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

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

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

**4. Define Computer graphics.**
Computer graphics remains one of the most existing andrapidly growing computer fields. Computer graphics maybe defined as a pictorial representation or graphicalrepresentation of objects in a computer.

**5. Name any four input devices?**
Four input devices are keyboard, mouse, image scanners,and trackball.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**21. What is persistence?**
The time it takes the emitted light from the screen to decay one tenth of its original intensity is called as persistence.
 **22. What is Aspect ratio?**
The ratio of vertical points to the horizontal points necessary to produce length of lines in both directions of the screen is called the Aspect ratio. Usually the aspect ratio is ¾.

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

**24. Define pixel?**
Pixel is shortened forms of picture element. Each screen point is referred to as pixel or pixel.

**25. What is frame buffer?**
Picture definition is stored in a memory area called frame buffer or refresh buffer.

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

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

**28. What is point in the computer graphics system?**
The point is a most basic graphical element & is completely defined by a pair of user coordinates (x, y).

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

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

**31. What are the various attributes of a line?**
The line type, width and color are the attributes of the line.The line type include solid line, dashed lines, and dotted lines.

**32. What is anti aliasing?**
The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called anti aliasing.

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

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

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

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

**37. What is shearing?**
The shearing transformation actually slants the object along the X direction or the Y direction as required. ie; this transformation slants the shape of an object along a required plane.

**38. What is reflection?**
The reflection is actually the transformation that produces a mirror image of an object. For this use some angles and lines of reflection.

**39. What are the two classifications of shear transformation?**
X shear, y shear

**40. A point (4,3) is rotated counterclockwise by an angle of 45°. Find the rotation matrix and the resultant**
point

**41. Name any three font editing tools.**
ResEdit, FONTo grapher

**42. Differentiate serif and sans serif fonts.**
Give one example Serif fonts has a little decoration at the end of the letter,but serif font has not. Times, new century schoolbook is the examples of serif fonts. Arial, optima are examples for sanserif fonts.

**43. Distinguish between window port & view port?**
A portion of a picture that is to be displayed by a window is known as window port. The display area of the part selected or the form in which the selected part is viewed is known as view port.

**44. Define clipping?**
Clipping is the method of cutting a graphics display to neatly fit a predefined graphics region or the view port.

**45. What is the need of homogeneous coordinates?**
To perform more than one transformation at a time, use homogeneous coordinates or matrixes. They reduce unwanted calculations intermediate steps saves time and memory and produce a sequence of transformations.

**46. Write The Important Applications Of Computer Graphic?**

**Answer :**

Following are the applications of computer graphic
1. computer graphics is used in the field of computer aided design.
2. It is used to produce illustrations for reports or to generate slide for with projections.
3. Computer graphic methods are widely used in both fine are and commercial are applications.
4. The artist uses a combination of 3D modeling packages, texture mapping, drawing programs and CAD software.
5. In the field of entertainment CG methods are now commonly used in making motion pictures. music videos and television shows.
6. Computer-generated models of physical, financial and economic systems are often used as educational aids.

 **47. What Are The Raster And Vector Graphics?**

**Answer :**

The Raster and Vector graphics can be explained as-
**RASTER**- In computer graphics image, or BITMAP, is a dot matrix data structure representing a generally rectangular grid of pixels or points of color, viewable via a monitor, paper, or other display medium. Raster image are stored in image files with varying formats.
**VECTOR**- Vector graphics is the use of geometrical primitives such as points, lines, curves, and shapes or polygon, which are all based on mathematical expressions, to represent image in computer graphics. “Vector”, in this context, implies more than a straight line.

**48. Write The Difference Between Vector And Raster Graphics?**

**Answer :**

Following are the differences between vector and raster graphics-
1. Raster or Bitmap images are resolution dependent because of this its not possible to increase or decrease their size without sacrificing on image quality.
While vector based image are not dependent on resolution. The size of vector image image can be increased or decreased without affecting image quality.
2. Raster or bitmap images are always rectangular in shape, Vector image, however, can have any shape.
3. Unlike raster image, vector image can’t be use for realistic images. This is because vector images are made up of solid color areas and mathematical gradients, so they can’t be used to show continuous tones of a colors in a natural photograph.

**49. What Is Scaling In Computer Graphics?**

**Answer :**

In computer graphic, image scaling is the process of resizing a digital image. scaling is a non-trivial process that involves a trade off between efficiency, smoothness and sharpness. With bitmap graphics, as the size of an image is reduced or enlarged, the pixels which comprise the image become increasingly visible, making the image appear “soft” if pixels are averaged, or jagged if not.

1. **. What Are The Hardware Devices Used For Computer Graphics?**

**Answer :**

The hardwares devices used for the computer graphics are

**Input Devices:** Keyboard, Mouse, Data tablet, Scanner, Light pen, Touch screen, Joystick
**Output Devices:** Raster Devices- CRT, LCD, LED, Plasma screens, Printers,Vector Devices- Plotters, Oscilloscopes

**51. What Are The Features Of Inkjet Printers?**

**Answer :**

Features of inkjet printers are
1. They can print 2 to 4 pages per minute.
2. resolution is about 360d.p.i. Therefor better print quality is achieved.
3. The operating cost is very low. The only part that requires replacement is ink cartridge.
4. Four colors cyan, yellow, magenta, black are available.

**52. Define Random And Raster Scan Displays?**

**Answer :**

Random scan is a method in which display is made by electronic beam, which is directed only to the points or parts of the screen where picture is to be drawn.

The Raster scan system is a scanning technique in which the electron sweep from top to bottom and from left to right. The intensity is turned on or off to light and un-light the pixel.

**53. Define Computer graphics?**Computer graphics are graphics created by computers and, more generally, the representation and manipulation of pictorial data by a computer

**54. Define Computer Animation?**Computer animation is the art of creating moving images via the use of computers. It is a subfield of computer graphics and animation

**55. Define Pixel?**The word pixel is based on a contraction of pix (“pictures”) and el (for “element”). Pixels are normally arranged in a 2-dimensional grid, and are often represented using dots, squares, or rectangles

**56. Define Raster graphics?**Raster Graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images.

**57. Define Image?**An image (from Latin imago) is an artifact, usually two-dimensional (a picture), that has a similar appearance to some subject—usually a physical object or a person.

**58. Define Rendering?**Rendering is the process of generating an image from a model, by means of computer programs.

**59. Define Ray Tracing?**Ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane.

**60. Define Projection?**Projection is a method of mapping points to a plane.

**61. Define 3D Projection?**3D projection is a method of mapping three dimensional points to a two dimensional plane.

**62. Define Texture Mapping?**Texture mapping is a method for adding detail, surface texture, or color to a computer-generated graphic or 3D model.

**63. Define Vector Graphics?**Vector graphics formats are complementary to raster graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images.

**64. Define Pinhole camera model?**The pinhole camera model describes the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light.

**65. Define digital image?**A digital image is a representation of a two-dimensional image using ones and zeros (binary). Without qualifications, the term “digital image” usually refers to raster images.

**66. Define Grayscale?**A grayscale or greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information.

**67. Define CCD?**A charge-coupled device (CCD) is an analog shift register that enables the transportation of analog signals (electric charges) through successive stages (capacitors), controlled by a clock signa.

**68. Define Image Resolution?**Image resolution describes the detail an image holds. The term applies equally to digital images, film images, and other types of images. Higher resolution means more image detail.

**69. Define Wavelength?**Wavelength is the distance between repeating units of a propagating wave of a given frequency.

**70. What is an Electromagnetic Spectrum?**The electromagnetic (EM) spectrum is the range of all possible electromagnetic radiation frequencies. The “electromagnetic spectrum” (usually just spectrum) of an object is the characteristic distribution of electromagnetic radiation from that particular object.

**71. Define Halftone?**Halftone is the reprographic technique that simulates continuous tone imagery through the use of dots, varying either in size or in spacing. ‘Halftone’ can also be used to refer specifically to the image that is produced by this process.

**72. Define WCS?**World Coordinate Systems. While every point in the FITS data array can be located in the coordinate system determined by the array axes. World Coordinate Systems (WCS) are any coordinate systems that describe the physical coordinate associated with a data array, such as sky coordinates.

**73. What is OpenGL?**OpenGL(R) is the software interface for graphics hardware that allows graphics programmers to produce high-quality color images of 3D objects. OpenGL is a rendering only, vendor neutral API providing 2D and 3D graphics functions, including modeling, transformations, color, lighting, smooth shading, as well as advanced features like texture mapping, NURBS, fog, alpha blending and motion blur. OpenGL works in both immediate and retained (display list) graphics modes.
OpenGL is window system and operating system independent. OpenGL has been integrated with Windows NT and with the X Window System under UNIX. Also, OpenGL is network transparent. A defined common extension to the X Window System allows an OpenGL client on one vendor’s platform to run across a network to another vendor’s OpenGL server

**74. What hardware supports the OpenGL API?**Many vendors have developed or are developing implementations of the OpenGL API for a variety of embedded hardware devices including aircraft avionics, PDAs (personal digital assistants such as PalmTM), cellular phones, game consoles (Sony Playstation® 2), television set-top boxes, and display devices (X-Terms and network computers). The small size of the OpenGL API, its open nature, and now free use of the sample implementation make the OpenGL API an ideal graphics library for these types of applications.

**75. What are the benefits of OpenGL for hardware and software developers?**∗ Industry standard
∗ Reliable and portable.
∗ Evolving.
∗ Scalable.
∗ Easy to use.
∗ Well-documented.

**76. What is the GLUT Toolkit?**GLUT is a portable toolkit which performs window and event operations to support OpenGL rendering

**77. How does the camera work in OpenGL?**As far as OpenGL is concerned, there is no camera. More specifically, the camera is always located at the eye space coordinate (0.0, 0.0, 0.0). To give the appearance of moving the camera, your OpenGL application must move the scene with the inverse of the camera transformation by placing it on the MODELVIEW matrix. This is commonly referred to as the viewing transformation. In practice this is mathematically equivalent to a camera transformation but more efficient because model transformations and camera transformations are concatenated to a single matrix. As a result though, certain operations must be performed when the camera and only the camera is on the MODELVIEW matrix. For example to position a light source in world space it most be positioned while the viewing transformation and only the viewing transformation is applied to the MODELVIEW matrix.

**78. How do I implement a zoom operation?**A simple method for zooming is to use a uniform scale on the ModelView matrix. However, this often results in clipping by the zNear and zFar clipping planes if the model is scaled too large. A better method is to restrict the width and height of the view volume in the Projection matrix. For example, your program might maintain a zoom factor based on user input, which is a floating-point number. When set to a value of 1.0, no zooming takes place. Larger values result in greater zooming or a more restricted field of view, while smaller values cause the opposite to occur.

**79. How do I get a specified point (XYZ) to appear at the center of the scene?**gluLookAt() is the easiest way to do this. Simply set the X, Y, and Z values of your point as the fourth, fifth, and sixth parameters to gluLookAt ().

**80. Define Instruction Pipeline?**An instruction pipeline is a technique used in the design of computers and other digital electronic devices to increase their instruction throughput (the number of instructions that can be executed in a unit of time).

30. Define Geometric Pipeline?
Geometric manipulation of modeling primitives, such as that performed by a Geometry Pipeline, is the first stage in computer graphics systems which perform image generation based on geometric models.

**81. Define Perspective Projection?**Perspective (from Latin perspicere, to see through) in the graphic arts, such as drawing, is an approximate representation, on a flat surface (such as paper), of an image as it is perceived by the eye. The two most characteristic features of perspective are that objects are drawn.

**82. Can you give some basic features of computer graphics?**
**Ans.** The salient feature of computer graphics is the creation and manipulation of graphics (artificial images) by computer.

**83. Can you tell which major components (hardware and software) are needed for computer graphics?**
**Ans.** Besides the basic computer, some special devices and software may be required especially for computer graphics. For hardware, a special high-resolution, color monitor is often demanded and some input tools, e.g. mouse and joy-sticker, and hard-copy devices, e.g. high-resolution color printer, may be required. For software, some special purpose utilities (device-dependent and device-independent) are needed for handling processing in computer graphics.

**84. What does refreshing of the screen mean?**
**Ans.** Some method is needed for maintaining the picture on the screen. Refreshing of screen is done by keeping the phosphorus glowing to redraw the picture repeatedly. i.e. by quickly directing the electronic beam back to the same points.

**85.** **Explain the merits and demerits of Penetration techniques.**
**Ans.** The merits and demerits of the Penetration techniques are as follows:

1. It is an inexpensive technique.
2. It has only four colors.
3. The quality of the picture is not good when it is compared to other techniques.
4. It can display color scans in monitors.

**86. Explain the merits and demerits of DVST.**
**Ans.** The merits and demerits of direct view storage tubes (DVST) are as follows:

1. It has a flat screen.
2. Refreshing of screen is not required.
3. Selective or part erasing of screen is not possible.
4. It has poor contrast.
5. Performance is inferior to the refresh CRT.

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

**88.** **Explain the merits and demerits of Plasma panel display.**
**Ans.** **ADVANTAGES:**

1. Refreshing is not required.
2. Produce a very steady image free of Flicker.
3. Less bulky than a CRT.

**DISADVANTAGES:**

1. Poor resolution of up to 60 d.p.i.
2. It requires complex addressing and wiring.
3. It is costlier than CRT.

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

**90.** **What is meant by Addressability?**
**Ans.** Addressability is the number of individual dots per inch (d.p.i.) that can be created. If the address of the current dot is (x, y) then the next dot will be (x + y), (x + y + 1) etc.

**91.** **What is a dot size?**
**Ans.** Dot size may be defined as the diameter of a single dot on the devices output. Dot size is also called as the Spot size.

**92.** **What is interdot distance?**
**Ans.** Interdot distance is the reciprocal of addressability. If the addressability is large, the interdot distance will be less. The interdot distance should be less to get smooth shapes.

**93.** **Explain the differences between a general graphics system designed for a programmer and one designed for a specific application, such as architectural design?**
**Ans.** Basically, packages designed for graphics programming contain functions for setting primitives, attributes, and parameters for various graphics operations such as viewing and transformations. Packages designed for applications allow a user to create scenes in terms of the particular application, rather than in terms of graphics functions.

**94.** **Consider three different raster systems with resolutions of 640 x 480, 1280 x 1024 and 2560 x 2048.**

1. **What size is frame buffer (in bytes) for each of these systems to store 12 bits per pixel?**
2. **How much storage (in bytes) is required for each system if 24 bits per pixel are to be stored?**

**Ans.** **1.**Because eight bits constitute a byte, frame-buffer sizes of the systems are as follows:
640 x 480 x 12 bits π 8 = 450KB
1280 x 1024 x 12 bits π 8 = 1920KB
2560 x 2048 x 12 bits π 8 = 7680KB
**2.** Similarly, each of the above results is just doubled for 24 (12x2) bits of storage per pixel.

**95.** **Consider two raster systems with the resolutions of 640 x 480 and 1280 x 1024.**

1. **How many pixels could be accessed per second in each of these systems by a display controller that refreshes the screen at a rate of 60 frames per second?**
2. **What is the access time per pixel in each system?**

**Ans. 1.** Since 60 frames are refreshed per second and each frame consists of 640 x 480 pixels, the access rate of such a system is:
(640 x 480) \* 60 = 1.8432 x 107 pixels/second.
Likewise, for the 1280 x 1024 system, the access rate is:
(1280 x 1024) \* 60 = 7.86432 x 107 pixels/second.

1. According to the definition of access rate, we know that the access time per pixel should be 1/(access rate). Therefore, the access time is around 54 nanoseconds/pixel for the 640 x 480 system, and the access time is around 12.7 nanoseconds/pixel for the 1280 x 1024 system.

**96.** **Consider a raster system with the resolution of 1024 x 768 pixels and the color palette calls for 65,536 colors. What is the minimum amount of video RAM that the computer must have to support the above-mentioned resolution and number of colors?**
**Ans.** Recall that the color of each pixel on a display is represented with some number of bits. Hence, a display capable of showing up to 256 colors is using 8 bits per pixels (i.e., “8-bit color”).
Notice, first that the color palette calls for 65,536 colors. This number is but 216, which implies that 16 bits are being used to represent the color of each pixel on the display. The display’s resolution is 1024 by 768 pixels, which implies that there is a total of 786,432 (1024 × 768) pixels on the display. Hence, the total number of bits required to display any of 65,536 colors on each of the screen’s 786,432 pixels is 12,582,912 (786,432 × 16). Dividing this value by 8 yields an answer of 1,572,864 bytes. Dividing that value by 1,024 yields an answer of 1,536 KB. Dividing that value by 1,024 yields an answer of 1.5 MB.

**97.What is OpenGL?
98.What are OpenGL utilities and Libraries?
99.What is the use of glut.h?
100. What is the method generation of sierpinski gasket?
101. Differentiate additive color model from subtractive color model
102. What are the two classes of primitives openGL supports?
103. What is opengl interface?
104. List out different opengl primitives, giving examples.
105. What are different opengi frames?
106. Define the following 2 dimensional transformations
      1) translation 2)rotation 3)scaling 4)reflection
107. What are the basic transformations in 3D?
108. What is concatenation?
109. What are the advantages of quaternion?
110. What is projection normalization?
111. Explain with the help of opengl functions perspective and parallel viewing opengl?
112. What is gluLookAt() function?
113. What are different types of light sources support by opengl?
114. What is aliasing?
115. What are four major tasks in a pipeline implementation?
116. What is Cohen Sutherland line clipping?
117. What do you understand by clipping?
118. What is Z-buffer algorithm for removing hidden faces?
119. What are orthographic projections? When do we need them?
120. What is a aliasing? Explain different methods of minimizing its effect?
121. What is polygon clipping?
122. What. is windowing and clipping?
123. List the advantages of interactive Graphics.
124. What do you mean by composite transformation? How it is useful?**

**125. What are projection, projector and plane of projection?**

Ans. projection is an image or a view. Projectors are the lines drawn from each and every point of the

object. These lines are perpendicular to the plane of projection & parallel to each other. Plane of

projection (POP) is the plane on which image is drawn.

**126. Differentiate first and third angle projection.**

Ans i.) Object is placed in first and third quadrant in first and third angle projection respectively.

ii.)Object is in between POP and observer in first angle projection, whereas in third angle projection POP

is in between object and observer.

iii.)In first angle projection, Plan (Top View) is below the Elevation (Front View).Right side view is on the

left of elevation and Left side view is on the right of the elevation.

iv) In third angle projection , Plan (Top View) is above the Elevation (Front View).Right side view is on the

right of front view (elevation) and Left side view is on the left of the elevation.

**127.What is orthographic projection and why it is called so?**

Ans. An orthographic projection is called so because projectors drawn from the corners of an object are

parallel to each other and perpendicular to the plane of projection(HP or VP)

.

**128. what is difference between orthographic and pictorial views?**

Ans. Orthographic projection is commonly used because it gives 100% details of an object by drawing

different views e.g. Elevation, Plan, side views, section views auxiliary views etc. It is 2-D, whereas

Pictorial views are 3-D and don’t give full details. Yes pictorial projection can be understood by a layman

easily, but orthographic projection can be understood by an engineer or concerned person only.

**129. Define point, line, plane & solid.**

Ans. Point- it is a geometrical entity which is dimensionless. We cannot give dimension to a point. It is

known as 0-D (zero dimension) entity.

Line- a line is shortest distance between two points. It is a geometrical entity which is 1-D (length only).

Plane- it is a geometrical entity which is 2-D (length x breadth) or have a area only. It has two types

1. Principle plane- HP, VP & PP (profile plane). orthographic views are drawn on it

2. Secondary plane(plane lamina)

Solid- it is a geometrical entity which is 3-D (length x breadth x height) or have a definite volume.

**130.In a Color CRT, What role does the shadow mask play?**

The shadow mask made of metal screen has small holes that ensures that an electron beam excites only phosphors of the proper color.

**131) Give an introduction to the types of geometric primitives that can be drawn with OpenGL?**

The type of primitive that needs to be drawn can be specified as parameter to glBegin(). We can draw points(GL\_POINTS), lines(GL\_LINES), polylines (GL\_LINE\_STRIP, GL\_LINE\_LOOP), polygons(GL\_POLYGON), triangles and quadrilaterals(GL\_TRIANGLES,

GL\_QUADS) and strips and fans(GL\_TRIANGLE\_STRIP, GL\_QUAD\_STRIP, GL\_TRIANGLE\_FAN). The vertices of the primitive can be specified using glVertex2f or glVertex3i where f stands for float, d for double and i for integer.

**132) How can you specify a viewer?**

If you are writing a 2D program then a viewer can be specified using gluOrtho2D with parameters left, right,bottom,top which specifies a viewing rectangle within the projection plane which is the z=0 plane or x-y plane. Using this function will cause all points outside the clipping rectangle to be invisible or clipped.

If it is a 3D program we need to specify a view volume. The view volume shape depends on the type of projection. If it is an Orthographic projection then view volume is a cuboid specified by glOrtho with parameters left,right,bottom,top,near,far. Any 3D object within this volume will be visible and any object outside will be not. The projection plane is again on z=0 plane.

If it is a perspective projection we can specify a view volume which is in the shape of a frustum of a pyramid using glFrustum with parameters left,right,bottom,top,near,far or using gluPerspective with parameters fieldofview, aspectratio , near, far.

We can specify a viewer position in such cases using gluLookAt with parameters eyex,eyey,eyez,atx,aty,atz,upx,upy,upz.

Remember that a 3D program can also draw 2D objects and not vice-versa.

**133) What is hidden surface removal?**

Hidden surface removal or visible surface determination is a technique used to achieve realism in 3D. When objects(polygons) are drawn on screen the final image generated will show polygons drawn last completely overlapping those drawn before. We should not draw polygons which are not visible. Hence there is an order in which if we draw the polygons they will look real. This method of enforcing an order while drawing polygons is called painter's algorithm. There are other techniques like z-buffer algorithm which rely on removing parts of the image which are farthest from the viewer and hence obscured by other polygons. While projecting points on a 2D surface the transformation is not invertible as all points lying on a projector map onto the same point in the image. To perform hidden surface removal we retain depth information(- distance along a projector-) as long as possible in the pipeline.

**134) what is projection normalization?**

We use a technique called projection normalization, which converts all projections into orthogonal projections by first distorting the objects such that the orthogonal projection of the distorted objects is the same as the desired projection of the original objects.

**135) What is the minimum refresh rate required for real-time video?**

Screen needs to be refreshed to draw slightly different images in a video. The small changes in the images is visible if the refresh rate is less than 24 frames per sec. Hence real-time video refers to a refresh rate of at least 24 images per sec. Images consequently need to be generated within 1/24th of a second or 41.67 millisecond.

**136) What are the stages of a graphics pipeline? How is pipelining useful?**

The graphics pipeline contains four stages of Vertex Processing, Clipping and Primitive Assembly, Rasterization and Fragment Processing. Pipelining is used whenever multiple sets of data need to be processed the same way. For example a complex graphics scene might consist of millions of polygons which need to be processed. If the vertices of the polygons are sent into the pipeline one at a time then the four stages mentioned earlier can process them in parallel. Pipelining increases the throughput of processing the data elements appreciably while the latency of processing each element increases slightly.

**137) What are the differences between additive colors and subtractive colors?**

Examples of additive color devices is a CRT monitor or projectors or Slide(positive) film. Examples of subtractive colors is color printers which use cyan, magenta and yellow colors. In additive color the primary colors like Red, Green and Blue add together to give the perceived color. With additive color, primaries add light to an initially black display yielding the desired color. In subtractive color we assume that white light hits the surface, a particular point will be red if all the components of the incoming light are absorbed by the surface except for wave lengths in the red part of the spectrum, which are reflected.

**138) What is a color look-up table?**

Suppose that the frame buffer has k-bits per pixel. Each pixel value or index is an integer between 0 and 2k-1. Suppose that we can display colors with a precision of m bits ie 2m reds,2m greens and 2m blues hence we can display any of the 23m colors but the frame buffer can specify only 2k of them. We handle this through a user defined lookup table that is of size 2k x 3m. The user program fills the 2k entries(rows) of the table with the desired colors. Once the LUT is populated, we can specify a color by its index in the LUT. For k=m=8 a common configuration we can choose 256 colors to be used in any image out of 16 million colors. These 256 colors are called the pallet.

**139) What is frame buffer? What value is stored in the frame buffer? What is color depth? What is double buffering?**

Frame buffer also called video RAM is a memory buffer which stores the image that is currently being displayed on the display screen.

The frame buffer stores the pixel values. The number of pixel values depends on the horizontal and vertical resolution of the display. Ex: 1024x768.

Color depth is the number of bits required to store a pixel value, usually mentioned as bits per pixel.

Double buffering is used to speed up animation. Animation requires us to draw objects with slight displacements. These calculations need to be done in real time at the rate of 24 frames per sec for smooth animations. Double buffering can help to refresh image displayed on screen with just a change in the base address of the VRAM(video RAM). Double bufferring is used when objects on screen are moving. Usually when double buffering is used the frame buffer size is twice that required to store one image.

**140) How are shadows generated in OpenGL?**

A shadow in order to be drawn needs to be generated by projection of light rays coming from a light source. In Viewing projections are already used to generate images. The same transformations can be reused to generate shadows. The Centre of Projection(COP) is now the position of the light source. The shadow is assumed to be produced on a projection plane say on the ground(y=0 plane). The shadow polygon so generated is drawn with shadow color and passes through graphics pipeline as would any other primitive.

**141) How can a polygon have a 2D-texture on its surface?**

At various stages we will work with screen coordinates, object coordinates, texture coordinates which we use to locate position within the texture and parametric coordinates which is used to describe curved surfaces.A 2D texture can be created by populating an array such as

GLubyte my\_texels[512][512][3];

This array is then passed as parameter to the function

glTexImage2D(GL\_TEXTURE\_2D, 0,GL\_RGB,512,512,0,GL\_RGB,GL\_UNSIGNED\_BYTE, my\_texels);

 the parameters meaning target,level,iformat,width,height,border,format,type,tarray. Level and border give us fine control over how texture is handled. We must enable texture mapping as we do other options using

glEnable(GL\_TEXTURE\_2D);

There are seperate memories like physical memory, frame buffer, texture memory. Texture coordinates should be mapped to the vertices by calling glTexCoord2f(s,t);

A following block of code will assign texture to a quadrilateral

glBegin(GL\_QUAD);

glTexCoord2f(0.0,0.0);

glVertex3f(x1,y1,z1);

glTexCoord2f(1.0,0.0);

glVertex3f(x2,y2,z2);

glTexCoord2f(1.0,1.0);

glVertex3f(x3,y3,z3);

glTexCoord2f(0.0,1.0);

glVertex3f(x4,y4,z4);

glEnd();

**142) Which transformations are called rigid body transformations?**

Translation and rotation do not change the shape of the body undergoing trasformation hence they are called rigid body transformations.

**143) What is composition of transformations?**

Basic transformations like translation and rotation when applied successively one after another then it is called composition of transformations. Ex: if object is denoted by O and translation and rotation by T,R then composition of transformations is when after applying T you apply R on the result ie R(T(O)) ie same as (RT) O. ie we can say the transformation matrices for translation and rotation can be multiplied to get a resultant matrix which when multiplied with the object(points) gives a transformed object(wrt position and orientation). Order of transformations is important as RT is not same as TR.

(plz remember rot\_mat of 4th program which was result of multiplying=> TRT)

**144) When do you use GLUT\_DEPTH in call to glutInitDisplayMode()?**

GLUT\_DEPTH is used in call to glutInitDisplayMode when the objects we draw are having three coordinates ie x,y,z. We would be calling glVertex3i or glVertex3f instead of glVertex2i or 2f. We have to call glEnable(GL\_DEPTH\_TEST); and clear the depth buffer in display function by calling glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

**145) How big are the two matrices GL\_PROJECTION and GL\_MODELVIEW?**

The two matrices are 4x4 in size with sixteen elements.

**146) Why do you need to call glFlush()?**

After drawing anything the object drawn does not appear on screen immediately untill glFlush is called.

**147) What is a logic operation?**

While drawing objects on screen the bits in the frame buffer are being modified. If a line is being drawn with white color then bits representing white are being set in specific positions corresponding to points on the line. If these bits representing white are operated bitwise on the pixel values already in the frame buffer then it becomes a logic operation. Various logic operations like AND, NOT, XOR ,OR can be performed on the bits in the frame buffer.

**148) How do you set material properties of objects?**

We can set material properties using call to glMaterialfv or glMaterialf.

**149) How do you set properties of light sources?**

Light sources (max 8 in a program) denoted by GL\_LIGHT0 to GL\_LIGHT7 can have properties such as position and intensities of red, green, and blue as well as ambient, diffuse and specular light. We can set these using call to glLightfv or glLightf.

**150) Why do we call glPushMatrix() and glPopMatrix()?**

Properties such as current color or raster position can all be pushed onto the stack and retrieved later. In between push and pop we can change these properties without affecting parts of the code following the pop.

**151) Why do we need to call glutSwapBuffers() when using double buffering?**

When using double bufferring, there are two buffers GL\_FRONT and GL\_BACK. We will usually be drawing in one of them while contents of the other buffer are being displayed. After drawing we make that buffer display and start drawing in the buffer which was earlier being displayed. This is accomplished using call to glutSwapBuffers();

**152) How do you set the size and position of a window?**

We can set the size and position of a window by calling glutInitWindowSize(width,height) and glutInitWindowPosition(x,y);

**OpenGL Functions**

**Libraries**

* gl - basic OpenGL functions. #include <GL/gl.h>
* glu - OpenGL Utility Library. Encapsulates frequently used combinations of gl calls plus setup inside its functions. #include <GL/glu.h>
* glut - OpenGL Utility Toolkit. Encapsulates window management functions via a set of window-system independent functions. #include <GL/glut.h>
* glx - OpenGL Extension to the X Window System. Wrappers for X functions. #include <X11/Xlib.h> and #include <GL/glx.h>

 **Window Manager Setup Using glut**

* glutInit(&argc, argv) - initializes GLUT, processes any command-line arguments for X functions. Call this before anything else.
* glutInitDisplayMode( unsigned int mode ) - sets the basic display modes. For more than one choice, OR the constants together. Usual mode constants are:

 GLUT\_SINGLE - single buffering

 GLUT\_DOUBLE- double buffering

 GLUT\_RGB - set to full color mode

 GLUT\_RGBA - same as RGB

 GLUT\_INDEX - set color index mode

 GLUT\_DEPTH - set up a depth buffer

* glutInitWindowSize( int width, int height ) - sets size of screen window in pixels
* glutInitWindowPosition( int x, int y ) - sets initial window position on the screen for upper left corner
* glutCreateWindow( char\* title ) creates a top-level window labeled with title.
* glutDisplayFunc( void (\*func)(void) ) - sets the callback function, provided by your program, that GLUT will call when a window must be redisplayed (e.g. on resize). Simply pass the name of a void function that takes no parameters.
* glutMainLoop( ) - start the event loop; never returns.

**Window Manager Input Handling Using glut**

* glutReshapeFunc( void (\*func)(int width, int height) ) - registers the callback function that GLUT calls on window reshape events.
* glutKeyboardFunc( void (\*func)(unsigned char key, int x, int y) - registers the callback function that GLUT calls on a keyboard event. The parameter key is the ASCII value of the key pressed. Parameters x and y indicate the mouse position at the time of the key press.
* glutMouseFunc( void (\*func)(int button, int state, int x, int y) ) - registers the callback function that GLUT calls on a mouse event. The parameter button may be GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON, or GLUT\_RIGHT\_BUTTON (a two button mouse may not generate the middle value). The parameter state may be GLUT\_UP or GLUT\_DOWN indicating button release or press, respectively. Parameters x and y indicate the mouse position.
* glutMotionFunc( void (\*func)(int x, int y) - registers the callback for a mouse move event while a button is pressed.
* glutIdleFunc( void (\*func)(void)) - registers the callback for idle time (no events)
* glutPostRedisplay( void ) - "nudges" the main loop to call display( ); sometimes useful after keyboard or mouse input changes some display parameters.
* glFlush( ) - flush the graphics output buffer.
* glFinsh( ) - waits here for flush to complete.

**Viewing and Modeling Transformations**

* gluLookat( GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble aimx, GLdouble aimy, GLdouble aimz, GLdouble upx, GLdouble upy, GLdouble upz) - defines the viewing matrix, setting the current eye position (eyex, eyey, eyez), the aim point along the desired line of sight (aimx, aimy, aimz), and the desired "up" direction (upx, upy, upz). Default values are (0,0,0), (0,0,-1), and (0,1,0).
* glScalef(float x, float y, float z) - scale by (x, y, z). Also: glScaled( ).
* glTranslatef(float x, float y, float z) - translate by (x, y, z). Also: glTranslated( ).
* glRotatef(float angle, float x, float y, float z) - rotate counterclockwise by angle degrees around the line through the origin with direction vector (x, y, z)
* glMatrixMode(GLenum mode) - sets the matrix to be worked on. Values for mode are:

 GL\_MODELVIEW - use the modelview matrix

 GL\_PROJECTION - use the projection matrix

 GL\_TEXTURE - use the texture matrix

* glLoadIdentity( void ) - set the current matrix to the identity

 **Projection and Viewport Transformations**

* glOrtho(double left, double right, double bottom, double top, double near, double far ) - sets the projection matrix to orthographic (parallel), with the near clipping plane defined by (left, bottom, -near) and (right, top, -near) and the far clipping plane defined by (left, bottom, -far) and (right, top, -far).
* gluOrtho2D(double left, double right, double bottom, double top) - sets the 2-dimensional clipping rectangle as (left, bottom) and (right, top).
* gluPerspective(double fieldofview, double aspectratio, double near, double far) - defines a perspective projection with the given field of view, aspect ration, and distance to the near and far clipping planes.
* gluFrustum(double left, double right, double bottom, double top, double near, double far) - defines a perspective projection with the near clipping plane as (left, bottom, -near) and (right, top, -near) and the far clipping plane at distance far from the eye point.

**Color and Clearing the Screen**

* glClearColor( float red, float green, float blue, float alpha ) - set the clear color to (red, green, blue) with value alpha.
* glClear( GLbitfield mask ) - clear the buffer indicated by mask using the current clear color. Values for mask are:

 GL\_COLOR\_BUFFER\_BIT - color buffer

 GL\_DEPTH\_BUFFER\_BIT - depth buffer

 GL\_ACCUM\_BUFFER\_BIT - accumulation buffer

 GL\_STENCIL\_BUFFER\_BIT - stencil buffer

* glColor3f( float red, float green, float blue ) - set the current drawing color to (red, green, blue).

**OpenGL Drawing Primitives**

* glVertex{234}{sifd}{v}( … ) - specify a vertex in either 2, 3, or 4-D coordinates, using either 16 or 32 bit integers or floats or doubles, and either listing all coordinates or using an array parameter.
* glBegin(GLenum mode) - start a vertex list. Parameter mode can be:

 GL\_POINTS - single points

 GL\_LINES - pairs of vertices are a line segment

 GL\_LINE\_STRIP - polyline

 GL\_LINE\_LOOP - closed polyline

 GL\_TRIANGLES - triples indicate a triangle

 GL\_TRIANGLE\_STRIP- linked strip of triangles

 GL\_TRIANGLE\_FAN - linked fan of triangles

 GL\_QUADS - quadruples indicate a quadrilateral

 GL\_QUAD\_STRIP - linked strip of quadrilaterals

 GL\_POLYGON - simple convex polygon

* glEnd( void ) - end of a vertex list
* glPointSize( float size ) - sets pixel width for a point; default is 1.0
* glLineWidth( float width ) - sets pixel width for a line; default is 1.0
* glLineStipple( int factor, ushort pattern ) - uses 16-by pattern to draw dashed or dotted lines at factor stretching.
* glRectf( float x1, float y1, float x2, float y2 ) - draw a rectangle with corners (x1, y1) and (x2, y2). Also in s, i, and d versions. Rectangle appearance depends on color, line, and fill attributes.
* glPolygonMode(GLenum face, GLenum mode) - sets how polygons, including triangles and quads, are drawn. Parameter face can be GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK (the default) to indicate which side should be drawn. Parameter mode can be GL\_POINT, GL\_LINE, or GL\_FILL (the default) to indicate the polygon should be drawn as points, outlined, or filled.
* glPolygonStipple(Glubyte\* array) - uses the array as a pattern to be drawn in a polygon that uses "filled" mode.
* glRasterPos{234}{sifd}{v}( … ) - sets the current raster position (for what comes next) in either 2, 3, or 4-D coordinates (see glVertex( ) ).
* glutBitmapCharacter( void\* font, char c ) - draws one bitmapped character using the indicated font; some fonts are: GLUT\_BITMAP\_TIMES\_ROMAN\_10 (also in 24), GLUT\_BITMAP\_HELVETICA\_10 (also in 12 and 18).
* glutStrokeCharacter( void\* font, char c ) - draws one stroke character using the indicated font; fonts include: GLUT\_STROKE\_ROMAN and GLUT\_STROKE\_MONO\_ROMAN. Stroke fonts can be scaled but slower than bitmaps.
* glEnable( constant ) - enables one of many modes; here, possibilities are GL\_LINE\_STIPPLE and GL\_POLYGON\_STIPPLE.
* glDisable( constant ) - disables the given mode.

**Lighting**

* glLights{if}{v}( light#, param, vector ) - specify a light# (from GL\_LIGHT0, … , GL\_LIGHT7) with some parameter and value(s). Choose from GL\_AMBIENT, GL\_DIFFUSE, or GL\_SPECULAR to set light colors; GL\_POSITION (vector indicates either within-scene position if vector[3] is non-zero, or directional light if vector[3] is 0), GL\_SPOT\_DIRECTION, GL\_SPOT\_EXPONENT, and GL\_SPOT\_CUTOFF (an angle) for a spotlight, or values for distance attenuation using GL\_CONSTANT\_ATTENUATION, GL\_LINEAR\_ATTENUATION, and GL\_QUADRATIC\_ATTENUATION.
* glLightModelf{v}( param, vector or value) - set the light model from GL\_LIGHT\_MODEL\_AMBIENT, GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, GL\_LIGHT\_MODEL\_TWO\_SIDE. The latter two use the values GL\_TRUE or GL\_FALSE.
* glEnable( GL\_LIGHTING ) - to enable the lighting system
* glEnable( GL\_LIGHT0 ) - or GL\_LIGHT1, etc., to enable specific lights

 **Materials Setting**

* glMaterial{f}{v}( face, param, vector or value ) - specify the properties of an object's surface. The face is either GL\_FRONT, GL\_BACK, or GL\_FRONT\_AND\_BACK. The following param values are used to specify color vectors: GL\_AMBIENT, GL\_DIFFUSE, GL\_AMBIENT\_AND\_DIFFUSE, GL\_SPECULAR, and GL\_EMISSION. The param GL\_SHININESS sets one float used as the cosine exponent for specular surfaces.
* glColorMaterial( face, param ) - causes the specified material property of the specified face to track the value of the current color at all times, i.e. any call to glColor\*( ) affects the material. The face and param are chosen as in glMaterial.
* glEnable( GL\_COLOR\_MATERIAL ) - enables color material mode.

**GLUT Function Key Constants**

* GLUT\_KEY\_F1 -F1 function key
* GLUT\_KEY\_F2-F2 function key
* GLUT\_KEY\_F3-F3 function key
* GLUT\_KEY\_F4-F4 function key
* GLUT\_KEY\_F5-F5 function key
* GLUT\_KEY\_F6-F6 function key
* GLUT\_KEY\_F7-F7 function key
* GLUT\_KEY\_F8-F8 function key
* GLUT\_KEY\_F9-F9 function key
* GLUT\_KEY\_F10-F10 function key
* GLUT\_KEY\_F11-F11 function key
* GLUT\_KEY\_F12-F12 function key
* GLUT\_KEY\_LEFT-Left function key
* GLUT\_KEY\_RIGHT-Up function key
* GLUT\_KEY\_UP- Right function key
* GLUT\_KEY\_DOWN-Down function key
* GLUT\_KEY\_PAGE\_UP-Page Up function key
* GLUT\_KEY\_PAGE\_DOWN-Page Down function key
* GLUT\_KEY\_HOME-Home function key
* GLUT\_KEY\_END-End function key
* GLUT\_KEY\_INSERT-Insert function key

**Texture Mapping**

* glTexImage2D( GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, GLvoid\* texels ); - specify a two dimensional texture map. The target is either GL\_TEXTURE\_2D or GL\_PROXY\_TEXTURE\_2D. The level is normally 0 unless using multilevel maps; internalFormat is usually GL\_RGBA, but there are many other constants available. The width and height are the texture map's size; border is 0 (no border) or 1 (1 pixel border). Both width and height must be of the form 2m + 2b; 64 x 64 is the minimum size. The format can be GL\_RGB, GL\_RGBA, or other constants; the type can be GL\_UNSIGNED\_BYTE, GL\_UNSIGNED\_SHORT, GL\_UNSIGNED\_INT, GL\_FLOAT, or other types. A typical call is:
* glTexImage2D( GL\_TEXTURE\_2D, 0, GL\_RGBA, 64, 64, 0, GL\_RGBA,GL\_UNSIGNED\_BYTE, myarray );
* glTexEnv{if}( GLenum target, GLenum pname, TYPE param ) - sets how a texture is applied. The target parameter must be GL\_TEXTURE\_ENV. The pname parameter is usually GL\_TEXTURE\_ENV\_MODE. The last parameter can be GL\_DECAL (texture replaces pixels), GL\_BLEND (texture blends with pixels), GL\_MODULATE (texture is blended by subtraction), or GL\_REPLACE (same as GL\_DECAL). These actions are for RGB texture maps; for RGBA maps, alpha channel maps, luminance maps, and intensity maps, the functions indicated by the last parameter, while similar, are not the same as stated here.
* glTexParameter{if}( GLenum target, GLenum pname, TYPE param ) - sets various parameters for applying a texture map. The target is usually GL\_TEXTURE\_2D. The pname parameter is associated with the last parameter. When the texture parameter falls outside [0, 1], use pname GL\_TEXTURE\_WRAP\_S or GL\_TEXTURE\_WRAP\_T to tell what to do: the last parameter can be GL\_REPEAT (to wrap the texture across the object) or GL\_CLAMP (to fix the outside values at 0 or 1). When a pixel is smaller than a texture element, use GL\_TEXTURE\_MIN\_FILTER to tell how to filter the map to the pixel; when the pixel is larger than the texture element, use GL\_TEXTURE\_MAG\_FILTER. The usual last parameters with these are GL\_NEAREST (to chose the nearest texture value) and GL\_LINEAR (to compute an average value). There are other values for MIN that deal with mipmapping.
* glTexCoord2f( coordinates ) - sets the current texture coordinates, and should immediately precede a call to glVertex\*( ). This associates texture parameters s and t with particular vertices.
* glEnable( GL\_TEXTURE\_2D ) - enable texture mapping.
* glDisable( GL\_TEXTURE\_2D ) - disable texture mapping.

**Bezier and NURBS Evaluators**

* glMap1{fd}( GLenum target, TYPE u1, TYPE u2, GLint stride, GLint order, TYPE\* controlpoints) - defines a 1-dimensional (curve) Bezier evaluator for a set of float or double control points. The target parameter is usually GL\_MAP1\_VERTEX\_3, although color, normal, and texture coordinates can be used instead. The u1 and u2 values indicate the range of the parameter and are typically 0.0 and 1.0. The stride tells the offset (the number of single or double precision values in each block) between control points, usually 3. The order is the degree plus 1 of the curve. The controlpoints variable points to the first control point in an array.
* glEvalCoord1{fd}(TYPE u) - evaluates the previously defined 1-D map; replaces glVertex calls in a loop, where the value of u is spaced between u1 and u2 (see above), and the loop is placed inside a glBegin(GL\_LINE\_STRIP) - glEnd( ) pair (note: other primitives may be used instead of GL\_LINE\_STRIP). Add "v" and make u a pointer to indicate an array of points instead.
* glMapGrid1{fd}( int #steps, TYPE u1, TYPE u2 ) - defines a grid from u1 to u2 in #steps, evenly spaced. Used with glEvalMesh1( ), instead of glEvalCoord1, for evenly spaced grids.
* glEvalMesh1( GLenum mode, int p1, int p2 ) - evaluates the 1 -D map according to glMapGrid1, from evaluation points numbered p1 to p2. The mode parameter is either GL\_POINT or GL\_LINE. Use instead of a loop containing glEvalCoord1.
* glEnable( GL\_MAP1\_VERTEX\_3 ) - enables the 1-D map.
* glMap2{fd}( GLenum target, TYPE u1, TYPE u2, GLint ustride, GLint uorder, TYPE v1, TYPE v2, GLint vstride, GLint vorder, TYPE\* controlpoints) - defines a 2-dimensional (patch) Bezier evaluator for a set of float or double control points. The parameters are similar to those for the 1-D function, but you must specify parameter ranges, strides, and orders in each dimension; for example, the target parameter is usually GL\_MAP2\_VERTEX\_3, although color, normal, and texture coordinates can be used instead. Note that ustride is typically 3, but vstride is typically 12, for a single order 4 (degree 3) patch; adjust accordingly if controlpoints is a larger array containing several patches.
* glEvalCoord2{fd}( TYPE u, TYPE v ) - evaluates the previously defined 2-D map; replaces glVertex calls inside loops. Note that you must loop over both parameter directions. Use glEnable( GL\_MAP2\_VERTEX\_3) and glEnable( GL\_AUTO\_NORMAL ) to auto-generate true surface normals.
* glMapGrid2{fd}(int #steps\_in\_u, TYPE u1, TYPE u2, int #steps\_in\_v, TYPE v1, TYPE v2) - defines a 2-D grid that goes from u1 to u2 in #steps\_in\_u evenly-spaced steps and from v1 to v2 in #steps\_in\_v evenly-spaced steps. Use with glEvalMesh2 to replace glEvalCoord2.
* glEvalMesh2( GLenum mode, int p1, int p2, int q1, int q2 ) - evaluates a 2-D map according to glMapGrid2, from p1 to p2 in parameter u and from q1 to q2 in parameter v. The mode is either GL\_POINT, GL\_LINE, or GL\_FILL. The latter generates filled polygons. Use glEnable( GL\_MAP2\_VERTEX\_3) and glEnable( GL\_AUTO\_NORMAL ) to auto-generate true surface normals.