Computer Graphics

Unit 1 – Part3

-By Manjula. S

OpenGL Introduction

- OpenGL is a Graphical Application Programming Interface(API).
- OpenGL allows to access and control GPU.
- It can be called as a specification and provides geometric primitives but it does not have specific code.
- Every Graphic card developer have implemented or Informative of graphic card in the form of OpenGL.
- OpenGL will be different from one Developer to others.
- OpenGL is not a Open source.
- OpenGL is very easy to use for users as it connect across different platform easily.

-By Manjula. S

Objects in OpenGL

- Defined by “set of vertices”.
- Example : A rectangle can be created in OpenGL using 4 vertices.
- In graphics, we form objects by specifying the positions in space of various geometric primitives (lines, points, polygons).

Objects in OpenGL

- OpenGL programs define primitives through list of vertices.

Ex., triangular polygon glBegin(GL_POLYGON);

```
    glVertex3f(0.0, 0.0, 0.0); /* vertex A*/
    glVertex3f(0.0, 1.0, 0.0); /* vertex B*/
    glVertex3f(0.0, 0.0, 1.0); /* vertex C*/
    glEnd();
```

Function glBegin specifies the type of primitive that the vertices define. Function glVertex3f specifies x,y,z coordinates of a location in space.

Function glEnd ends the list of vertices.

Can also have GL_LINE_STRIP and GL_POINTS.

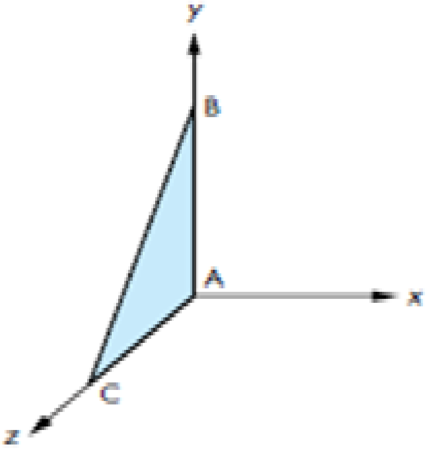


FIGURE 1.31 A triangle.

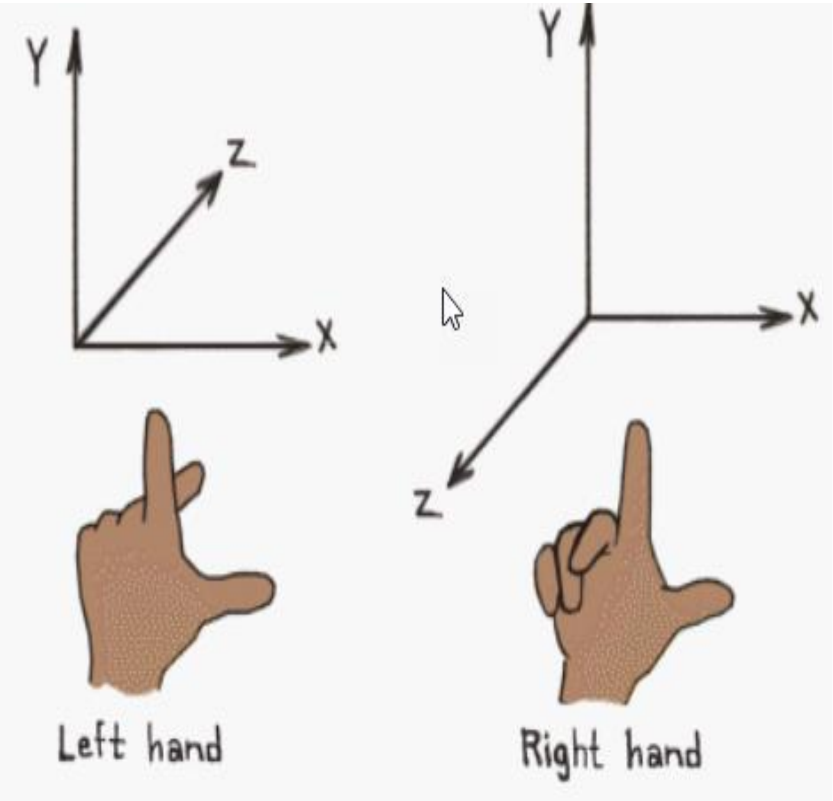
-By Manjula. S

Coordination

- **coordinate system** is a way of assigning numbers to points. In two dimensions, you need a pair of numbers to specify a point. The 2D coordinates are often referred to as *x* and *y and 3D as x, y and z.*
- There are a total of 5 different coordinate systems that are of importance to us:
  1. Local space (or Object space)
  2. World space.
  3. View space (or Eye space)
  4. Clip space.
  5. Screen space.
- **Frame of reference** – a coordinate system in which we are currently representing our objects in.
- **Terms we come across in Frame of reference**
  1. **Object space** – a frame of reference from a specific object.
  2. **World space** – a frame of reference from our world coordinate system.
  3. **Camera space** – a frame of reference from our camera.
  4. **Clip space** – a frame of reference used to define a portion of our world that is actually visible from a camera.
  5. **Screen space** – a frame of reference from our screen's coordinate system.

-By Manjula. S

Coordinate System of OpenGL



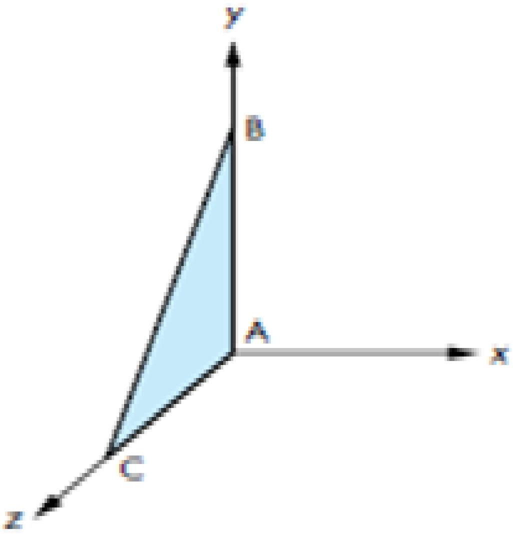
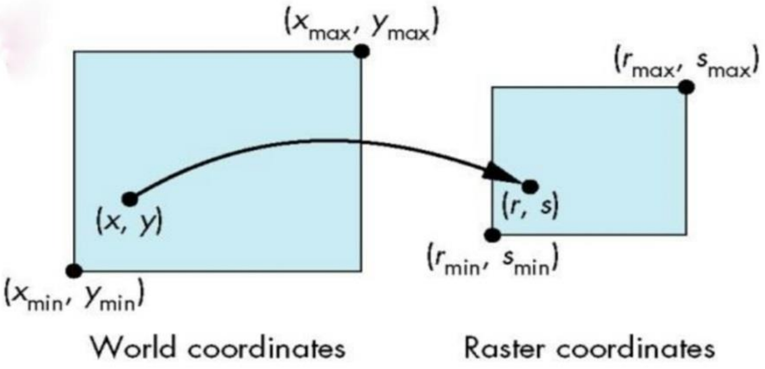


FIGURE 1.31 A triangle.

-By Manjula. S

Coordinate Systems

- **World co-ordinates** user' co-ordinate system became known as world coordinate system OR application, model or object coordinate system.



- **Screen co-ordinates** Also known as window co-ordinates OR raster coordinates. This 2D coordinate system refers to the physical coordinates of the pixels on the computer screen, based on current screen resolution.

Viewer or camera

- OpenGL provides access to the frame buffer. We can define a viewer or camera in a variety of ways.
- Available APIs differ both in how much flexibility they provide in camera selection and in how many different methods they allow. If we look at the camera in Figure, we can identify four types of necessary specifications:

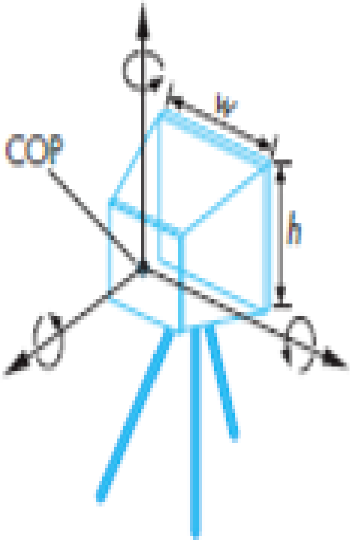


FIGURE 1.32 Camera specification.

Viewer or camera

- A viewer can be specified in a variety of ways. The 4 types of specifications are

  1) position

  2) Orientation (rotation)

  3) Focal length

  4) Film plane

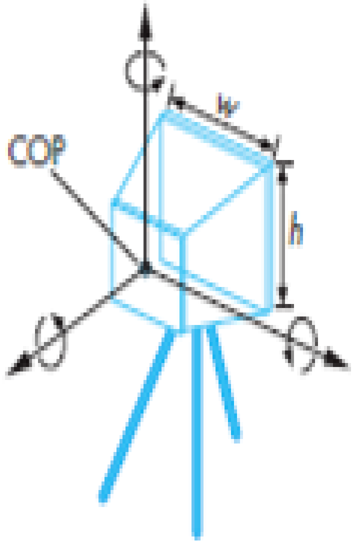


FIGURE 1.32 Camera specification.

Viewer or camera

- **Position**: The camera location usually is given by the position of the center of the lens (the center of projection).
- **Orientation:** Once the camera is positioned, a camera coordinate system can be placed with its origin at the center of projection.

Then the camera can be rotated independently around the three axes of this system.

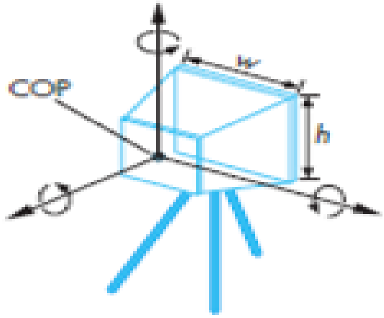


FIGURE 1.32 Camera specification.

-By Manjula. S

Viewer or camera

- **Focal length:** The focal length of the lens determines the size of the image on the film plane or, equivalently, the portion of the world the camera sees.
- **Film plane:** The back of the camera has a height and a width. The orientation of the back of the camera can be adjusted independently of the orientation of the lens.

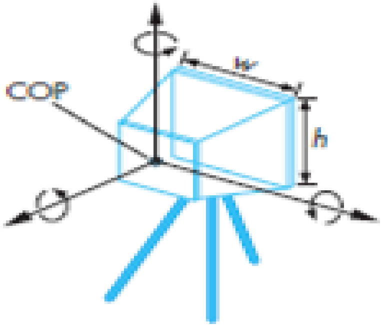


FIGURE 1.32 Camera specification.

Viewer Specification in OpenGL

- OpenGL provides the following function for specifying the camera position and the view. /* points the camera from COP toward desired point with specified up direction for the camera */

gluLookAt(cop_x, cop_y,cop_z, at_x, at_y, at_z, up_x, up_y, up_z);

cop_x,cop_y, cop_z : eyePosition3D is a XYZ position. This is where you are (your eye is).

at_x, at_y, at_z : center3D is the XYZ position where you want to look at.

up_x, up_y, up_z : upVector3D is a XYZ normalized vector. Often 0.0, 1.0, 0.0

glPerspective(field_of_view, aspect ratio, near, far); selects a lens for a perspective view and how much of the world the camera should image.

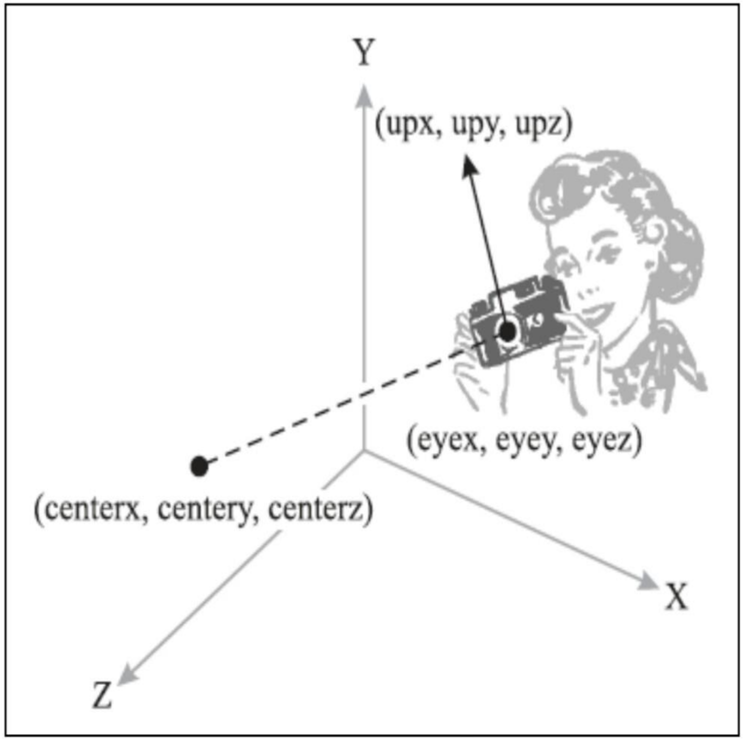
-By Manjula. S

Viewer Specification in OpenGL

```
void gluLookAt (
  GLdouble eyex, GLdouble eyey, GLdouble eyez,
 GLdouble centerx, GLdouble centery, GLdouble centerz,
 GLdouble upx, GLdouble upy, GLdouble upz
  );
```

If you don’t call gluLookAt(), the OpenGL camera is given some default settings:

- it’s located at the origin, (0, 0, 0);
- it looks down the negative Z axis;
- its “up” direction is parallel to the Y axis.



-By Manjula. S

Basic OpenGL types

- GLfloat and GLint, are used rather than the C types, such as float and int.
- These types are defined in the header files and usually in the obvious way.
- For example, #define GLfloat float
- However, use of the OpenGL types allows additional flexibility for implementations.
- For example, suppose the floats are to be changed to doubles without altering existing application programs.

Basic OpenGL types

- Returning to the vertex function, if the user wants to work in 2-D with integers, then the form

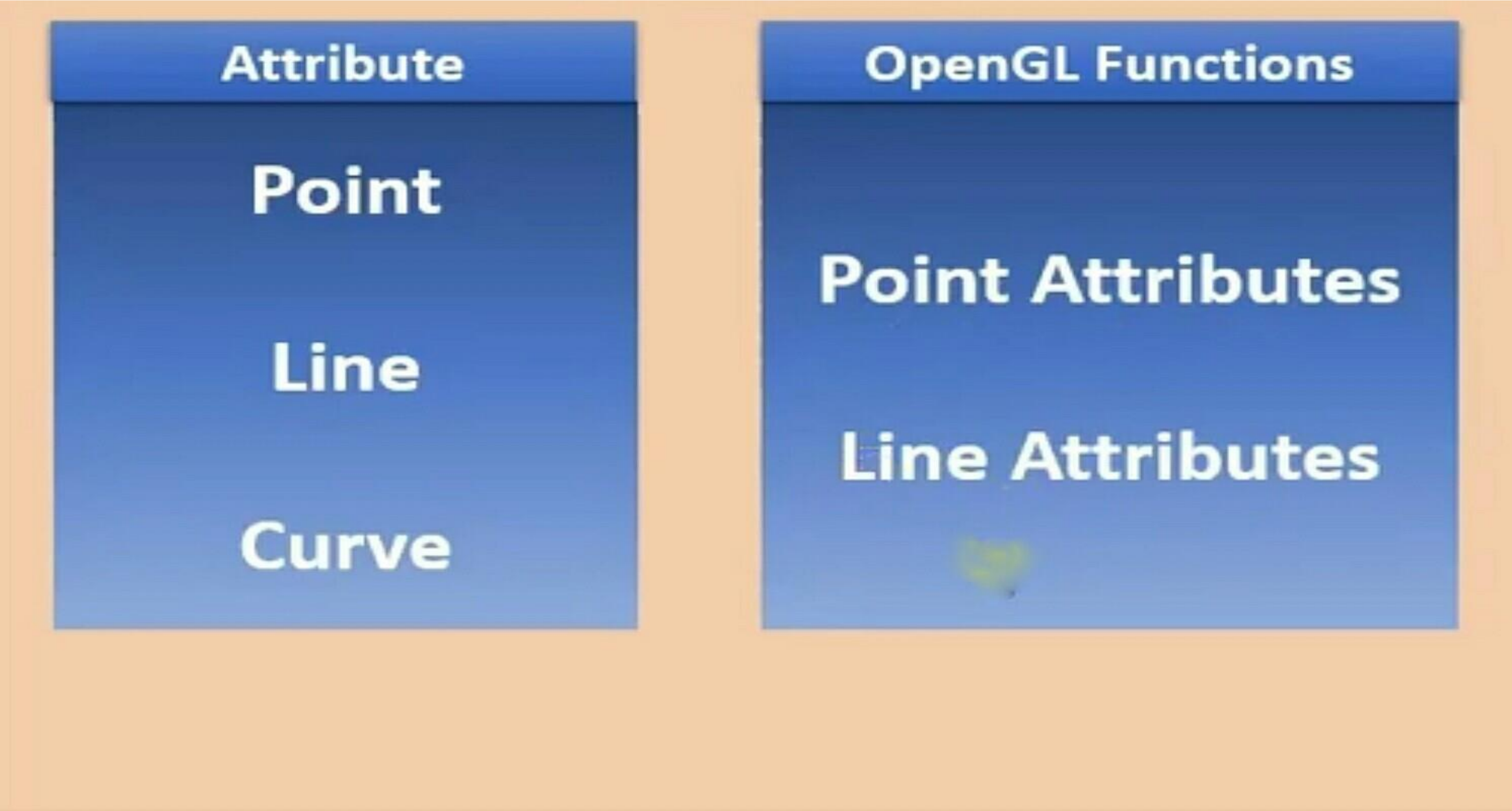
  glVertex2i(GLint xi, GLint yi) is appropriate
- glVertex3f(GLfloat x, GLfloat y, GLfloat z) specifies a position in 3-D space using floating-point numbers.
- If an array is used to store the information for a 3-D vertex, GLfloat vertex[3] then glVertex3fv(vertex) can be used.

Geometric primitives

- Different numbers of vertices are required depending on the object.
- Any number of vertices can be grouped using the functions glBegin and glEnd.
- The argument of glBegin specifies the geometric type that the vertices define. Hence, a line segment can be specified by

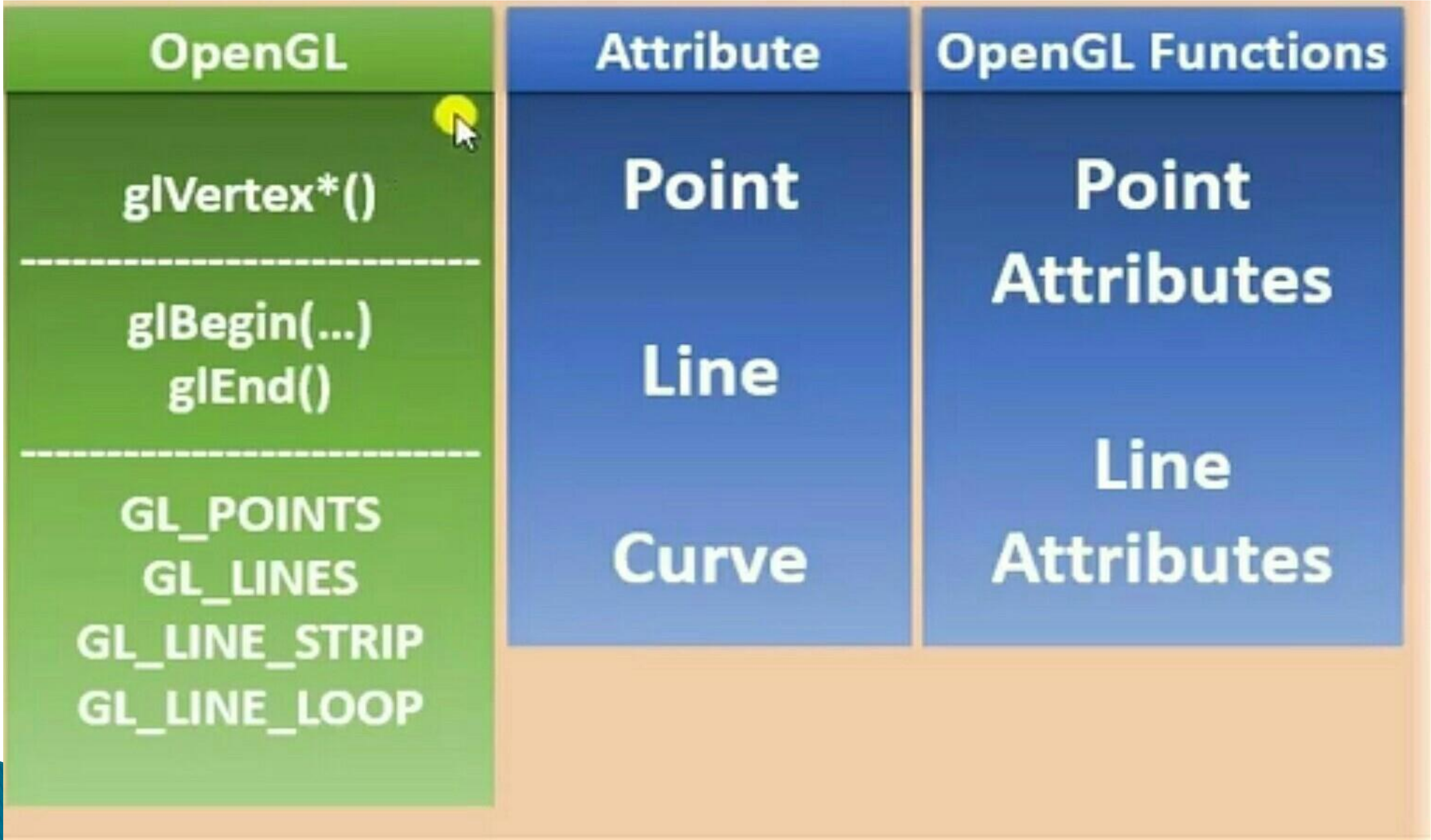
```
glBegin(GL_LINES);
glVertex2f(xl,yl);
glVertex2f(x2,y2);
glEnd();
```

OpenGL Attribute Functions



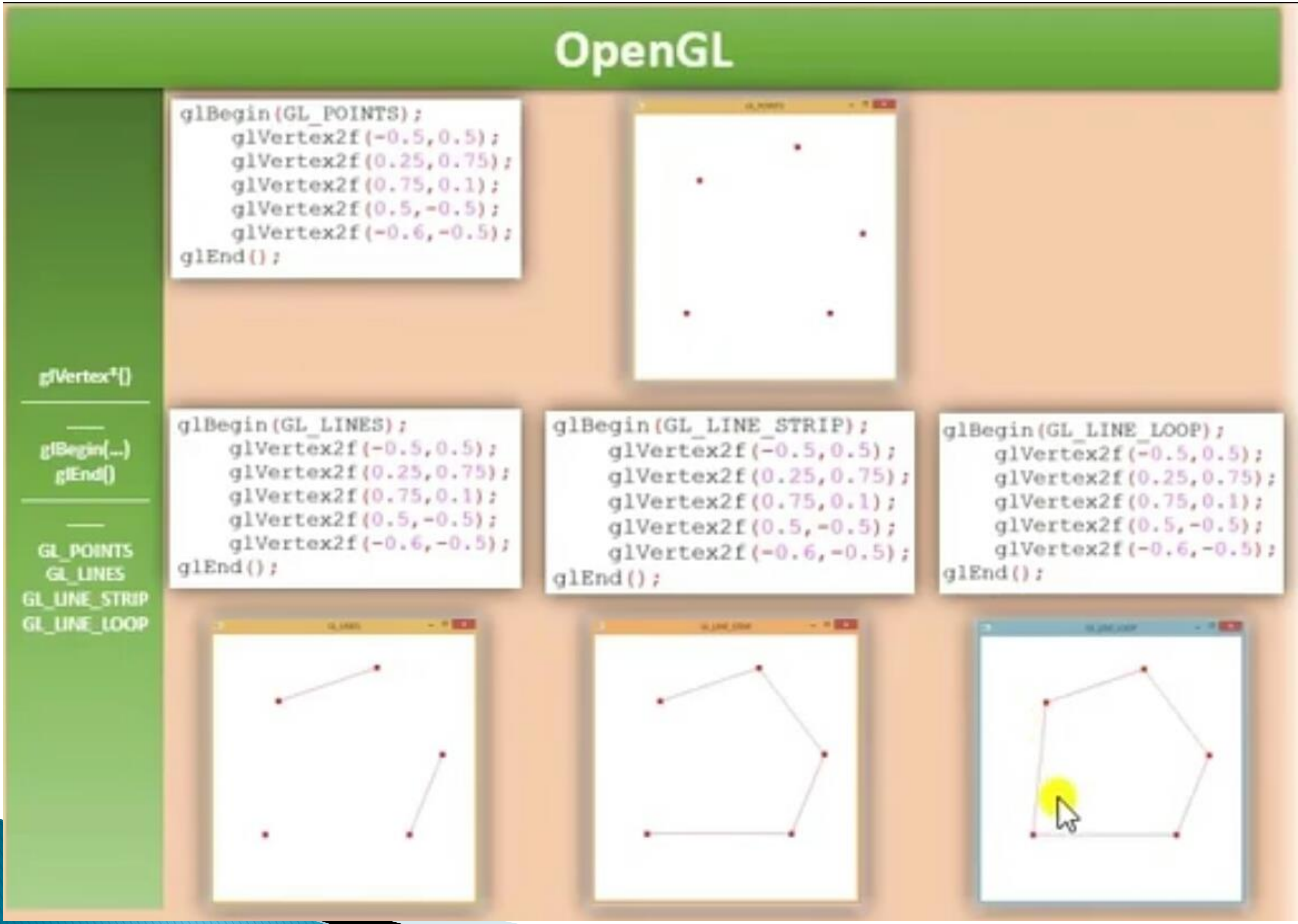
Attribute	OpenGL Functions
Point	Point Attributes
Line	Line Attributes
Curve	

OpenGL Attribute Functions



OpenGL	Attribute	OpenGL Functions
glVertex*() glBegin(...) glEnd() GL_POINTS GL_LINES GL_LINE_STRIP GL_LINE_LOOP	Point Line Curve	Point Attributes Line Attributes

-By Manjula. S



OpenGL

glVertex*()

glBegin(...)
glEnd()

GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_LINE_LOOP

```
glBegin(GL_POINTS);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.25,0.75);
    glVertex2f(0.75,0.1);
    glVertex2f(0.5,-0.5);
    glVertex2f(-0.6,-0.5);
glEnd();
```

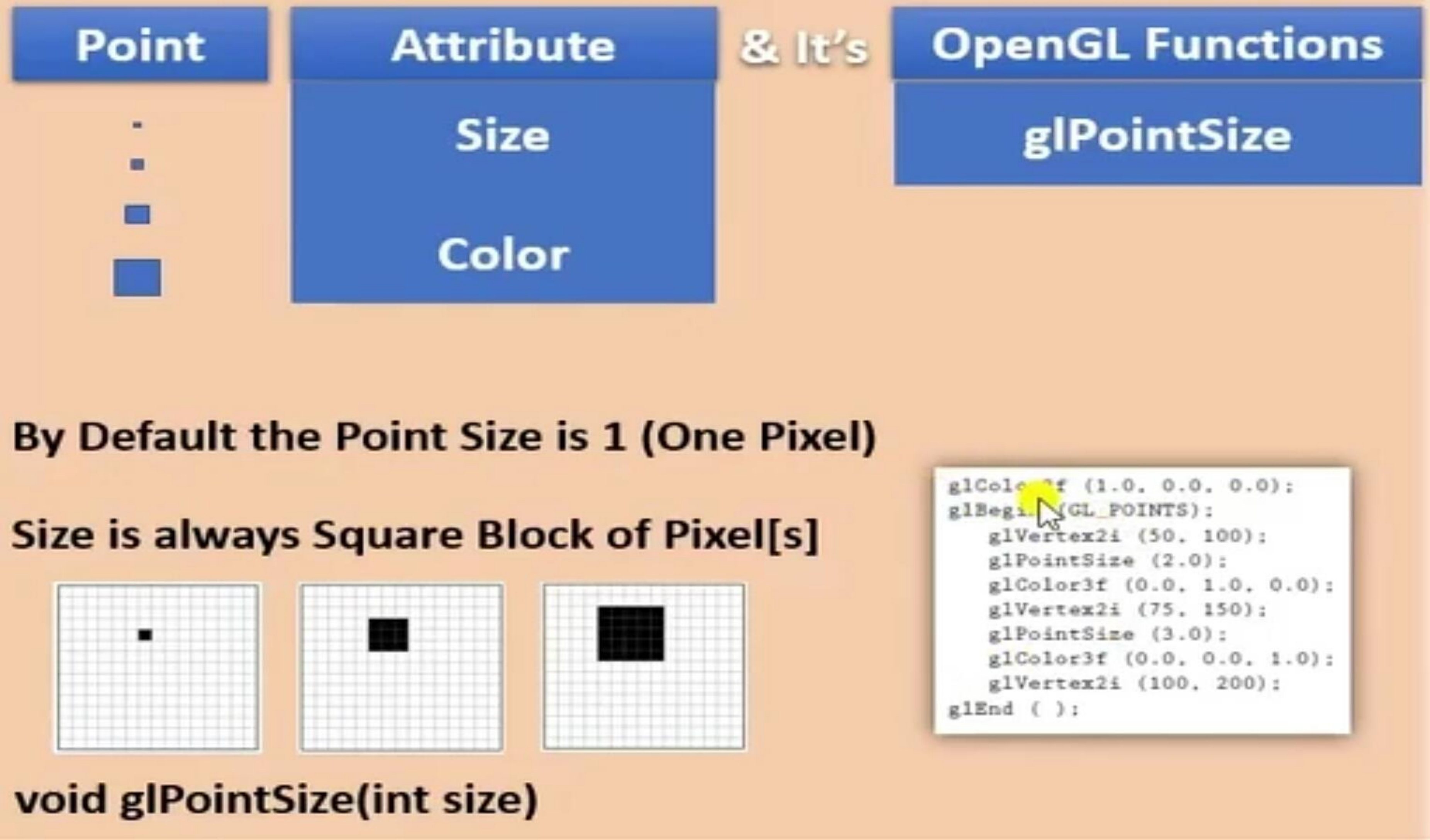
```
glBegin(GL_LINES);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.25,0.75);
    glVertex2f(0.75,0.1);
    glVertex2f(0.5,-0.5);
    glVertex2f(-0.6,-0.5);
glEnd();
```

```
glBegin(GL_LINE_STRIP);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.25,0.75);
    glVertex2f(0.75,0.1);
    glVertex2f(0.5,-0.5);
    glVertex2f(-0.6,-0.5);
glEnd();
```

```
glBegin(GL_LINE_LOOP);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.25,0.75);
    glVertex2f(0.75,0.1);
    glVertex2f(0.5,-0.5);
    glVertex2f(-0.6,-0.5);
glEnd();
```

-By Manjula. S

OpenGL Point Attribute Function



Point	Attribute	& It's	OpenGL Functions
	Size		glPointSize
	Color		

By Default the Point Size is 1 (One Pixel)

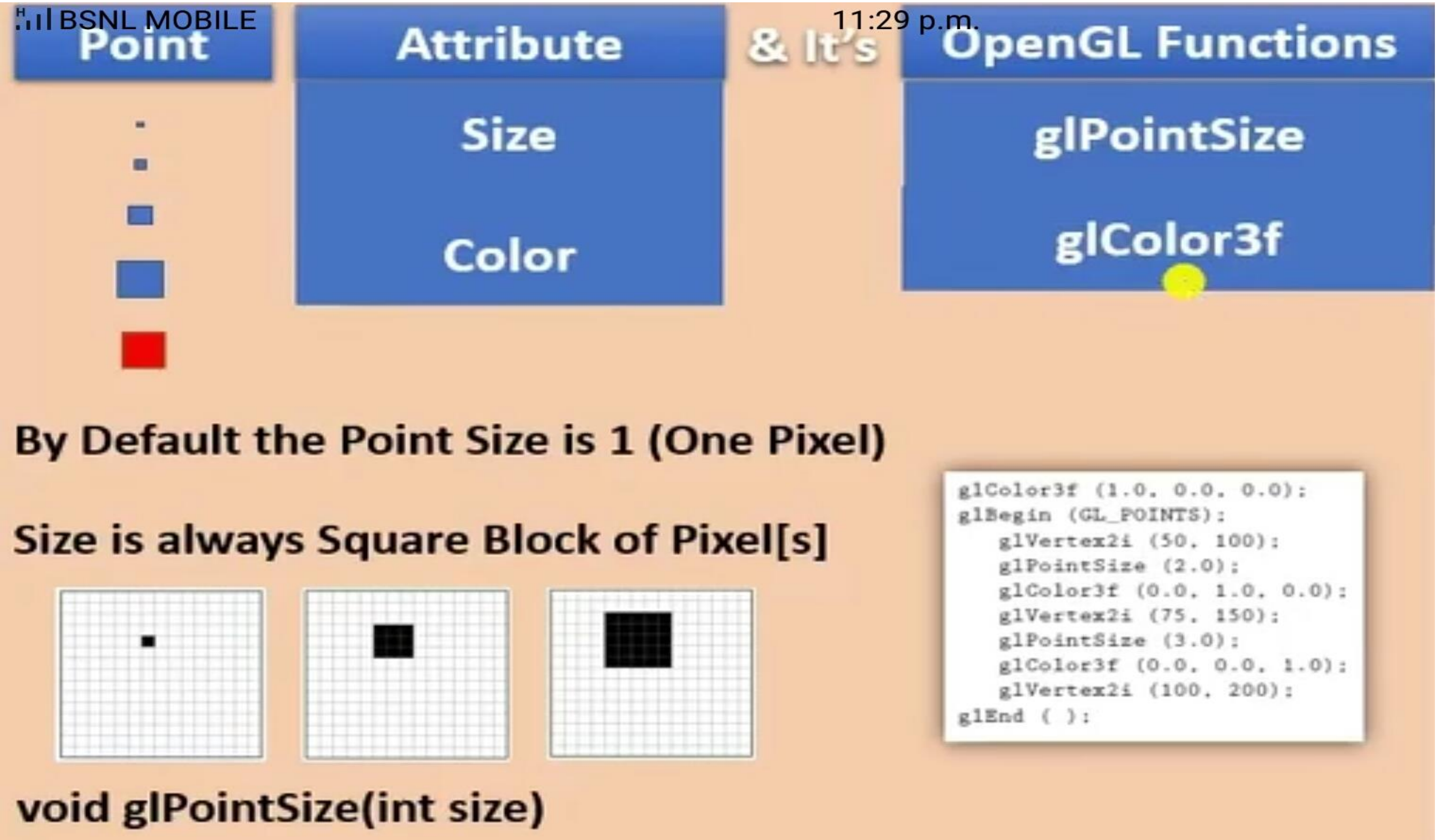
Size is always Square Block of Pixel[s]

```
glColor3f (1.0, 0.0, 0.0);
glBegin (GL_POINTS);
   glVertex2i (50, 100);
   glPointSize (2.0);
   glColor3f (0.0, 1.0, 0.0);
   glVertex2i (75, 150);
   glPointSize (3.0);
   glColor3f (0.0, 0.0, 1.0);
   glVertex2i (100, 200);
glEnd ( );
```

void glPointSize(int size)

-By Manjula. S

OpenGL Point Attribute Function



BSNL MOBILE 11:29 p.m.

Point	Attribute	& It's	OpenGL Functions
	Size		glPointSize
	Color		glColor3f

By Default the Point Size is 1 (One Pixel)

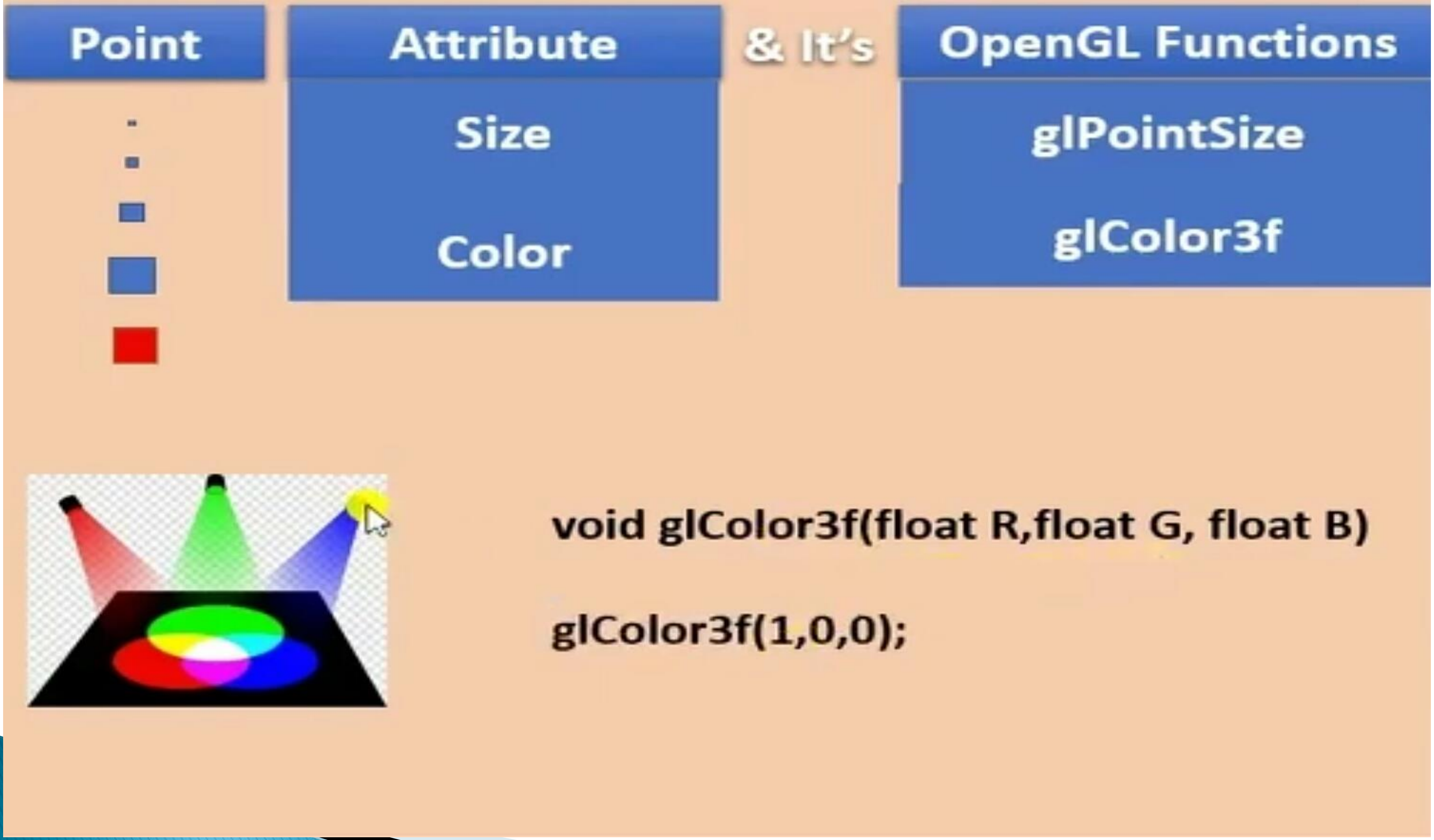
Size is always Square Block of Pixel[s]

```
glColor3f (1.0, 0.0, 0.0);
glBegin (GL_POINTS);
   glVertex2i (50, 100);
   glPointSize (2.0);
   glColor3f (0.0, 1.0, 0.0);
   glVertex2i (75, 150);
   glPointSize (3.0);
   glColor3f (0.0, 0.0, 1.0);
   glVertex2i (100, 200);
glEnd ( );
```

void glPointSize(int size)

-By Manjula. S

OpenGL Point Attribute Function



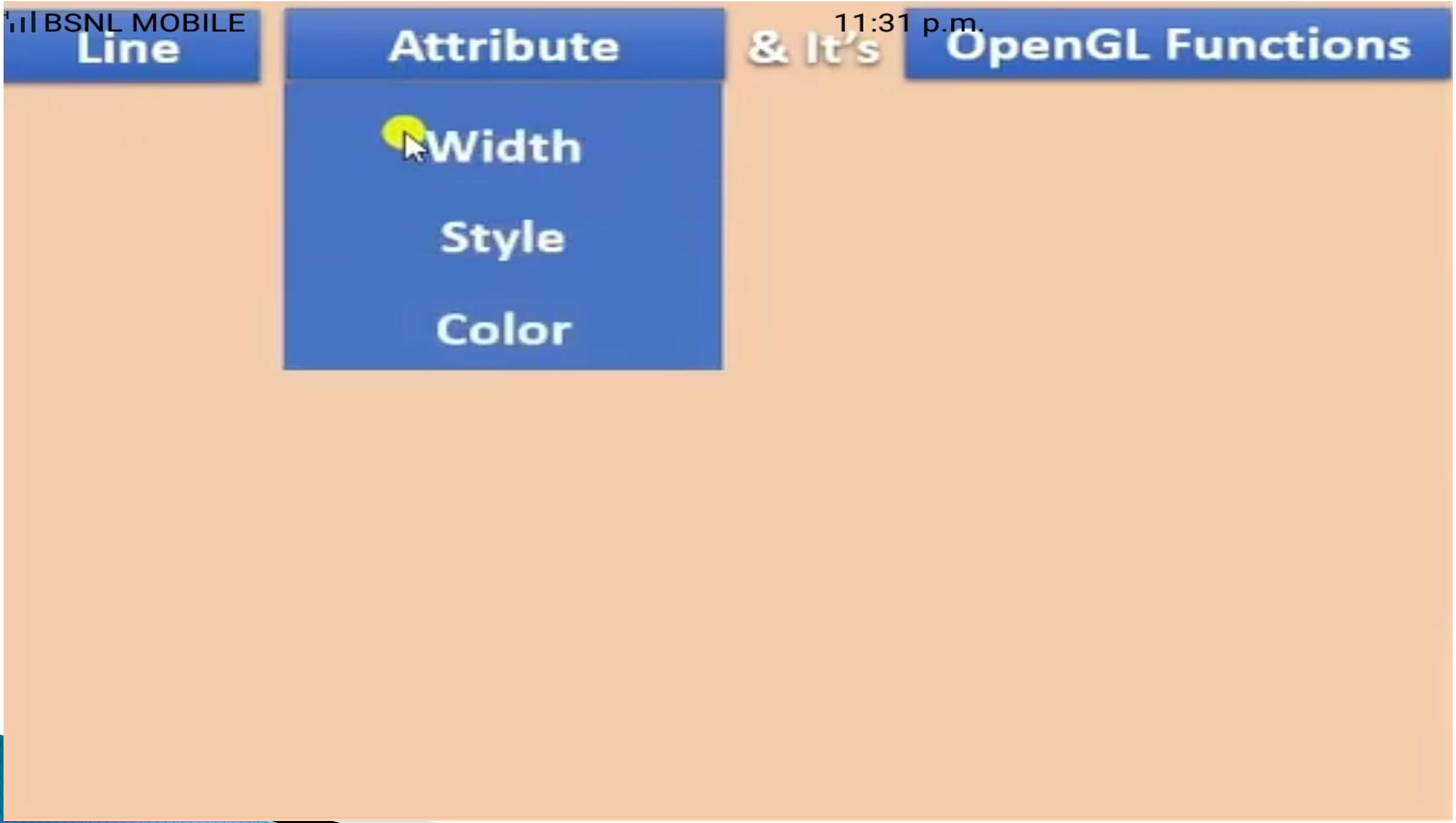
Point	Attribute	& It's	OpenGL Functions
	Size		glPointSize
	Color		glColor3f

```
void glColor3f(float R,float G, float B)

glColor3f(1,0,0);
```

-By Manjula. S

OpenGL Line Attribute Function



BSNL MOBILE 11:31 p.m.

Line Attribute & It's OpenGL Functions

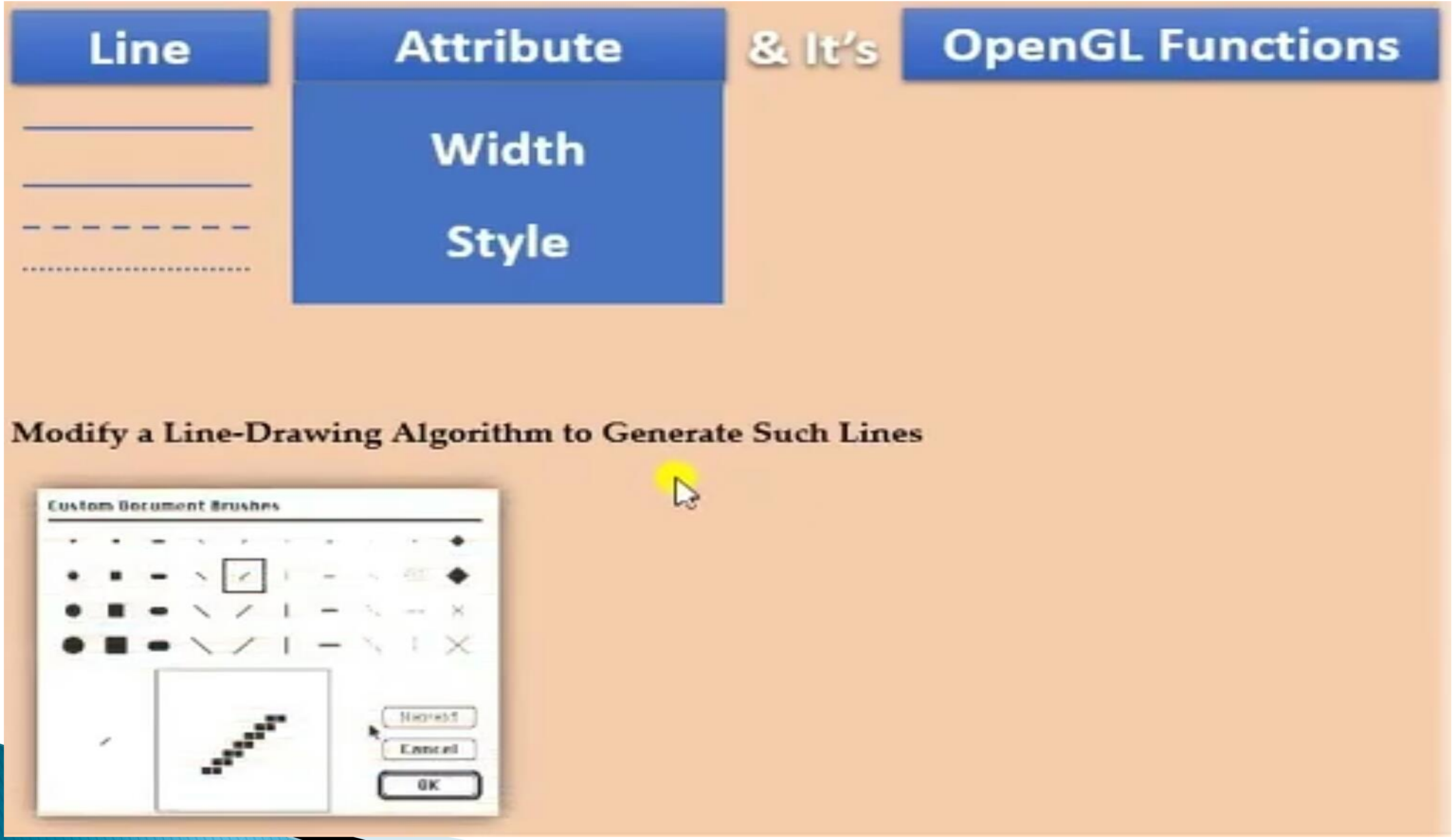
Width

Style

Color

-By Manjula. S

OpenGL Line Attribute Function



Line

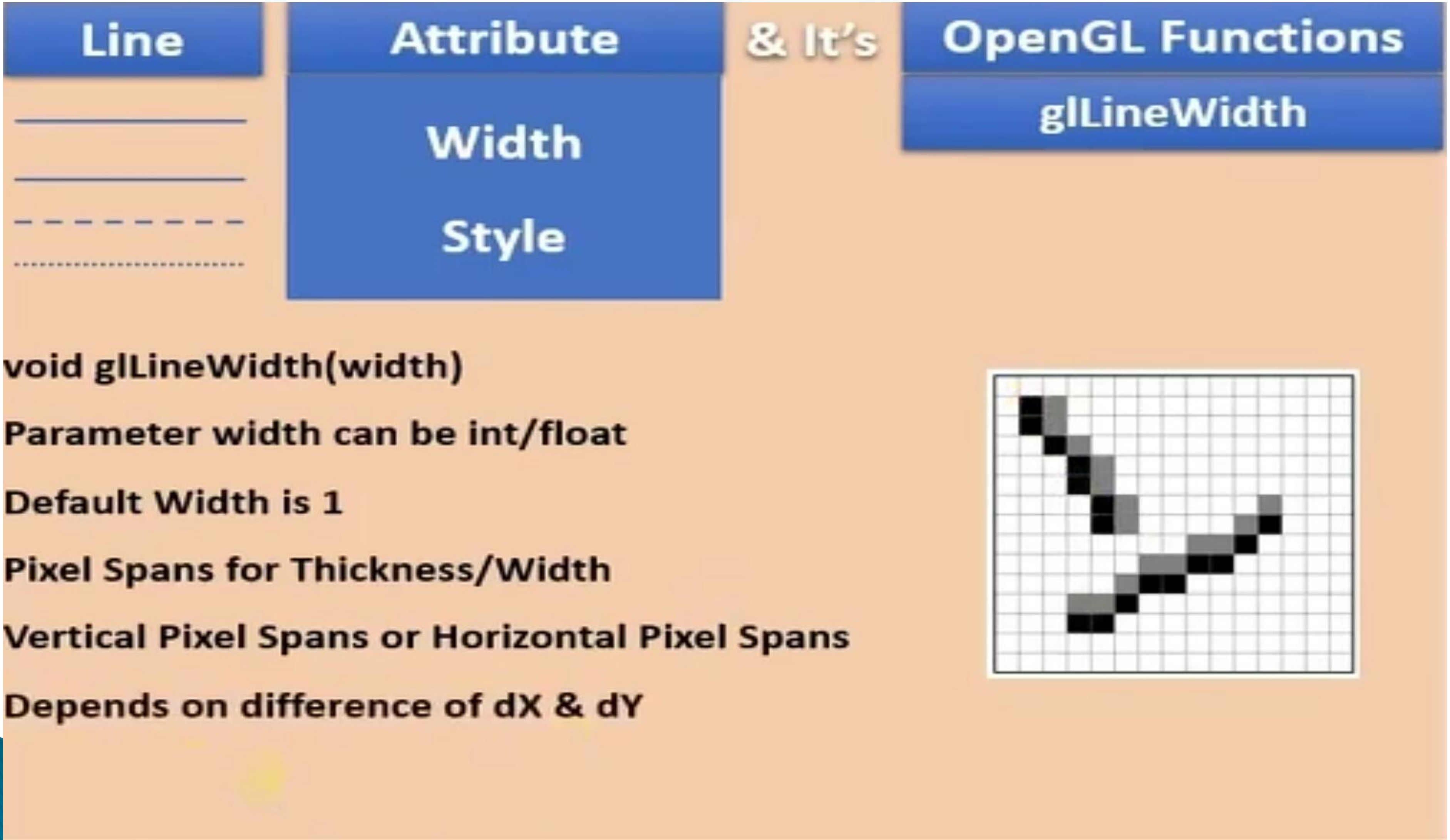
Attribute & It's OpenGL Functions

Width

Style

Modify a Line-Drawing Algorithm to Generate Such Lines

OpenGL Line Attribute Function



Line

Attribute

Width

Style

& It's

OpenGL Functions

glLineWidth

void glLineWidth(width)

Parameter width can be int/float

Default Width is 1

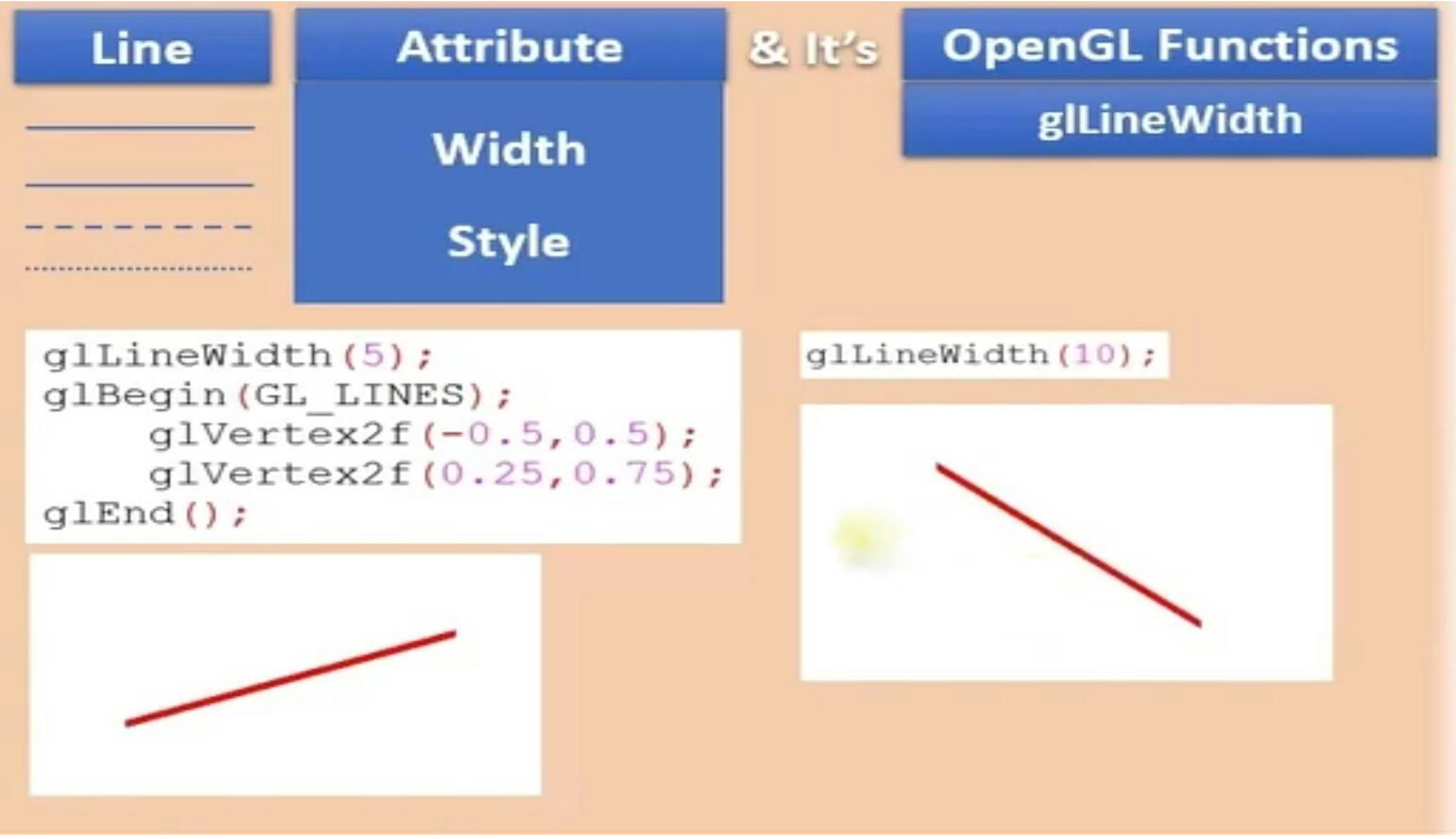
Pixel Spans for Thickness/Width

Vertical Pixel Spans or Horizontal Pixel Spans

Depends on difference of dX & dY

-By Manjula. S

OpenGL Line Attribute Function



Line

Attribute

& It's

OpenGL Functions

Width

Style

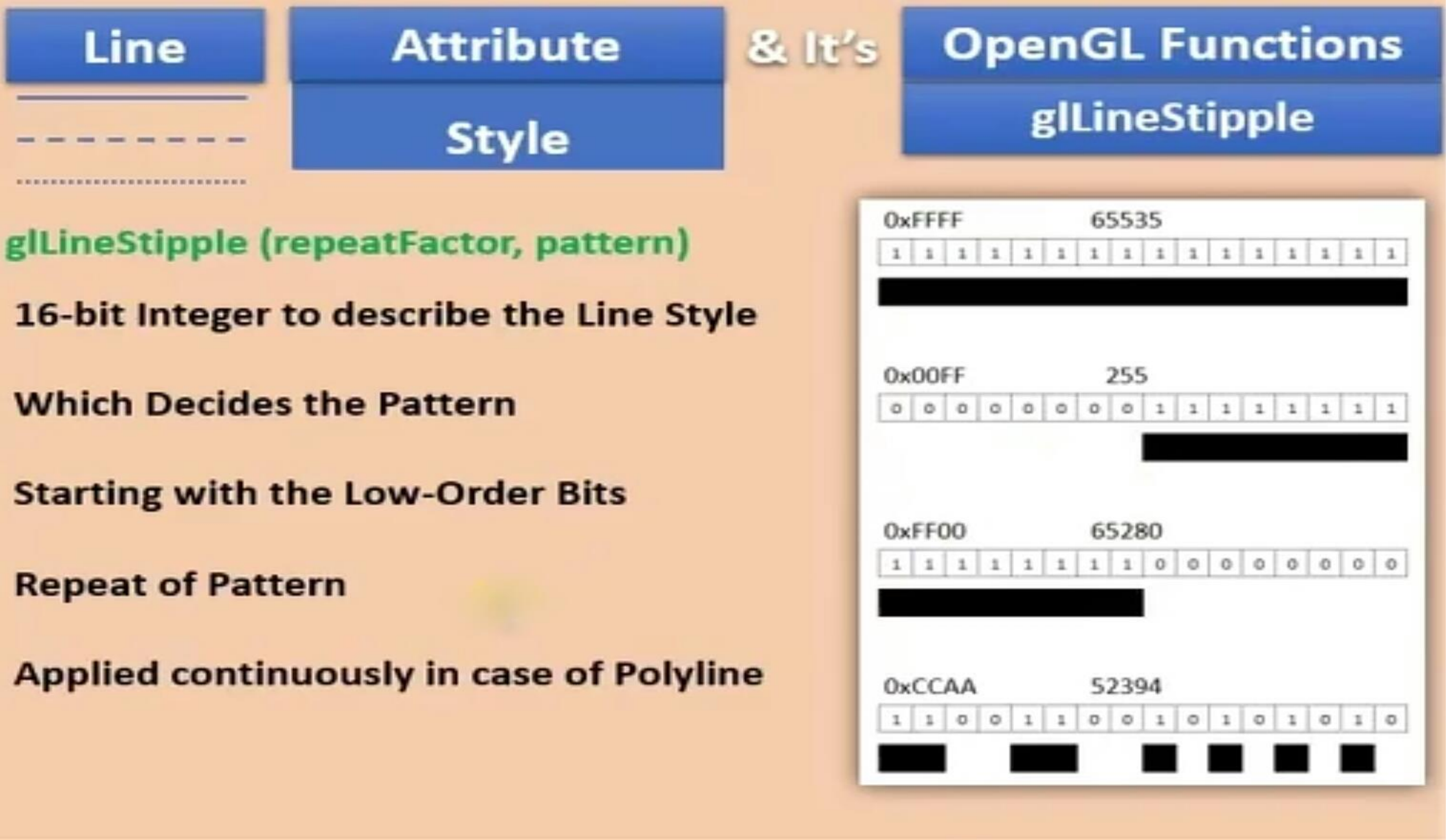
glLineWidth

```
glLineWidth(5);
glBegin(GL_LINES);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.25,0.75);
glEnd();
```

```
glLineWidth(10);
```

-By Manjula. S

OpenGL Line Attribute Function



Line | Attribute | Style | & It's | OpenGL Functions | glLineStipple

glLineStipple (repeatFactor, pattern)

16-bit Integer to describe the Line Style

Which Decides the Pattern

Starting with the Low-Order Bits

Repeat of Pattern

Applied continuously in case of Polyline

0xFFFF 65535

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0x00FF 255

0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

0xFF00 65280

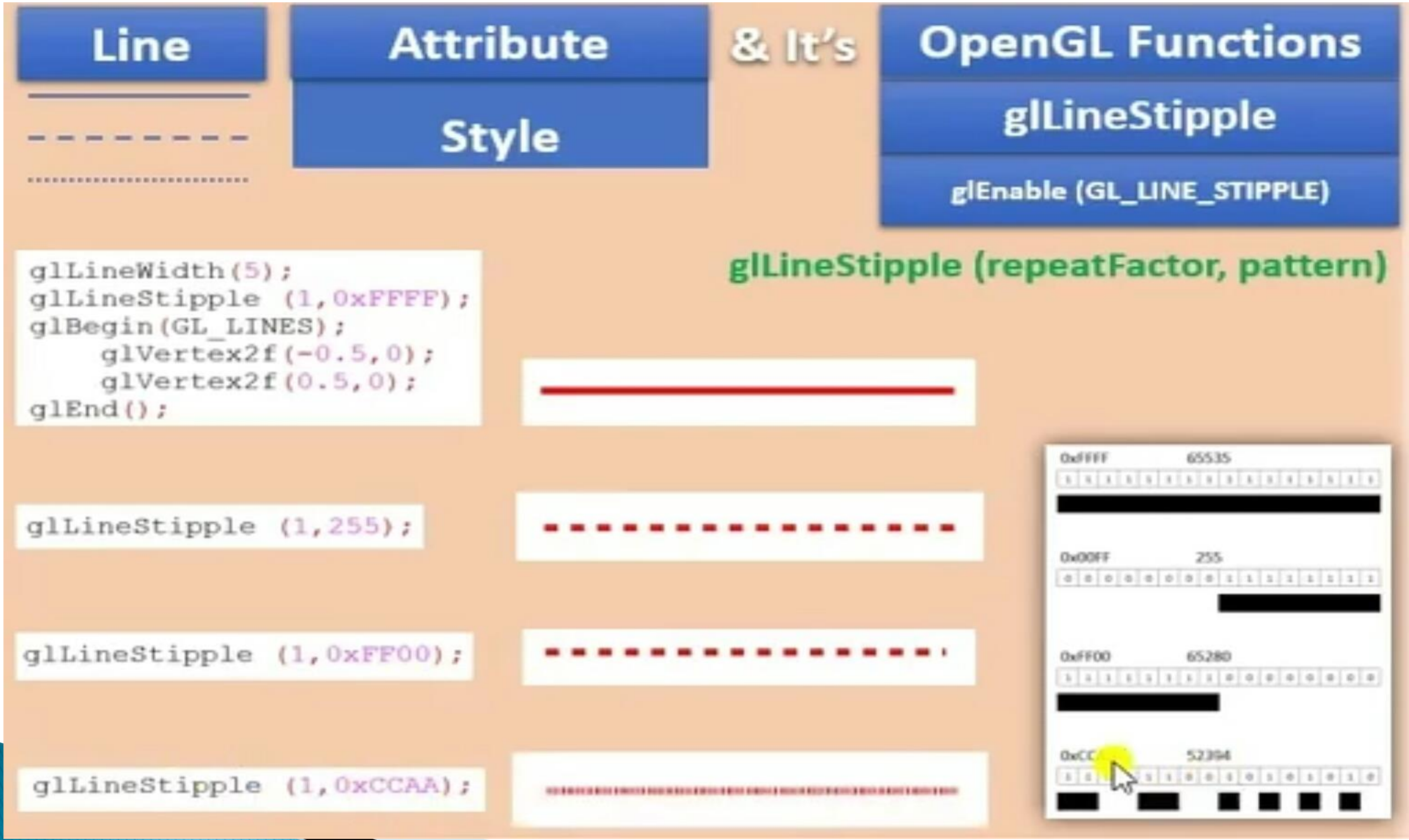
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

0xCCAA 52394

1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0

-By Manjula. S

OpenGL Line Attribute Function



Line | Attribute Style | & It's | OpenGL Functions

glLineStipple

glEnable (GL_LINE_STIPPLE)

```
glLineWidth(5);
glLineStipple (1,0xFFFF);
glBegin(GL_LINES);
    glVertex2f(-0.5,0);
    glVertex2f(0.5,0);
glEnd();
```

glLineStipple (repeatFactor, pattern)

```
glLineStipple (1,255);
```

```
glLineStipple (1,0xFF00);
```

```
glLineStipple (1,0xCCAA);
```

0xFFFF 65535

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0x00FF 255

0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

0xFF00 65280

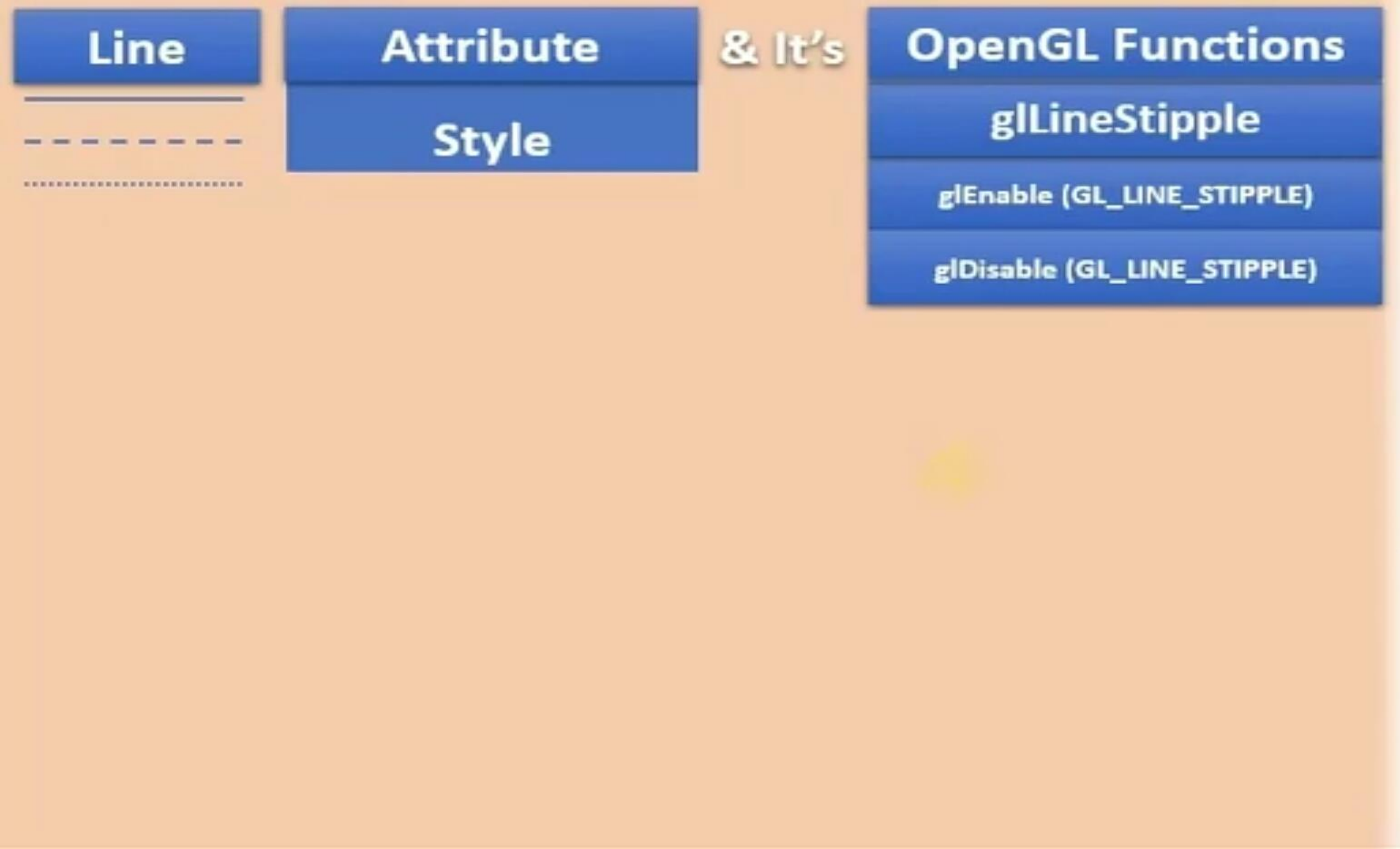
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

0xCC 52394

1 1 [illegible] [illegible] 1 1 0 0 1 0 1 0 1 0 1 0

-By Manjula. S

OpenGL Line Attribute Function



Line

Attribute & It's OpenGL Functions

Style

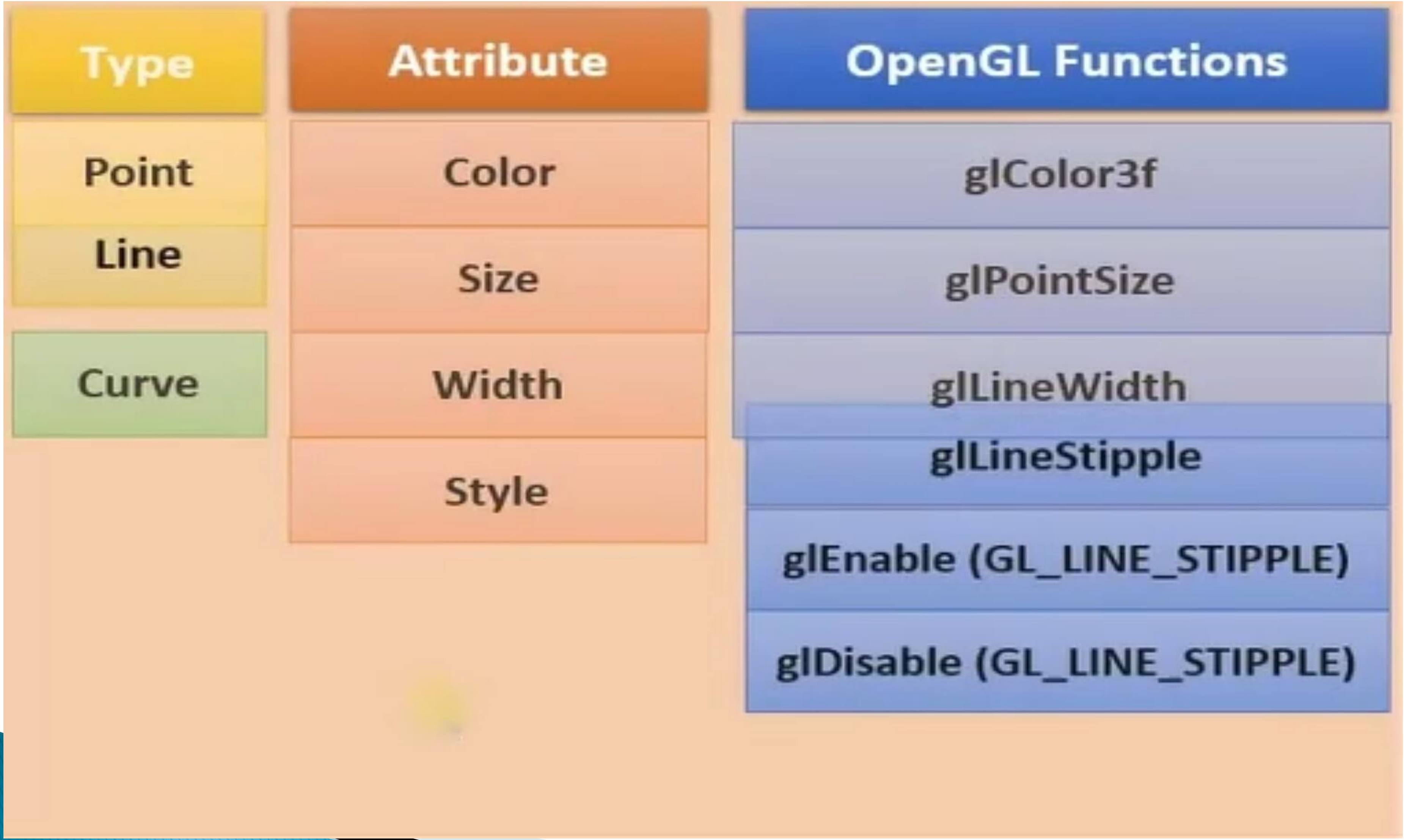
glLineStipple

glEnable (GL_LINE_STIPPLE)

glDisable (GL_LINE_STIPPLE)

-By Manjula. S

OpenGL Curve Attribute Function



Type	Attribute	OpenGL Functions
Point	Color	glColor3f
Line	Size	glPointSize
Curve	Width	glLineWidth
	Style	glLineStipple
		glEnable (GL_LINE_STIPPLE)
		glDisable (GL_LINE_STIPPLE)

-By Manjula. S

Thank You

-By Manjula. S