**RGB color components**

Color information can be stored in the frame buffer in two ways:

* Store RGB color code in frame buffer
* Put color codes in separate table and use pixel locations to store index values for color table entries.

Color tables are an alternate for providing extended color capabilities to user without requiring large frame buffers.

**Color table**

Values in the frame buffer are used as indices into the color table.

Color tables provide reasonable number of simultaneous colors without requiring large frame buffers.

Without color table we can change color of a pixel by storing the new color at frame buffer location.

**OpenGL RGB and RGBA color modes**

The difference between RGB and RGBA is whether we are using alpha value for blending.

When we change the color in OpenGL we define color state of OpenGL, which is applied to all subsequent primitives until color setting is changed.

Important application of color blending is the simulation of transparency effect.

glColor3f (0.0, 0.0, 1.0);

glColor3i (0,255,0);

frame buffer positions store integer values, specifying the color values as integers avoid the conversion necessary when floating point values are given.

**OpenGL color Index Mode**

Color specification in OpenGL can be given in color index mode.

Specify color using glIndex \*(colorIndex)

e.g., glIndexi(196);

we can set the color into the table at a given index position

glutSetColor( index, red, green, blue);

**OpenGL point attributes**

Display the point as a square block of pixels, default point size is 1.0

glPointSize(size);

**Polygon edge table (ch4, section 4-10, page 199)**

For efficient polygon fill store polygon boundary in sorted edge table.

Each entry contains maximum y value for the edge, x-intercept value for the lower y, and the inverse slope.

For each scan line, the edges are sorted from left to right.

See page 200 for example

**Antialiasing**

Line segments and other graphics primitives generated by raster algorithms in ch3 have a jagged or stair-step appearance. This distortion of information due to low frequency sampling is called aliasing. We can improve the appearance of displayed raster lines by applying antialiasing methods.

One way to increase sampling rate with raster systems is to display objects at higher resolution. Unless hardware technology is developed to handle arbitrarily large frame buffers, increased screen resolution is not a complete solution to the aliasing problem.

**Supersampling**: Antialiasing method in which we increase sampling rate by treating the screen as if it were covered with a finer grid than is actually available. Then we use multiple sample points across this finer grid to determine appropriate intensity level for screen pixel (combine the result from subpixels).

**OpenGL Query functions**

We can retrieve current values for any of the state parameters using OpenGL query functions. These functions copy specified state values into an array.

glGetBolleanv(); glGetFloatv(); glGetIntegerv(); glGetDoublev();

each function take 2 parameters, OpenGL constant that identify attribute or state and a pointer to array of data type indicated by the function name.

glGetFloatv (GL\_CURRENT\_COLOR, colorValues);// return current color components

we can check point size, frame buffer bits per pixel, check if the system support double buffering and many other stats.

**OpenGL Attribute Groups**

Attributes and other OpenGL state parameters are arranged in attribute groups, and each group contains a set of related state parameters.

About twenty different attribute groups are available in OpenGL, and all parameters in one or more groups can be saved or reset with a single function.

We save all the group parameters using the command:

glPushAttrib(attGroup);

e.g., glPushAttribu(GL\_POINT\_BIT);// size and point smooth parameters are saved

we can reinstate all the values on the attribute stack with the function

glPopAttrib();