



bitmap is a matrix of 0s and 1s (bits)

image is a matrix of pixels (sometimes called *pixmap*)

OpenGL does NOT support any specific format (GIF, JPG).

Image must be ready as a matrix in the memory

How can we read an image?

...use some external reader...



OpenGL uses a virtual pointer (raster position)
for an image reading and rendering

```
glRasterPos{234}{sifd}[v](TYPE x,TYPE y,TYPE z,TYPE w)
```

coordinates are transformed exactly as a vertex

```
glGetFloatv(GL_CURRENT_RASTER_POSITION, pos)
```

returns current raster position

```
glGetBoolean(GL_CURRENT_RASTER_POSITION_VALID,result)
```

says, whether the position has been clipped out or not



```
void glBitmap(GLsizei w,GLsizei h,GLfloat x, GLfloat y,  
GLfloat xx, GLfloat yy,const GLubyte *bitmap);
```

w and **h** determines width and height of **bitmap**

x and **y** is location of origin in the bitmap (useful for fonts)

xx and **yy** is the new raster position

bitmaps are intended mainly for font definition



```
void glDrawPixels(GLsizei w, GLsizei h,  
GLenum format, GLenum type, GLvoid *dta)
```

w and **h** determines width and height of **bitmap**

format indicates the kind of pixel data element i.e.,

GL_RGB, GL_RGBA, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA
GL_LUMINANCE - a single luminance component

GL_LUMINANCE_ALPHA - luminance followed by alpha

GL_STENCIL_INDEX - a single stencil index

GL_DEPTH_COMPONENT - a single depth component

type defines a data type i.e.,

GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP,

GL_UNSIGNED_SHORT, GL_SHORT,

GL_UNSIGNED_INT, GL_INT, GL_FLOAT



```
void glReadPixels(GLint x,GLint y,GLsizei w,
GLsizei h, GLenum format, GLenum type, GLvoid *dta)
```

x and **y** determines the current position
w and **h** determines width and height of **bitmap**
the rest is equal

- ~ It *reads* the screen and saves it into the memory
- ~ Does not allocate the memory!
- ~ x and y can be *outside* the current window



```
void glCopyPixels(GLint x,GLint y,
GLsizei w, GLsizei h, GLenum type)
```

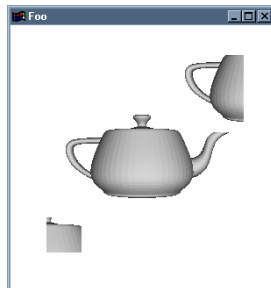
x and **y** determines the current position of *reading*
w and **h** determines width and height of **bitmap**
image is written from the current raster position

- ~ It *reads* the pixels and displays them immediately
- ~ x and y can be *outside* the current window

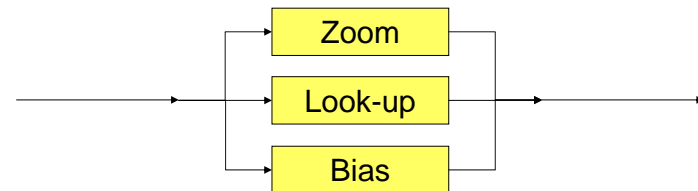


Example:

```
glutSolidTeapot(0.8);
glRasterPos2f(0.5,0.5); //set position for writing
glCopyPixels(50,100,80,80,GL_COLOR); //copy image
glRasterPos2f(-1.5,-1.5);
glCopyPixels(150,150,40,40,GL_COLOR);
```



Pixels while read, copied, or written,
can be manipulated in some way



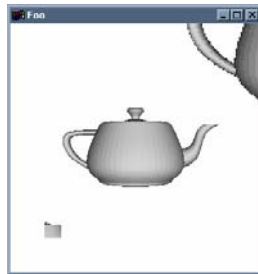


```
void glPixelZoom(GLfloat x, GLfloat y)
```

the magnification or reduction factor pixel-write operation

Example:

```
glutSolidTeapot(0.8); //render teapot
glRasterPos2f(0.5,0.5);
glPixelZoom(2,2); //zoom area
glCopyPixels(50,100,80,80,GL_COLOR);
glPixelZoom(0.5,0.5);
glRasterPos2f(-1.5,-1.5);
glCopyPixels(150,150,40,40,GL_COLOR);
```



```
void glPixelTransfer{if}(GLenum pname,TYPE param)
```

step 1) Enabling the color mapping

```
GL_MAP_COLOR,_STENCIL param is GLboolean
```

step 2) Defining the color map

```
GL_RED(GREEN,BLUE,DEPTH,ALPHA)_SCALE
```

param is GLfloat specifies scale of the component

```
GL_RED(GREEN,BLUE,DEPTH,ALPHA)_BIAS
```

param is GLfloat specifies bias of the component



Example:

```
//upper image
glRasterPos2f(0.5,0.5);
glPixelTransferf(GL_BLUE_SCALE,2.7);
glCopyPixels(50,100,80,80,GL_COLOR)
```

```
//lower image
glRasterPos2f(-1.5,-1.5);
glPixelTransferf(GL_BLUE_SCALE, 1);
glPixelTransferf(GL_RED_BIAS,-0.2);
glPixelTransferf(GL_BLUE_BIAS,-2);
glCopyPixels(150,150,40,40,GL_COLOR);
```



Setting the look-up table

```
void glPixelMap{ui us f}v(GLenum map,
                           GLint size, const TYPE *array)
```

map is

```
GL_PIXEL_MAP_R_TO_R(G_TO_G, B_TO_B, A_TO_A)
```

specifies what is mapped where or not

size specifies array extent

this operation must be enabled

```
glPixelTransferf(GL_MAP_COLOR, GL_TRUE);
```

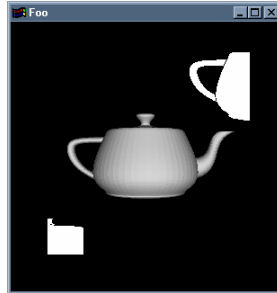


Example - thresholding:

```
static GLfloat array[256]; //the look-up table
```

```
for (i=128;i<256;i++) array[i]=1.0; //half ones
glPixelTransferi(GL_MAP_COLOR, GL_TRUE);
glPixelMapfv(GL_PIXEL_MAP_R_TO_R,256,array);
glPixelMapfv(GL_PIXEL_MAP_G_TO_G,256,array);
glPixelMapfv(GL_PIXEL_MAP_B_TO_B,256,array);
```

```
glRasterPos2f(0.5,0.5);
glCopyPixels(50,100,80,80,GL_COLOR);
glRasterPos2f(-1.5,-1.5);
glCopyPixels(150,150,40,40,GL_COLOR);
```



13



- Jackie Neider, Tom Davis, Mason Woo
OpenGL Programming Guide,
Addison-Wesley Publication Company
ON LINE at <http://www.opengl.org.ru/docs/>
- www.opengl.org/developers/code/tutorials.html
- SIGGRAPH 2001
An Interactive Introduction To OpenGL Programming
www.opengl.org/developers/code/s2001/index.html
- SIGGRAPH '99
Lighting and Shading Techniques for Interactive Applications
www.opengl.org/developers/code/sig99/index.html