1. What is OPENGL?

- OpenGL (Open Graphics Library) is the computer industry's standard application program interface (API) for defining 2-D and 3-D graphic images.
- OpenGL comes with a large number of built-in capabilities requestable through the API.
- These include hidden surface removal, alpha blending (transparency), antialiasing , texture mapping, pixel operations, viewing and modelling transformations, and atmospheric effects (fog, smoke, and haze).

2. What is the purpose / meaning of following APIs in OpenGL?

a. glClearColor(0.0, 1.0, 0.0, 1.0);

- specify clear values for the color buffers.
- void **glClearColor**(GLclampf *red*, GLclampf *green*, GLclampf *blue*, G Lclampf *alpha*);

b. glClear(GL_COLOR_BUFFER_BIT); //Clearing the screen …clrscr()

- clear buffers to pre-set values
- void **glClear**(GLbitfield *mask*);
- The three masks are as follows:
  GL_COLOR_BUFFER_BIT: Indicates the buffers currently enabled for color writing.
  GL_DEPTH_BUFFER_BIT: Indicates the depth buffer.
  GL_STENCIL_BUFFER_BIT: Indicates the stencil buffer.

c. gluOrtho2D(-1.0,1.0,-1.0,1.0); or gluOrtho2D(-100,100,-100,100);

- define a 2D orthographic projection matrix
- void gluOrtho2D(GLdouble **left**, GLdouble **right**, GLdouble **bottom**, GLdouble **top**);
  left, right: Specify the coordinates for the left and right vertical clipping planes.
  bottom, top: Specify the coordinates for the bottom and top horizontal clipping planes.

d. glVertex2f(-0.5, -0.5);

- specify a vertex
- void glVertex2f(GLfloat x,GLfloat y);

- Specify *x, y* coordinates of a vertex. Not all parameters are present in all forms of the command.

e. glVertex2i(-5, -5)
- specify a vertex
- void glVertex2i(GLint x, GLint y);
- Specify *x, y* coordinates of a vertex. Not all parameters are present in all forms of the command.

f. glBegin(GL_LINE); …. glEnd();
- Delimit the vertices of a primitive or a group of like primitives.
- Treats each pair of vertices as an independent line segment. Vertices 2 n - 1 and 2 n define line n. N 2 lines are drawn

g. glBegin(GL_POLYGON); …. glEnd();
- Delimit the vertices of a primitive or a group of like primitives
- Draws a single, convex polygon. Vertices 1 through N define this polygon.

h. glutInit(&argc, argv);
- used to initialize the GLUT library.
- void glutInit(int *argcp, char **argv);

argcp:
A pointer to the program's *unmodified* argc variable from main. Upon return, the value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library.
argv:
The program's *unmodified* argv variable from main. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

i. glutInitDisplayMode(GLUT_SINGLE);
- sets the *initial display mode*.
- void glutInitDisplayMode(unsigned int mode);

GLUT_SINGLE:
Bit mask to select a single buffered window. This is the default if neither GLUT_DOUBLE or GLUT_SINGLE are specified.

j. glutInitWindowSize(500,500);
- set the *initial window size* respectively.
- void glutInitWindowSize(int width, int height);

width: Width in pixels.
height: Height in pixels.

k. glutInitWindowPosition(100,100);
- set the *initial window position* respectively.
- void glutInitWindowPosition(int x, int y);

x: Window X location in pixels.
y: Window Y location in pixels.

l. glutCreateWindow("Menu Driven Program");
- creates a top-level window.
- int glutCreateWindow(char *name);
  name: ASCII character string for use as window name.

m. glutDisplayFunc(myLine);
- sets the display callback for the *current window*.
- void glutDisplayFunc(void (*func)(void));

func: The new display callback function.

n. glColor3f(1.0, 1.0, 1.0);
- set the current color
- void glColor3f(GLfloat *red*, GLfloat *green*, GLfloat *blue*);

o. glutMouseFunc(mymouse);
- sets the mouse call-back for the *current window*.
- When a user presses and releases mouse buttons in the window, each press and each release generate a mouse call-back.
- void glutMouseFunc(void (*func)(int button, int state, int x, int y));
  func: The new mouse callback function.

p. void mymouse(int button, int state, int x, int y)
- The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON.

- The state parameter is either GLUT_UP or GLUT_DOWN indicating whether the call-back was due to a release or press respectively.
- The x and y call-back parameters indicate the window relative coordinates when the mouse button state changed

q. glutKeyboardFunc(key); //void key(unsigned char key)

- sets the keyboard callback for the *current window*.
- void glutKeyboardFunc(void (*func)(unsigned char key,int x, int y));

  func: The new keyboard callback function.

3. Which command is used to compile openGL Program?

- g++ main.c -lglut -lGL -lGLEW -lGLU -o OpenGLExample

Installation*: *https://www.wikihow.com/Install-Mesa-(OpenGL)-on-Linux-Mint

4. Which APIs are used to get maximum values of x and y extent of openGL window?

xmax=glutGet(GLUT_WINDOW_WIDTH);
ymax=glutGet(GLUT_WINDOW_HEIGHT);

5. What is scan conversion?

- It is a process of representing graphics objects a collection of pixels. The graphics objects are continuous.
- The pixels used are discrete.
- Each pixel can have either on or off state.
- Examples of objects which can be scan converted

1. Point 2. Line 3. Sector 4. Arc 5. Ellipse	6. Rectangle 7. Polygon 8. Characters 9. Filled Regions

- Advantage:

a. Algorithms can generate graphics objects at a faster rate.
b. Using algorithms memory can be used efficiently.
c. Algorithms can develop a higher level of graphical objects.

6. What are the benefits and drawbacks of DDA line drawing algorithm?

Benefits:

- It is a simple algorithm.
- It is easy to implement.
- It avoids using the multiplication operation which is costly in terms of time complexity.

Drawbacks:

- There is an extra overhead of using round off() function.
- Using round off() function increases time complexity of the algorithm.
- Resulted lines are not smooth because of round off() function.
- The points generated by this algorithm are not accurate.

7. What are the benefits and drawbacks of Bresenham's line drawing algorithm?

The advantages of Bresenham Line Drawing Algorithm are-

- It is easy to implement.
- It is fast and incremental.
- It executes fast but less faster than DDA Algorithm.
- The points generated by this algorithm are more accurate than DDA Algorithm.
- It uses fixed points only.

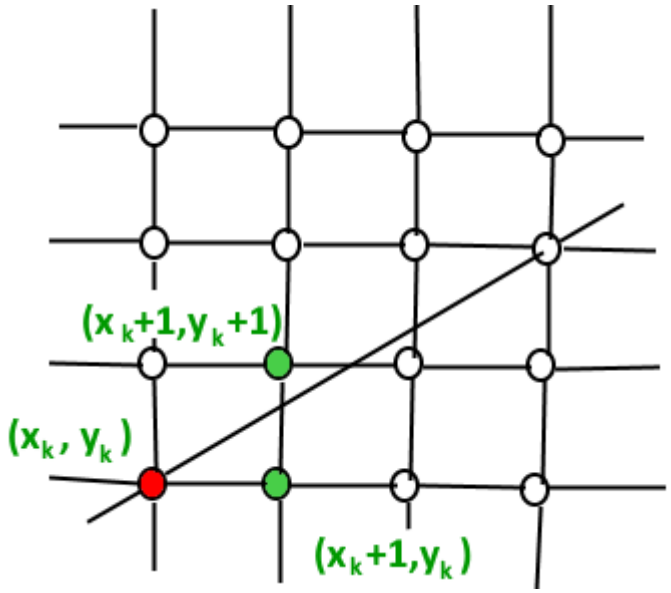
The disadvantages of Bresenham Line Drawing Algorithm are-

- Though it improves the accuracy of generated points but still the resulted line is not smooth.
- This algorithm is for the basic line drawing.
- It can not handle diminishing jaggies.

8. What is decision parameter in Bresenham's line drawing algorithm? How it is useful?

It is used to make decision for selection of next pixel point in both the phases.

We always increase x by 1, and we choose about next y, whether we need to go to y+1 or remain on y. In other words, from any position (X_k, Y_k) we need to choose between $(X_k + 1, Y_k)$ and $(X_k + 1, Y_k + 1)$.



We would like to pick the y value (among $Y_k + 1$ and Y_k) corresponding to a point that is closer to the original line.
We need to a decision parameter to decide whether to pick $Y_k + 1$ or Y_k as next point.

9. What is decision parameter in Bresenham's / Mid-point Circle drawing algorithm? How it is useful?

In Bresenham's algorithm at any point (x, y) we have two option either to choose the next pixel in the east i.e. (x+1, y) or in the south east i.e. (x+1, y-1). And this can be decided by using the decision parameter as:

- If d > 0, then (x+1, y-1) is to be chosen as the next pixel as it will be closer to the arc.
- else (x+1, y) is to be chosen as next pixel.

In Short - Bresenham Circle Algorithm makes the use of decision parameter to calculate the next pixel location from a previously known pixel location (x, y). In Bresenham's algorithm at any point (x, y) we have two option to choose the next pixel in the X–direction either (x+1, y) or (x+1, y-1).

10. What is eight-way Symmetry? How it is beneficial in Circle drawing algorithm?

Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants. In each quadrant, there are two **octants**. If the calculation of the point of one octant is done, **then the other seven points can be calculated easily by using the concept of eight-way symmetry.**

11. Benefits / Drawbacks of Boundary Fill.

Benefits of Boundary Fill Algorithm

1. This method is good for a polygon with the multicolour area and single-coloured boundary.
2. In this method, pixels may be visited more than once.
3. Memory consumption is relatively low in Boundary-fill algorithm.

Drawbacks of Boundary Fill Algorithm

1. If the polygon is having boundaries with different colours then boundary fill algorithm fails.
2. The algorithm takes time and memory because some recursive calls are needed.
3. It is a complicated algorithm.
4. It may not fill regions sometimes correctly when some interior pixel is already filled with colour. The algorithm will check this boundary pixel for filling and will found already filled so recursive process will terminate. This may vary because of another interior pixel unfilled. So check all pixels colour before applying the algorithm.

12**. Benefits / Drawbacks of Flood Fill**.

Benefits of Flood Fill Algorithm

1. It is an easy way to fill colour in the graphics. One just takes the shape and starts flood fill.
2. When boundary is of many colours and interior is to be filled with one colour, then this algorithm is suitable.
3. Flood-fill algorithms are simple and efficient.

Drawbacks of Flood Fill Algorithm

1. Very slow algorithm.

2. May be fail for large polygons.
3. Initial pixel required more knowledge about surrounding pixels.
4. It requires huge amount of memory.

13. **Compare - Boundary and Flood Fill**.

Criteria	**Boundary Fill Algorithm**	**Flood Fill Algorithm**
Basic	It defines the area with a single colour.	It can have an area containing several colours.
Painting Process	Interior points are coloured by continuously searching for the boundary colour.	A random colour can be used to colour the interior portion then the old one is replaced with a new one.
Memory Consumption	Low.	High.
Speed	Fast.	Comparatively slower.
Algorithm complexity	Complicated.	Relatively simple.

14. **Compare - Scan Line VS Boundary / Flood Fill.**

Criteria	**Scan Line Algorithm**	**Boundary Fill Algorithm**
Basic	It can have an area containing several colours.	It defines the area with a single colour.
Painting Process	It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence.	Interior points are coloured by continuously searching for the boundary colour.
Memory Consumption	Relatively low.	Low.
Speed	Comparatively fast.	Fast.

Criteria	**Scan Line Algorithm**	**Flood Fill Algorithm**
Basic	It can have an area containing several colours.	It can have an area containing several colours.
Painting Process	It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence.	A random colour can be used to colour the interior portion then the old one is replaced with a new one.

Memory Consumption	Relatively low.	High.
Speed	Comparatively fast.	Comparatively slower.

15. Which API have been used to read pixel colors in polygon filling program?

16. What is viewport? What is window? What is clipping?

Ans. Viewport:

An area on display device to which a window is mapped [where it is to displayed].

Basically, the window is an area in object space. It encloses the object. After the user selects this, space is mapped on the whole area of the viewport. Almost all 2D and 3D graphics packages provide means of defining viewport size on the screen. It is possible to determine many viewports on different areas of display and view the same object in a different angle in each viewport.

Window:

A window is a separate viewing area on a computer display screen in a system that allows multiple viewing areas as part of a graphical user interface (GUI). Windows are managed by a windows manager as part of a windowing system . A window can usually be resized by the user.

Clipping:

When we have to display a large portion of the picture, then not only scaling & translation is necessary, the visible part of picture is also identified. This process is not easy. Certain parts of the image are inside, while others are partially inside. The lines or elements which are partially visible will be omitted.

For deciding the visible and invisible portion, a particular process called clipping is used. Clipping determines each element into the visible

and invisible portion. Visible portion is selected. An invisible portion is discarded.

17. What are main steps in Cohen Sutherland line clipping?

Ans. **Step 1 :** Assign a region code for two endpoints of given line

Step 2 : If both endpoints have a region code 0000 then given line is completely inside and we will keep this line

Step 3 : If step 2 fails, perform the logical AND operation for both region codes.

Step 3.1 : If the result is not 0000, then given line is completely outside.

Step 3.2 : Else line is partially inside.

Step 3.2.a : Choose an endpoint of the line that is outside the given rectangle.

Step 3.2.b : Find the intersection point of the rectangular boundary (based on region code).

Step 3.2.c : Replace endpoint with the intersection point and update the region code.

Step 3.2.d : Repeat step 2 until we find a clipped line either trivially accepted or rejected.

Step 4 : Repeat step 1 for all lines

18. What is four-bit region code in Cohen Sutherland line clipping?

19. What are applications of clipping?

Ans.

a. It will extract part we desire.
b. For identifying the visible and invisible area in the 3D object.

c. For creating objects using solid modeling.
d. For drawing operations.
e. Operations related to the pointing of an object.
f. For deleting, copying, moving part of an object.

20.What are applications of 2D transformation?

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

21. What is the significant of homogeneous coordinate systems?

- Homogeneous coordinates are ubiquitous in computer graphics because they allow common vector operations such as translation, rotation, scaling and perspective projection to be represented as a matrix by which the vector is multiplied.
- By the chain rule, any sequence of such operations can be multiplied out into a single matrix, allowing simple and efficient processing.
- Modern OpenGL and Direct3D graphics cards take advantage of homogeneous coordinates to implement a vertex shader efficiently using vector processors with 4-element registers

22 .Main steps in composite transformations?

The enlargement is with respect to center. For this following sequence of transformations will be performed and all will be combined to a single one

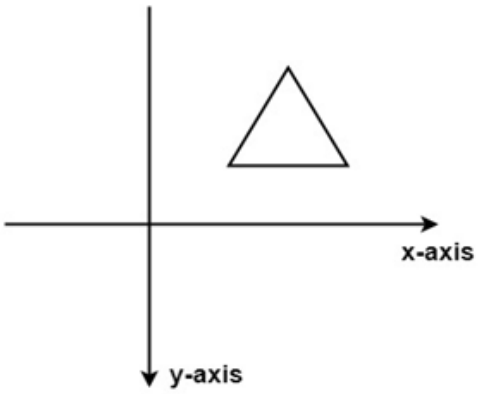
Step1: The object is kept at its position as in fig (a)

Step2: The object is translated so that its center coincides with the origin as in fig (b)

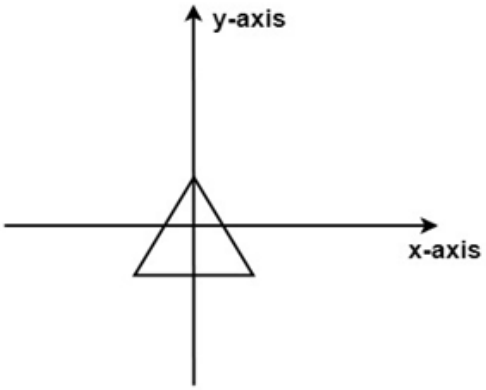
Step3: Scaling of an object by keeping the object at origin is done in fig (c)

Step4: Again translation is done. This second translation is called a reverse translation. It will position the object at the origin location.

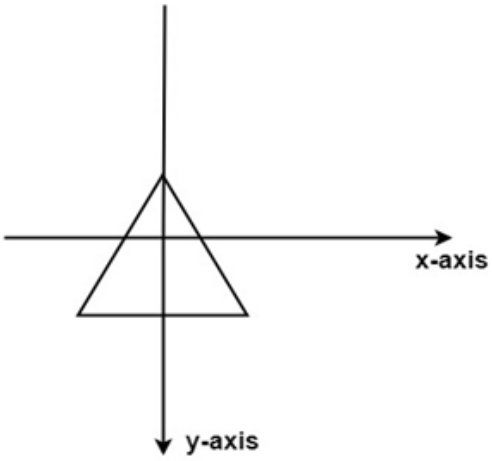
Above transformation can be represented as $T_V.ST_V^{-1}$



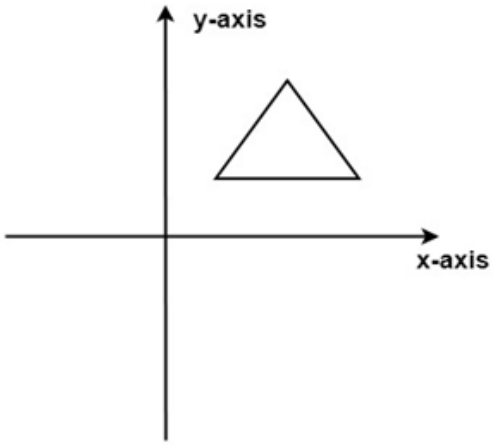
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

23.Which are basic and advanced transformations?

- Basic transformations:

1) Translation
2) Scaling
3) Rotation

- Advanced transformation

1) Reflection
2) Shearing