Lectures
glTranslate Case Studies

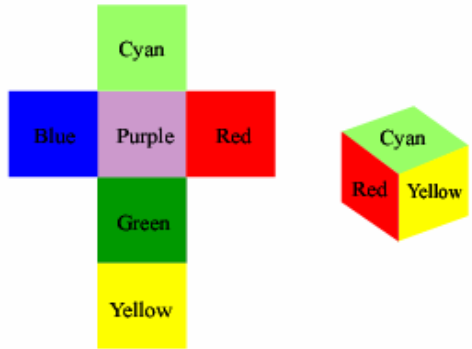
By
Tom Duff
Pixar Animation Studios
Emeryville, California
and
George Ledin Jr
Sonoma State University
Rohnert Park, California

Case Study 1

Case Study Setup:

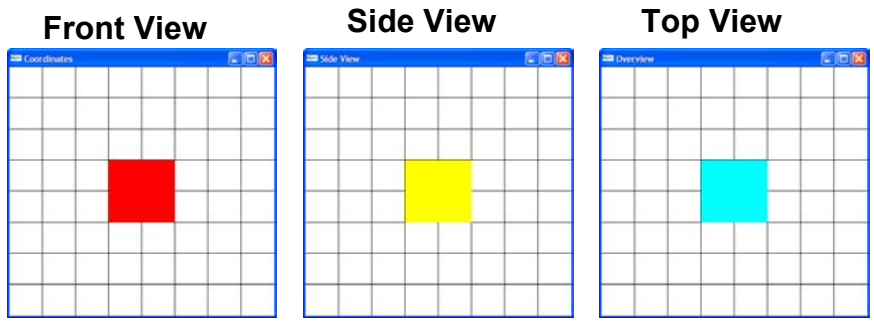
Assume in the world coordinates we have one color cube of size two, whose front is red.

colorcube (): centered at (0,0,0)



Goal:

We will move the position of camera or cube to find out what the camera will see (which is also what will show up in the screen).



Files Used:
glTranslate.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

First, Set up the Projection View

```
glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glOrtho(-4.0,4.0,-4.0,
            4.0,-4.0,4.0);
```

- **This set up means two things:**
  - If the object is outside its bounding box, it cannot be viewed. If part of an object is outside, that part cannot be viewed.
  - The camera is always in the geometric center of its bounding box. The camera cannot see anything outside its box.

Second, Set up the Camera's View and the glTranslate in the World Coordinates

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
     gluLookAt(…);
     glTranslate{f,d}(…);
// Draw cube afterwards
```

- Under the model view, we can decide how camera views the objects, which angle, how far away.
- **What will be translated, Camera or Objects?**

  This depends on the order of specification of gluLookAt and glTranslate*.

  – If we specify gluLookAt before glTranslate{f,d}, the objects drawn after glRotate{f,d} will be translated.

  – If we specify gluLookAt after glTranslate{f,d}, we will be moving the camera instead of the objects.
- Camera's translation is opposite of Objects.
- If you don't specify gluLookAt, the camera will take its default position and aim. Then the translation applies only to the objects.

How glTranslate{f,d} works?

```
void glTranslate{d,f}
        (GLdouble x,
         GLdouble y,
         GLdouble z)
```

- **glTranslate{f,d}** moves the coordinate system origin to the point specified by (x,y,z).
  - *x*, *y*, *z*
    Specify the *x*, *y*, and *z* coordinates of a vector, respectively.

- Objects and Camera are Translated in opposite directions.
  - If we are moving the object to (x,y,z) position, we can achieve the same result of translating the camera to (-x,-y,-z) position.

Question

Why the following three pieces of code produce the same results?

- **Code 1: Front View**

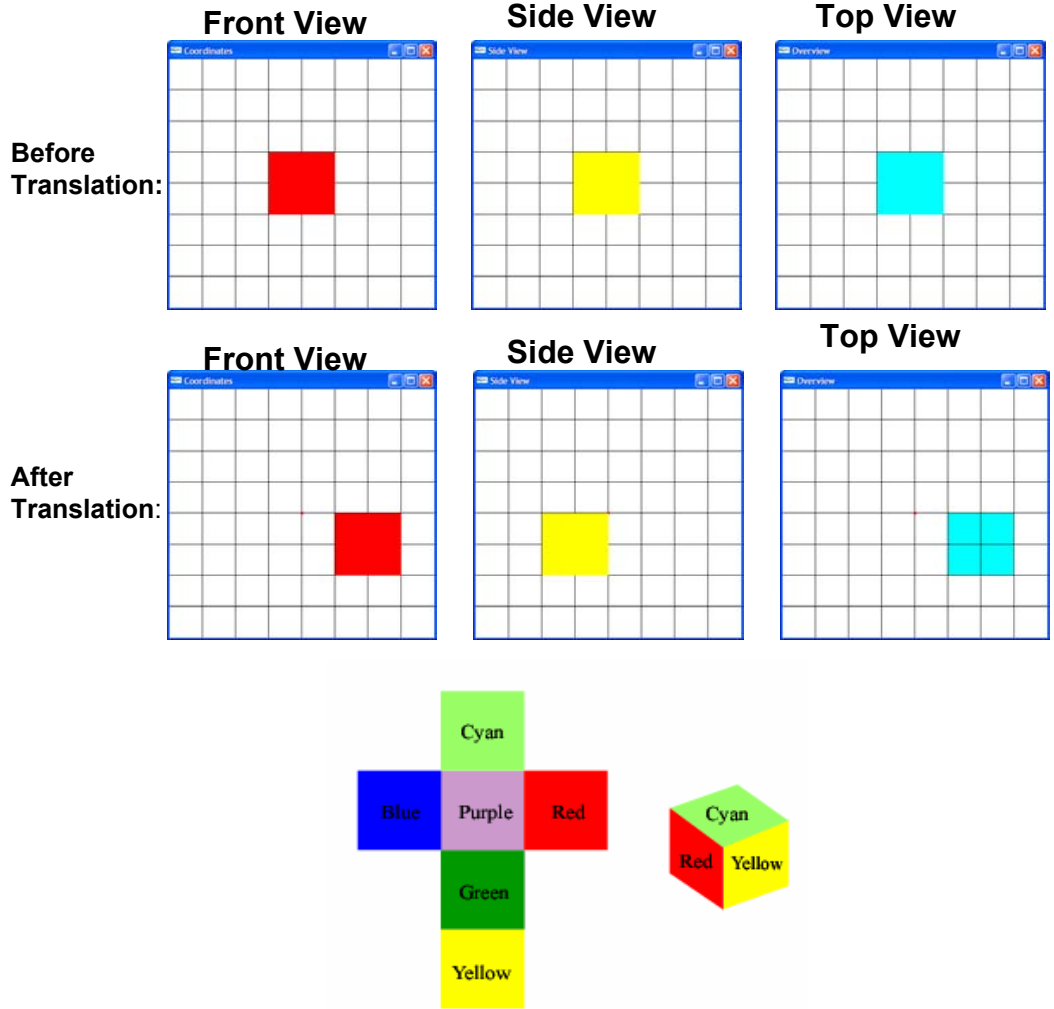
```
glLoadIdentity();
glTranslatef(2,-1,1);
colorcube0();
```

- **Code 2: Front View**

```
glLoadIdentity();
glLookAt(0,0,0,
         0,0,-1,
         0,1,0);
 glTranslatef(2,-1,1);
colorcube0();
```

- Code 3: Front View

```
glLookAt(0,0,-2,
          0,0,-3,
          0,1,0);
glLoadIdentity();
glTranslatef(2,-1,1);
colorcube0();
```



Answers

Code 1:

```
glLoadIdentity();
glTranslatef(2,-1,1);
colorcube0();
```

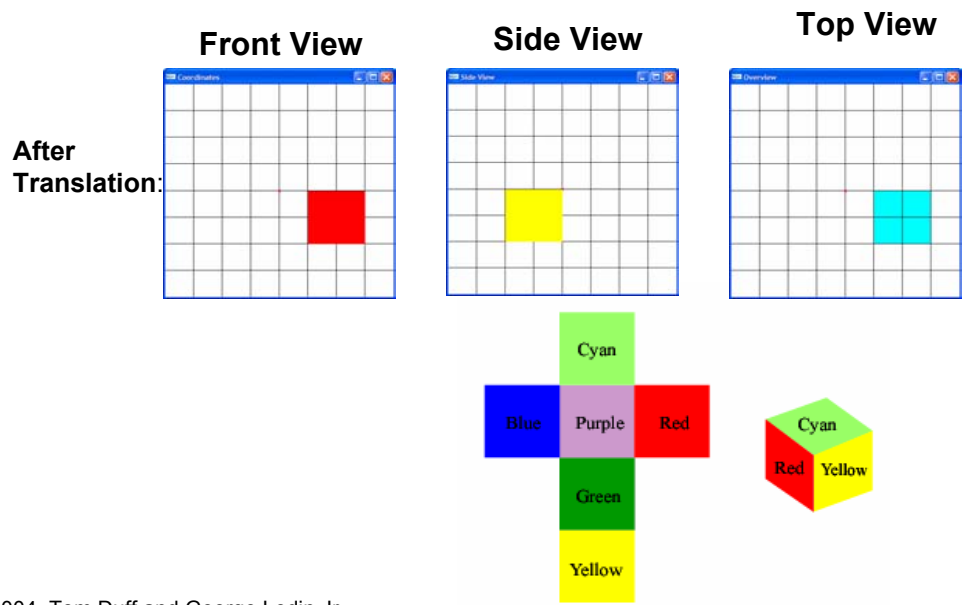
- **Code 2:**

```
glLoadIdentity();
glLookAt(0,0,0,
         0,0,-1,
         0,1,0);
 glTranslatef(2,-1,1);
colorcube0();
```

- **Code 3:**

```
glLookAt(0,0,-2,
          0,0,-3,
          0,1,0);
glLoadIdentity();
glRotatef(2,-1,1);
colorcube0();
```

1. **These three pieces of codes produce the same results:**
   - Code 1 and Code 2 produce the same result because the default of glLookAt is glLookAt(0,0,0,0,0,-1,0,1,0);
   - Code 2 and Code 3 produce the same result because the first glLookAt(0,0,-2,0,0,-3,0,1,0) is ignored. Therefore code 2 and 3 produces the same results.

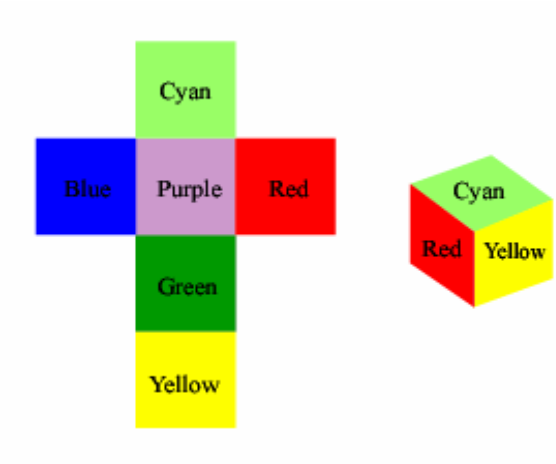


Question

1. **What is being translated? Camera or Object?**
2. **How much units were translated in x,y,z direction?**

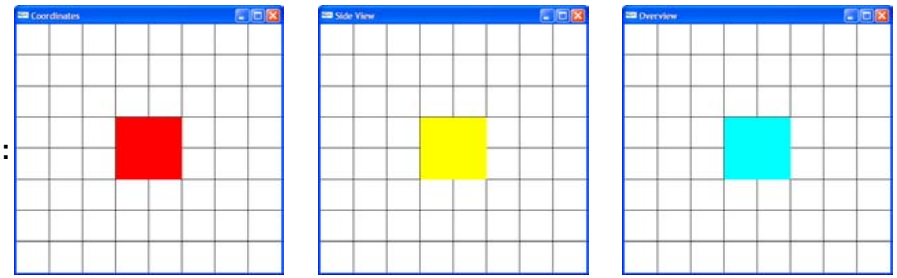
- **Code A: Front View**

```
glLoadIdentity();
glTranslatef(2,-1,1);
colorcube0();
```



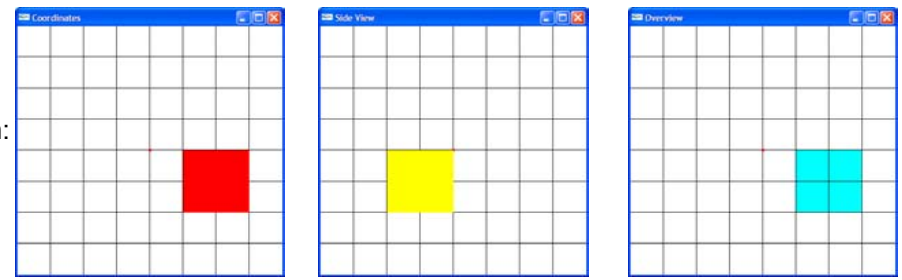
Before Translation:

Front View **Side View** **Top View**



After Translation:

Front View **Side View** **Top View**



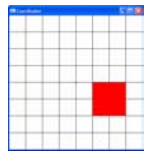
Answers

- **Code A: Front View**

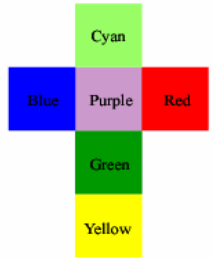
```
glLoadIdentity();
glTranslatef(2,-1,1);
colorcube0();
```

After Translation:

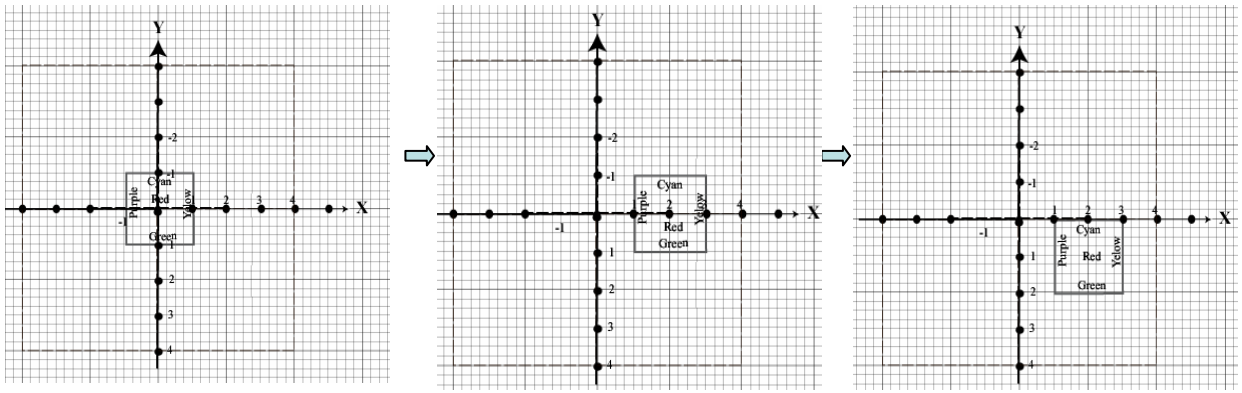
Front View



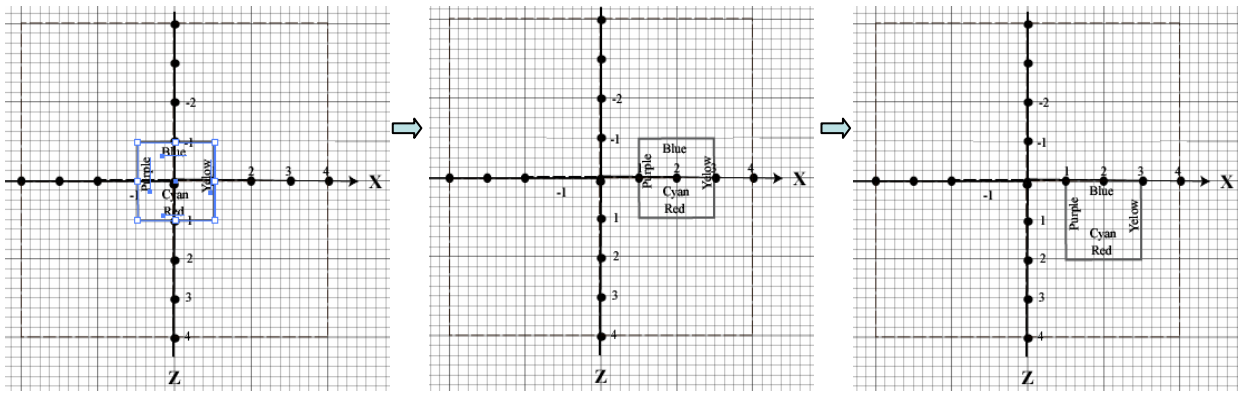
Top View



Original Position **+2 in X direction** **-1 in Y direction**



Original Position **+2 in X direction** **+1 in Z direction**



1. **The object is being translated:**
   - Since gluLookAt is not set, camera takes its default position, (0,0,0), aiming at (0,0,-1)
   - Since after glTranslate and before the drawing of objects, there is no gluLookAt specification, therefore prior to glTranslate's execution, there is no camera specification. Therefore, glTranslate will translate the objects.

4. **The units of translation in X,Y,Z direction:**
   - The object is translated 2 units in the positive X direction, -1 unit in the negative Y direction, and 1 unit in the positive Z direction.

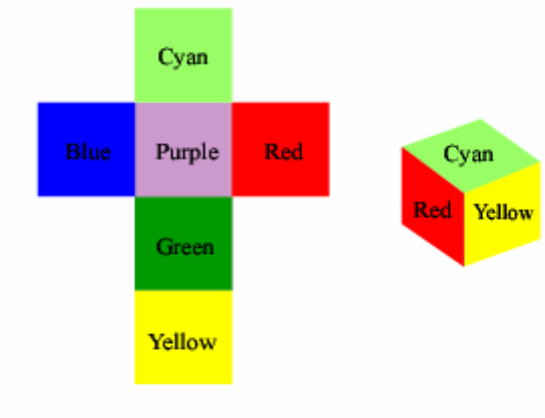
Case Study 2

- glTranslate.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

1. **What is being translated? Camera or Object?**
2. **How much units were translated in x,y,z direction?**

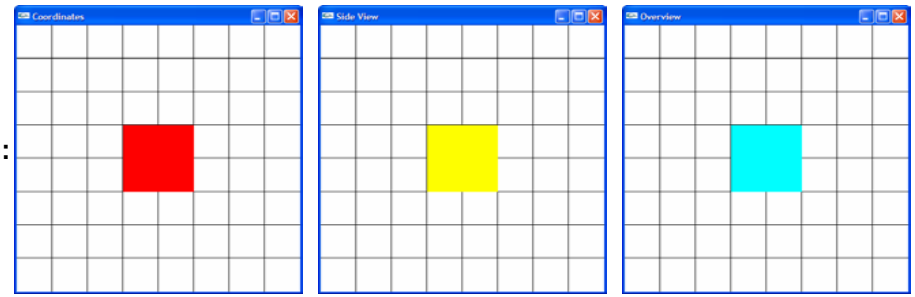
- **Code B: Front View**

```
glLoadIdentity();
glTranslatef(2,-1,1);
      glLookAt(0,0,0,
       0,0,-1,
       0,1,0);
colorcube0();
```



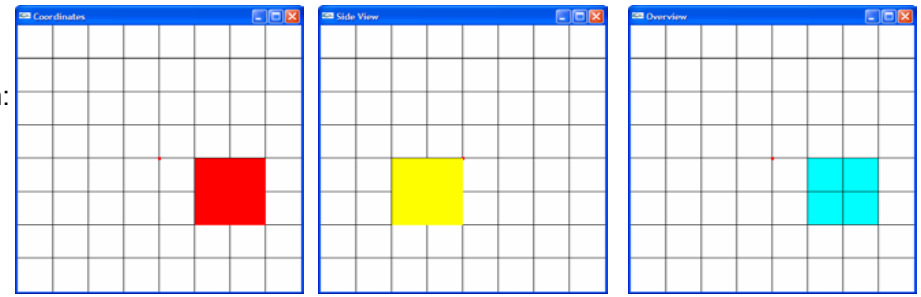
Before Translation:

Front View **Side View** **Top View**



After Translation:

Front View **Side View** **Top View**



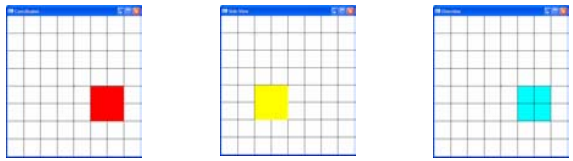
Answers

- **Code B: Front View**

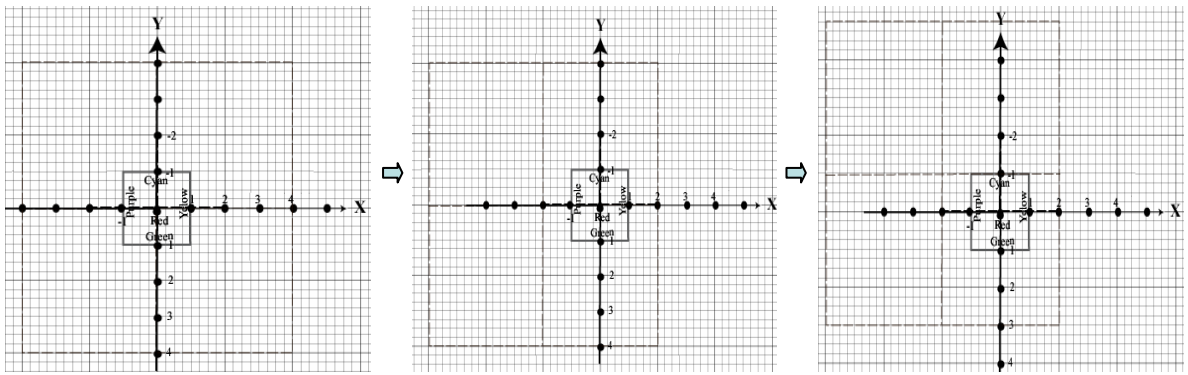
```
glLoadIdentity();
glTranslatef(2,-1,1,0);
glLookAt(0,0,0,
         0,0,-1,
         0,1,0);
colorcube0();
```

Front View **Side View** **Top View**

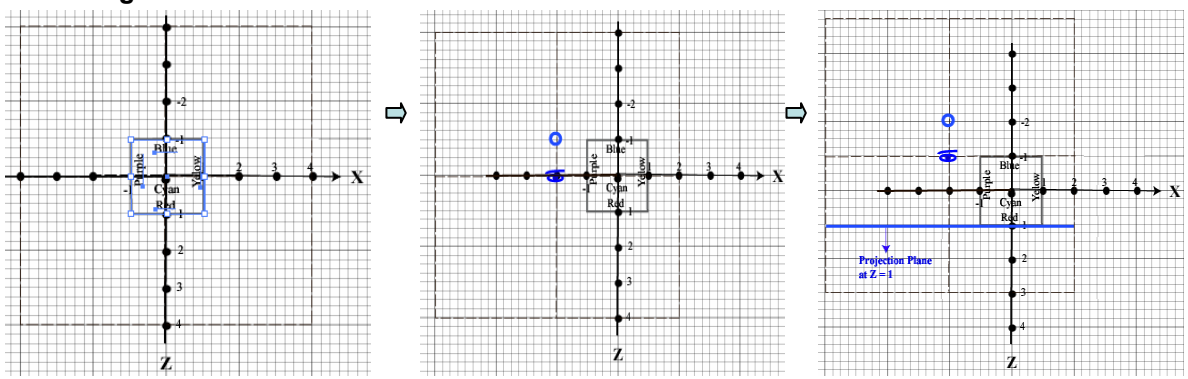
After Translation:



Original Position **-2 in X direction** **+1 in Y direction**

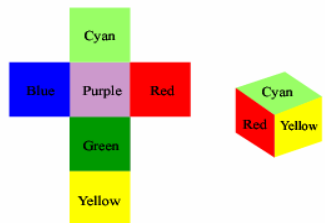


Original Position **-2 in X direction** **-1 in Z direction**



1. **The Camera is being translated:**
   - Since gluLookAt is set after glTranslate and before drawing of the cube, gluLookAt will be executed first. Therefore, glTranslate will translate the Camera.

4. **The units of translation in X,Y,Z direction:**
   - The Camera is translated -2 units in the positive X direction, 1 units in the negative Y direction, and -1 units in the positive Z direction.



Case Study 3
What's the accumulative effect of several glTranslate?

General Formula

```
glTranslate{f,d} (x1,y1,z1);
glTranslate{f,d} (x2,y2,z2);
```

```
glTranslate{f,d} (x2+x1,y2+y1,z2+z1);
```

Example

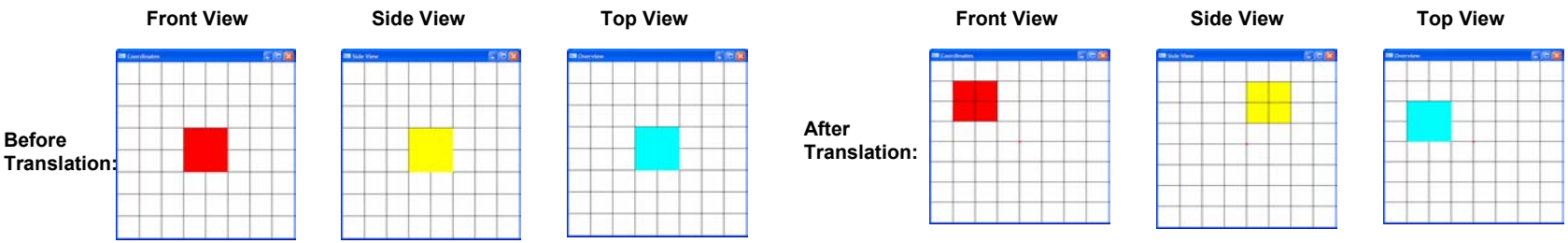
Code C:

```
glLoadIdentity();
glTranslatef(2,-1,1);
glTranslate(-4,3,-2);
colorcube0();
```

Code D:

```
glLoadIdentity();
glTranslatef(-2,2,-1);
colorcube0();
```

This above two pieces of code produces the same result:



The Actual Matrix Glut used in its Implementation

The following code will allow you to see what matrix Glut uses in its implementation:

```
// Print out the current matrix state after
// each execution of glut functions.
float CT[16];
glLoadIdentity();

glTranslatef(2,-1,1);
glGetFloatv(GL_MODELVIEW_MATRIX,CT);
PrintMToFile(CT,
     "After glTranslatef(2,-1,1);");

glTranslatef(-4,3,-2);
glGetFloatv(GL_MODELVIEW_MATRIX,CT);
PrintMToFile(CT,
     "After glTranslatef(2,-1,1);glTranslatef(-
     4,3,-2);");

Colorcube0();
```

```
// Print Matrix in Row Major Style
void PrintM(float m[], char *s) {
  int i = 0; int j = 0; int t = 0;

   printf("Current Matrix: %s\n",s);

   for (j = 0; j < 4; j++) {
      i = j;
      for (t = 0; t < 4; t++) {
         printf("%2f\t", m[i]);
         i=i+4;
         }
      printf("\n");
      }
      printf("End of printing Current Matrix \n\n");
     }
```

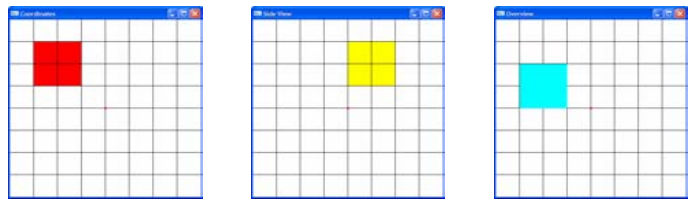
```
The Output of printed Current Matrices

Current Matrix, After glTranslatef(2,-1,1);
1.00          0.00          0.00          2.00
0.00          1.00          0.00          -1.00
0.00          0.00          1.00          1.00
0.00          0.00          0.00          1.00

Current Matrix, After glTranslatef(2,-1,1);glTranslatef(-4,3,-2);
1.00          0.00          0.00          -2.00
0.00          1.00          0.00          2.00
0.00          0.00          1.00          -1.00
0.00          0.00          0.00          1.00
```

After Translation:

Front View **Side View** **Top View**



Case Study 3, continued
Generic Matrix for translation

Recall glLoadIdentity() is defined as follows:

```
1,0,0,0
0,1,0,0
0,0,1,0
0,0,0,1
```

Recall that translation Matrix for
`glTranslate{f,d}(x,y,z)is as follows:`

```
1,0,0,x
0,1,0,y
0,0,1,z
0,0,0,1
```

Therefore, we can write our own Matrix to do the same work that glut Function does.

```
glLoadIdentity();
glTranslatef(2,-1,1);
glTranslate(-4,3,-2);
colorcube0();                    // Draw a cube centered at (0,0,0)
```

Case Study 3, continued

Using Matrix Multiplication:

```
float *m0,m1,*m2,*m3;
...
glLoadMatrixf(m0);
glMultMatrixf(m1);
glMultMatrixf(m2);
```

Using Glut Functions:

```
glLoadIdentity();
glTranslatef(2,-1,1);
glRotatef(-4, 3,-2);
glTranslatef(2,1,0);
colorcube3();
```

```
m0:
1, 0, 0, 0
0, 1, 0, 0
0, 0, 1, 0
0, 0, 0, 1
```

```
m1:
1, 0, 0, 2
0, 1, 0, -1
0, 0, 1, 1
0, 0, 0, 1
```

```
m2:
1, 0, 0, -4
0, 1, 0, 3
0, 0, 1, -2
0, 0, 0, 1
```

We shall achieve the same output by using glut functions or constructing our own load matrix and multiply matrix functions.

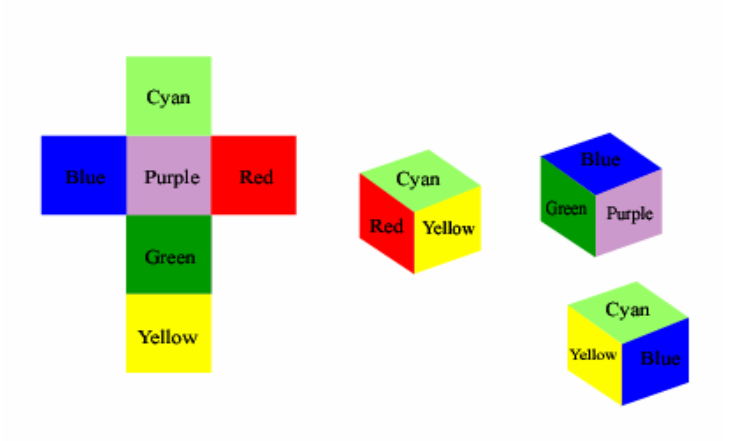
Note: The m0,m1,m3 were displayed in row major.
But the matrix array must be built in column major.

Case Study 4
Translation of Three Cubes

Case Study Setup:

Assume in the world coordinates we have three cubes of size two. One cube's front is red, the second one's front is green, the third one's front is yellow.

Cube (1): centered at (-3,0,-1)
Cube (2): centered at (3,0,-1)
Cube (3): centered at (0,0,-3)

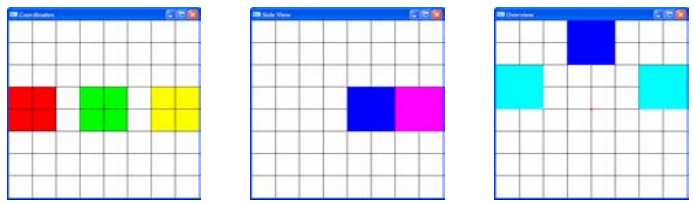


Goal:

We will test the effect of glTranslate on the view of three colored cubes.

Before Translation:

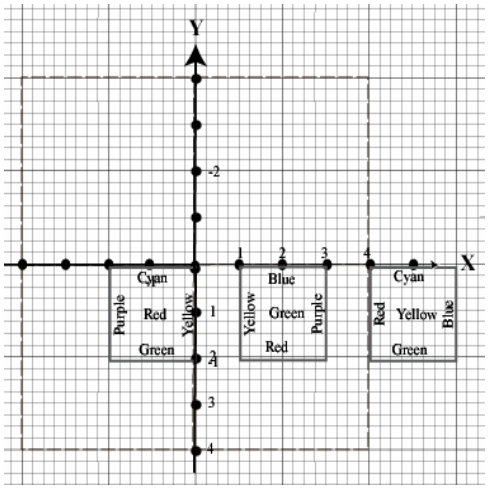
Front View **Side View** **Top View**



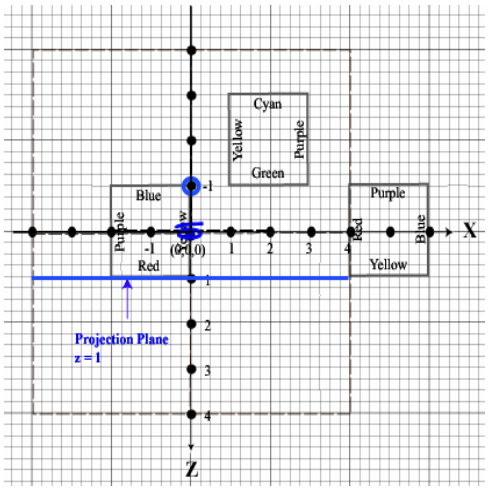
Files Used:
glRotate.c, DrawCubes.c, DrawCubes.h

Case Study 4
- Translation of 3 cubes

Front View



Top View



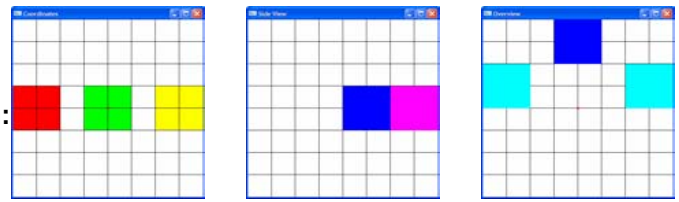
• **Code: Front View**

```
glLoadIdentity();
// camera default: located at (0,0,0, ref point (0,0,-1)
glTranslate(2,-1,1);  // Translate three objects

colorcube1();
colorcube2();
colorcube3();
```

Front View **Side View** **Top View**

Before Translation:



Front View **Side View** **Top View**

After Translation: