{k+1} = p_k + 2\Delta y - 2 \Delta x(y_{k+1} - y_k)$$

Where the term y_{k+1}-y_k is either 0 or 1, depending on the sign of *parameter* p_k

The first parameter p_0 is directly computed

$$p_0 = 2 \Delta y x_k - 2 \Delta x y_k + c = 2 \Delta y x_k - 2 \Delta x y_k + 2 \Delta y + \Delta x (2b-1)$$

Since (x_0,y_0) satisfies the line equation , we also have

$$y_0 = \Delta y/ \Delta x * x_0 + b$$

Combining the above 2 equations , we will have

$$p_0 = 2\Delta y - \Delta x$$

The constants $2\Delta y$ and $2\Delta y$-$2\Delta x$ are calculated once for each time to be scan converted.

Stick Data sheets of Program-1 here:

OUTPUT:

PROGRAM – 2

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

Use the concept of multi dimensional array to define a triangle , define a rotation matrix and find the resultant matrix.

Stick Data sheets of Program-2 here:

OUTPUT:

PROGRAM – 3

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

1. Define global arrays for vertices and colors
2. Draw cube from polygon faces.
3. Define function to spin the cube and use mouse operation.

Stick Data sheets of Program-3 here:

OUTPUT:

PROGRAM – 4

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

1. Define global arrays for vertices and colors
2. Draw cube from polygon faces.
3. Initialize the theta value and initial viewer location and axis.
4. Define display function to update viewer position in model view matrix
5. Define mouse function for rotating cube and key function for adjusting the viewer position.

Stick Data sheets of Program-4 here:

OUTPUT:

PROGRAM – 5

Apply Cohen-Sutherland algorithm for line clipping.

The Cohen-Sutherland algorithm uses a divide-and-conquer strategy. The line segment's endpoints are tested to see if the line can be trivially accepted or rejected. If the line cannot be trivally accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions. Each end point of the line segment is then assigned the code of the region in which it lies.

Stick Data sheets of Program-5 here:

OUTPUT:

PROGRAM – 6

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.

Sierpinski's Triangle is a very famous fractal that's been seen by most advanced math students. This fractal consists of one large triangle, which contains an infinite amount of smaller triangles within. The infinite amount of triangles is easily understood if the fractal is zoomed in many levels. Each zoom will show yet more previously unseen triangles embedded in the visible ones.

Creating the fractal requires little computational power. Even simple graphing calculators can easily make this image. The fractal is created pixel by pixel, using random numbers; the fractal will be slightly different each time due to this. Although, if you were to run the program repeatedly, and allow each to use an infinite amount of time, the results would be always identical. No one has an infinite amount of time, but the differences in the finite versions are very small.

A **fractal** is generally "a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole“.

Stick Data sheets of Program-6 here:

OUTPUT:

Menu creation example for program 7 and 8

int glutCreateMenu(void (*f)(int value))

Creates top level menu that uses callback f( ) which is passed an integer value from the menu entry. Returns a unique menu identifier

void glutSetMenu(int id)

Sets the current menu to id

void glutAddMenuEntry(char *name, int value)

Adds entry to the current menu. Name is the entry name and value is returned to the menu callback

int glutAttachMenu(int button)

Attaches current menu to the specified mouse button. Button is GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON or GLUT_MIDDLE_BUTTON

void glutAddSubMenu(char *name, int menu)

Adds a submenu entry name as the next entry in the current menu. The value of menu is the id of the submenu returned when the submenu was created.

Example:

```
glutCreateMenu(mymenu);  // single menu, no need for id glutAddMenuEntry(“ClearScreen”, 1);
glutAddMenuEntry(“Exit”, 2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Callback:

```
void mymenu(int value)
{
 if(value== 1) glClear( );
if(value== 2) exit(0); }
```

PROGRAM – 7

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

Stick Data sheets of Program-7 here:

OUTPUT:

PROGRAM – 8

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

Stick Data sheets of Program-8 here:

OUTPUT:

OUTPUT:

PART – B

You need to submit the hard copy of the course project report.