Colors in OpenGL

- Two color models:
  RGB or RGBA (GLUT_RGB) and **Color-Index** (GLUT_INDEX)
- Both use the Color Cube to hide differences in hardware
  - (R,G,B) triplet where 0.0 = none & 1.0 = full value
- **A** or **alpha** in RGB**A** expresses ***opacity***
  - 0.0 = fully transparent object and 1.0= fully opaque object
- glColor3tv(), glColor4tv(): sets the present drawing color
  - t captures data type of arguments (**b**,i,f,d) **b**: byte
  - v passing arguments by reference rather than by value
- glClearColor(): sets the present clear color used by glClear()
  - E.g., glClearColor(1.0,1.0,0.0,1.0)

Indexed Color

- Overcomes limited-depth pixels
- Use the pixels as indices to a ***color-lookup table*** (***palette***)
- *K=M=8, there are 256* colors in the table

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

- glIndexi(element)
  - selects the present drawing color
- glutSetColor(int element, GLfloat R, Glfloat G, GLFloat B)
  - sets the entries in a color table for each window

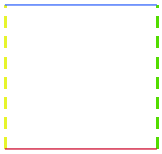
"White Rectangle"

```
glLineWidth(1.0);
glLineStipple(1, 0xFFFF);
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINES);
  glVertex2f(-30, -30);
  glVertex2f(30, -30);
glEnd();
```

```
glLineWidth(1.0);
glLineStipple(1, 0xFFFF);
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINES);
  glVertex2f(30, 30);
  glVertex2f(-30, 30);
glEnd();
```

```
glLineWidth(2.0);
glLineStipple(2, 0xCCCC);
glColor3f(0.0, 1.0, 0.0);
glBegin(GL_LINES);
  glVertex2f(30, -30);
  glVertex2f(30, 30);
glEnd();
```

```
glLineWidth(2.0);
glLineStipple(2, 0xCCCC);
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_LINES);
  glVertex2f(-30, 30);
  glVertex2f(-30, -30);
glEnd();
```



Color Interpolation

```
glBegin(GL_LINES);
  glColor3f(1.0, 0.0, 0.0);
  glVertex2f(0.0, 0.0);
  glColor3f(0.0, 0.0, 1.0);
  glVertex2f(0.1,0.0);
glEnd();
```

- Vertex color
  - Smooth shading (GL_SMOOTH)
    - interpolate the vertex colors
  - Flat shading (GL_flat)
    - Use the color of the first vertex
- Polygons (convex): vertex colors are interpolated for the interior
  - *bilinear interpolation* applied on triangles
- glShadeMode(Glenum mode);
  - GL_SMOOTH is the default

Two-Dimensional Viewing

- What objects are displayed is determined by the ***viewing*** or ***clipping rectangle***.
- Clipping window can be specified using the GLU function:

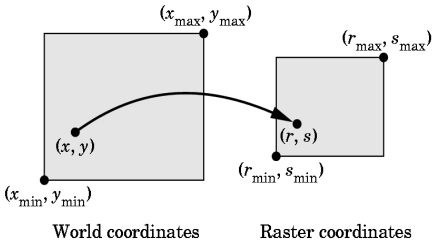
```
void gluOrtho2D(GLdouble left, GLdouble right,
                GLdouble bottom, GLdouble top)
```

- Default is a square 2x2:
  - (left,bottom) = (-1.0, -1.0) and (right, top) = (1.0, 1.0)
- Clipping window is a special case of 3D ***viewing volume*** defined by the orthographic projection on the plane z=0

```
void glOrtho (GLdouble left, GLdouble right,
              GLdouble bottom, GLdouble top,
              GLdouble near, GLdouble far)
```


Coordinate Systems & Matrix Modes

- ***Object*** or ***world*** coordinates
- ***Window*** coordinates, ***screen*** or **raster** coordinates
- OpenGL automatically transforms *object* to *window* coordinates
  - glutInitWindowSize(), gluOrtho2D()



- Two transformation matrices: ***model-view*** and ***projection***

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-1.0,1.0,-1.0,1.0);
glMatrixMode(GL_MODELVIEW);
```

Saving the State

- Matrix stacks – one for each type

  `void glPushMatrix();  void glPopMatrix();`

- E.g.,

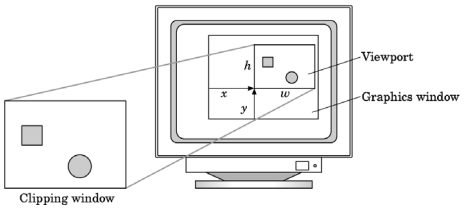
```
glMatrixMode(GL_PROJECTION);
glIndentity();
glutPostRedisplay();
glPushMatrix();
ZOOM();
glutPostRedisplay();
glPopMatrix();
```

- Attribute stack – 20 groups of attributes
  - `GL_POLYCON_BIT, GL_LINE_BIT, GL_ALL_ATTRIBUTE_BITS`

  `void glPushAttrib(Glbitfield mask);`
  `void glPopAttrib();`

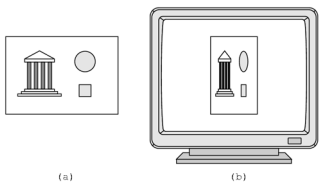
Viewport

- Viewport allows us to split a window into different parts and restrict drawing within a given part with its own `gluOrtho2D()` properties.



- This is very useful to maintain proportions (same aspect ratio).

- `void glViewport(GLint x, GLint y, Glsizei w, Glsizei h);`
  - `w`x`h` pixels in the window with lower-left at (x,y)

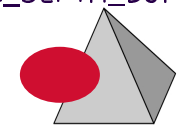


Hidden-Surface Removal

- *Visible-surface* or *hidden-surface-removal* algorithms
  - They order the objects so that they are drawn correctly
- **Z-buffer** algorithm (GL_DEPTH_TEST)

  glEnable(GL_DEPTH_TEST);
- glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH);
  - Allocates ***z-buffer*** or ***depth-buffer***
- Before redraw the display we need to clear the z-buffer

```
void display () {
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   tetrahedron(n);
   glflush();
}
```



Typical Non-interactive Main Program

```
#include <windows.h>
#include <gl/glut.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    glutInit(&argc,argv);
     glutInitDisplayMode(GLUT_RGB| GLUT_DOUBLE | GLUT_DEPTH);
     glutInitWindowSize(500,500);
     glutInitWindowPosition(0,0);
     glutCreateWindow("My White Rectangle");
     glutDisplayFunc(display);
     Init();  /* set up attributes, set up viewing */
     glutMainLoop();
}
```

Display Lists (DLs)

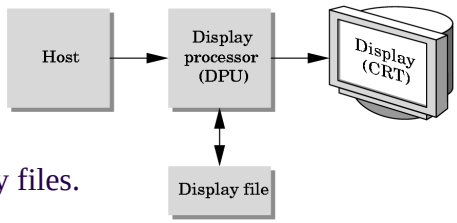
- DLs are a storage mechanism similar to ordinary files,
- DLs have an *identifier* (int) and a *content* (glNewList, glEndlist)

  glNewlist(ID, execution flag)
  - flag = GL_COMPILE, GL_COMPILE_AND_EXECUTE
- *E.g.*,

```
#define BlueBox 1    /* or some unused integer */
glNewList (BlueBox, GL_COMPILE);
  glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(-1.0,-1.0); glVertex2f(1.0,-1.0);
    glVertex2f(1.0, 1.0);   glVertex2f(-1.0,1.0);
   glEnd();
 glEndList();
```


Display Lists (DLs)

- Two methods of operations:
  - *Immediate mode*
  - *Retained mode*



- **Display Lists** are a storage mechanism similar to ordinary files.
- DLs have an *identifier* (int) and a *content* (glNewList, glEndlist)

  glNewlist(ID, execution flag)
  - flag = GL_COMPILE, GL_COMPILE_AND_EXECUTE

Example: Blue Box

```
#define BlueBox 1    /* or some unused integer */
 glNewList (BlueBox, GL_COMPILE);
  glBegin(GL_POLYGON);
    glColor3f(0.0, 0.0, 1.0);
    glVertex2f(-1.0,-1.0); glVertex2f(1.0,-1.0);
    glVertex2f(1.0, 1.0);   glVertex2f(-1.0,1.0);
   glEnd();
 glEndList();
```


Display Lists (Cont.)

- A DL is recall (drawn) using `glCallList (int identifier)`:

```
glPushAttrib(GL_ALL_ATTRIBUTE_BITS);
glPushMatrix();
glCallList (BlueBox);
glPopMatrix();
glPopAttrib();
```

- Additional functions:
  - `glGenLists(number)`
    returns consecutive integers that are unused labels
  - `glCallLists(n, t, p)`
    executes multiple lists, where
    n: number of lists; p: pointer to the array of lists;
    t: type of pointer p