SARANATHAN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

Subject Code / Name	**:**	**CS6513 - Computer Graphics Laboratory**
Year/Sem	**:**	**III/V**
Academic Year	**:**	**2017-2018**

CS6513 COMPUTER GRAPHICS LABORATORY L T P C

0 0 3 2

1. Implementation of Algorithms for drawing 2D Primitives –

   Line (DDA, Bresenham) – All slopes

   Circle (Midpoint)

2. 2D Geometric transformations – Translation Rotation Scaling Reflection Shear Window-Viewport

3. Composite 2D Transformations

4. Line Clipping

5. 3D Transformations - Translation, Rotation, Scaling.

6. 3D Projections – Parallel, Perspective.

7. Creating 3D Scenes.

8. Image Editing and Manipulation - Basic Operations on image using any image editing software, Creating gif animated images, Image optimization.

9. 2D Animation – To create Interactive animation using any authoring tool.

LIST OF EXPERIMENTS

1) Basic Graphics Functions

2) a) Implementation of Digital Differential Analyser (DDA) Algorithm

   b) Implementation of Bresenham's Line Drawing Algorithm

   c) Implementation of Mid-Point Circle Drawing Algorithm

3) a) Implementation of 2D Transformations (Translation, Rotation, Scaling, Reflection and Shearing)

   b) Window to Viewport Mapping

4) Implementation of Composite 2D Transformations

5) Implementation of Cohen-Sutherland 2D Clipping

6) Implementation of 3D Transformations (Translation, Rotation, Scaling, Reflection and Shearing)

7) Implementation of 3D Projection

8) Three Dimensional Scenes using OpenGL

9) Basic Operations on Image using Photoshop

10) Creating an animation scene using Micromedia Flash

Ex.No. : 2a **IMPLEMENTATION OF DIGITAL DIFFERENTIAL ANALYZER ALGORITHM**

Aim:

To write a C Program for implementing Digital Differential Analyzer algorithm for line drawing.

Algorithm:

1. Call the initgraph() function.
2. Initialize the graphic mode with the path location in TurboC3 folder.
3. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also declare gdriver=DETECT, mode.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1, y1) into the frame buffer; that is, plot the first point. put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs (dx) > abs (dy), do s=abs(dx). Otherwise s= abs(dy).
8. Compute xi=dx/s and yi=dy/s.
9. Plot the successive pixel from k=1 to length s
    i. x=x+xi.
    ii. Y=y+yi.
10. Plot pixels using putpixel at points (x,y) in specified colour.
11. Call the closegraph() function.

Result:

Thus a C Program to implement Digital Differential Analyzer algorithm for drawing a line was written and executed.

Ex.No. : 2 b **IMPLEMENTATION OF BRESENHAM'S LINE DRAWING ALGORITHM**

Aim:

To write a C Program for implementing Bresenham's algorithm for line drawing.

Algorithm:

1. Call the initgraph() function.
2. Read the two end points of the line say (x1,y1), (x2,y2)
3. Assign the starting coordinates to (x,y)
4. Using putpixel() function plot the first point (x,y)
5. Compute dx,dy.
6. Determine the decision parameter P = 2dy-dx
7. At each point along the line starting , perform the following test
    a. If P< 0, the next point to plot is (x +1, y) and P = P +2dy
    b. Otherwise the next point to plot is (x , y+1) and P = P +2dy- 2dx
8. Repeat step 4 up to dx times

Result:

Thus a C Program to implement Bresenham's algorithm for drawing a line was written and executed.

Ex. No: 2 c **IMPLEMENTATION OF MID POINT CIRCLE DRAWING ALGORITHM**

AIM:

To implement the Midpoint circle drawing algorithm using c language.

ALGORITHM:

STEP 1: include the graphics and other header files.

STEP 2: initialize graphics mode and graphics driver.

STEP 3: get the radius of a circle.

STEP 4: initialize x=0 and y=r.

STEP 5: calculate p.

STEP 6: putpixel(x,y).

STEP 7: if p<0 then calculate x=x+1,y=y and p=p+2*x+1 else calculate x=x+1,y=y-1 and

p= p+2*(x-y)+1.

STEP 8: repeat step 6 and step 7 until x<y.

STEP 9: Display a circle.

RESULT:

Thus the above program for implementation of Midpoint Circle drawing algorithm was successfully executed.

Ex. No: 3 a **IMPLEMENTATION OF 2D TRANSFORMATIONS**

AIM:

To write a c program for 2D transformation.

ALGORITHM:

STEP 1: include the graphics and other header files.

STEP 2: initialize graphics mode and graphics driver.

STEP 3: Display a Menu and get choice from user.

STEP 4: get the object (triangle) coordinates.

STEP 5: Display the original object position.

STEP 6: a. if the choice is 1 then get translation vertex from user and translate the object to given vertex.

b. if the choice is 2 then get rotation angle and pivot point from user and rotate the object about pivot point.

c. if the choice is 3 then get rotation angle from user and rotate the object about origin.

d. if the choice is 4 then get scaling factor and fixed point from user and scale the object from fixed point.

e. if the choice is 5 then get shear value and fixed point from user and shear the object from fixed point.

f. If the choice is 6 then reflect the object.

g. if the choice is 7 then exit the program.

STEP 7: Display the transformed object.

RESULT:

Thus the above program for 2D transformation was successfully executed.

Ex. No: 3 b **WINDOW TO VIEWPORT MAPPING**

AIM:

To write a c program for window to viewport mapping.

ALGORITHM:

STEP 1: include the graphics and other header files.

STEP 2: initialize graphics mode and graphics driver.

STEP 3: get the object (triangle) coordinates.

STEP 4: set the window boundary.

STEP 5: Display the object in window.

STEP 6: set the view port boundary.

STEP 7: display the object in view port.

RESULT:

Thus the above program for window viewport mapping was successfully executed.

Ex.No: 4 **IMPLEMENTATION OF COMPOSITE 2D TRANSFORMATIONS**

Aim:

To write a C program to perform 2D composite transformations.

Algorithm:

1. Read the coordinate values to draw the 2D figure and perform the following steps to achieve the necessary composite transformations

2. Translation : Read the translation vector tx and ty and compute the new position using x1 = x + tx , y1 = y + ty and display the new position

3. Scaling : Read the scaling factor sx and sy and compute the new position using x1 = x. sx , y1 = y. sy and display the new position

4. InverseTranslation : Use the translation vector tx and ty and compute the new position using x1 = x – tx , y1 = y – ty and display scaled figure in its original position (i.e) the initial position before translation

Result:

Thus a C program to perform 2D composite transformations was written and executed.

Ex.No.5 IMPLEMENTATION OF COHEN SUTHERLAND LINE CLIPPING ALGORITHM

Aim:

To write a C program to implement Cohen Sutherland line clipping algorithm.

Algorithm:

1. Read two end points of the line say P1(x2-y2).
2. Read two corners (left-top and right-bottom) of the window, say (Wx1, Wy1and Wx2, Wy2).
3. Assign the region codes for two endpoints P1 and P2 using following steps :
4. Initialize code with bits 0000
    a) Set Bit 1- if (x<Wx1)
    b) Set Bit 2- if (x>Wx2)
    c) Set Bit 3- if (y<Wy2)
    d) Set Bit 4- if (y>Wy1)
5. Check for visibility of line P1 and P2
    a. If region codes for both endpoints P1and P2 are zero then the line is completely visible. Hence draw the line.
    b. If region codes for endpoints are not zero and the logical ending of them is also nonzero then the line is completely invisible, so reject the line.
    c. If region codes for two end point do not satisfy the conditions in 4a) and 4b)the line is partially visible.
6. Determine the intersecting edge of the clipping window by inspecting the region codes of two end points.

a. If region codes for both end points are non-zero, find intersection points P1 and P2 with boundary edges of clipping window with respect to it.

b. If region code for any one end point is non zero then find intersection points P1 or P2 with the boundary edge of the clipping window with respect to it.

7. Intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation.

   m=(y2-y1)/(x2-x1)

   a. The y coordinate of the intersection point at vertical line

      y=y1+m(x-x1)

   b. The x coordinate of the intersection point at horizontal

      line x=x1+(y-y1)/m

8. Divide the line segment considering intersection points.

9. Reject the line segment if any one end point of it appears outsides the clipping window.

10. Draw the remaining line segments.

Result:

Thus a C program to implement Cohen Sutherland line clipping algorithm was written and executed.

EX.NO: 6 **IMPLEMENTATION OF 3D TRANSFORMATIONS**

Aim:

To perform the various 3-Dimensional Transformations such as translation, scaling, rotation, shearing, etc.

Algorithm:

1. Input the object coordinates.

2. Display the menu as 1.Translation 2.Scaling 3.Rotation 4.Exit

3. Get the choice from the user.

4. If the choice is 1,an object is translated from position P to position P' with the operation P'=T.P where tx,ty and tz specifying translation distances.

x'=x+ tx

y'=y+ ty

z'=z+ tz

5. If the choice is 2, the scaling transformation of a position P can be written as P'=S.P where scaling parameters sx,sy and sz are assigned any positive values.

x'=x.sx

y'=y.sy

z'=z.sz

6. If the choice is 3 get the rotation angle. Rotate the figure with respect to the axis of rotation.

a. About z axis rotation

x'=xcosϴ-ysinϴ

y'=xsinΘ+ycosΘ

z'=z

Rotation can be expressed as P'=Rz(Θ).P

b. About x axis rotation

y'=ycosΘ-zsinΘ

z'=ysinΘ+zcosΘ

x'=x

Rotation can be expressed as P'=Rx(Θ).P

c. About y axis rotation

z'=zcosΘ-xsinΘ

x'=zsinΘ+xcosΘ

y'=y

Rotation can be expressed as P'=Ry(Θ).P

7. If choice is 4 exit the program.

Result:

Thus a C Program to implement 3D transformations such as translation, rotation and scaling was written and executed.

Ex. No: 7 **IMPLEMENTATION OF 3D PROJECTION**

AIM:

Write a ‘c’ program to perform projection of 3D image.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare the variables using structure.

STEP 3: Create the function and get the function using create() function.

STEP 4: Get the coordinates and assign the values for the coordinates.

STEP 5: Define the drawcube() function.

STEP 6: Using rotate() function, rotation is performed.

STEP 7: End the program.

RESULT:

Thus the c program for visualizing projections of 3D images using graphics was successfully executed.

Ex.No.8 **THREE DIMENSIONAL SCENES USING OpenGL**

Aim:

To generate 3D scene using OpenGL.

Algorithm:

1. Declare necessary Open Graphics Library (OpenGL) header files.
2. Initialize graphics utility toolkit using glutInit() function.
3. Set the display mode of window by calling glutInitDisplayMode() function.
4. Using glutInitWindowSize(), set the size of the window.
5. Create a window using glutCreateWindow() function.
6. Register the callback function “display” using glutDisplayFunc(display) function.
7. Register the callback function “keyboard” using glutKeyboardFunc(keyboard) function.
8. Draw the scene by calling below function:
    a. glutSolidTeapot() – To draw solid teapot
    b. glutWireTeapot() - To draw wire-frame teapot
    c. glutSolidCube() – To draw solid cube
    d. glutSolidSphere() – To draw solid sphere
9. Use various transformation, projection, shading and lighting functions to add realism for the 3D object (teapot).

Result:

Thus generation of 3D scene using the OpenGL has been executed and verified.

Ex. No: 9 **BASIC OPERATIONS ON IMAGE USING PHOTOSHOP**

AIM:

The main objective is to perform basic operations on image using Photoshop.

PROCEDURE:

The Brush Popup Palette

Allow quick, convenient access to a range of standard, preset brushes.

The Brushes Palette

Provides access to a wide variety of options for controlling the appearance and characteristics of brush.

The Pencil Tool

Used to draw free form lines. These draws with the foreground color.

The Gradient Tool

Used to color pixels with the foreground color based on tolerance setting.

The Paint Bucket Tool

Used to color pixels with the foreground color on the tolerance setting.

Cropping An Image

Digital editors are used to crop images. Cropping creates a new image by selecting a desired rectangular portion from the image being cropped. The unwanted part of the image is discarded. Image cropping does not reduce the resolution of the area cropped. A primary reason for cropping is to improve the image composition in the new image. Removal Of Unwanted Elements Most image editors can be used to remove unwanted branches, etc., using a "clone" tool.

Image Orientation

Image orientation (from left to right): original, -30° CCW rotation, and flipped.

Sharpening And Softening Images

Graphics programs can be used to both sharpen and blur images in a number of ways, such as unsharp masking or deconvolution.

Editing

- Open your project file and create a duplicate.
- Crop the image using crop tool.
- Change the image size using canvas technique.

Back Ground Changing

- Select the area to change the back ground using magic wand tool.
- Select the back ground image for your image.
- Move the shape of the back ground using marquee tool.
- Using selection tool, move the back ground.

Changing Color

- Select the area using Lasso tool .
- Go to image tab and adjustments and select the Hue / saturation option.

Change the color using RGB mode

RESULT:

The image editing was done using Photoshop and the result is as shown above.

Ex. No: 10 CREATING AN ANIMATION SCENE USING MICROMEDIA FLASH

AIM:

To perform animation using any animation software.

ALGORITHM:

MOTION TWEENING

Step 1: Create an object in the first layer of first key frame.

Step 2: Create the last key frame and move the object to it.

Step 3: Right click on the first key frame and select create motion tween.

Step 4: Play the picture.

SHAPE TWEENING

Step 1: Create an object in the first layer of first key frame.

Step 2: Create the last key frame.

Step 3: In first key frame's properties and select shape tween.

Step 4: Change the shape of the object at the last frame.

Step 5: Play the picture.

GUIDE LAYER

Step 1: Create an object in the first layer of first key frame.

Step 2: Create the last key frame and move the object to it.

Step 3: Right click on the first key frame and select create motion tween.

Step 4: Right click on the object's layer select add motion guide.

Step 5: In motion guide draw your guide

line. Step 6: Play the picture.

MASKING

Step 1: Create an object in the first layer of first key frame

Step 2: Add new layer and draw the shape of the view.

Step 3: Add the motion guide to the view tool.

Step 4: Right click on the second layer (view layer) and select

mask. Step 5: Play the picture

RESULT:

The animation was done using flash and the the result is as shown above.