SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

OpenGL

By

Prof. Vaibhav P. Vasani

Assistant Professor

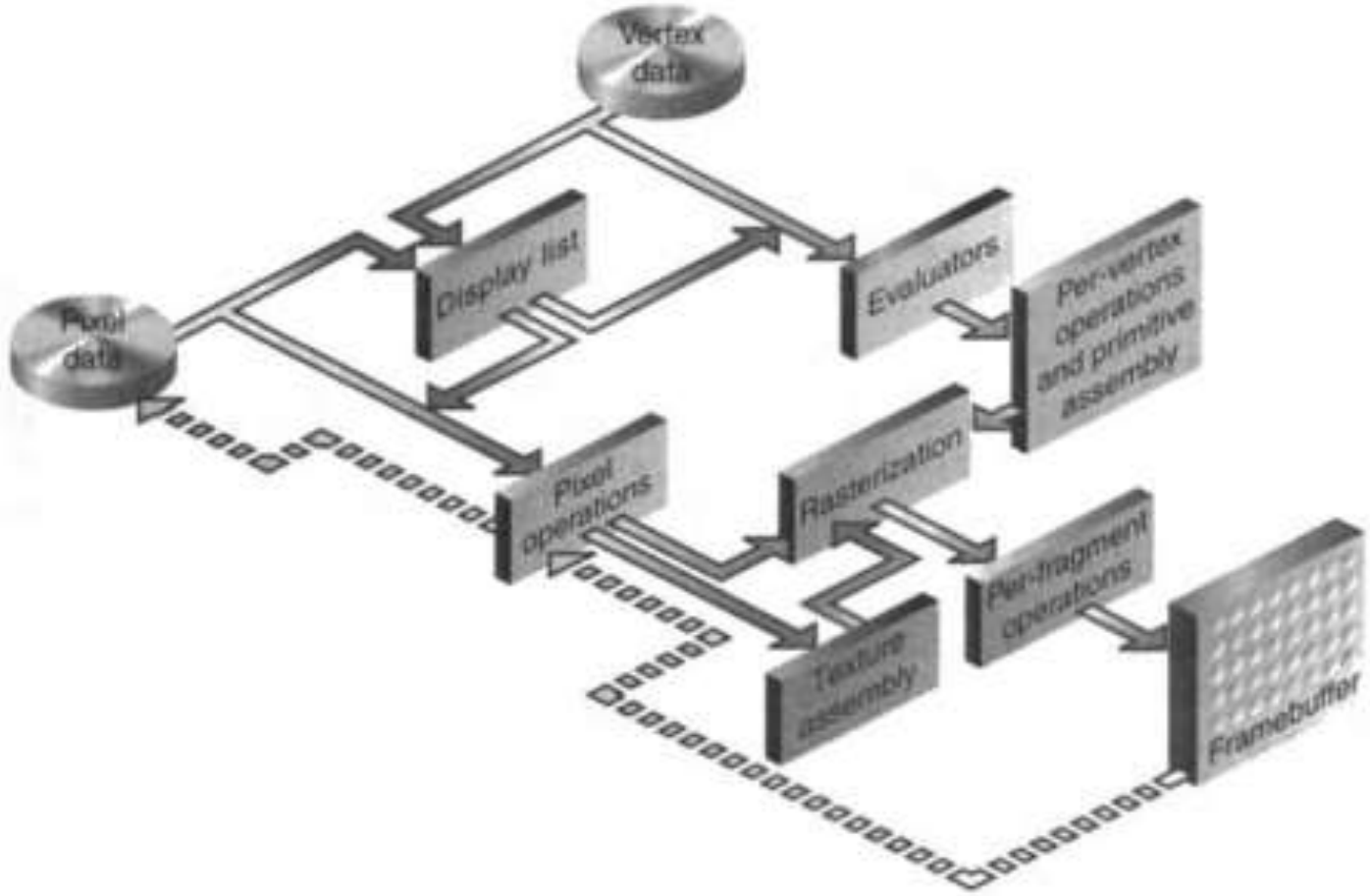
Department of Computer Engineering

K. J. Somaiya College of Engineering

Somaiya Vidyavihar University

Somaiya TRUST

OpenGL Rendering Pipeline



COLOR

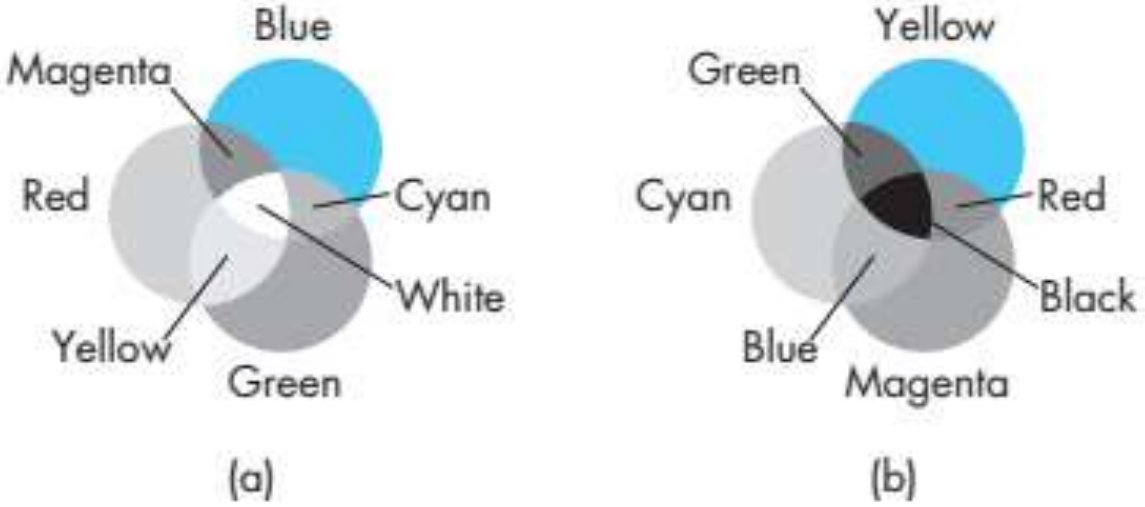


FIGURE 2.25 Color formation. (a) Additive color. (b) Subtractive color.

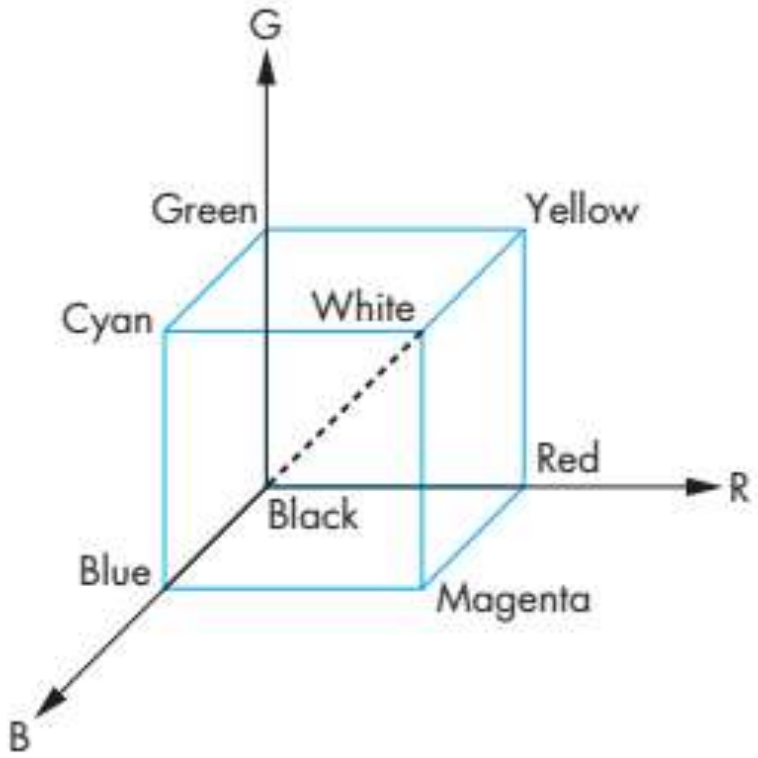
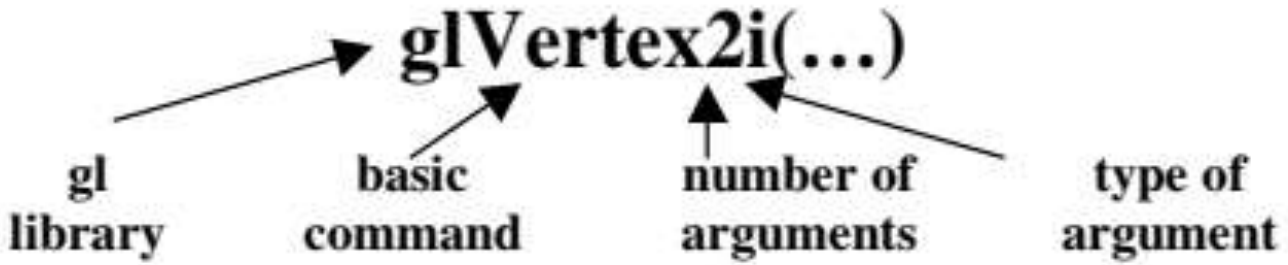


FIGURE 2.26 Color solid.

The vertices of the cube correspond to black (no primaries on);
red, green, and blue (one primary fully on);
the pairs of primaries, cyan (green and blue fully on),
magenta (red and blue fully on), and yellow (red and green fully on); and white (all primaries fully on).
The principal diagonal of the cube connects the origin (black) with white.

RGB Color model

The such function calls are formatted.



the type of argument, (i for an integer, f for a floating point value, etc

Command suffixes and argument data types.

suffix	data type	typical C or C++ type	OpenGL type name
b	**8-bit integer**	**signed char**	**GLbyte**
s	**16-bit integer**	**short**	**GLshort**
i	**32-bit integer**	**int or long**	**GLint, GLsizei**
f	**32-bit floating point**	**float**	**GLfloat, GLclampf**
d	**64-bit floating point**	**double**	**GLdouble,GLclampd**
ub	**8-bit unsigned number**	**unsigned char**	**GLubyte,GLboolean**
us	**16-bit unsigned number**	**unsigned short**	**GLushort**
ui	**32-bit unsigned number**	**unsigned int or unsigned long**	**GLuint,Glenum,GLbitfield**

The OpenGL "State"

- OpenGL keeps track of many *state variables*, such as the current "size" of points, the current color of drawing, the current background color, etc.
- The value of a state variable remains active until a new value

- glColor3f(red, green, blue)
  - where the values of red, green, and blue vary between 0.0 and 1.0.

- glColor3f(1.0, 0.0, 0.0) // set drawing color to red
- glColor3f(0.0, 0.0, 0.0) // set drawing color to black
- glColor3f(1.0, 1.0, 1.0) // set drawing color to white
- glColor3f(1.0, 1.0, 0.0) // set drawing color to yellow

- glClearColor(red, green, blue, alpha)
- The background color is set with glClearColor, where alpha specifies a degree of transparency or **opacity**
- Opacity values can range from
  - fully transparent (A=0.0) to
  - fully opaque (A=1.0).

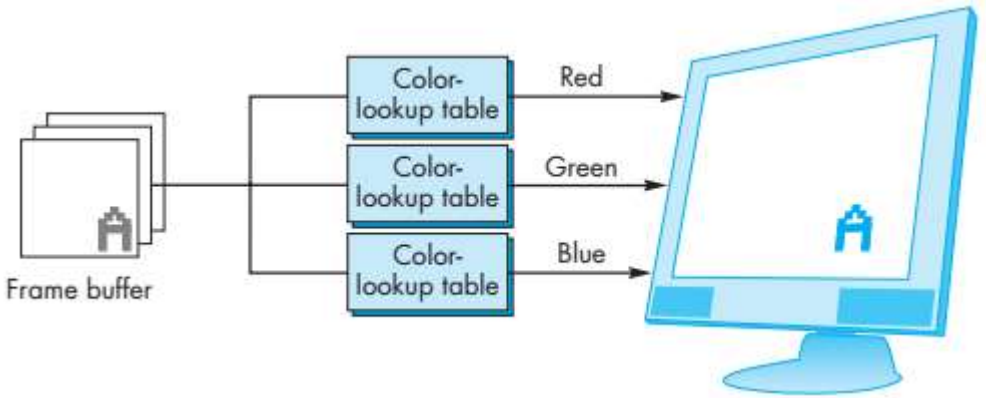
- To clear the entire window to the background color, use glClear(GL_COLOR_BUFFER_BIT). The argument GL_COLOR_BUFFER_BIT is another constant built into OpenGL.

Indexed Color

- Each pixel value or index is an integer between 0 and (2^*k*)- 1.
- Suppose each color component with a precision of *m* bits; then 2^*m* reds, 2^*m* greens, and 2^*m* blues.
- Hence, we can produce any of 2^(3*m)* colors on the display, but the frame buffer can specify only 2^*k* of them.

Input	Red	Green	Blue
0	0	0	0
1	$2^m - 1$	0	0
.	0	$2^m - 1$	0
.	.	.	.
.	.	.	.
$2^k - 1$	.	.	.

m bits | *m* bits | *m* bits



- Set the size of rendered points to be 2 pixels wide by using the following OpenGL function:
  - glPointSize(2.0)

The Orthographic View

- The simplest and OpenGL's default view is the orthographic projection.

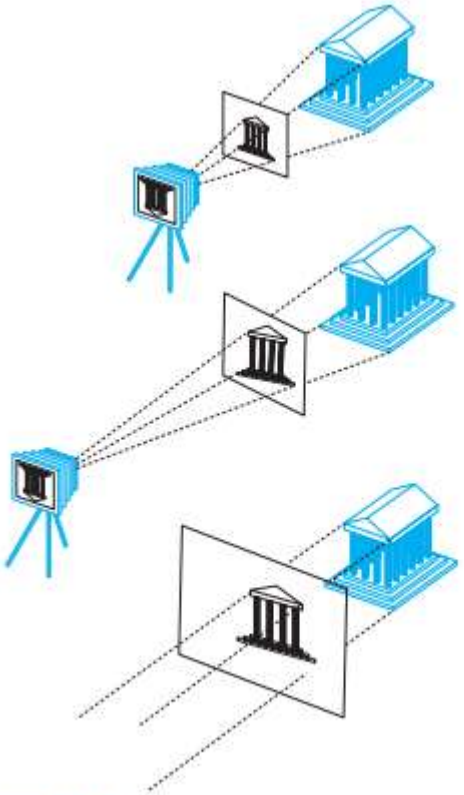


FIGURE 2.30 Creating an orthographic view by moving the camera away from the projection plane.

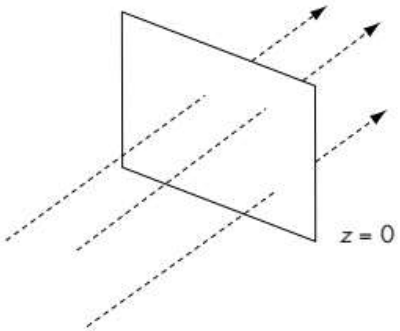


FIGURE 2.31 Orthographic projectors with projection plane $z = 0$.

Two-Dimensional Viewing

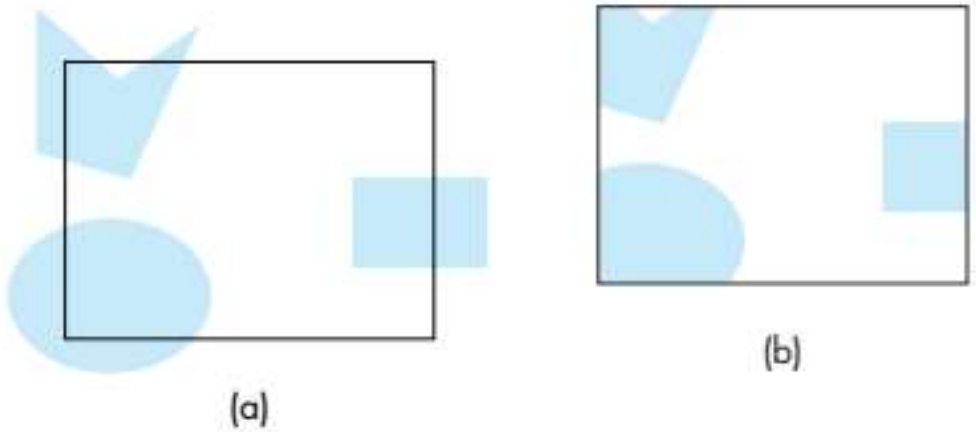
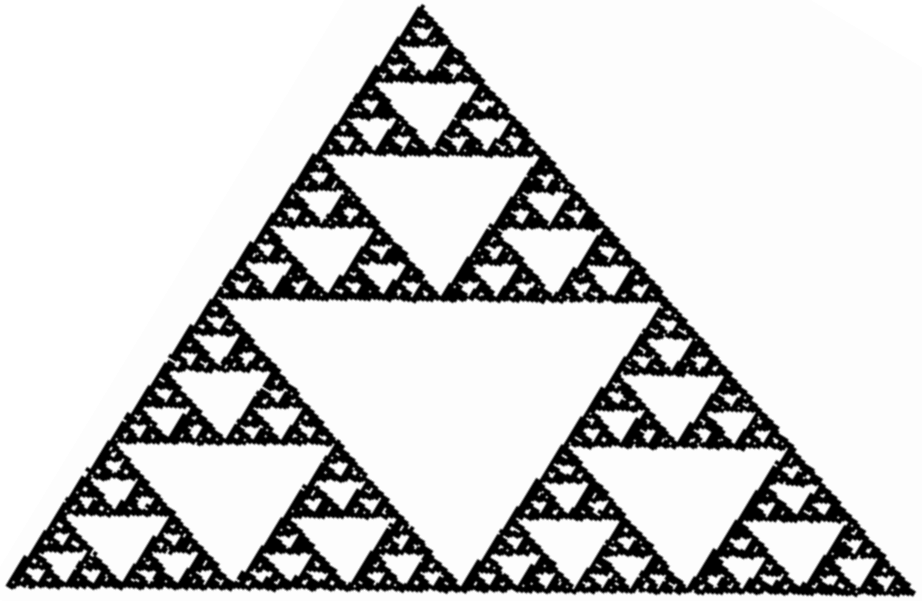


FIGURE 2.35 Two-dimensional viewing. (a) Objects before clipping. (b) Image after clipping.

OpenGL Utility Toolkit (GLUT)

- it is a library of functions that provides a simple interface between the systems.
- Details specific to the underlying windowing or operating system are inside the implementation, rather than being part of its API.

Drawing the Sierpinski Gasket



The Sierpinski gasket is produced by calling `drawDot()` many times with dot positions (x_0, y_0), (x_1, y_1), (x_2, y_2),.... determined by a simple algorithm. Denote the k-th point $p_k = (x_k, y_k)$. Each point is based on the previous point p_{k-1}.The procedure is:

1. Choose three fixed points T_0, T_1, and T_2 to form some triangle, as shown in Figure 2.13a.
2. Choose the initial point p_0 to be drawn by selecting one of the points T_0, T_1, and T_2 at random.

Now iterate steps 3-5 until the pattern is satisfyingly filled in:

3. Chose one of the three points T_0, T_1, and T_2 at random; call it T.
4. Construct the next point p_k as the **midpoint**[7] between T and the previously found point p_{k-1}. Hence

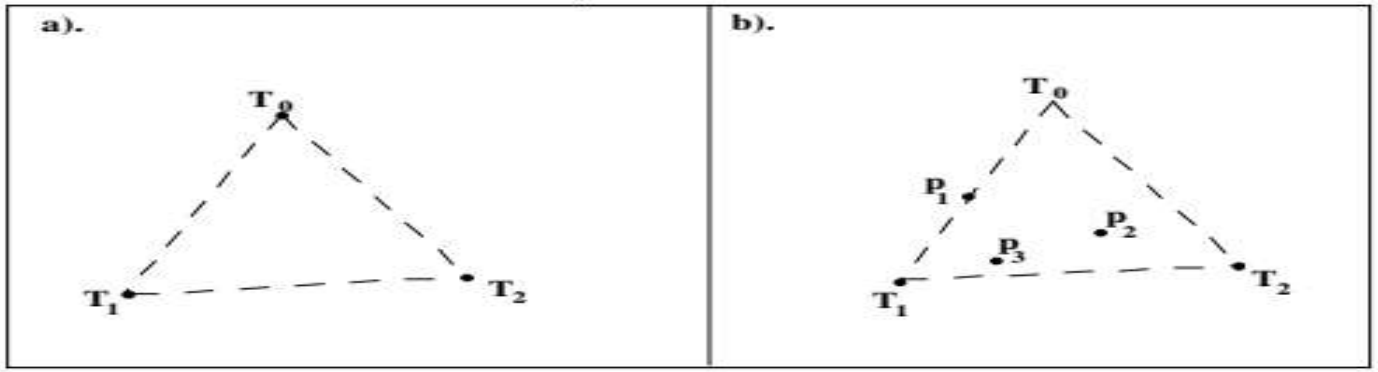


Figure 2.13. Building the Sierpinski gasket.

p_k = midpoint of p_{k-1} and T,

5. Draw p_k using `drawDot()`.

Establishing the Coordinate System

- glVertex: a function that defines a vertex
- glVertex3: a function that defines a vertex using 3 coordinates
- glVertex3f: a function that defines a vertex using 3 coordinates of type GLfloat
- glVertex3fv: a function that defines a vertex using 3 coordinates of type GLfloat which are put inside a vector (tuple) (the alternative would be glVertex3fl which uses a list of arguments instead of a vector)

Thank you