1. Explain the working details of DDA algorithm? Explain. Digitize a line with end points A(6,12) and B(10,5) using Bresenham's line drawing algorithm.

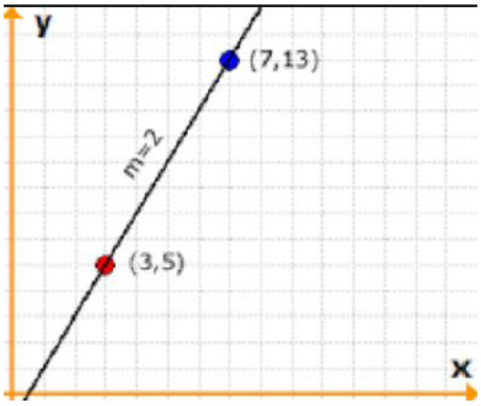
Answer.

DDA(Digital Differential Analyzer) Line Algorithm

- It is a scan conversion line algorithm based on calculation either $\triangle x$ or $\triangle y$ using equation $m=\triangle y/\triangle x$.
- We sample the line at unit interval in one direction (X if $\triangle x$ is greater than $\triangle y$ otherwise in y direction) and determine the corresponding integer values nearest the line path for the other coordinate.

Algorithm

Consider a line with positive slope as shown in the figure below



The equation of the line is given,

$Y = m.x+b$ .....(i)

$m = (y2-y1)/(x2-x1)$ .....(ii)

For any interval $\triangle x$, corresponding interval is given by $\triangle y = m.\triangle x$.

Case I:

If $|m| \leq 1$, we sample at unit x interval i.e $\triangle x = 1$.

$x_{k+1} = x_k+1$ .....(iii)

Then we compute each successive y-values, by setting $\triangle y = m$ (since, $m=\triangle y/\triangle x$ and $\triangle x=1$).

So, $y_{k+1} = y_k + m$ .....(iv)

The calculated y value must be rounded to the nearest integer

Case II:

If $|m| > 1$, we sample at unit y-interval i.e $\triangle y = 1$ and compute each successive x-values.

$y_{k+1} = y_k + 1$.....(v)

Therefore, $1 = m.\Delta x$, and $\Delta x = 1/m$ (since, $m=\Delta y/\Delta x$ and $\Delta y=1$).

$x_{k+1} = x_k + 1/m$... .....(vi)

The calculated x value must be rounded to the nearest integer

<u>Digitizing a line with end points A(6,12) and B(10,5) using Bresenham's line drawing algorithm.</u>

Digitization

Given end points A(6, 12) & B(10,5)

Slope $(m) = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5-12}{10-6} = -\frac{7}{4}$ $\therefore |m| = 1.75 > 1$

For $|m| > 1$

$\Delta x = 4$
$\Delta y = 7$

Acc. to Bresenham algorithm

$P_0 = 2\Delta x - \Delta y$ *

if $P_k > 0$

$x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2\Delta x - 2\Delta y$

else

$x_{k+1} = x_k$ → $y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2\Delta x$

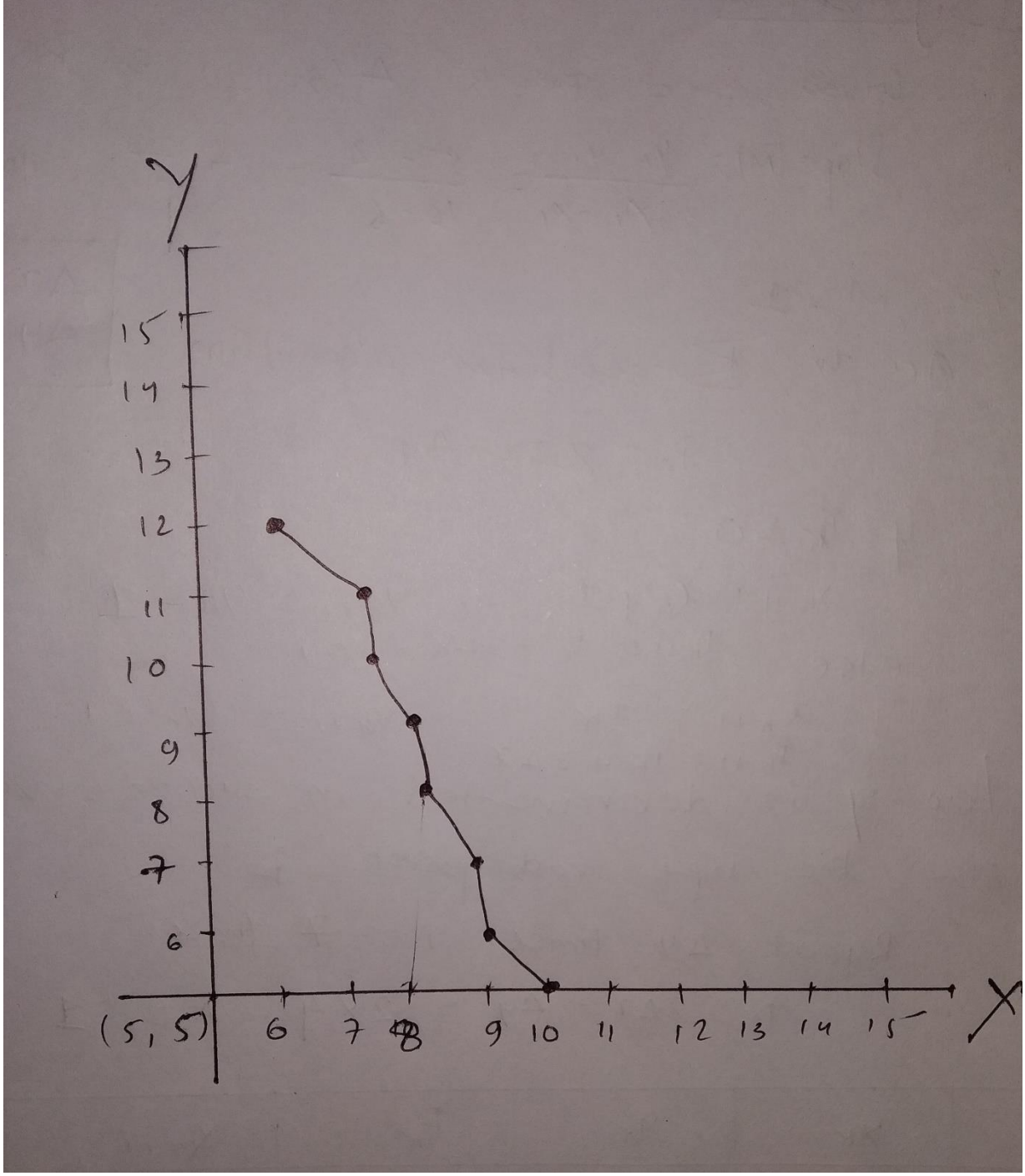
Here 'y' is decremented in successive step

~~since the left endpoint is~~

Repeat Δy times i.e 7 times.

$P_0 = 2\Delta x - \Delta y = 2 \times 4 - 7 = 1$

K	x_k	y_k	P_k	X_{k+1}	Y_{k+1}
0	6	12	1	7	11
1	7	11	-5	7	10
2	7	10	3	8	9
3	8	9	-3	8	8
4	8	8	5	9	~~8~~
5	9	7	-1	9	6
6	9	6	7	10	5
	10	5			



2. How can polygons be clipped? Why is Phong shading also called Normal Vector Interpolation scheme? Explain.

Polygon Clipping: Sutherland – Hodgeman

The Sutherland–Hodgeman algorithm is used for clipping polygons. A single polygon can actually be split into multiple polygons.

There are four possible cases for any given edge of given polygon against clipping edge.

1. **Both vertices are inside** :

Only the second vertex is added to the output list

2. **First vertex is outside while second one is inside** :

Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list

3. **First vertex is inside while second one is outside** :

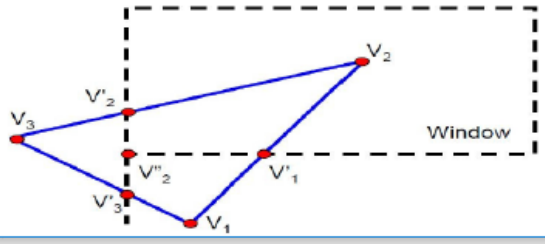
Only the point of intersection of the edge with the clip boundary is added to the output list

4. **Both vertices are outside** :

No vertices are added to the output list

Case	1st vertex	2nd vertex	output
1	inside	inside	2nd vertex
2	inside	outside	intersection
3	outside	outside	none
4	outside	inside	2nd and intersection

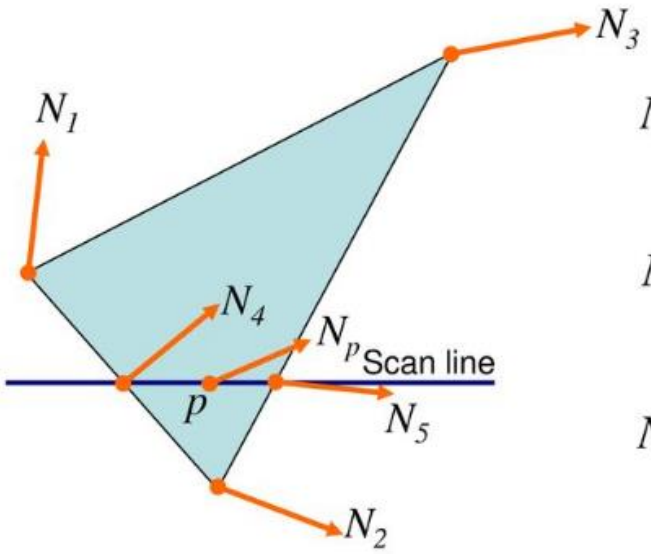
Example:



From	To	1st point	2nd point	Case	Output list
V_1	V_2	Outside	Inside	4	V'_1 and V_2
V_2	V_3	Inside	Outside	2	V'_2
V_3	V_1	Outside	Outside	3	

Phong shading,It is **also called Phong interpolation** or **normal-vector interpolation shading**. Specifically, it interpolates surface normals across rasterized polygons and computes pixel colors based on the **interpolated** normals and a reflection model.

- Interpolating Normal



$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

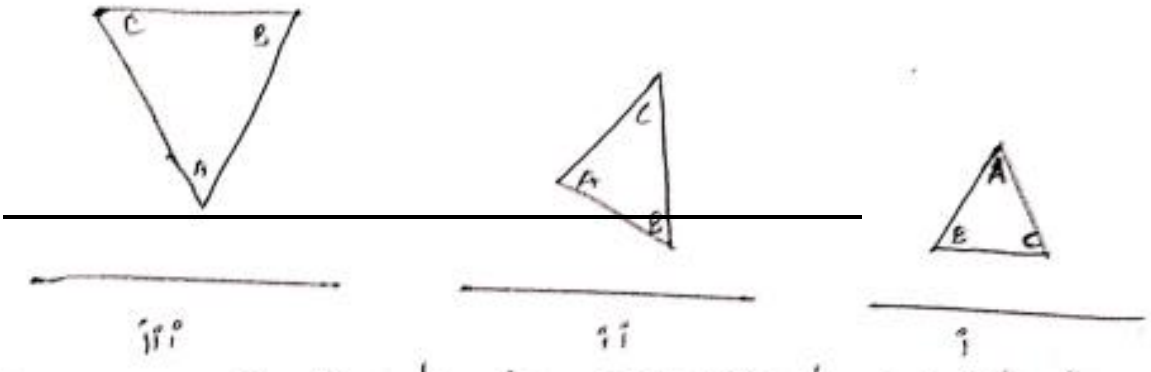
4. How to animate a two dimensional figure using transformations? Explain with example.

Transformation is the process of changing some graphics into something else using certain rules. There are various type of transformations, such as

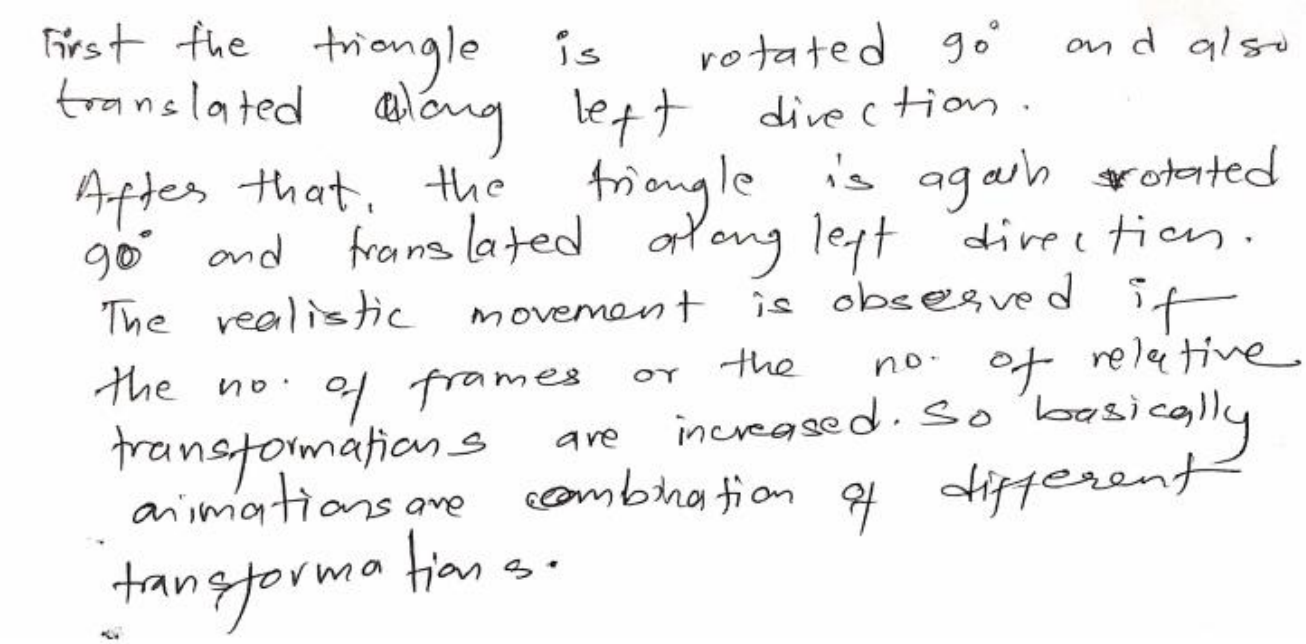
- translation
- rotation
- scaling
- shearing
- reflection.

Above are five major transformations which can be used for animation. Animation of 2D figure means movement of the 2D objects in window along with change in it's size and angles.

For Example



Let's say, a triangle is animated which is rolling in left direction and increasing its size also. For simplicity the process is completed in 3 frames i, ii & iii.



5.What are the key issues prevalent in producing a Virtual reality scene? Describe the Binary Space Partition tree

Answer.

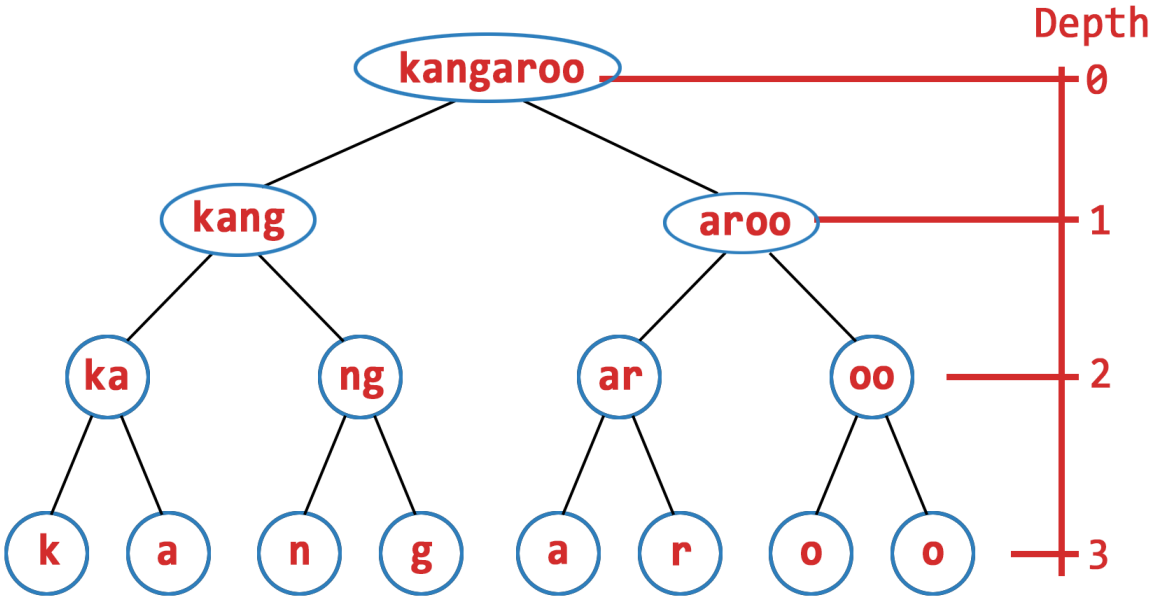
There is a number of challenges that are hampering the growth of VR. First, hardware for capturing 3D images is rudimentary at best, unless users seek out the uppermost level of hardware at extremely high price points.

Additionally, access to, and quality of, images that are currently available is limited at best. Developers are forced to either use low-quality images or develop the images themselves at a substantial cost, often pricing the bulk of the VR market out of their product.

Where high quality images exist, image depositories are able to charge substantial surcharges to users, profiting from the work of the creators, but not compensating them accordingly.

In computer science, **binary space partitioning** (BSP) is a method for recursively subdividing a **space** into convex sets by hyper planes. This subdivision gives rise to a representation of objects within the **space** by means of a **tree** data structure known as a BSP **tree.** A BSP tree is a recursive sub-division of space that treats each line segment (or polygon, in 3D) as a cutting plane which is used to categorize all remaining objects in the space as either being in "front" or in "back" of that plane.

In other words, when a partition is inserted into the tree, it is first categorized with respect to the root node, and then recursively with respect to each appropriate child

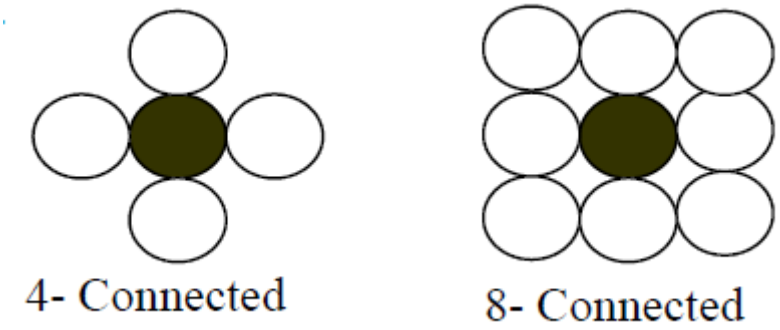


6. How can a polygon surface be filled using the Flood fill approach? Explain.

Answer.
Flood-fill Algorithm is applicable when we want to fill an area that is not defined within a single color boundary. If fill area is bounded with different color, we can paint that area by replacing a specified interior color instead of searching of boundary color value. This approach is called flood fill algorithm. We start from a specified interior pixel (x,y) and reassign all pixel values that are currently set to a given interior color with desired fill_color. Using either 4-connected or 8-connected region recursively starting from input position, the algorithm fills the area by desired color.
Algorithm for 4-connected:

```
void flood_fill4(int x,int y,int fill_color,int old_color)
{
int current;
current = getpixel (x,y);
if (current == old_color)
{
putpixel (x,y,fill_color);
flood_fill4(x-1,y, fill_color, old_color);
flood_fill4(x,y-1, fill_color, old_color);
flood_fill4(x,y+1, fill_color, old_color);
flood_fill4(x+1,y, fill_color, old_color);
```

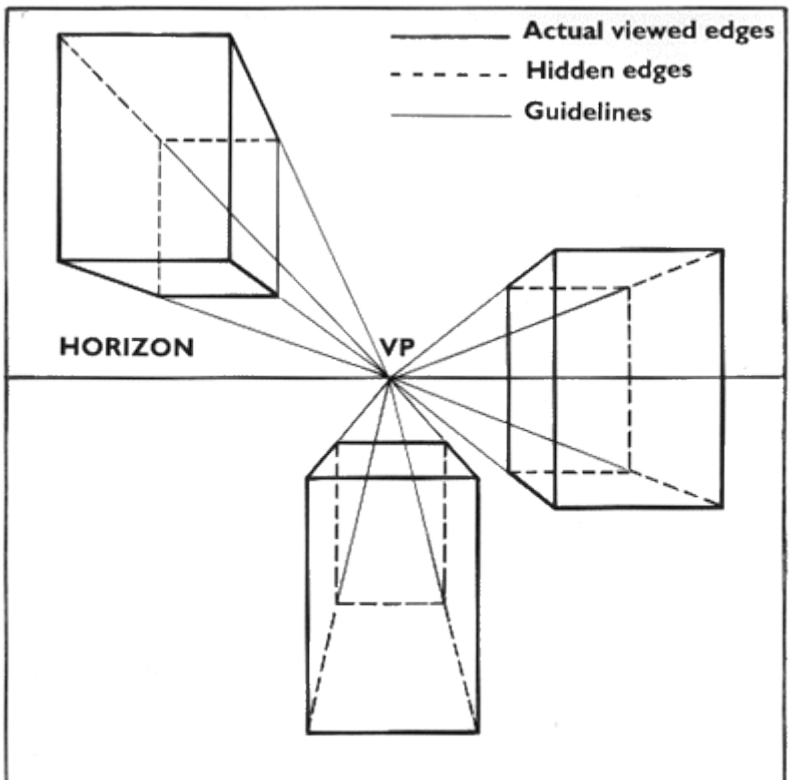
}
}



7. What is the significance of vanishing points in Perspective Projection? Explain.

Answer

A **vanishing point** is a **point** on the image plane of a **perspective** drawing where the two-dimensional **perspective projections** (or drawings) of mutually **parallel** lines in three-dimensional space appear to converge. It is what allows us to create **drawings**, paintings, and photographs that have a three-dimensional look.



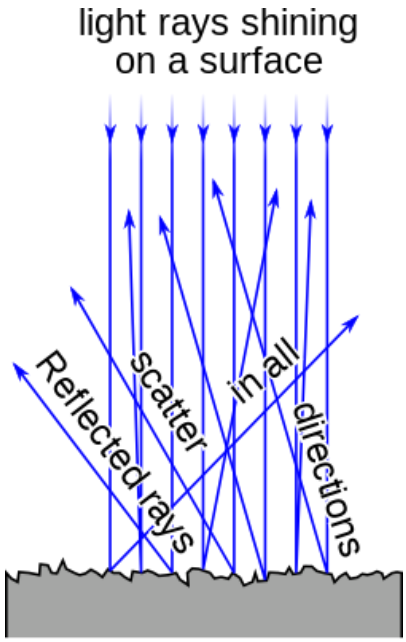
Vanishing Point Perspective is used in Graphic editing and 3D video games. It can be used to render 3D shapes (3D Buildings and objects), add perspective to a background scene (road, train track) or add shadow effects. Hence the significance of vanishing point is to make the projection more realistic in 3-D look.

8. Explain ambient light, diffuse reflection and specular reflection with examples

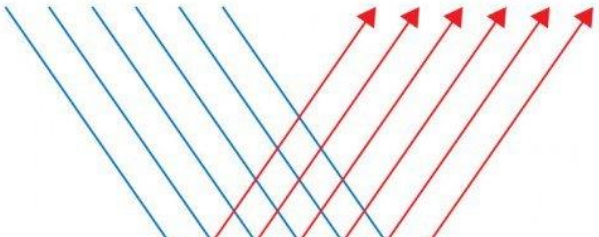
Answer

Ambient light means the **light** that is already present in a scene, before any additional **lighting** is added. It usually refers to natural **light**, either outdoors or coming through windows etc. It can also mean artificial **lights** such as normal room **lights**

Diffuse reflection is the **reflection** of light or other waves or particles from a surface such that a ray incident on the surface is scattered at many angles rather than at just one angle as in the case of specular **reflection**



Specular reflection is a type of surface **reflectance**often described as a mirror-like **reflection** of light from the surface. In **specular reflection**, the incident light is **reflected** into a single outgoing direction.



9. Compute the midpoint of the Bezier Curve with control points p0 = (0,0,1), p1 = (1,0,1) and p2 = (1,2,0).

9. Compute the midpoint of the Bezier curve with control points $P_0 = (0,0,1)$; $P_1 = (1,0,1)$ and $P_2 = (1,2,0)$.

→ Solution

Given control points $P_0 = (0,0,1)$
$P_1 = (1,0,1)$
$P_2 = (1,2,0)$

No. of control points = 3

So, $n = 3 - 1 = 2$

We know, $P(u) = \sum_{k=0}^{n} P_k \, BF_{2k,n}(u)$

And, $BF_{2k,n}(u) = C(n,k) u^k \cdot (1-u)^{n-k}$

Now

$BF_{2,0,2}(u) = \frac{2!}{0!\,2!} u^0 (1-u)^2$

$= (1-u)^2$

$BF_{2,1,2}(u) = \frac{2!}{1!\,1!} u^1 \cdot (1-u)$

$= 2u(1-u)$

$BF_{2,2,2}(u) = \frac{2!}{0!\,2!} u^2 (1-u)^0$

$= u^2$

Now

$P(u) = P_0 (1-u)^2 + P_1 2u(1-u)^2 + P_2 u^2$

For mid point, no. of segment = 2

No. of points = 3

i.e $u = 0, 1/2, 1$

$P(1/2) = P_0 (1-1/2)^2 + P_1 \cdot 2 \times \frac{1}{2} (1-1/2)^2 + P_2 (1/2)^2$

$= P_0 \cdot 1/4 + P_1 \, 1/4 + P_2 \, 1/4$

$= \frac{1}{4} [P_0 + P_1 + P_2)$

$= \frac{1}{4} [(0,0,1) + (1,0,1) + (1,2,0)]$

$= \frac{1}{4}(2, 2, 2)$

$= (1/2, 1/2, 1/2)$

Hence the midpoint of Bezier Curve is $(1/2, 1/2, 1/2)$ i.e $(0.5, 0.5, 0.5)$

10. How does a polygon can be created in OpenGL? Illustrate with an example.

Answer.
Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but you can also draw them as outlined polygons or simply as points at the vertices. A filled polygon might be solidly filled or stippled with a certain pattern.

A program to draw simple polygon.

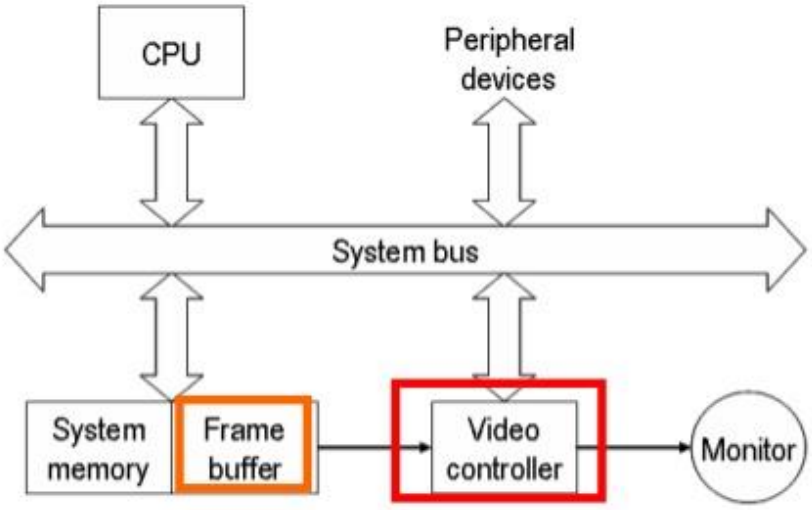
```
#include <stdio.h>
#include <GL/glut.h>

void display(void)
{
 glClear( GL_COLOR_BUFFER_BIT);
 glColor3f(0.0, 1.0, 0.0);
 glBegin(GL_POLYGON);
  glVertex3f(2.0, 4.0, 0.0);
  glVertex3f(8.0, 4.0, 0.0);
  glVertex3f(8.0, 6.0, 0.0);
  glVertex3f(2.0, 6.0, 0.0);
 glEnd();
```

```
 glFlush();
}

int main(int argc, char **argv)
{
 printf("hello world\n");
 glutInit(&argc, argv);
 glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB |
GLUT_DEPTH);

 glutInitWindowPosition(100,100);
 glutInitWindowSize(300,300);
 glutCreateWindow ("square");

 glClearColor(0.0, 0.0, 0.0, 0.0);          // black
background
 glMatrixMode(GL_PROJECTION);               // setup
viewing projection
 glLoadIdentity();                            // start
with identity matrix
 glOrtho(0.0, 10.0, 0.0, 10.0, -1.0, 1.0);   // setup a
10x10x2 viewing world

 glutDisplayFunc(display);
 glutMainLoop();

 return 0;
}
```

11. How does a video controller and a frame buffer jointly collaborate to produce graphical display on the screen, in case of a Raster Display?

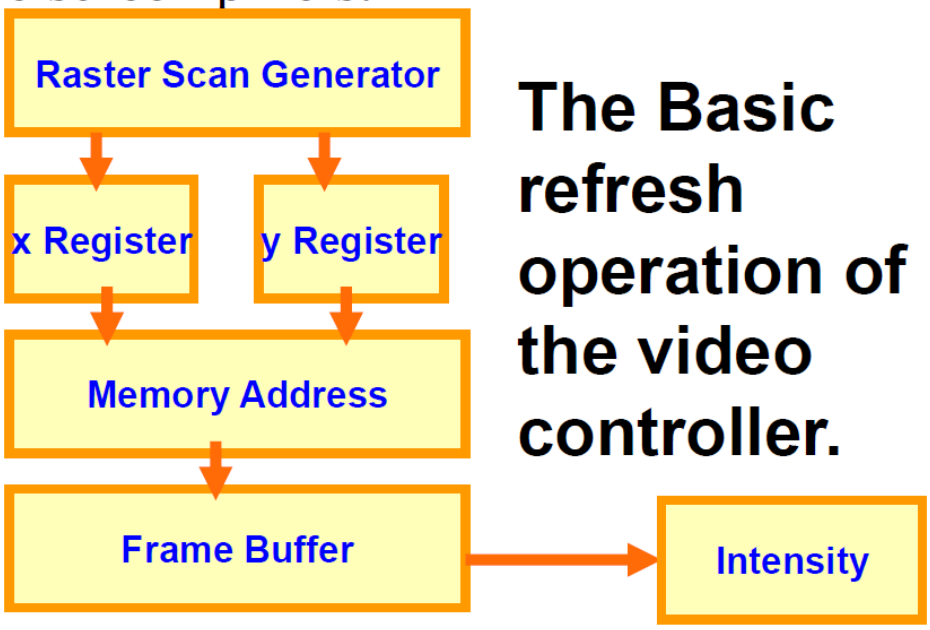
Video Controller

- A fixed area of the system memory is reserved for the frame buffer, and the video controller is given direct access to the frame buffer memory



Simple Organization of the Video Controller

- Two registers are used to store the coordinates of the screen pixels.



The Basic refresh operation of the video controller.

- A fixed area of memory is reserved for frame buffer, and the video controller is given direct access to the frame buffer
- When a particular command is called by the application program, the graphics subroutine package sets the appropriate pixels in the frame buffer.
- The video controller then cycles through the frame buffer, one scan line at a time, typically 50 times per second.
- It will bring a value of each pixel contained in the frame buffer and uses it to control the intensity of the CRT electron beam.
- So there exists a one to one relationship between the pixel in frame buffer and that on the CRT screen.
- Frame buffer locations, and the corresponding screen position are referenced in Cartesian co-ordinates.
- For most of the system, the coordinate origin is referenced at the lower left corner of the screen, with positive X value increasing to the right and positive y value increasing from bottom to top. However, in some PC, the coordinate origin is referenced at the upper left corner of the screen, so the y values are inverted.

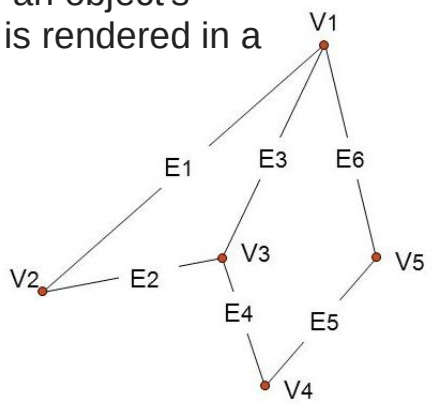
12. Write short notes on (Any TWO)

a. Polygon Tables

Polygon (**computer graphics**) **Polygons** are used in **computer graphics** to compose images that are three-dimensional in appearance. Usually (but not always) triangular, **polygons** arise when an object's surface is modeled, vertices are selected, and the object is rendered in a wire frame model

Polygon Tables

- Vertex Table
- Edge Table
- Surface Table



VERTEX TABLE	
V1	x1,y1,z1
V2	x2,y2,z2
V3	x3,y3,z3
V4	x4,y4,z4
V5	x5,y5,z5

EDGE TABLE	
E1	V1,V2
E2	V2,V3
E3	V3,V1
E4	V3,V4
E5	V4,V5
E6	V5,V1

POLYGON-SURFACE TABLE	
S1	E1,E2,E3
S2	E3,E4,E5,E6

b. Augmented Reality

Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real-world are "**augmented**" by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory.

c. Painter's algorithm

The painter's algorithm, also known as a priority fill, is one of the simplest solutions to the visibility problem in 3D computer graphics. When projecting a 3D scene onto a 2D plane, it is necessary at some point to decide which polygons are **visible**, and which are **hidden**.