Lectures
Camera set up Case Studies

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

How does OpenGL View the Scene?

- OpenGL uses two kinds of views:

  1. Projection view
     - Which determines either Orthogonal Projection or Perspective Projection
  2. Model view
     - Under the model view, we can decide how the camera views the objects, which angle, how far away.

- You need to first determine what kind of projection you want and then what kind of model view in order to view the scene.

- OpenGL has default setup if you don't specify anything.
  - The default projection view in OpenGL is the orthogonal view.
  - The default camera position is (0,0,0), aimed at (0,0,-1) in world coordinates.

The Orthogonal view world

- **First**, we use glOrtho to set up orthogonal view
- glOrtho — Set up the viewing volume by multiplying the current matrix with an orthographic matrix
- glOrtho (*left* , *right* , *bottom* , *top* , *zNear* , *zFar*)

- **Example:**

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

- This example sets up two 8x8x8 bounding boxes in world coordinates: one bounding box for the objects, and another bounding box, of the same dimensions, for the camera.
- **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 can not be viewed.
  - The camera is always in the geometric center of its bounding box. The camera cannot see anything outside its box.

The Orthogonal View World, continued…

- **Second**, we use gluLookAt to set up the camera.
- gluLookAt — define a viewing transformation
- gluLookAt (*eyeX* , *eyeY* , *eyeZ*,
  lookAtX , lookAtY , lookAtZ ,
  upX , *upY* , *upZ*)
  - eyeX, eyeY, eyeZ is the point where the camera is located.
  - lookAtX , lookAtY , lookAtZ is the point toward which the camera is aimed at. It is also a point that is centered at the camera's view.
    (The LookAt point can be outside the bounding box.)
  - upX, upY, upZ specifies the upright direction of the camera, which fixes the camera to be perpendicular to the viewing plane.

- **Example:**

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,0, 0,0,-1, 0,1,0);
// Any other Transformations
// Draw anything afterwards
```

- This example specifies that the camera is located at (0,0,0), aimed at (0,0,-1) in the world coordinates, which means camera is looking in the negative Z direction.
- If you don't specify gluLookAt, the camera default position and aim are as above (the default values).

What's the purpose of glLoadIdentity()?

- In both code segments below, before we do the viewing of a transformation, we use glLoadIdentity to reset the state of OpenGL.

- **Code Segment 1: reset projection**

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

- **Code Segment 2: reset camera viewing**

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,0, 0,0,-1, 0,1,0);
// Any other Transformations
// Draw anything afterwards
```

Case Study

– gluLookAt.c, DrawCubes.c, DrawCubes.h

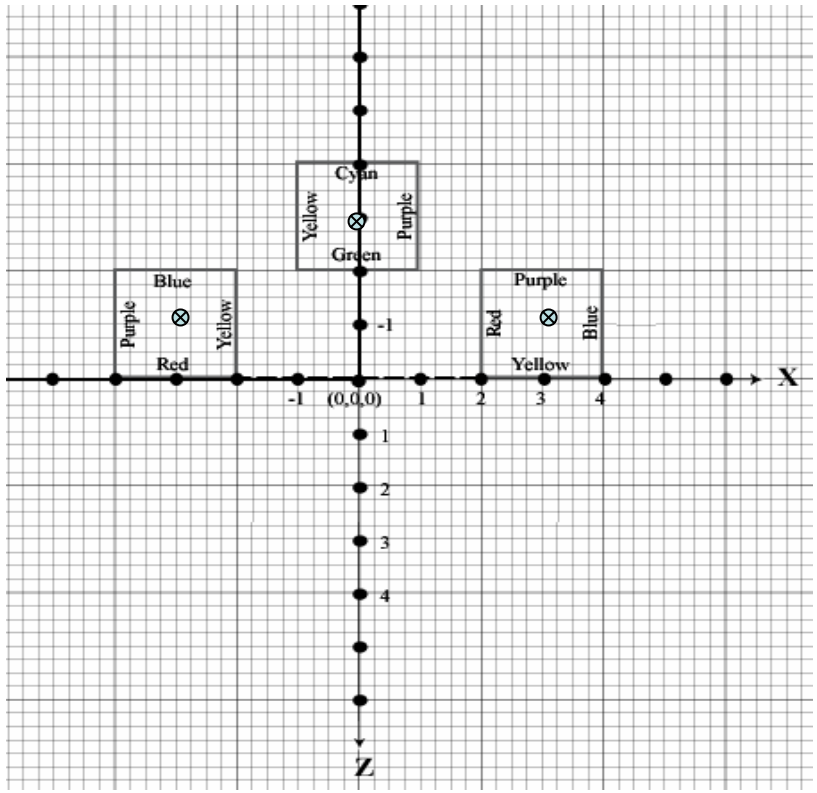
Case Study Setup:

Assume in 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)

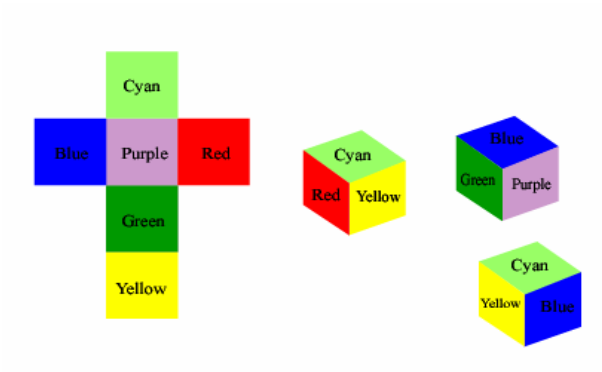
Cube (2): centered at (3,0,-1)

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



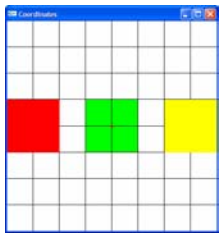
Goal:

We will change the camera's position and lookat point to find out what the camera will see (which is also what will show up on the screen).

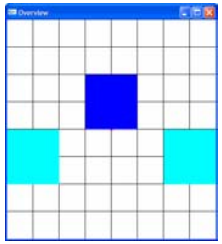


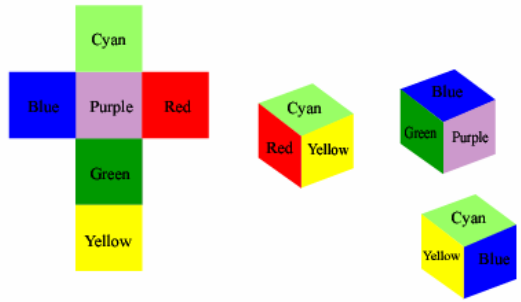
Camera default Position and LookAt point

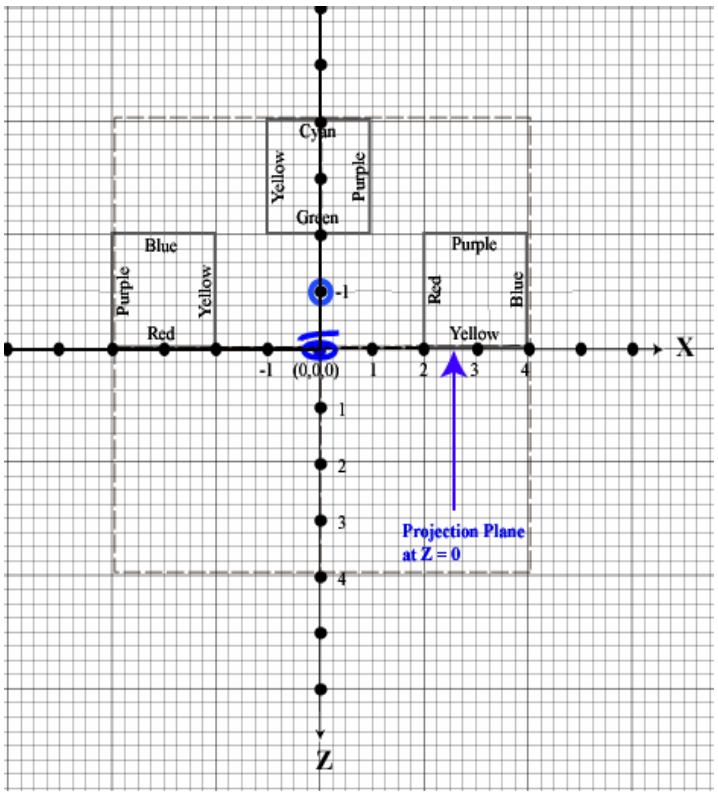
Front view



Top view



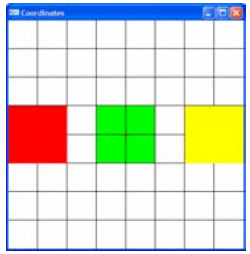




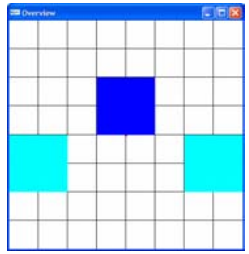

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-4.0, 4.0, -4.0, 4.0, -4.0, 4.0);
```

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

// This is camera default, if you don't
// specify gluLookAt
gluLookAt(0,0,0, 0,0,-1, 0,1,0);

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

Front View

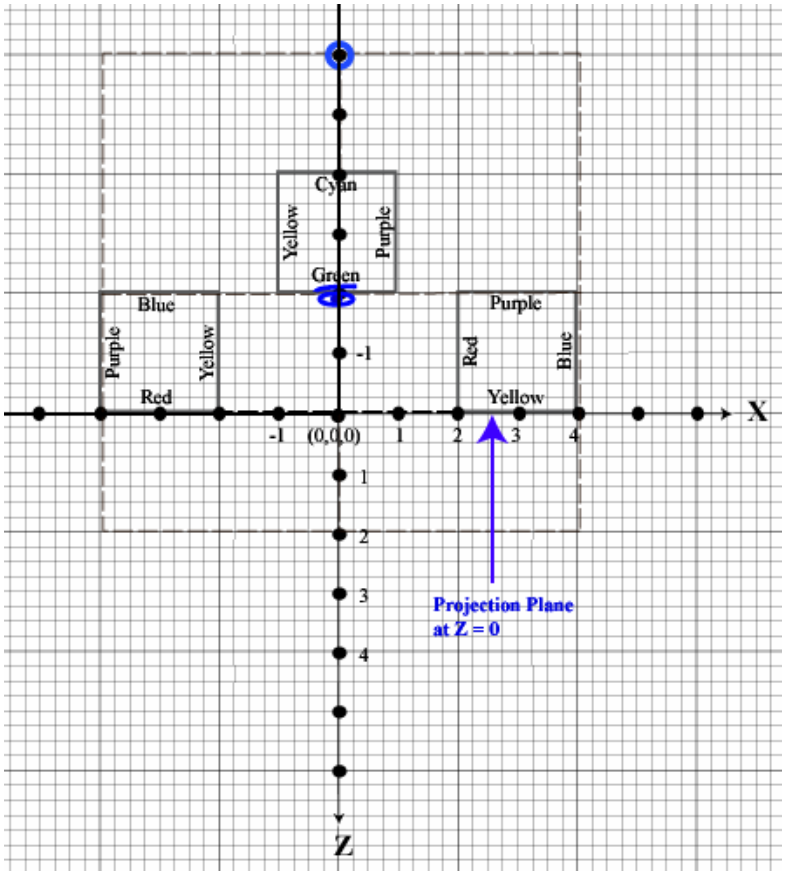


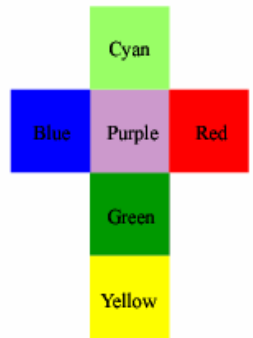
Top View

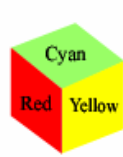


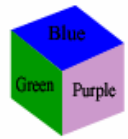
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-4.0, 4.0, -4.0, 4.0, -4.0, 4.0);
```

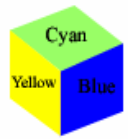
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0,0,-2, 0,0,-6, 0,1,0);

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



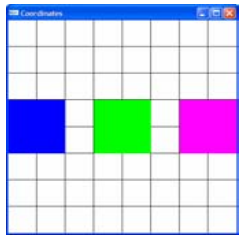




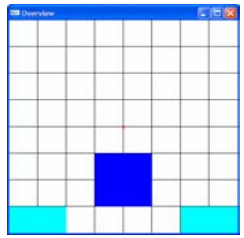




Front View

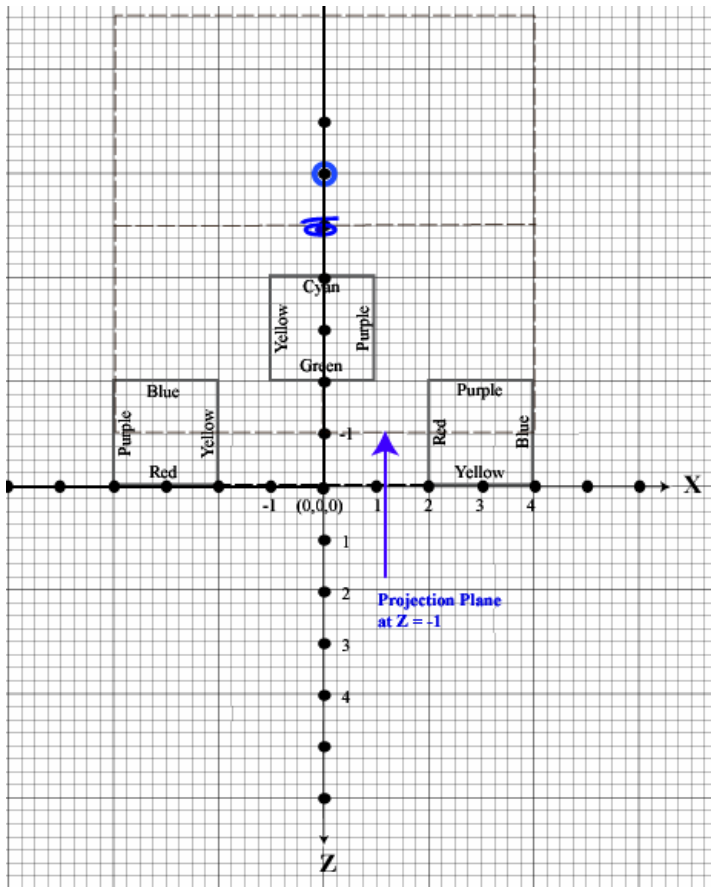


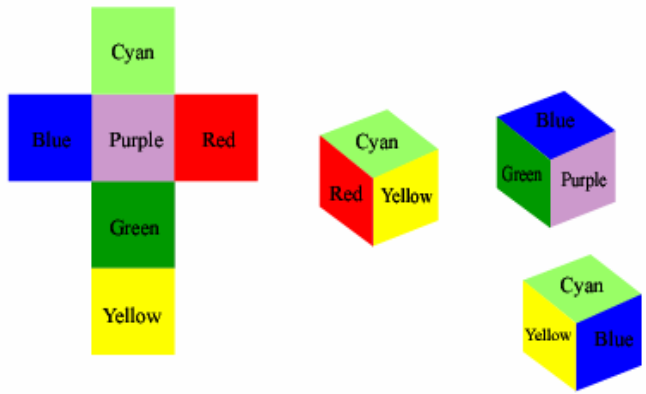
Top View



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

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,-5, 0,0,-6, 0,1,0);

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

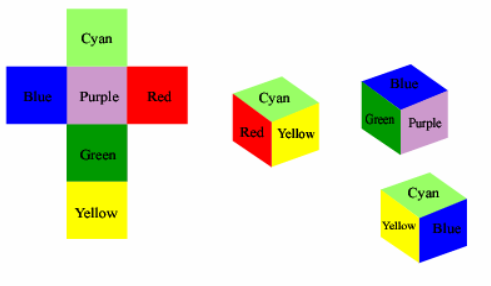




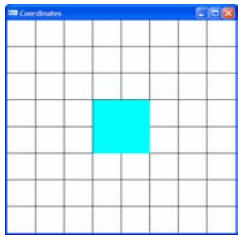
What are the consequences of Concatenating gluLookAt?

- For example:
- gluLookAt(0,0,a,0,0,b,0,1,0);
- gluLookAt(0,0,c,0,0,d,0,1,0);

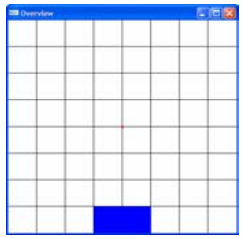
- We will demonstrate the cumulative effect of gluLookAt commands.
  - Recall that in OpenGL “the last transformation defined is the first executed”.
  - The gluLookAt commands will be executed the same way.

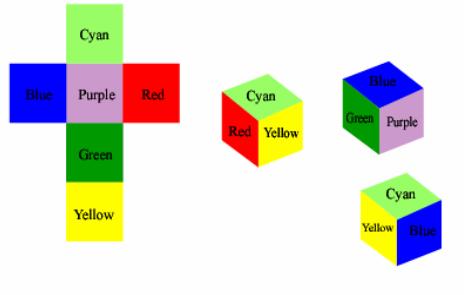


Front View



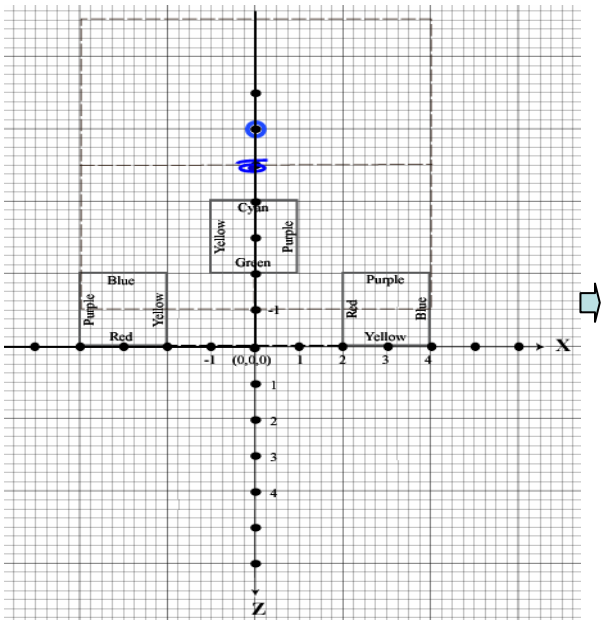
Top View

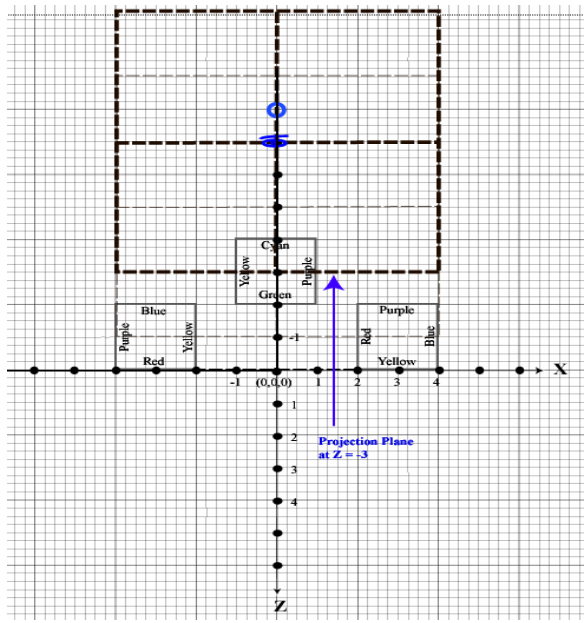





```
gluLookAt(0,0,-5, 0,0,-6, 0,1,0);
```

```
gluLookAt(0,0,-2, 0,0,-3, 0,1,0);
```






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

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0,0,-2, 0,0,-3, 0,1,0);
gluLookAt(0,0,-5, 0,0,-6, 0,1,0);


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

```
Cumulative Transformation effect:
            gluLookAt(0,  0,  -7,      // Point where camera is located
                      0,  0,  -8,      // Point toward which camera is aimed
                      0,  1,  0);      // Upright position of camera
                                       // specified by the vector from (0,0,0) to (0,1,0)
```

Concatenating gluLookAt

```
gluLookAt(0,0,a,0,0,b,0,1,0);
gluLookAt(0,0,c,0,0,d,0,1,0);
```

```
gluLookAt(0, 0, c + a,
          0, 0, c + b,
          0, 1, 0);
```

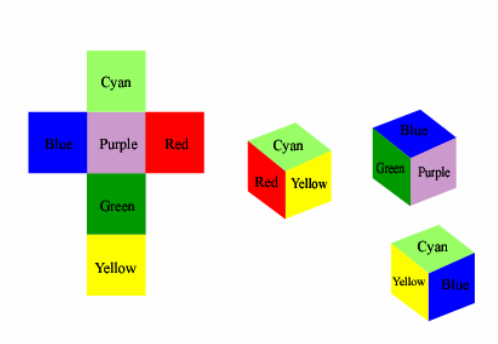
Note: When concatenating gluLookAt, only the first gluLookAt's lookAt point matters.

Example:

```
gluLookAt(0,0,-2,0,0,-3,0,1,0);
gluLookAt(0,0,-5,0,0,-6,0,1,0);
```

```
gluLookAt(0, 0, -5+(-2),
          0, 0, -5 - 3,
          0, 1, 0);
```

```
gluLookAt(0, 0, -7,
          0, 0, -8,
          0, 1, 0);
```



Concatenating gluLookAt

```
gluLookAt(0,0,a,0,0,b,0,1,0);
gluLookAt(0,0,c,0,0,d,0,1,0);
```

```
gluLookAt(0,  0,  c + a,
          0,  0,  c + b,
          0,  1,  0);
```

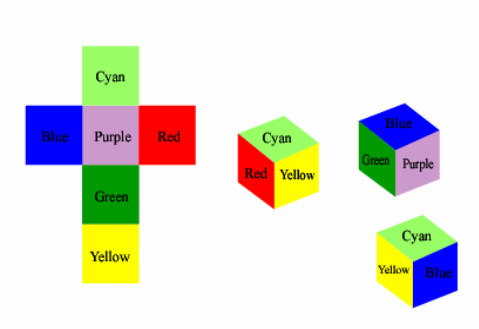
Note: When concatenating gluLookAt, only the first gluLookAt's lookAt point matters.

Example:

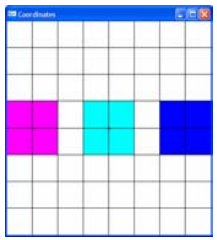
```
gluLookAt(0,0,-2,0,0,3,0,1,0);
gluLookAt(0,0,-3,0,0,-4,0,1,0);
```

```
gluLookAt(0,  0,  -2+(-3),
          0,  0,  -3 +3,
          0,  1,  0);
```

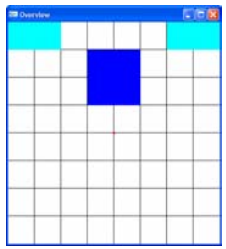
```
gluLookAt(0,  0,  -5,
          0,  0,  0,
          0,  1,  0);
```

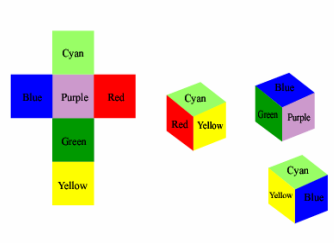


Front View

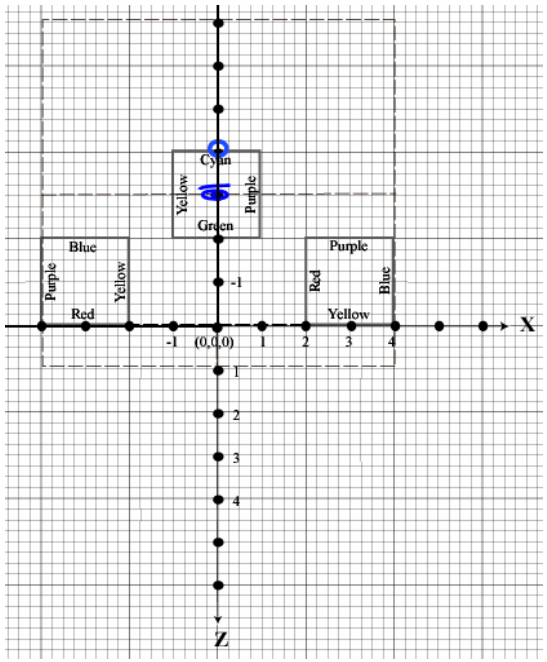


Top View

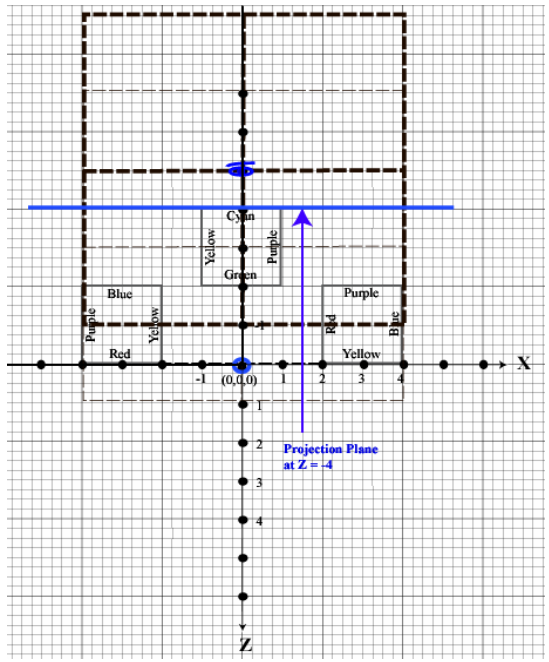




gluLookAt(0,0,-3, 0,0,-4, 0,1,0);



gluLookAt(0,0,-2, 0,0,3, 0,1,0);




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

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0,0,-2, 0,0,3, 0,1,0);
gluLookAt(0,0,-3, 0,0,-4, 0,1,0);


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

```
Cumulative Transformation effect:
          gluLookAt(0, 0, -5,     // Point where camera is located
                    0, 0, 0,      // Point toward which camera is aimed at
                    0, 1, 0);     // Upright position of camera specified by the
                                  // vector from (0,0,0) to (0,1,0)
```

Concatenating gluLookAt Special Cases

```
gluLookAt(0,0,a,0,0,b,0,1,0);
gluLookAt(0,0,c,0,0,d,0,1,0);
```

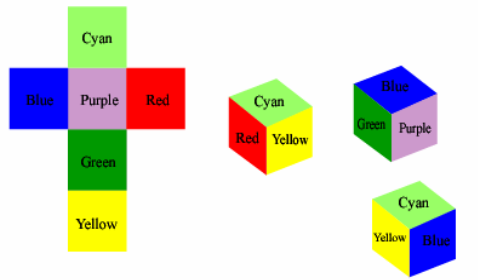
```
gluLookAt(0, 0, c + a,
          0, 0, c + b,
          0, 1, 0);
```

Note: When concatenating gluLookAt, only the first gluLookAt's lookAt point matters.

Example:

```
gluLookAt(0,0,0,0,0,-1,0,1,0);
gluLookAt(0,0,0,0,0,-1,0,1,0);
gluLookAt(0,0,0,0,0,-1,0,1,0);
gluLookAt(0,0,0,0,0,-1,0,1,0);
....
```

```
gluLookAt(0, 0, 0,
          0, 0, -1,
          0, 1, 0);
```



What will happen if the second gluLookAt cuts out some scene, but the first does not?

- In real life, if you change the camera position from not able to see the scene to the position where the camera can view the scene, all that counts is the last position of the camera.
- Therefore in OpenGL, if the second gluLookAt cuts out some scene, but the first gluLookAt can view the objects that were cut by the previous gluLookAt executed, you will still see the objects in the scene.

What will happen if the second gluLookAt aimes in the opposite direction from the first one?

- In real life, if you change the camera look at Point from one direction to another, all that counts is the last “Look At” point of camera, which determines the direction toward which the camera is aimed.
- Therefore in OpenGL, only the last executed “Look At” point determines the direction of the camera.

Concatenating gluLookAt Special Cases

```
gluLookAt(0,0,a,0,0,b,0,1,0);
gluLookAt(0,0,c,0,0,d,0,1,0);
```

```
gluLookAt(0, 0, c + a,
          0, 0, c + b,
          0, 1, 0);
```

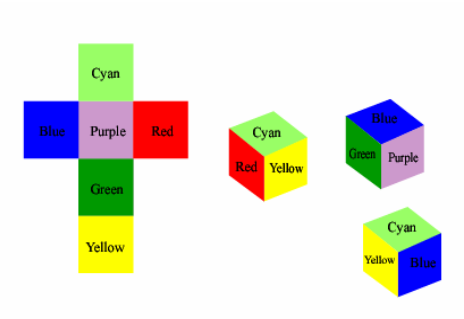
Note: When concatenating gluLookAt, only the first gluLookAt's lookAt point matters.

Example:

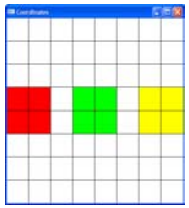
```
gluLookAt(0,0,6,0,0,0,0,1,0);
gluLookAt(0,0,-5,0,0,-6,0,1,0);
```

```
gluLookAt(0, 0, -5+6,
          0, 0, -5+0,
          0, 1, 0);
```

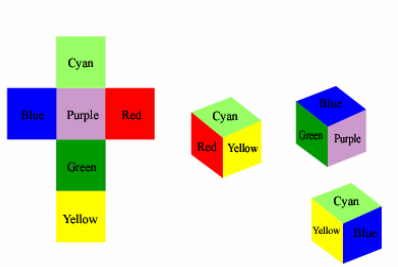
```
gluLookAt(0, 0, 1,
          0, 0, -5,
          0, 1, 0);
```



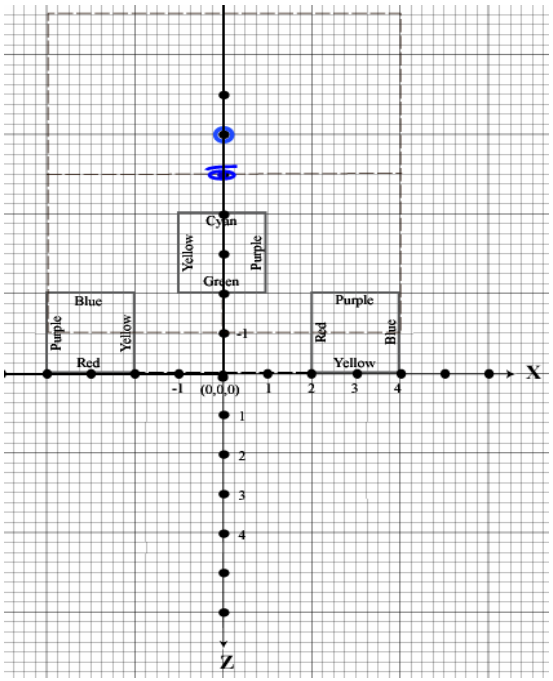
Front View



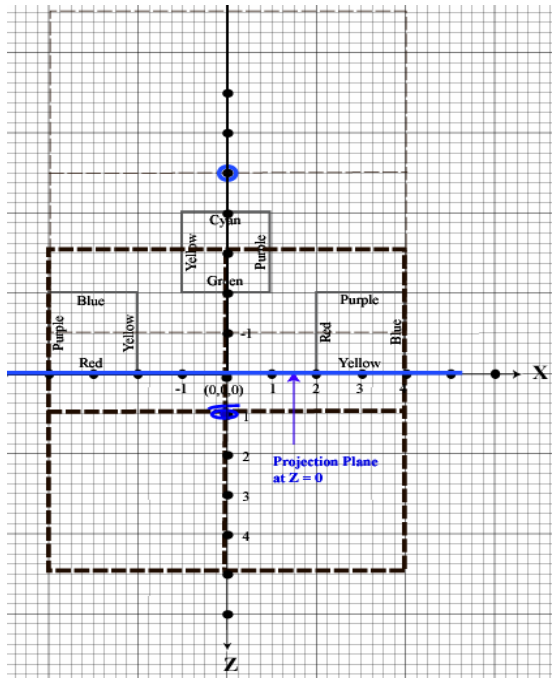
Top View




```
gluLookAt(0,0,-5, 0,0,-6, 0,1,0);
```




```
gluLookAt(0,0,6, 0,0,0, 0,1,0);
```




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

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0,0,6, 0,0,0, 0,1,0);
gluLookAt(0,0,-5, 0,0,-6, 0,1,0);

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

```
Cumulative Transformation effect:
                gluLookAt(0, 0, 1,          // Point where camera is located
                          0, 0, -5,         // Point toward which camera is aimed
                          0, 1, 0);         // Upright position of the camera
                                            // specified by the vector from (0,0,0) to (0,1,0)
```

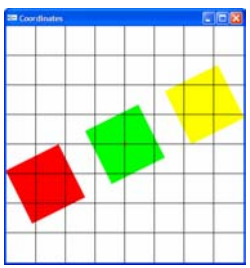
Experimenting with the Upper Vector of the Camera

- Case Study

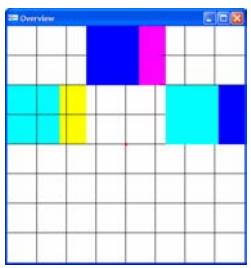
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-4.0, 4.0, -4.0, 4.0, -4.0, 4.0);
```

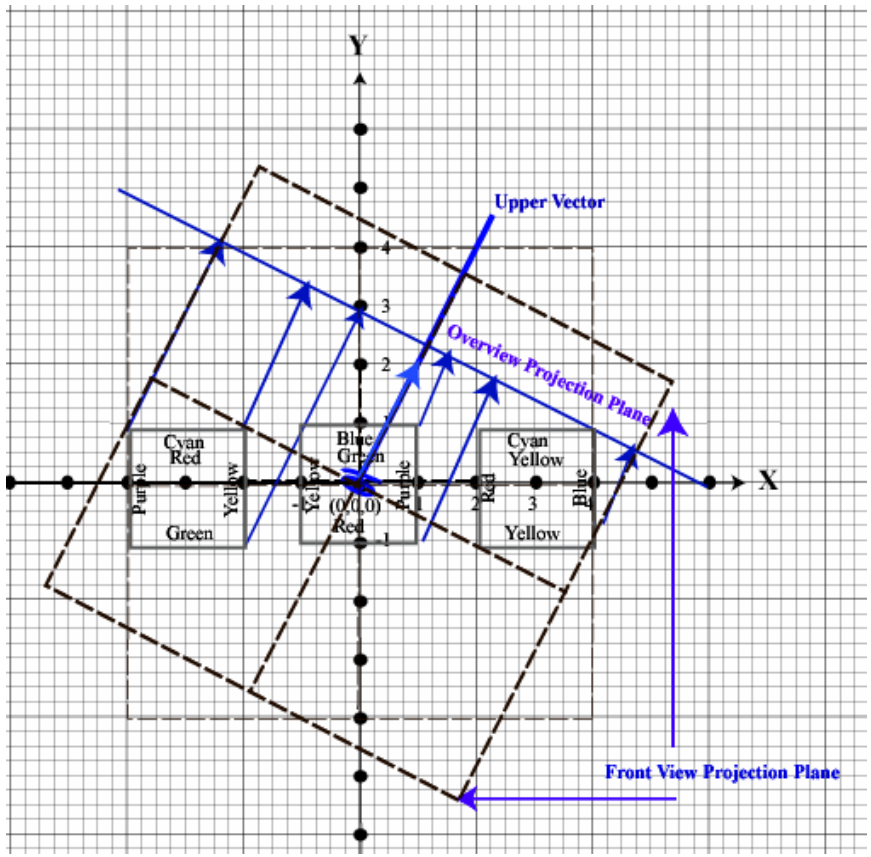
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

gluLookAt(0,0,0, 0,0,-1, 1,2,0);

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

Front View



Over View





What will happen if we concatenating two gluLookAt which have different upper vector?

Special Case Study:

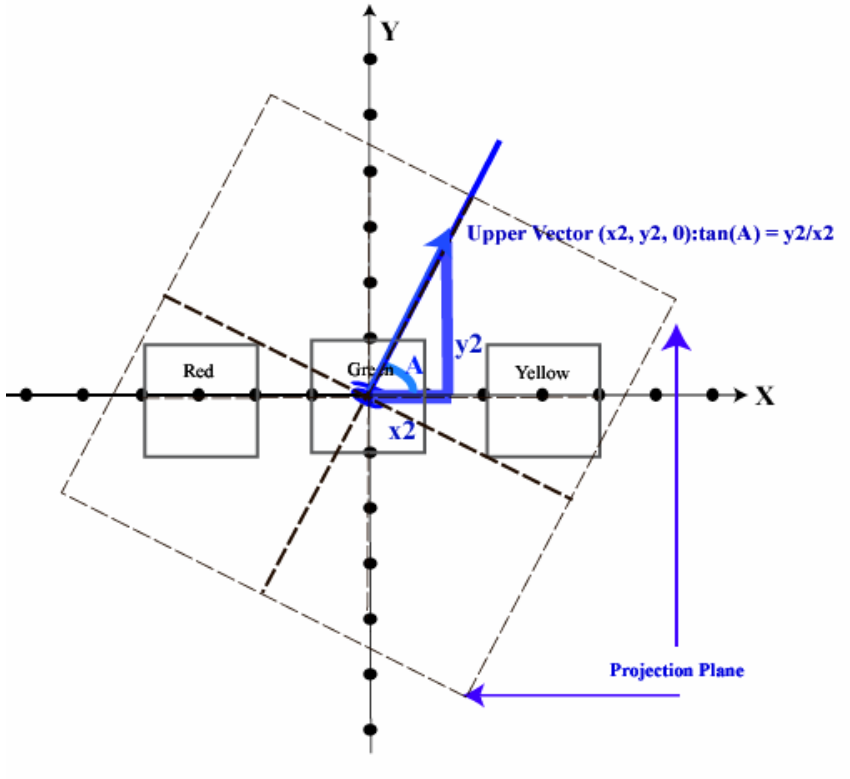
```
gluLookAt(0,0,0, 0,0,-1, x1,y1,z1);
gluLookAt(0,0,0, 0,0,-1, x2,y2,z2);
```

➔ `gluLookAt(0,0,0, 0,0,-1, x,y,z)`

Question: What's x, y z?

Steps to do:

(1) Determine the camera's relation with objects after executing second gluLookAt.



What will happen if we concatenating two gluLookAt which have different upper vector?

Special Case Study:

```
gluLookAt(0,0,0, 0,0,-1, x1,y1,z1);
gluLookAt(0,0,0, 0,0,-1, x2,y2,0);
➔      gluLookAt(0,0,0, 0,0,-1, x,y,z2)
```

Question: What's x, y z?

Steps to do:

(1) Determine the camera's relation with objects after executing second gluLookAt.

(2) Determine the camera's relation with objects after executing the first gluLookAt.
Note: the first gluLookAt is based on the position of camera after the second gluLookAt.

The accumulative result of combining two upper vector's angles is angle C:

C = A + (90 - (180 - B)) = A + B - 90

Therefore,

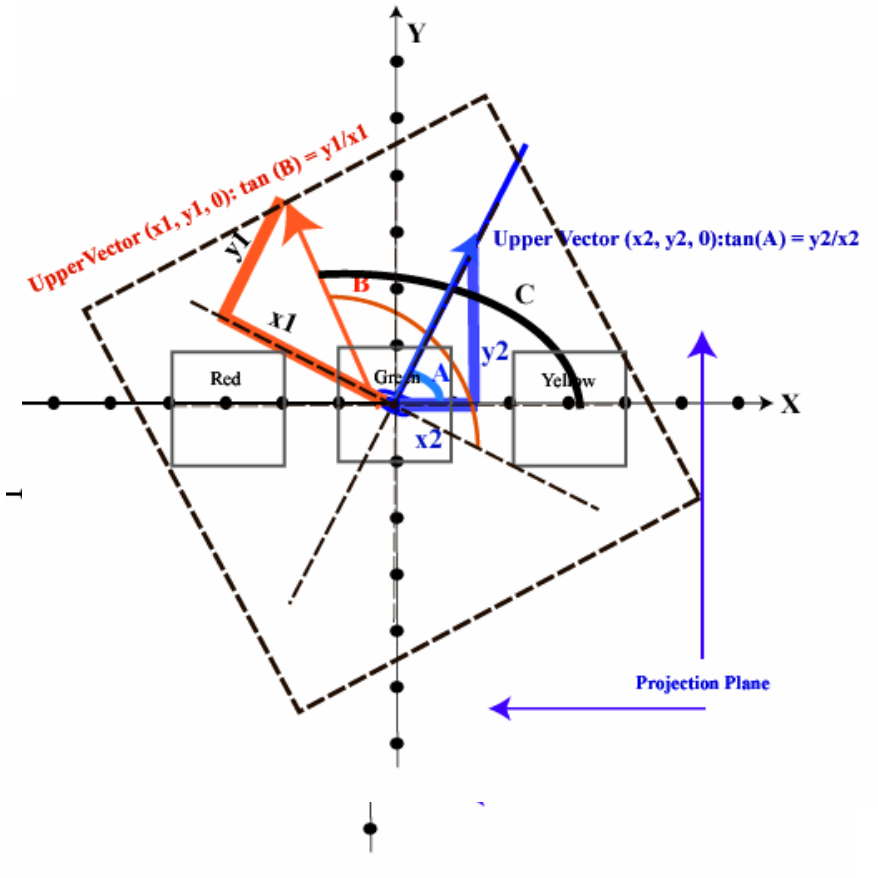
tan(C) = y / x

= tan(arctan(A) + arctan(B) - 90)

= tan(arctan(y2/x2) + arctan(y1/x1) - 90)

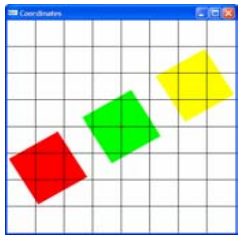
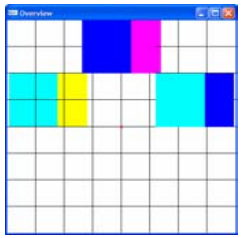
= (y2y1 - x2x1) / (y2x1 + y1x2)

Note: Z does not participate in the calculation because camera is looking at Z direction in this case.
Therefore, Z can be anything.

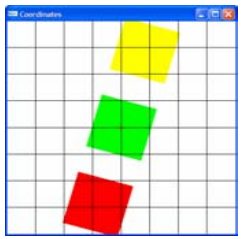
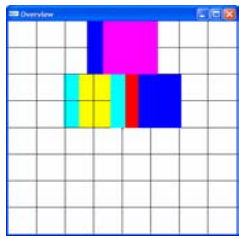


Case 1

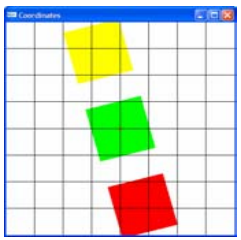
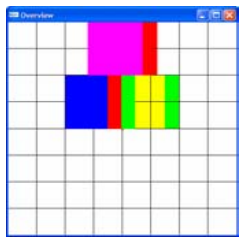
Front View **Top View**

```
gluLookAt(0,0,0, 0,0,-1, 3,5,0);
```

```
gluLookAt(0,0,0, 0,0,-1, 7,2,0);
```

```
gluLookAt(0,0,0, 0,0,-1, 3,5,0);
gluLookAt(0,0,0, 0,0,-1, 7,2,0);
```

=

Experiment shows that indeed the results are the same for these two different pieces of code.

Using previously deduced formular:

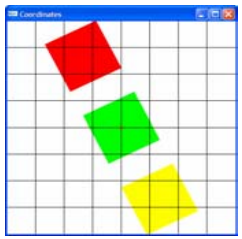
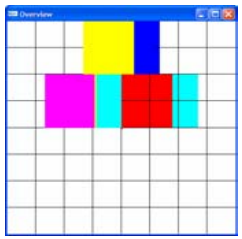
y / x = (y2y1 - x2x1) / (y2x1 + y1x2)

x1 = 3, y1 = 5, x2 = 7, y2 = 2 ➔ y = -11, x = 41

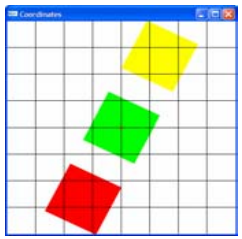
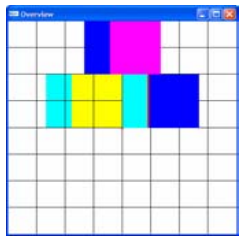
```
gluLookAt(0,0,0, 0,0,-1, 41,-11,0);
```

Case 2

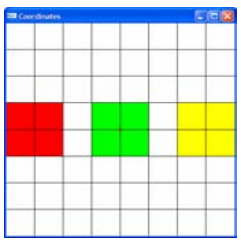
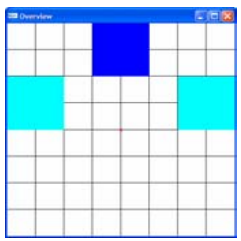
Front View **Top View**

```
gluLookAt(0,0,0, 0,0,-1, -4,2,0);
```

```
gluLookAt(0,0,0, 0,0,-1, 2,1,0);
```

```
gluLookAt(0,0,0, 0,0,-1, -4,2,0);
gluLookAt(0,0,0, 0,0,-1, 2,1,0);
```

||

Experiment shows that indeed the results are the same for these two different pieces of code.

Using previously deduced formular:

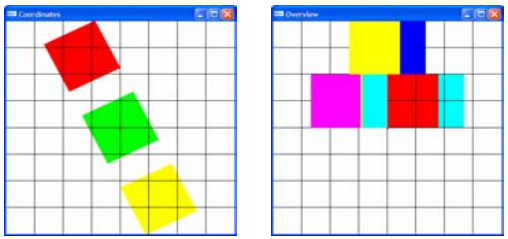
y / x = (y2y1 - x2x1) / (y2x1 + y1x2)

x1 = -4, y1 = 2, x2 = 2, y2 = 1 ➔ y = 10, x = 0

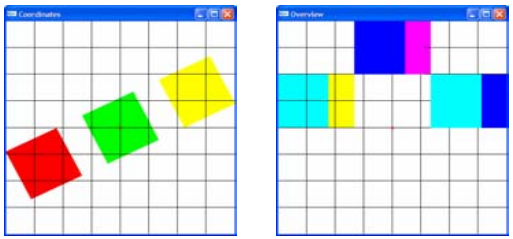
```
gluLookAt(0,0,0, 0,0,-1, 0,10,0);
```

Case 3

Front View **Top View**



```
gluLookAt(0,0,0, 0,0,-1, -1000,500,0);
```

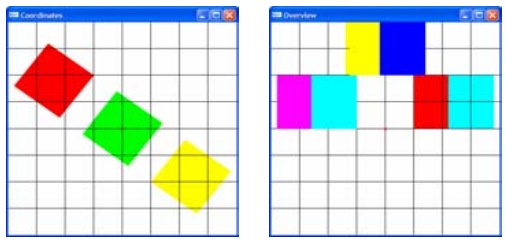


```
gluLookAt(0,0,0, 0,0,-1, 0.5,1,0);
```

```
gluLookAt(0,0,0, 0,0,-1, -1000,500,0);
gluLookAt(0,0,0, 0,0,-1, 0.5,1,0);
```

=

Experiment shows that indeed the results are the same for these two different pieces of code.



Using previously deduced formular:

y / x = (y2y1 - x2x1) / (y2x1 + y1/x2)

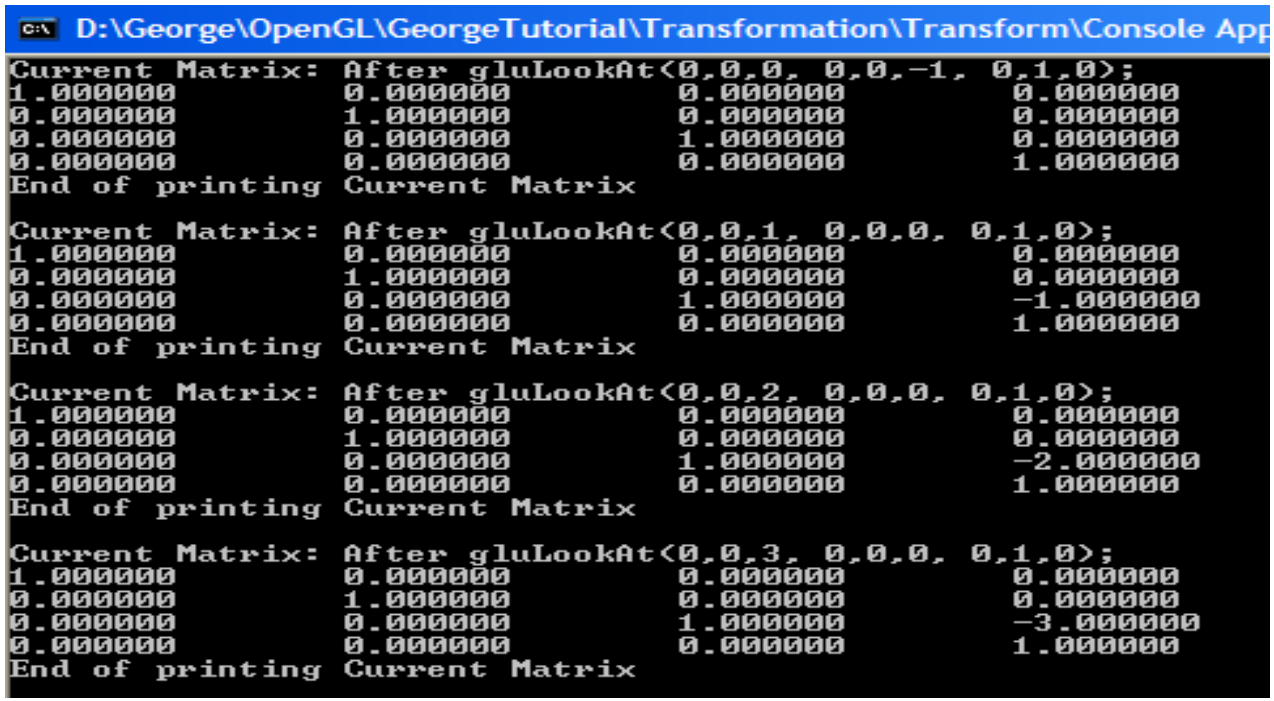
x1 = -1000, y1 = 500, x2 = 0.5, y2 = 1 ➔ y = 1000, x = -750

```
gluLookAt(0,0,0, 0,0,-1, -750,1000,0);
```

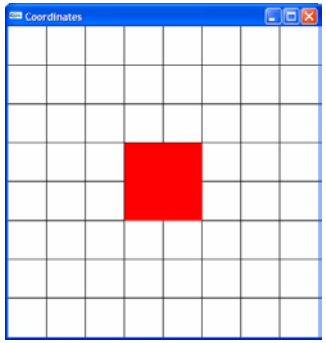
What happens when we concatenate two gluLookAt which have different positions, different “look at” points, different upper arrows?

- In previous cases we studied special cases, such as replace two gluLookAt which has the same upper arrow directions with one, or replace two gluLookAt which both look at negative Z direction and only differ in the upper arrow’s directions with one gluLookAt.
- We have produced formula to use one gluLookAt to replace two different gluLookAt for the above special cases.
- The general formula of using one gluLookAt to replace two consecutive gluLookAt is difficult and is outside our discussion scope.

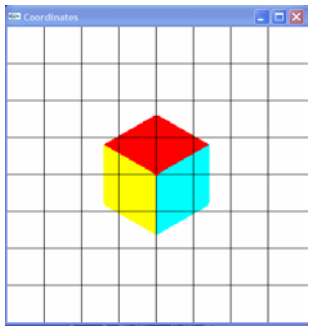
gluLookAt Matrixes



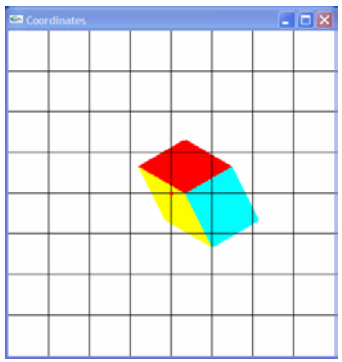
Cube: centered at (0,0,0), size=2



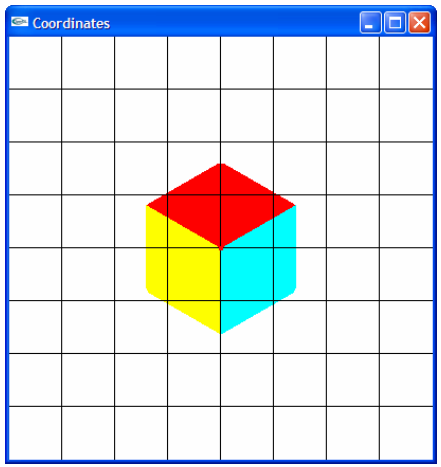

```
Current Matrix: After gluLookAt(2,2,2,  0,0,0,  0,0,1);
-0.71    0.71    0.00    0.00
-0.41    -0.41   0.82    0.00
0.58     0.58    0.58    -3.46
0.00     0.00    0.00    1.00
```




```
Current Matrix: After gluLookAt(2,2,2,  0,0,0,  0,0,1);
-0.71   0.71    0.00    0.00
-0.41   -0.41   0.82    0.00
0.58    0.58    0.58    -3.46
0.00    0.00    0.00    1.00
```




```
Current Matrix: After MyGluLookAt(2,2,2,        0,0,0,  0,0,1);
                       glTranslate(-2, -2,-2);
-0.58    0.58    -0.33    0.67
-0.33    -0.33   0.67     0.00
0.58     0.58    0.58     -3.46
0.00     0.00    0.00     1.00
```




```
Current Matrix: After gluLookAt(2,2,2,  0,0,0,  -1,-1,0);
-0.71   0.71    0.00    0.00
-0.41   -0.41   0.82    0.00
0.58    0.58    0.58    -3.46
0.00    0.00    0.00    1.00
```

```
Current Matrix: After gluLookAt(2,2,2,  0,0,0,  0,0,1);
-0.71   0.71    0.00    0.00
-0.41   -0.41   0.82    0.00
0.58    0.58    0.58    -3.46
0.00    0.00    0.00    1.00
```