Lectures
gluPerspective Case Studies

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

How does glPerspective work?

```
void gluPerspective(
          GLdouble fovy,
          GLdouble aspect,
          GLdouble zNear,
          GLdouble zFar)
```

- gluPerspective computes a viewing frustum in the world coordinates.
    - fovy
        - Specifies the Field Of Viewing Angle.
    - Aspect
        - The aspect ratio of x (width) to y (height).
    - zNear specifies the distance from camera (viewer) to the near clipping plane.
    - zFar specifies the distance from camera (viewer) to the far clipping plane.

Note: zNear and zFar are only clipping plane. Neither of them have anything to do with the projection plane.

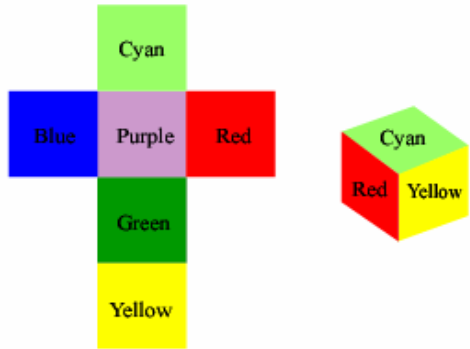
Case Study 1
Fovy Angle of gluPerspective

Case Study Setup:

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

colorcube0 ():

centered at (0,0,0)

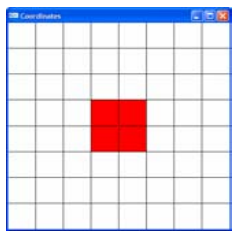


Goal:

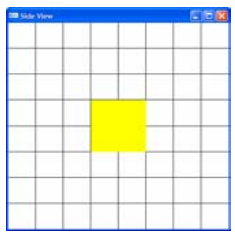
We will observe how varying fovy angle of gluPerspective will change the camera's view of the cube.

Orthogonal View

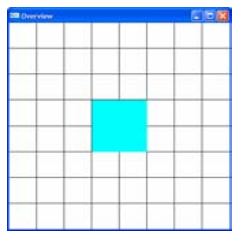
Front View



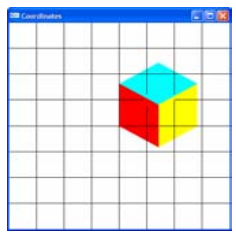
Side View



Top View



Diagonal View



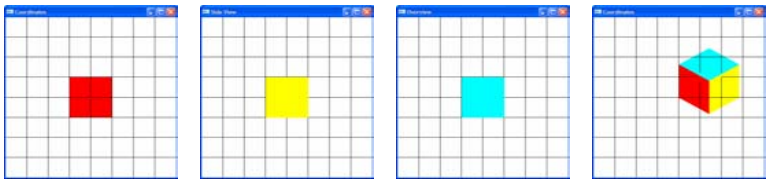
Files Used:
Perspective.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Example 1: Fovy = 90^0

Set Up Viewing Angle (Fovy) and Camera

Orthogonal View

Front View | Side View | Top View | Diagonal View

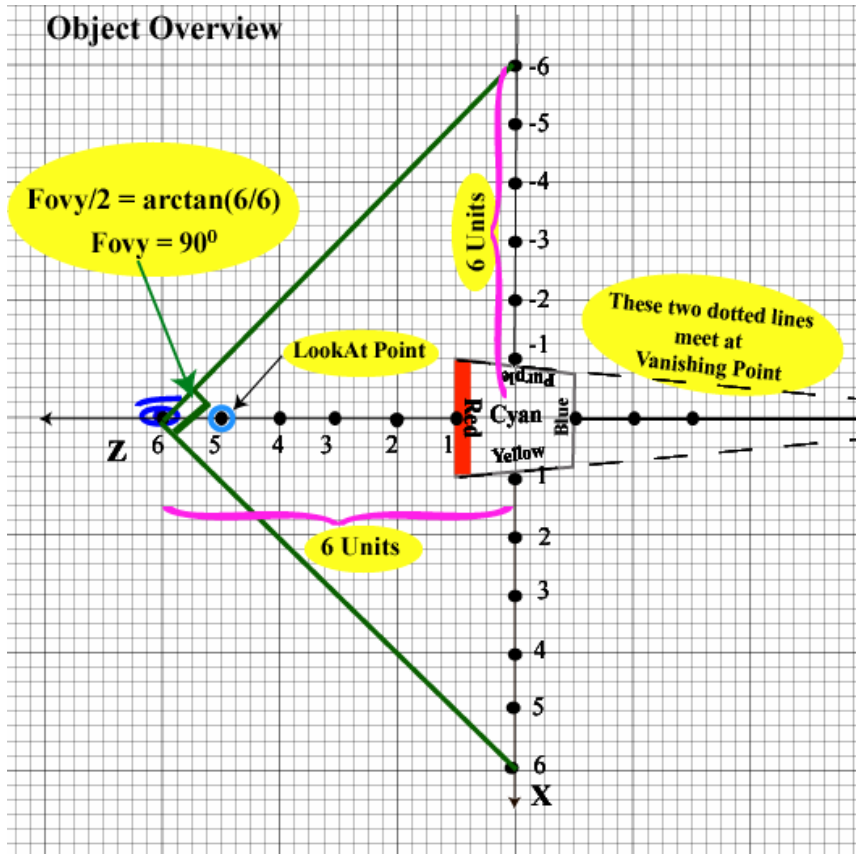


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View



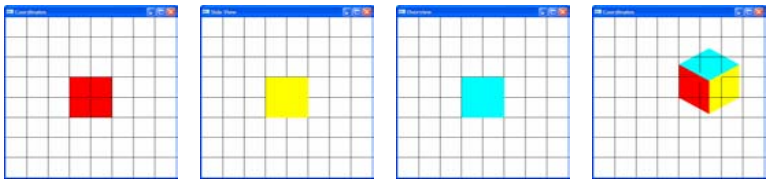
Note:

- **Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.**

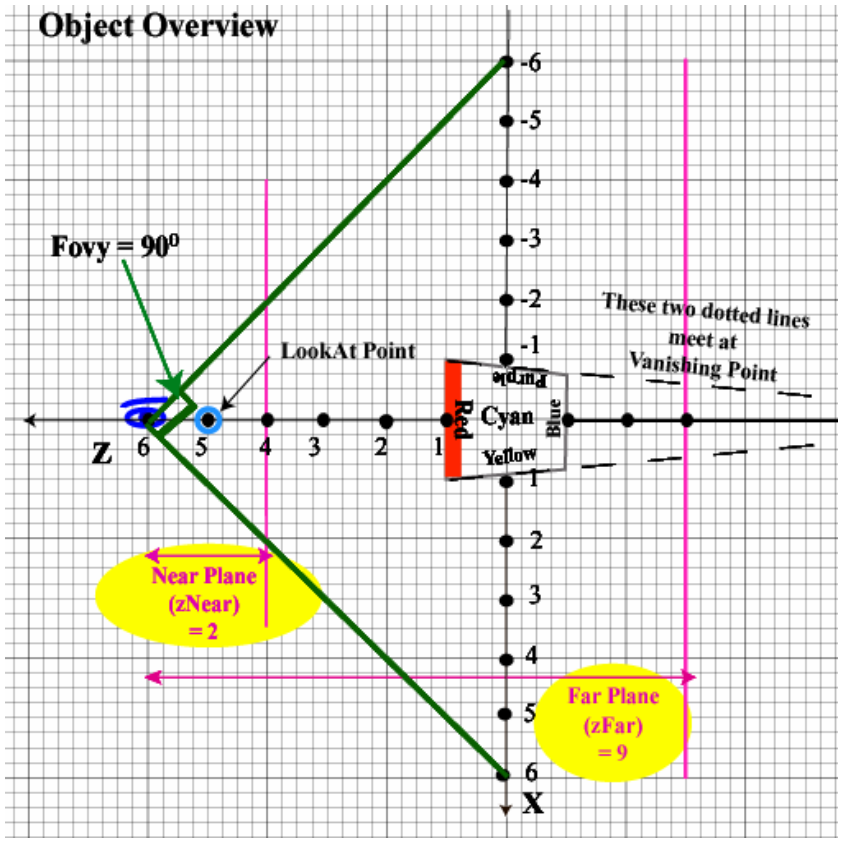
Example 1: Fovy = 90^0

Orthogonal View

Front View | Side View | Top View | Diagonal View



Set Up Near and Far Planes



Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View

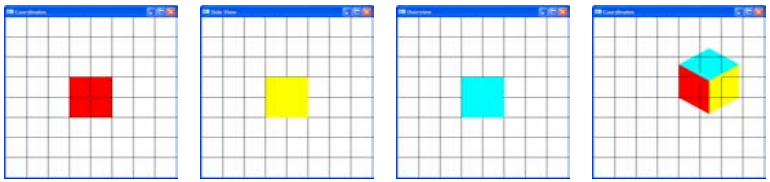
Note:

- Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.
- **Near plane and far plane together wich the fovy angle decide what is viewable to the camera.**

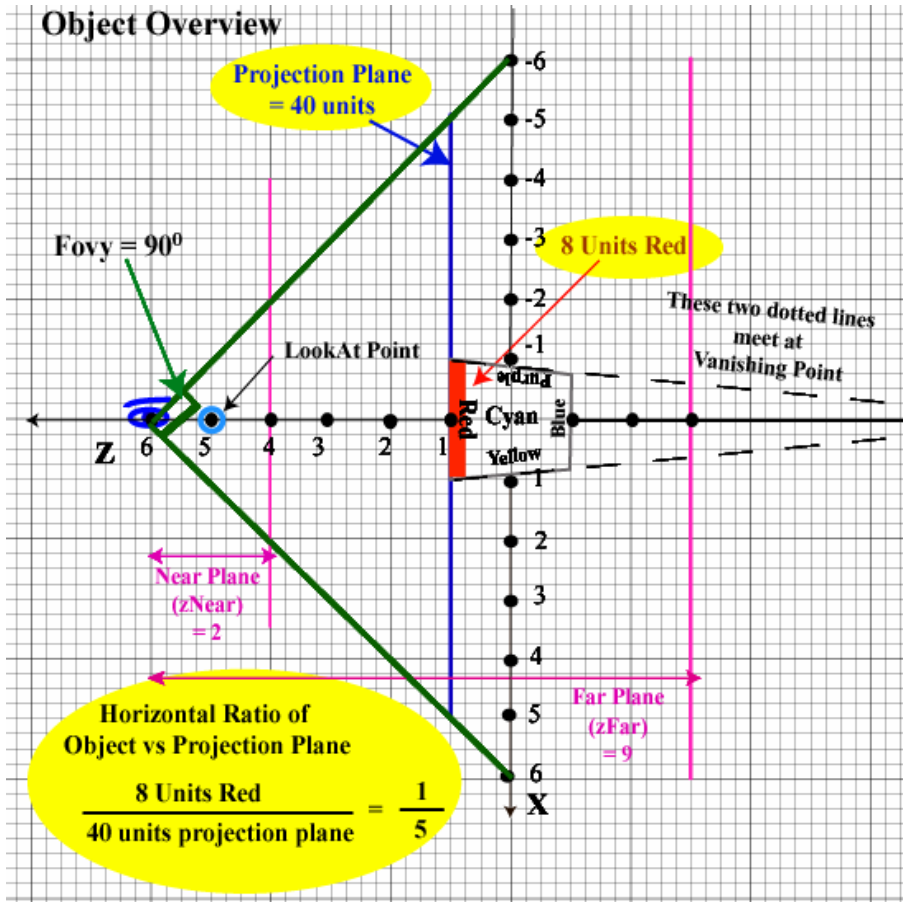
Example 1: Fovy = 90^0

Orthogonal View

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



Calculate Object vs Projection Plane Ratio



Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View

Note:

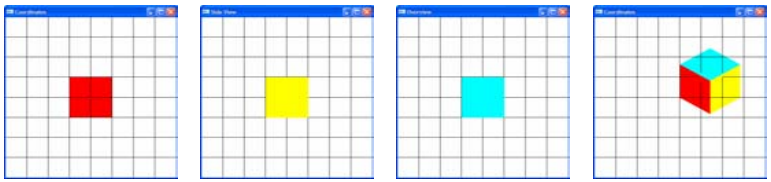
- Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.
- Near plane and far plane together wich the fovy angle decide what is viewable to the camera.

Example 2: Fovy ≈ 43.6^0

Set Up Viewing Angle (Fovy) and Camera

Orthogonal View

Front View | Side View | Top View | Diagonal View

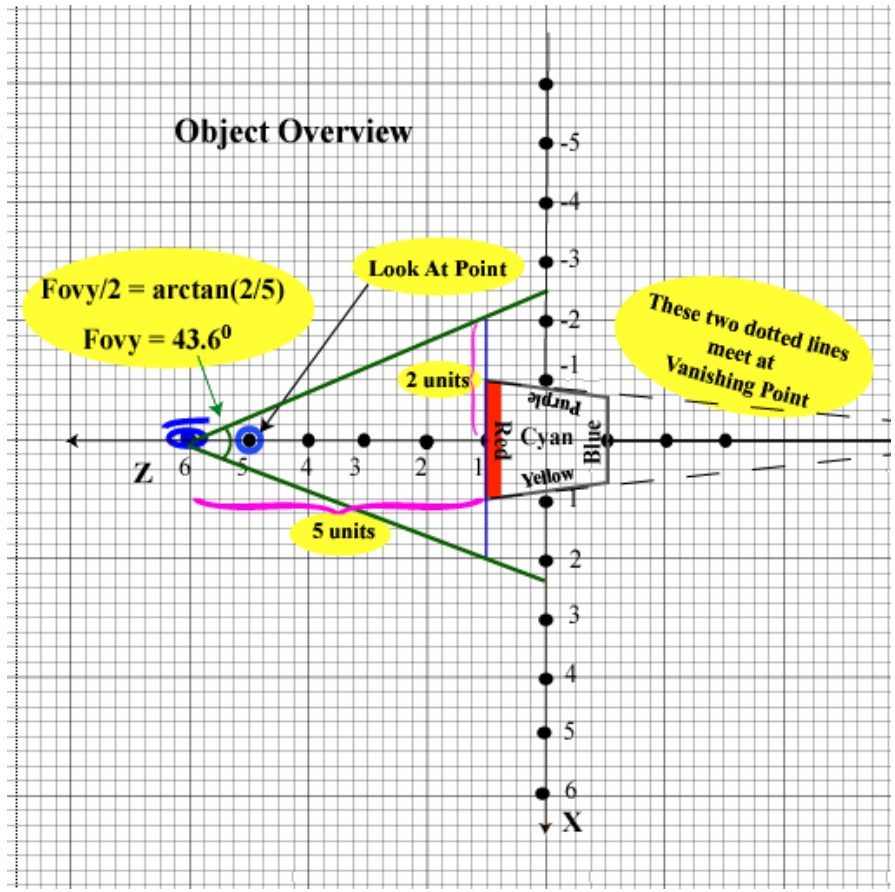


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

In Perspective View

Front View

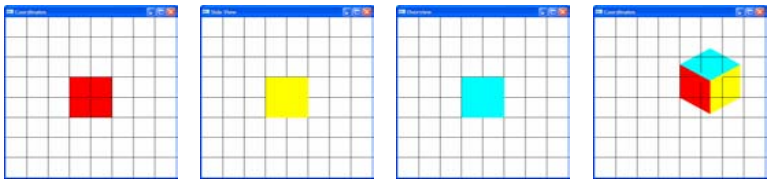


Example 2: Fovy ≈ 43.6^0

Set Up Near and Far Clipping Planes

Orthogonal View

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

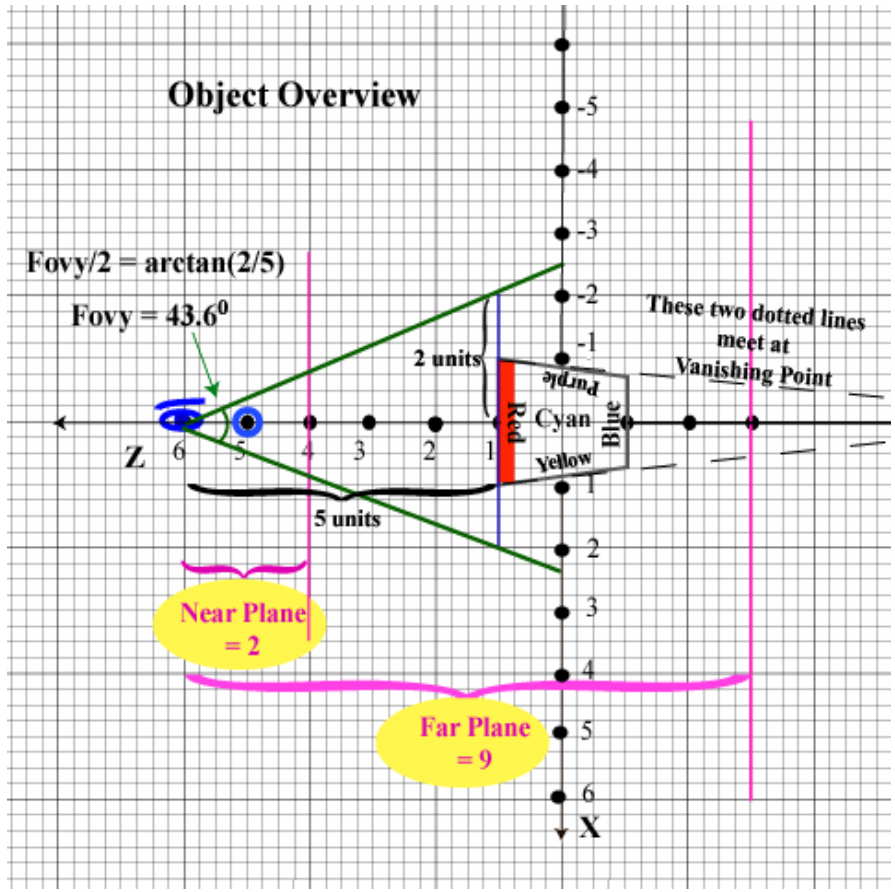


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View

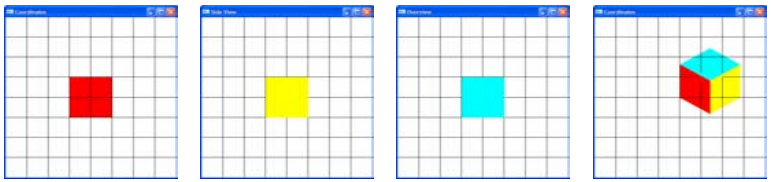


Example 2: Fovy ≈ 43.6^0

Calculate Object vs Projection Plane Ratio

Orthogonal View

Front View | Side View | Top View | Diagonal View

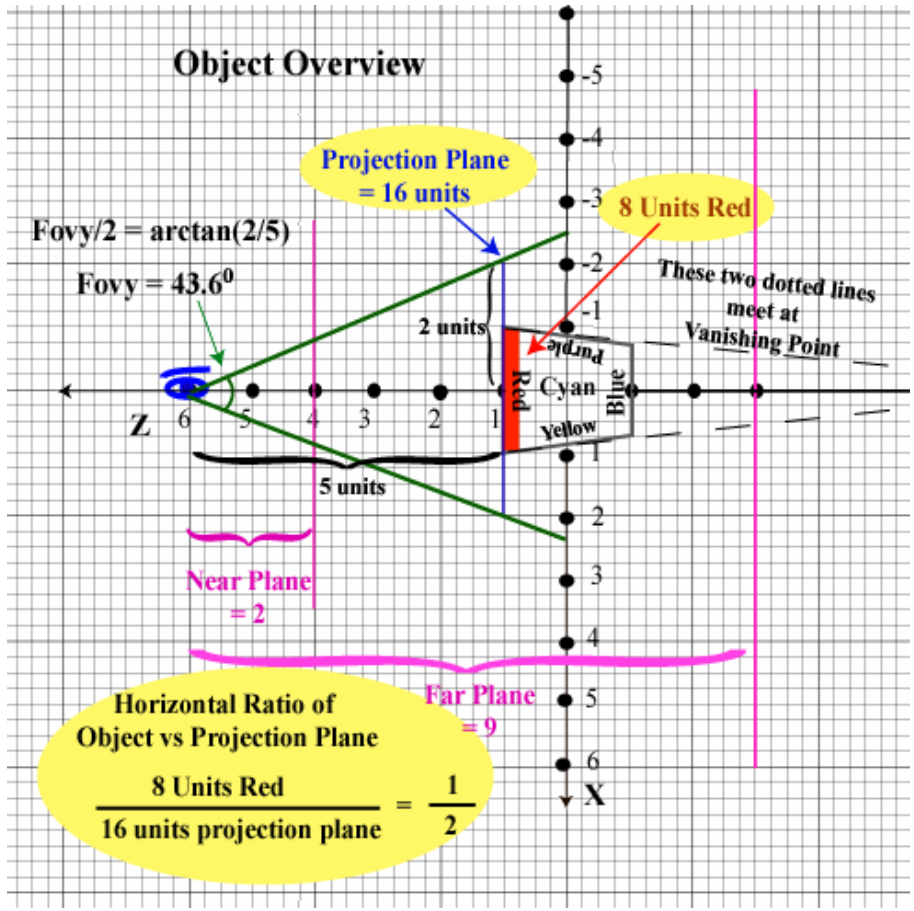


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View



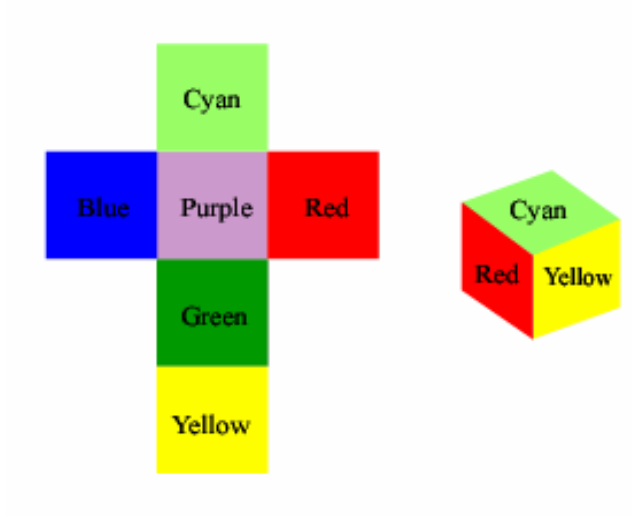
Case Study 2
Fovy Angle of gluPerspective

Case Study Setup:

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

colorcube5 ():

centered at (0,0,-1)

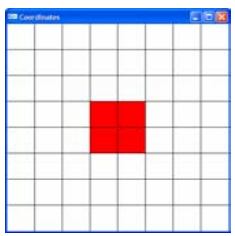


Goal:

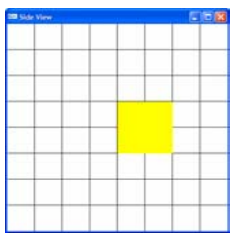
We will observe how varying fovy angle of gluPerspective will change the camera's view of the cube.

Orthogonal View

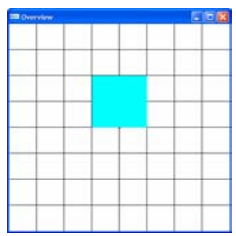
Front View



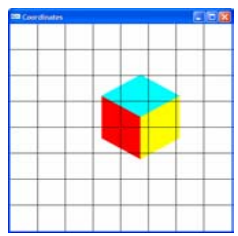
Side View



Top View



Diagonal View



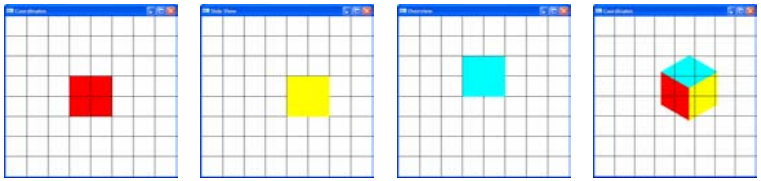
Files Used:
Shearing.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Example 1: Fovy = 90^0

Set Up View Angle (Fovy) and Camera

Orthogonal View

Front View | Side View | Top View | Diagonal View

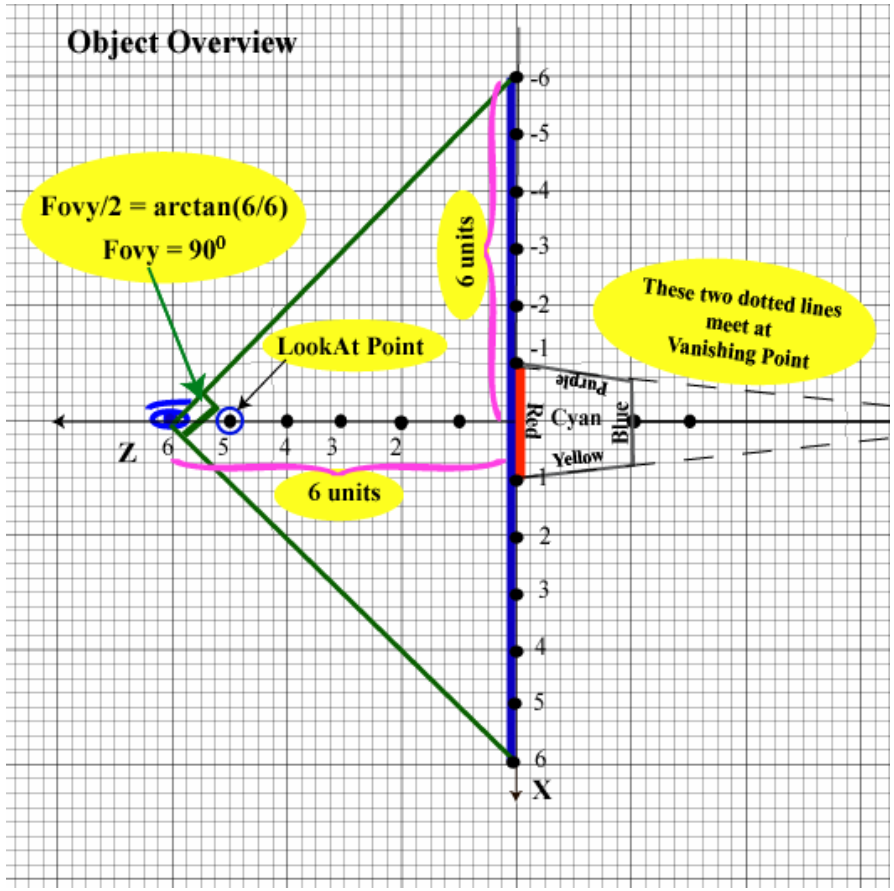


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View

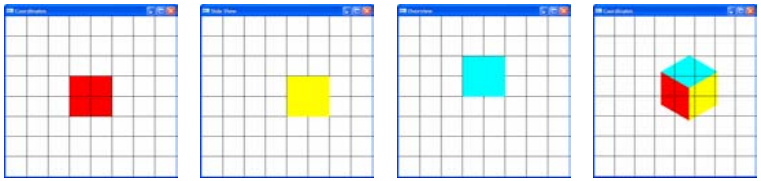


Example 1: Fovy = 90^0

Set Up Near and Far Clipping Planes

Orthogonal View

Front View | Side View | Top View | Diagonal View

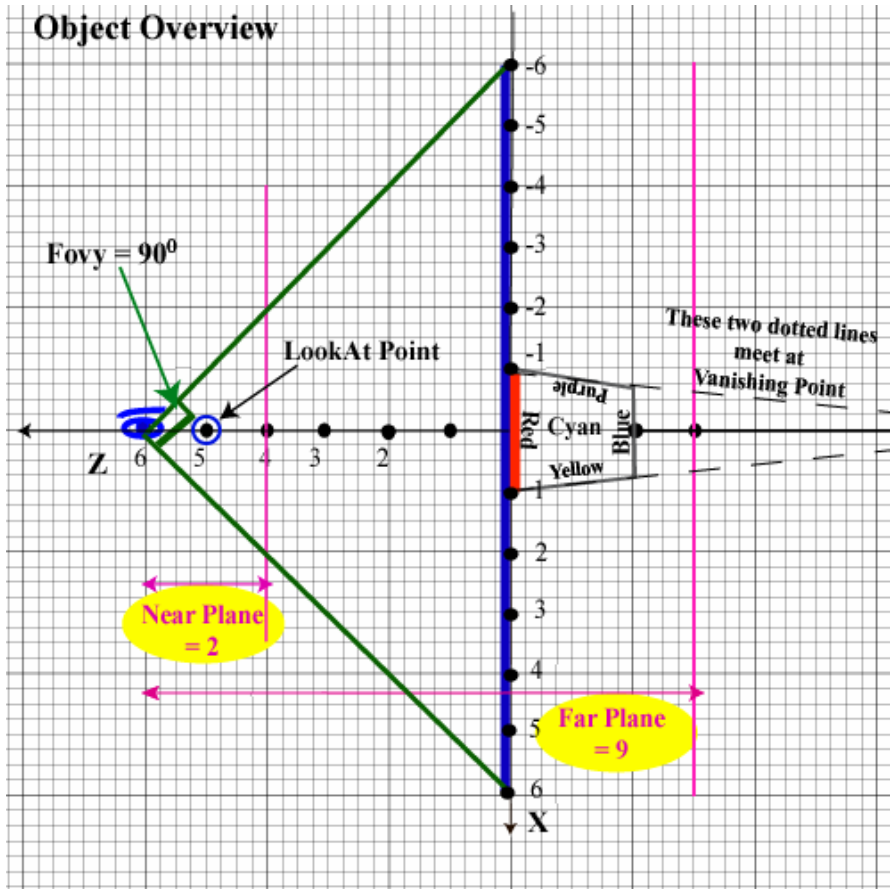


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View

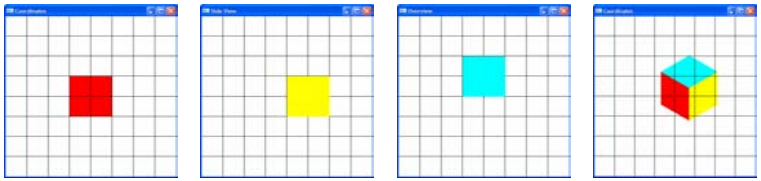


Example 1: Fovy = 90^0

Calculate Object vs Projection Plane Ratio

Orthogonal View

Front View | Side View | Top View | Diagonal View

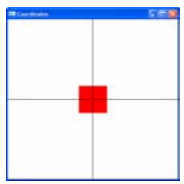


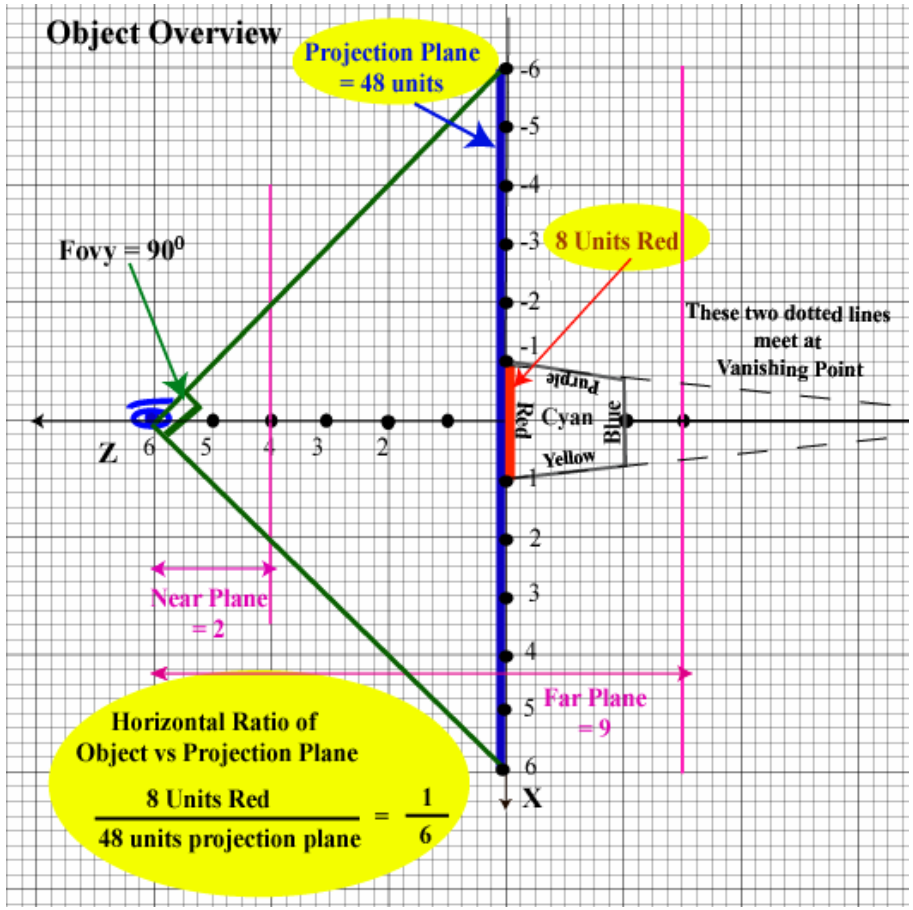
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View



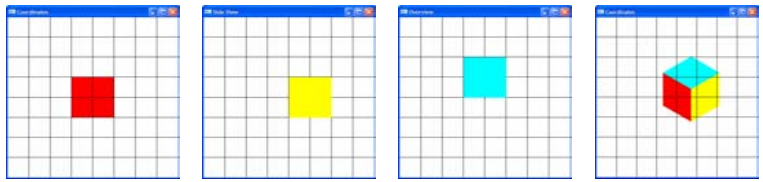


Example 2: Fovy ≈ 36.8^0

Set Up Viewing Angle (Fovy) and Camera

Orthogonal View

Front View | Side View | Top View | Diagonal View

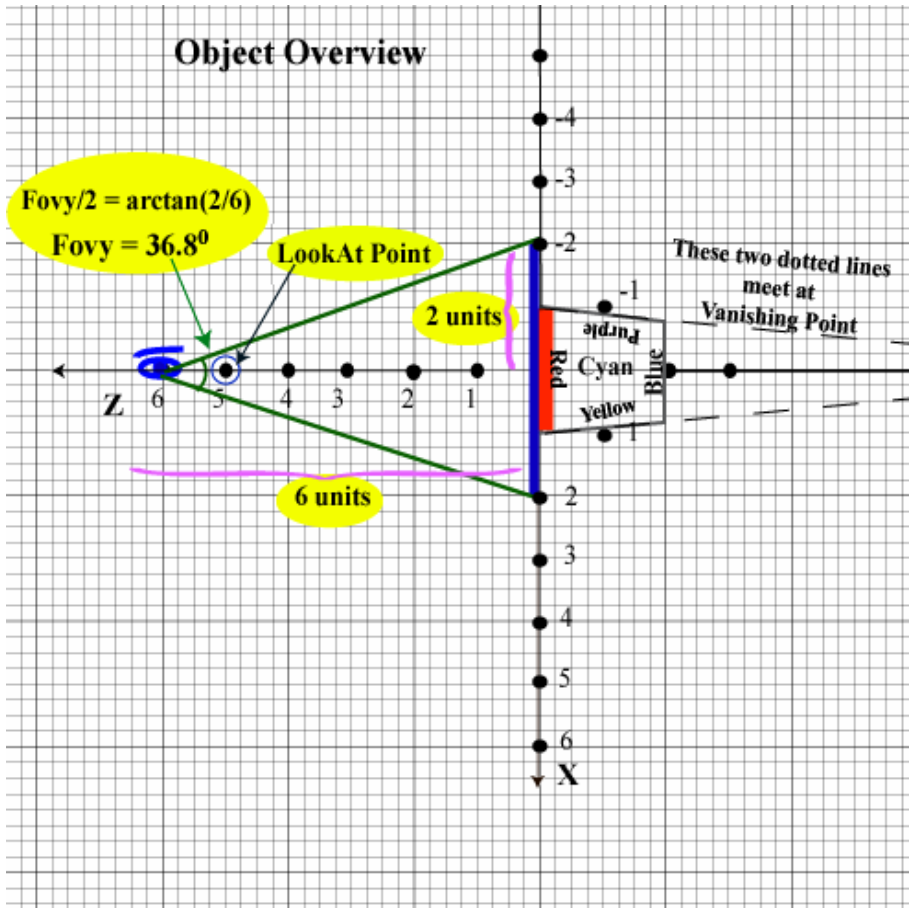


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View

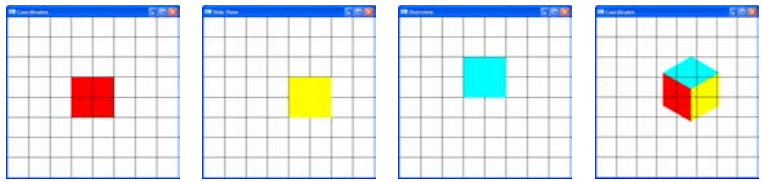


Example 2: Fovy ≈ 36.8^0

Set Up Near and Far Clipping Planes

Orthogonal View

Front View | Side View | Top View | Diagonal View

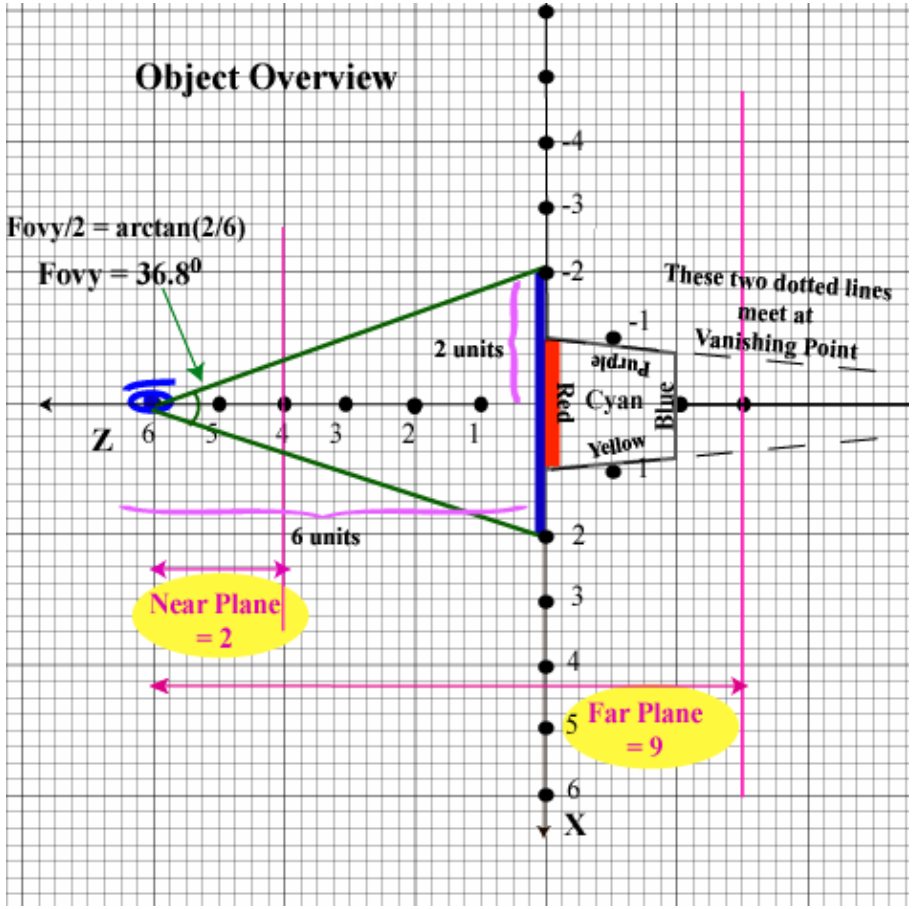


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View

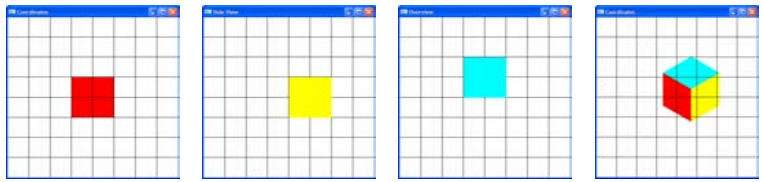


Example 2: Fovy ≈ 36.8^0

Calculate the Object vs Projection Plane Ratio

Orthogonal View

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

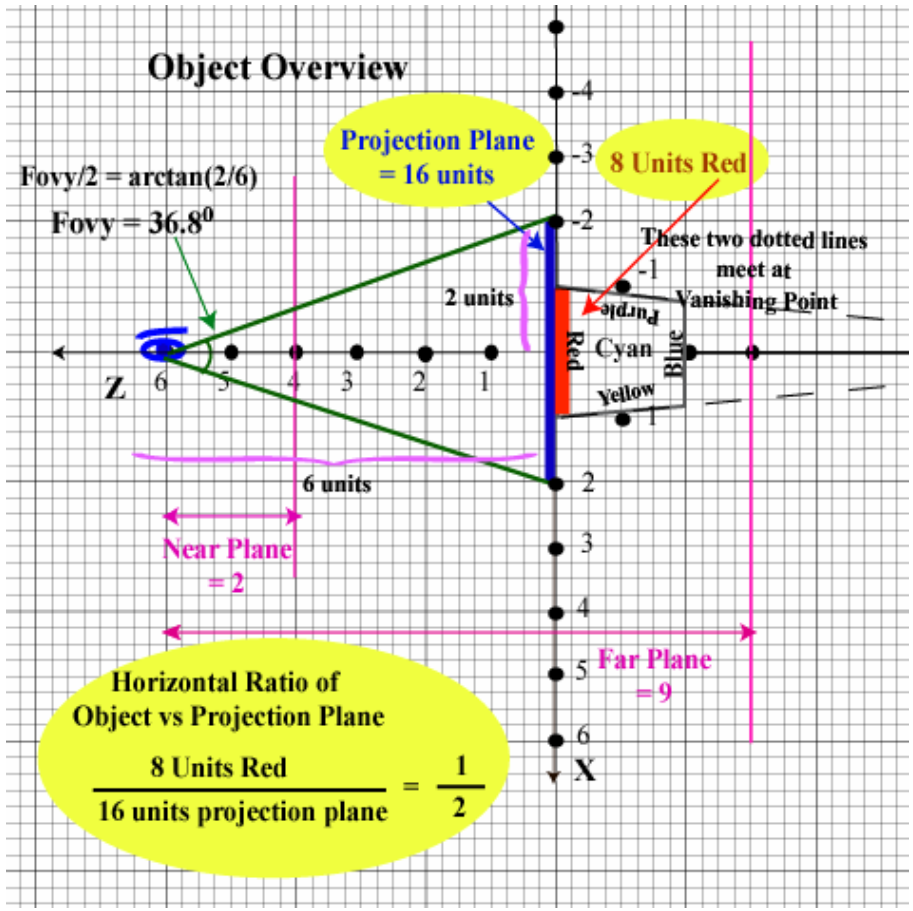


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube5();
```

Perspective View

Front View



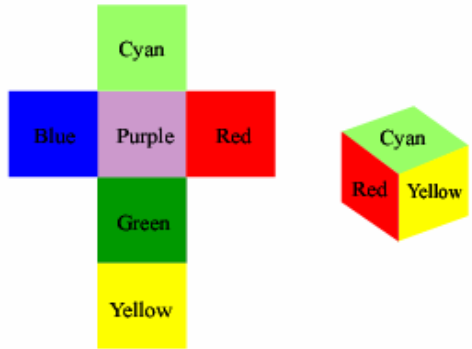
Case Study 3
zNear (Near Clipping Plane) of gluPerspective

Case Study Setup:

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

colorcube0 ():

centered at (0,0,0)

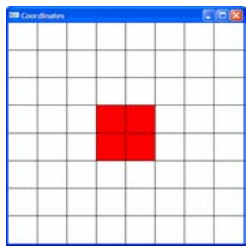


Goal:

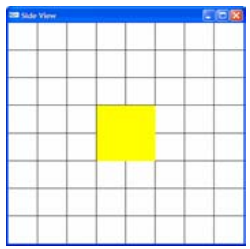
We will observe how varying zNear will change the camera's view of the cube.

Orthogonal View

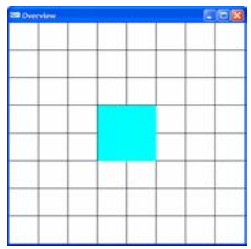
Front View



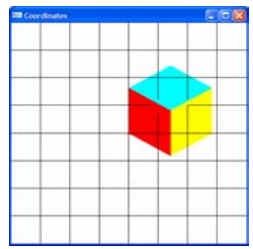
Side View



Top View



Diagonal View

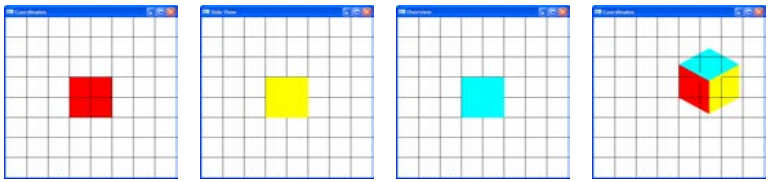


Files Used:
Perspective.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Object is Within the Viewing Frustum

Orthogonal View

Front View | Side View | Top View | Diagonal View



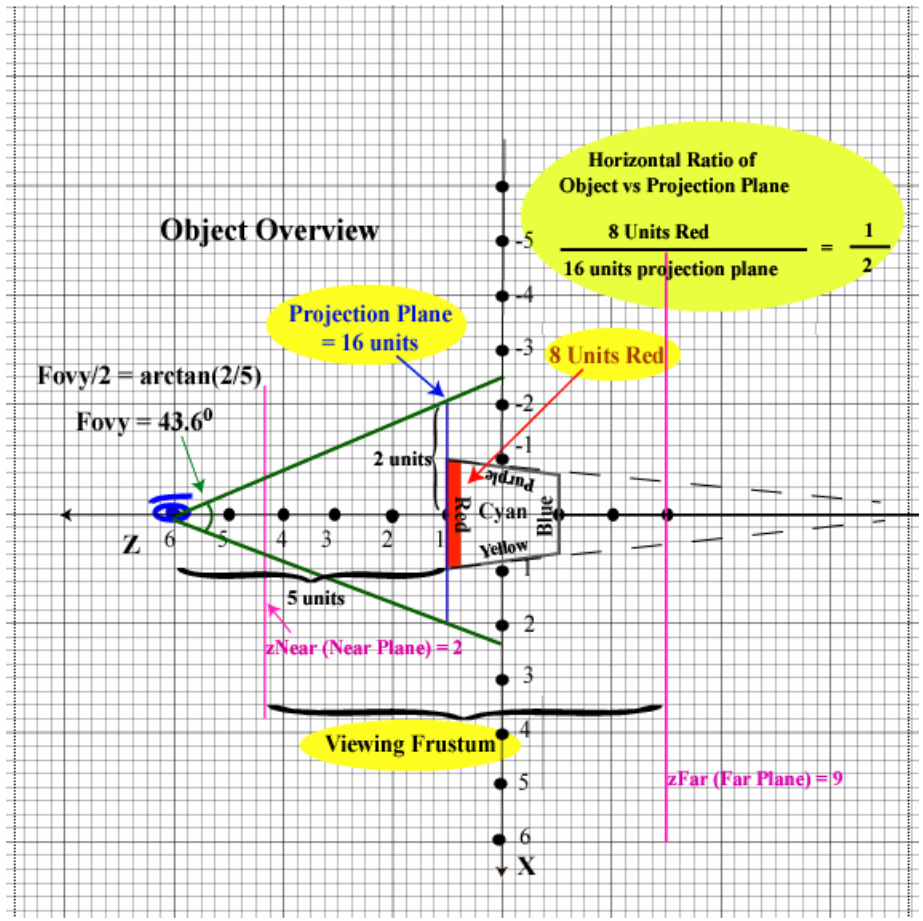
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View

Calculation of Object vs Projection Plane Ratio



Note:

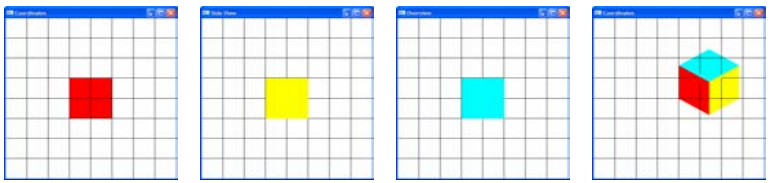
- **When object is within the viewing frustum, camera sees the front side of the cube, the red square.**
- **We will only move the zNear later on in this study case.**

Example 1: zNear = (0 ~ 5]

Changing Near Clipping Plane will not change the Front View

Orthogonal View

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

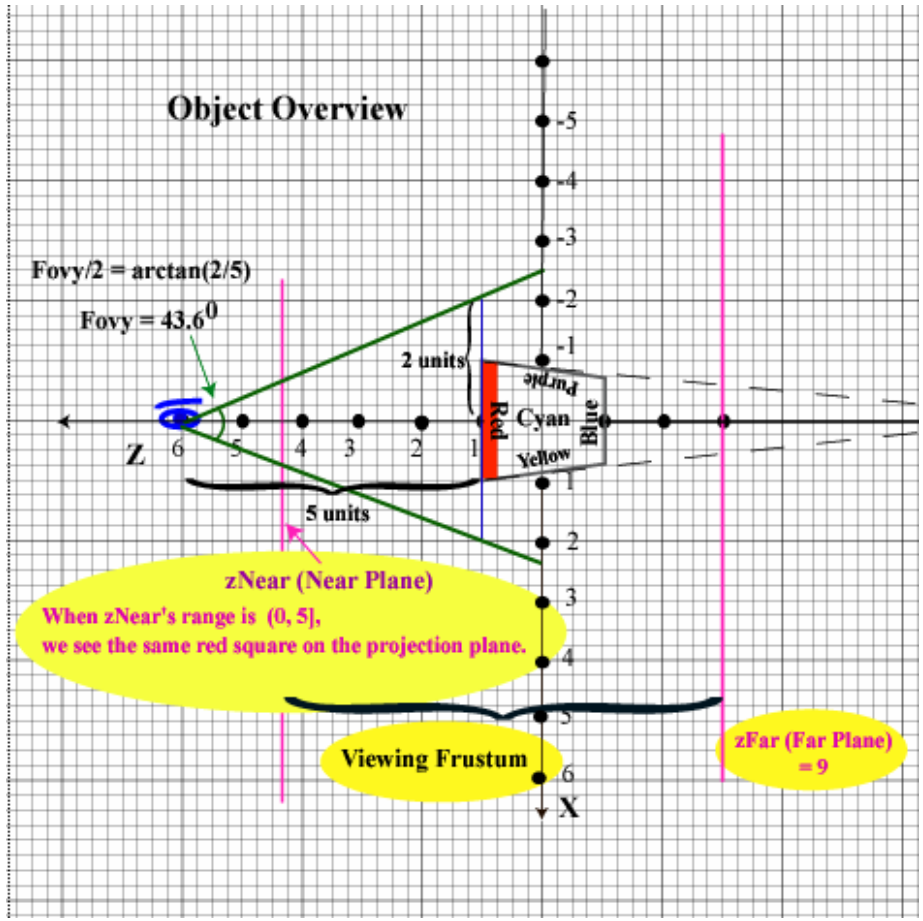


Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View



Note:

- **zNear >0 (gluPerspective Condition)**
- **When zNear = (0.1 ~ 5], the object is within the viewing frustum, the camera sees the front side of the cube, the red square.**

Example 2: zNear = (5 ~ 7)

Orthogonal View

Front View | **Side View** | **Top View** | **Diagonal View**

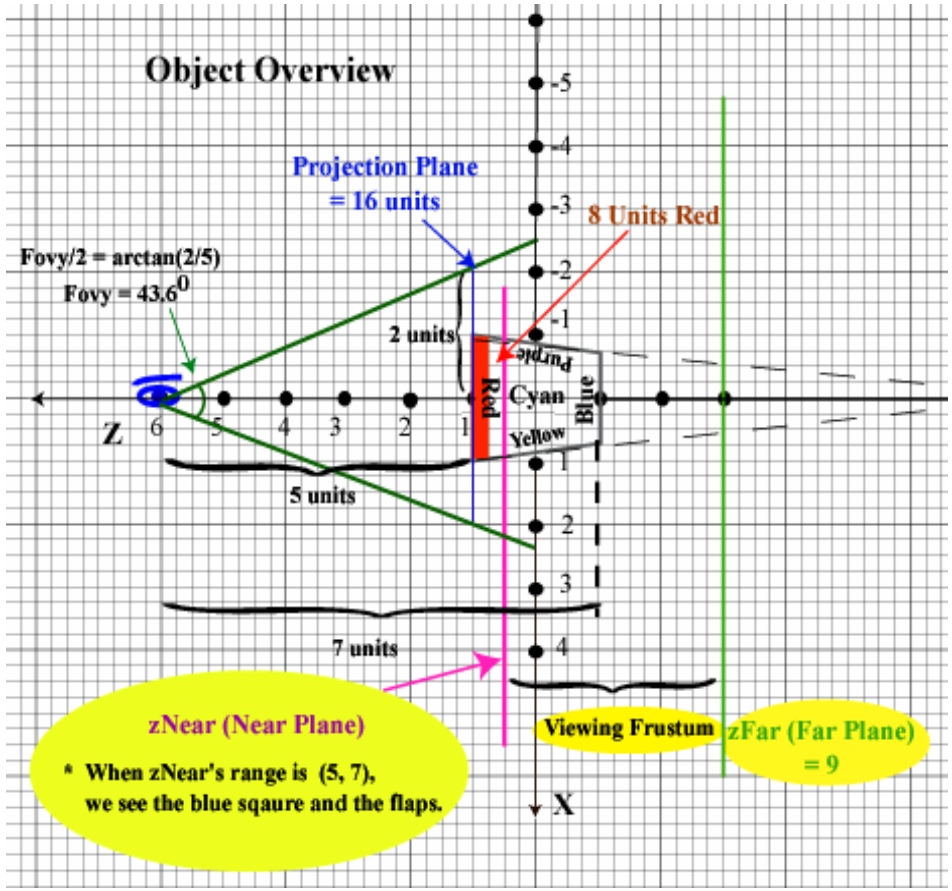
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, zNear,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View, Front View

zNear = 5 | zNear = 5.1 | zNear = 6

zNear = 6.5 | zNear = 7 | zNear = 8



Note:

• When zNear = (5 ~ 7), object is partially inside the viewing frustum, the camera sees the flaps of the cube and the back side of the cube (blue square), because the cube is vanishing towards the vanishing point. (See dashed line)

Example 2: zNear = 7

Orthogonal View

Front View | **Side View** | **Top View** | **Diagonal View**

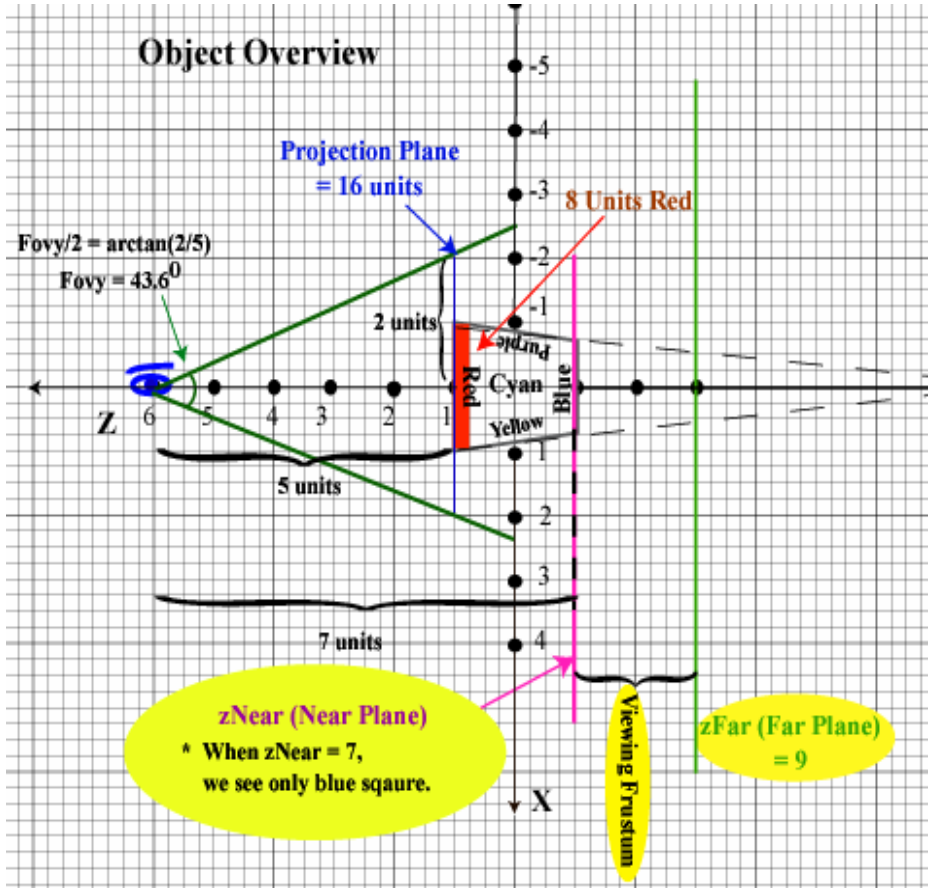
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, zNear,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View, Front View

zNear = 5 | zNear = 5.1 | zNear = 6

zNear = 6.5 | zNear = 7 | zNear = 8



Note:

• When zNear = 7, only the back side of the object is within the viewing frustum, therefore camera sees only the blue square.

Example 2: zNear > 7

Orthogonal View

Front View | **Side View** | **Top View** | **Diagonal View**

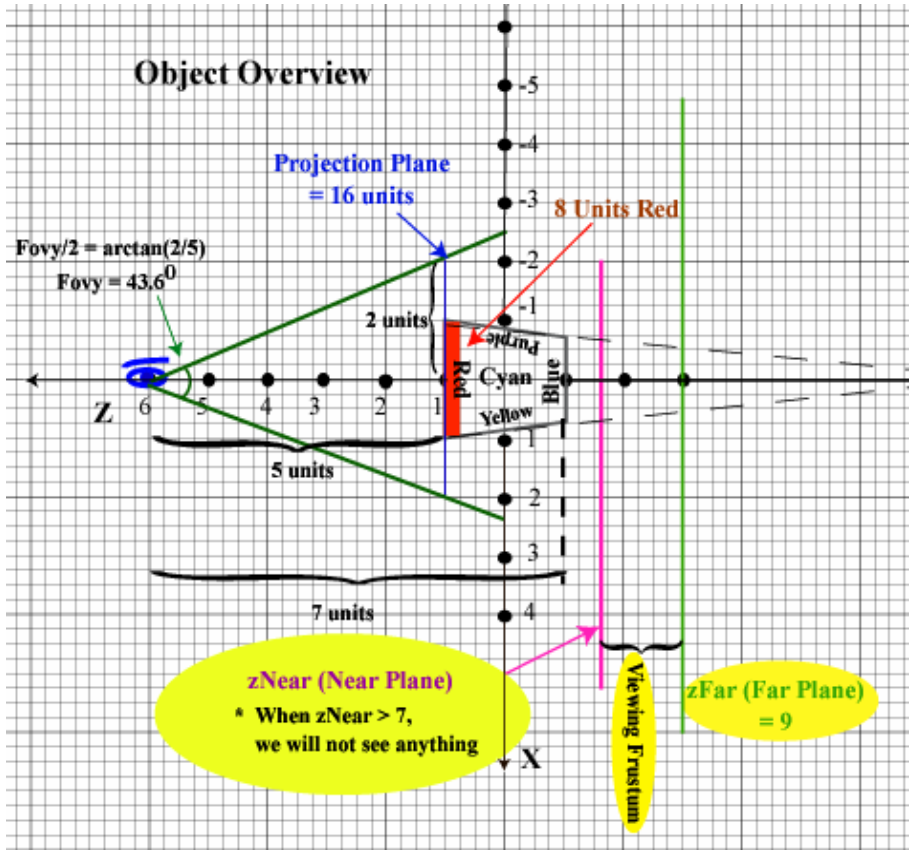
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, zNear,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View, Front View

zNear = 5 | zNear = 5.1 | zNear = 6

zNear = 6.5 | zNear = 7 | zNear = 8



Note:

• When zNear > 7, the object is completely outside the viewing frustum between zNear and zFar, therefore, the camera does not see any object.

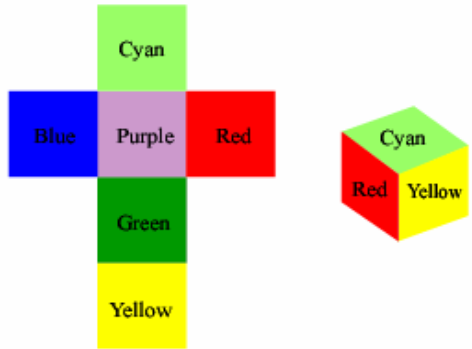
Case Study 4
zFar (Far Clipping Plane) of gluPerspective

Case Study Setup:

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

colorcube0 ():

centered at (0,0,0)

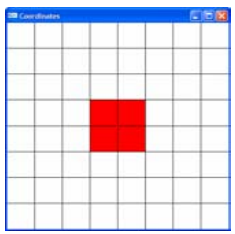


Goal:

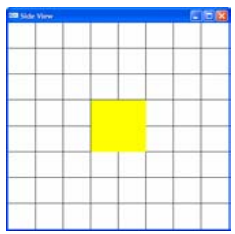
We will observe how varying zFar of gluPerspective will change the camera's view of the cube.

Orthogonal View

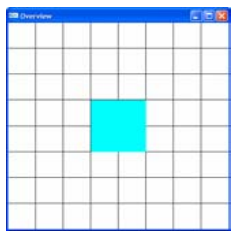
Front View



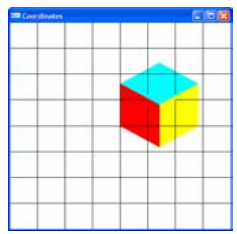
Side View



Top View



Diagonal View

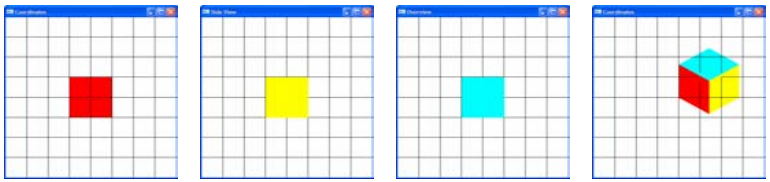


Files Used:
Perspective.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Object is Within the Viewing Frustum

Orthogonal View

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



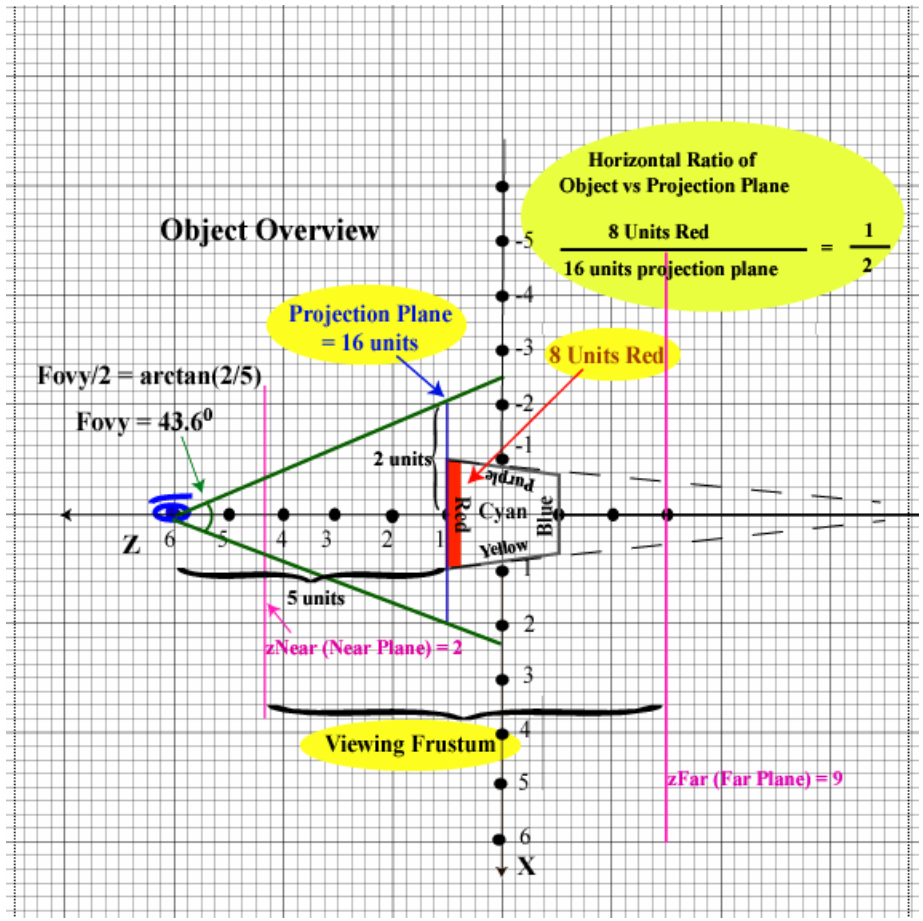
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View

Front View

Calculation of Object vs Projection Plane Ratio



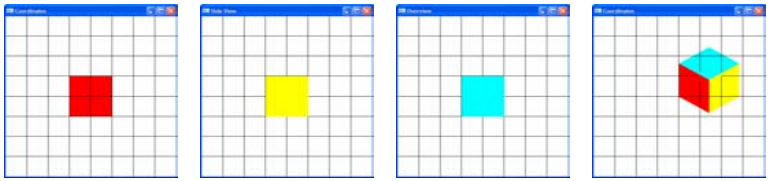
Note:

- **When object is within the viewing frustum, camera sees the front side of the cube, the red square.**
- **We will try to move only the zFar later on in this study case.**

Example 1: zFar < 5

Orthogonal View

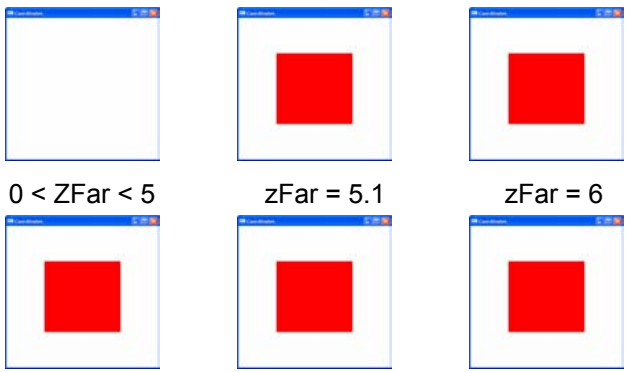
Front View **Side View** **Top View** **Diagonal View**



Code :

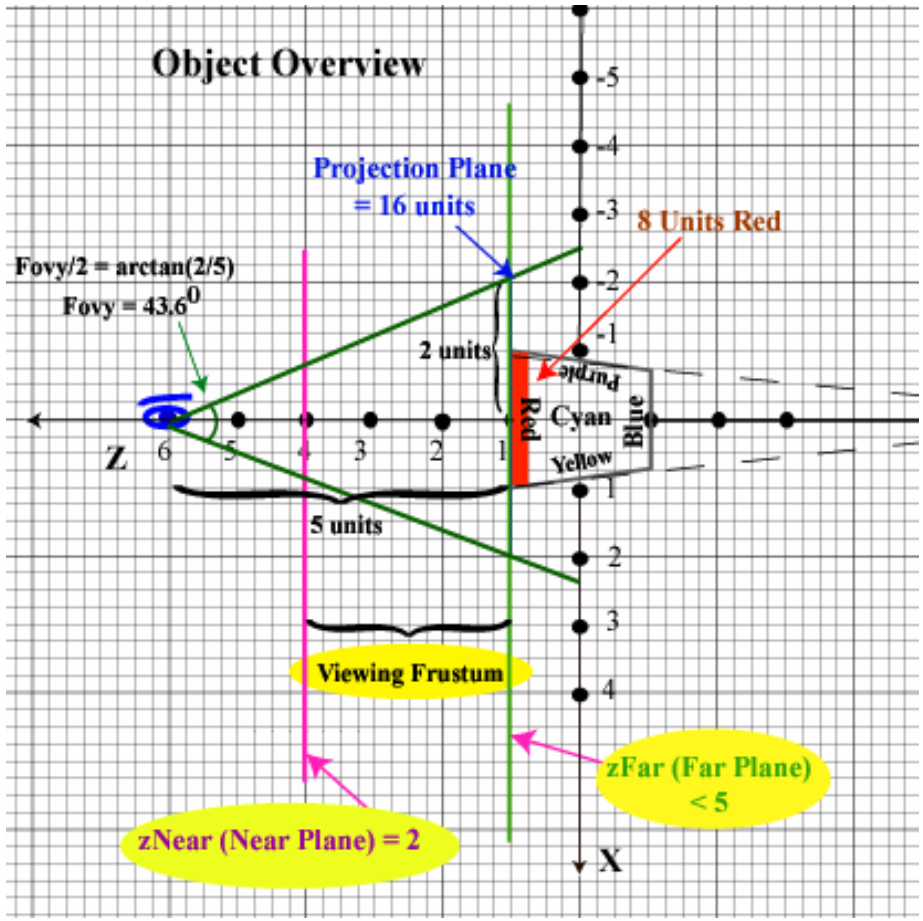
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View, Front View



0 < ZFar < 5 zFar = 5.1 zFar = 6

zFar = 7 zFar = 10 zFar = 10000



Note:

- **zFar >0 (gluPerspective Condition)**
- **When zFar < 5, object is outside the viewing frustum, the camera does not see any object.**

Example 1: zFar = [5 ~ ∞)

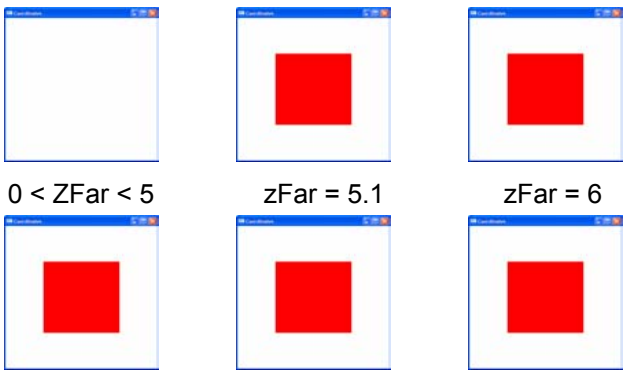
Orthogonal View

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

Code :

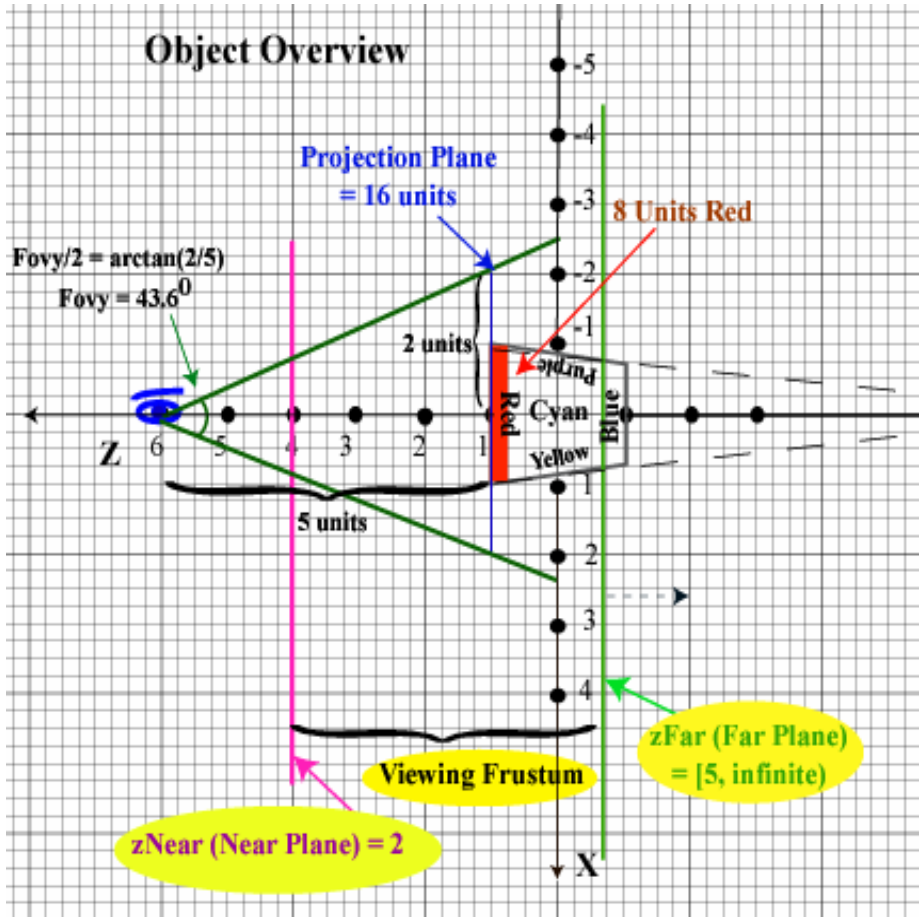
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(43.6, 1,2,9);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0();
```

Perspective View, Front View



0 < ZFar < 5 zFar = 5.1 zFar = 6

zFar = 7 zFar = 10 zFar = 10000



Note:

- **zFar >0 (gluPerspective Condition)**
- **When zFar is [5, ∞), camera sees the front part of the cube, the red square. Changing the zFar plane will not change the camera's view, because the front part of object is inside the viewing frustum.**

Case Study 5 - Where is the Vanishing Point?

Case Study Setup:

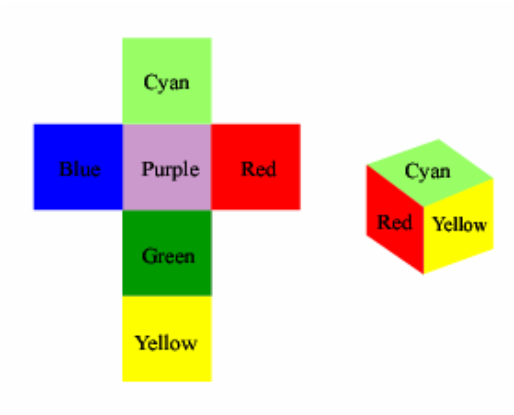
Assume in the world coordinates we have one color cube of size two, whose front is red.

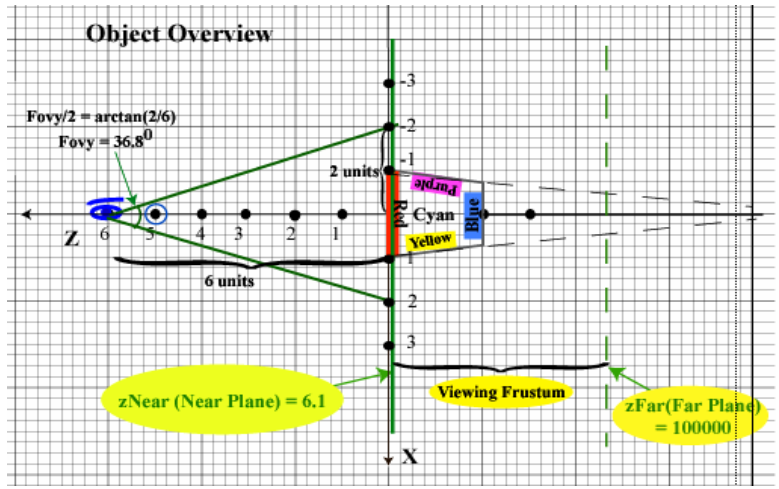
colorcube5 ():

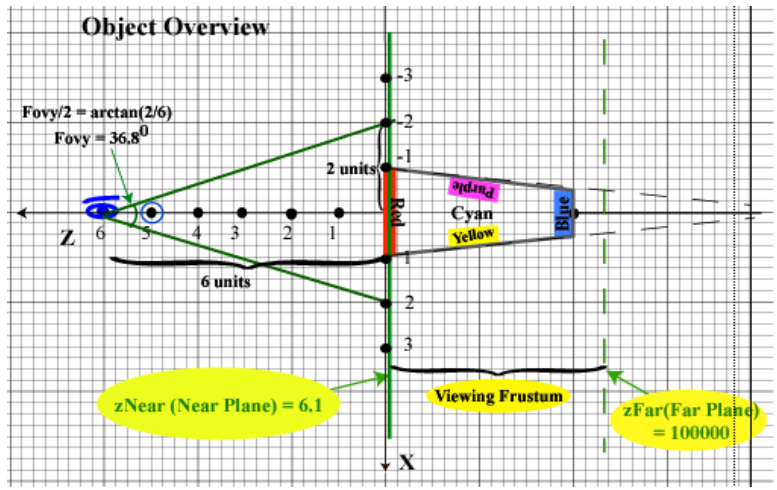
centered at (0,0,-1)

Goal:

- When the near plane cuts off part of the object, the object is partially within the viewing frustum; therefore, the camera sees flaps and the back plane (in this case, blue square).
- We will change the length of the object, and observe the ratio of total object seen by the camera (flaps plus blue square) and the back plane (blue square) of the cube to estimate the vanishing point.



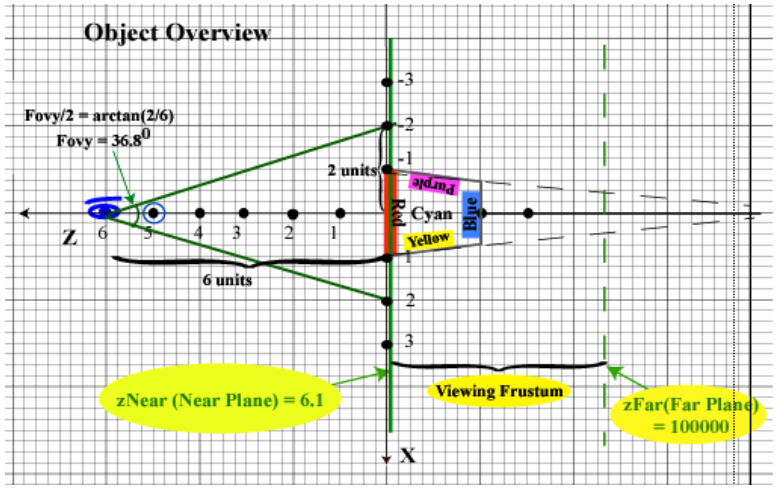




Length of object = 2 (no scaling)

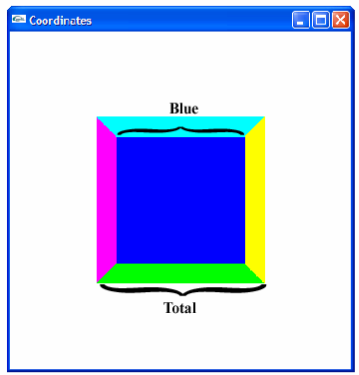
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,5.1,100000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
glScale(1,1,a);
colorcube5();
```



Note:

- **a = 1 ➔ Length = 2 X 1 = 2 (inches)**
- **Window Size = 10.5cmX10.5cm**
  - **"Total" is about half of window size.**
- **We do the actual measurements of "Blue" vs "Total" in "cm" for greater accuracy.**

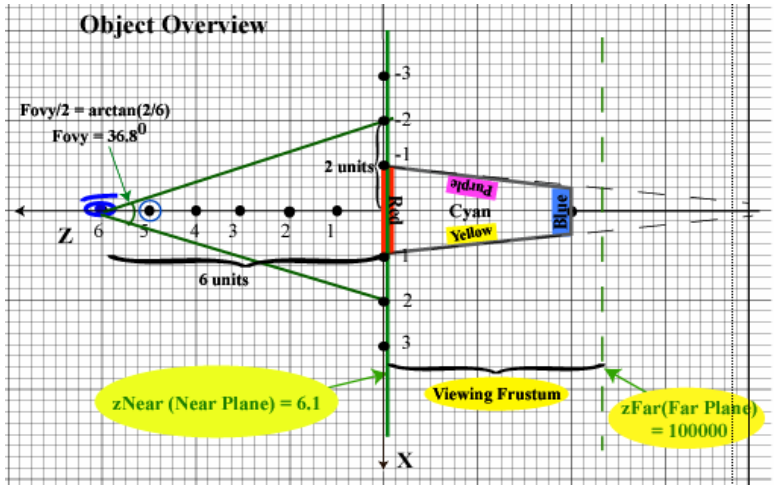


$$\frac{\text{Blue}}{\text{Total}} = \frac{3.9\text{cm}}{5.2\text{cm}} = 0.75$$

Length of object = 4

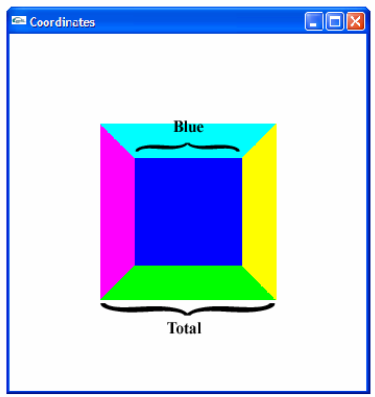
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,5.1,100000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
glScale(1,1,a);
colorcube5();
```



Note:

- **a = 2 ➔ Length = 2 X 2 = 4 (inches)**
- **Window Size = 10.5cmX10.5cm**
  - **"Total" is about half of window size.**
- **We do the actual measurements of "Blue" vs "Total" in "cm" for greater accuracy.**

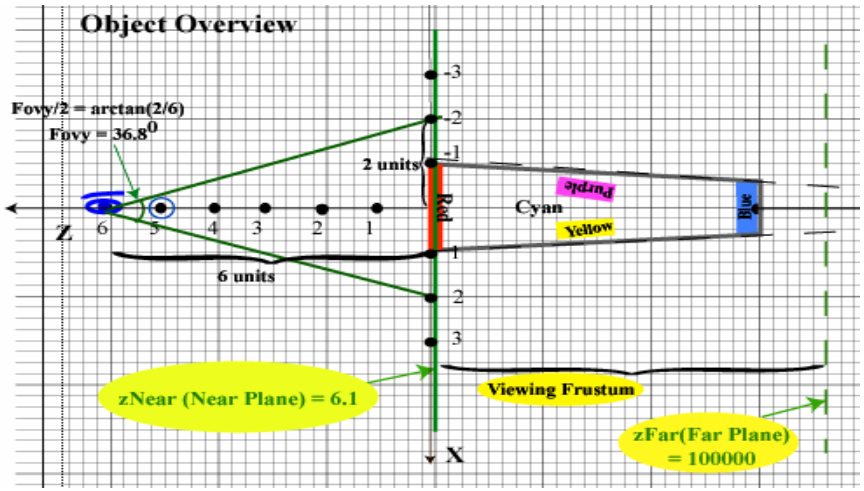


$$\frac{\text{Blue}}{\text{Total}} = \frac{3.1\text{cm}}{5.2\text{cm}} \approx 0.5962$$

Length of object = 6

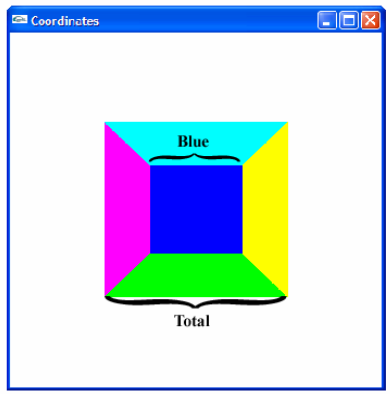
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,5.1,100000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
glScale(1,1,a);
colorcube5();
```



Note:

- **a = 3 ➔ Length = 2 X 3 = 6 (inches)**
- **Window Size = 10.5cmX10.5cm**
  - **"Total" is about half of window size.**
- **We do the actual measurements of "Blue" vs "Total" in "cm" for greater accuracy.**

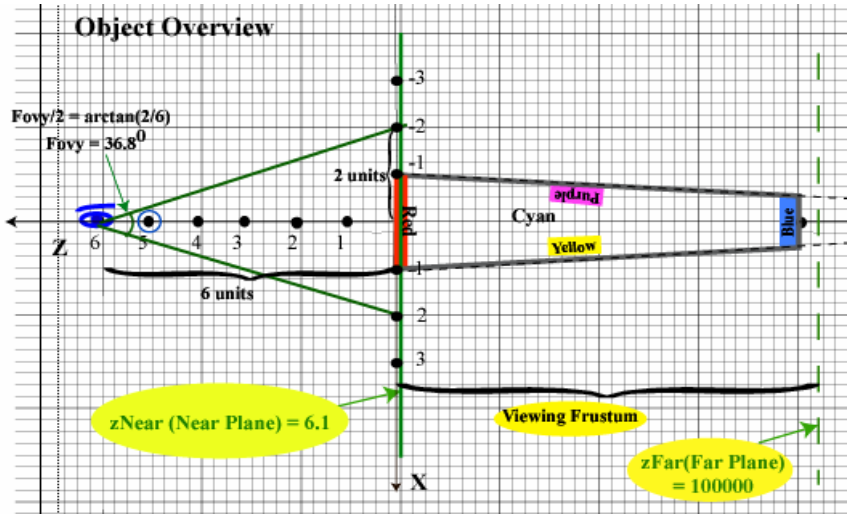


$$\frac{\text{Blue}}{\text{Total}} = \frac{2.6\text{cm}}{5.2\text{cm}} \approx 0.5$$

Length of object = 8

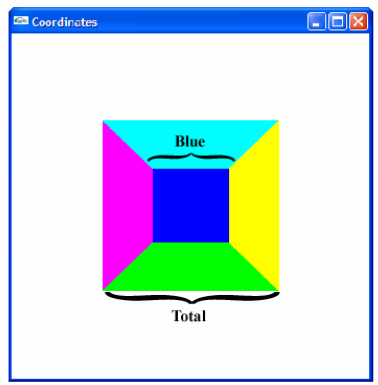
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(36.8, 1,5.1,100000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,6,0,0,5,0,1,0);
glScale(1,1,a);
colorcube5();
```



Note:

- **a = 4 ➔ Length = 2 X 4 = 8 (inches)**
- **Window Size = 10.5cmX10.5cm**
  - **"Total" is about half of window size.**
- **We do the actual measurements of "Blue" vs "Total" in "cm" for greater accuracy.**

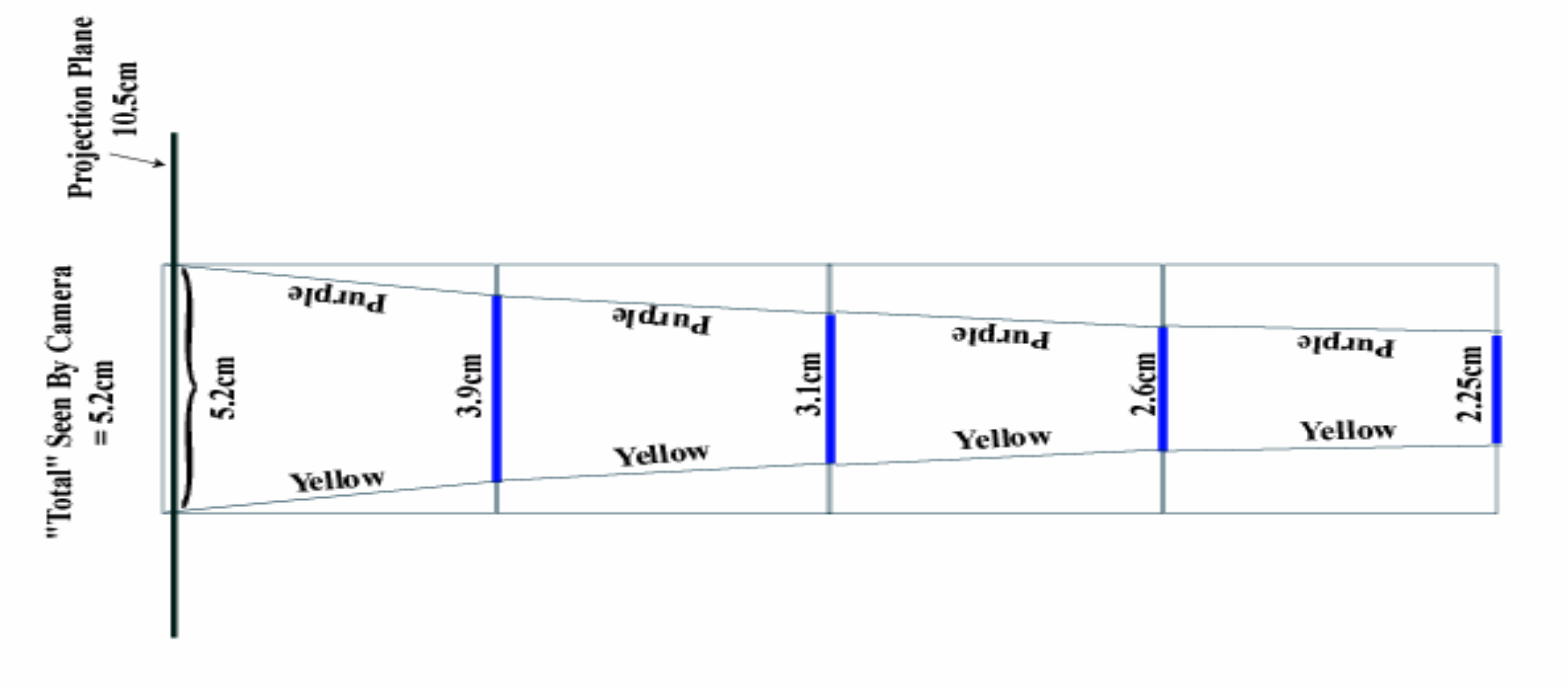


$$\frac{\text{Blue}}{\text{Total}} = \frac{2.25\text{cm}}{5.2\text{cm}} \approx 0.43269$$

Summary of Data Collected

- After Plotting the Data, we find OpenGL does not calculate vanishing point with straight lines.

- When designing the scene, we don't have to worry about the exact location of the the vanishing point, OpenGL does the magic for us and "foreshortens" the objects (makes the far objects look smaller).



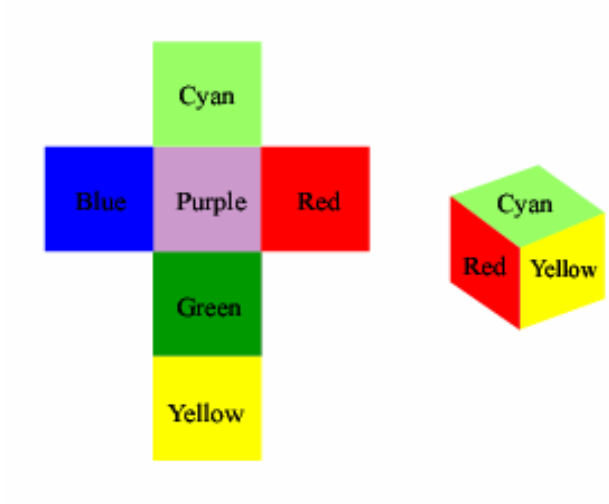
Case Study 6
Aspect Ratio of gluPerspective

Case Study Setup:

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

colorcube0 ():

centered at (0,0,0)

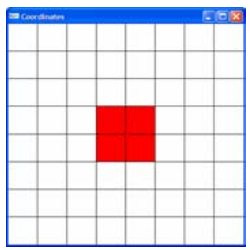


Goal:

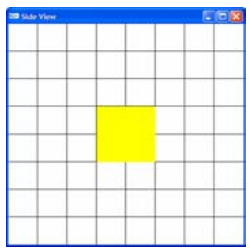
We will observe how varying Aspect Ratio of gluPerspective while fixing the window's Aspect Ratio will change the camera's view of the cube.

Orthogonal View

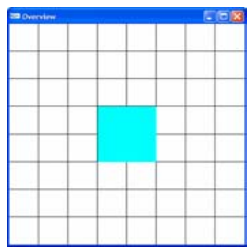
Front View



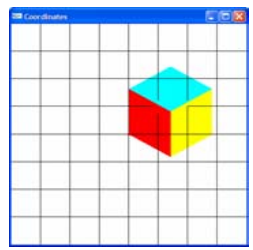
Side View



Top View



Diagonal View

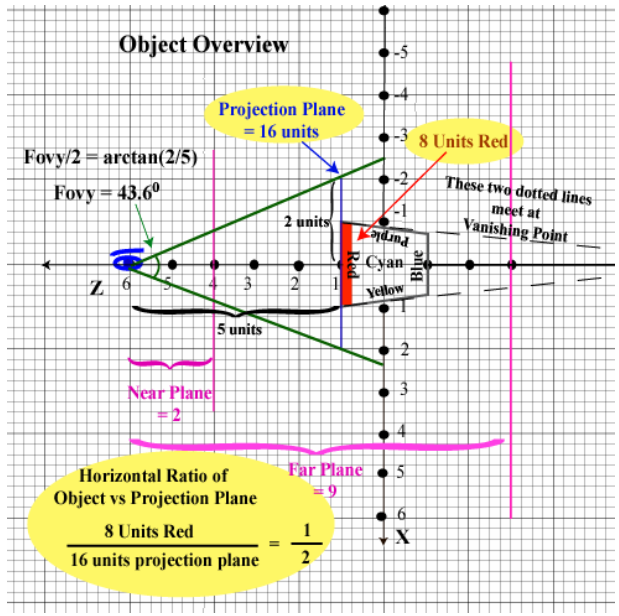


Files Used:
Perspective.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Case Study 6 – Example 1
gluPerspective Aspect Ratio matches Window's Aspect Ratio

Code :

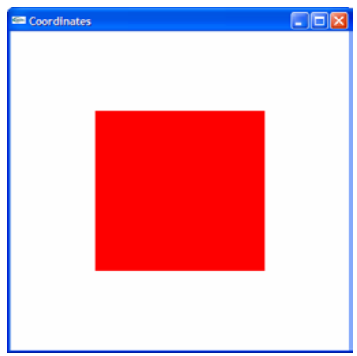
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(43.6, 1,2,9); //aspect ratio 1

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     gluLookAt(0,0,6,0,0,5,0,1,0);
     colorcube0(); // a 2x2x2 cube,
                   // centered at (0,0,0)

// in main
glutInitWindowSize(400,400);
glutReshapeFunc(reshape);
```



Note:

- **Aspect Ratio: width/height = ratio**
- **When aspect ratio of gluPerspective matches the window's Aspect Ratio, object will not be distorted.**
  - **In this case:**
    - **Window's Aspect Ratio: 400px/400px = 1**
    - **gluPerspectiive's Aspect Ratio: 1**
  - **Therefore, object is not distorted.**

Perspective View, Front View



Case Study 6 – Example 2
gluPerspective Aspect Ratio is twice of Window's Aspect Ratio

Code :

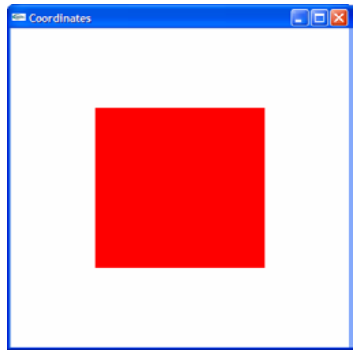
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(43.6, 2,2,9); //aspect ratio 2

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     gluLookAt(0,0,6,0,0,5,0,1,0);
     colorcube0(); // a 2x2x2 cube,
                   // centered at (0,0,0)

// in main
glutInitWindowSize(400,400);
glutReshapeFunc(reshape);
```

Note:

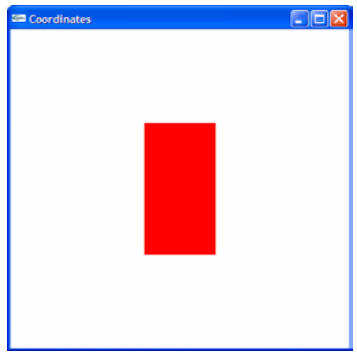
- **Aspect Ratio: width/height ratio**
- **When aspect ratio of gluPerspective matches the window's Aspect Ratio, object will not be distorted.**
  - **In this case:**
    - **Window's Aspect Ratio: 400px/400px = 1**
    - **gluPerspective's Aspect Ratio: 2**
  - **When gluPerspective's Aspect Ratio is twice the Window', object's height remains the same, but its width shrinks by half.**

Perspective View, Front View

gluPerspective Aspect Ratio = 1



gluPerspective Aspect Ratio = 2



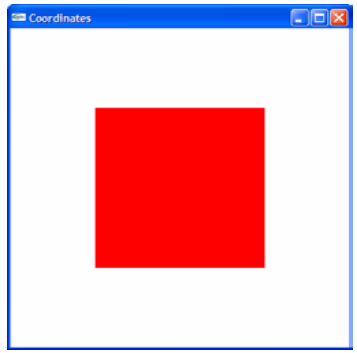
Case Study 6 – Example 3
gluPerspective Aspect Ratio is half of Window's Aspect Ratio

Code :

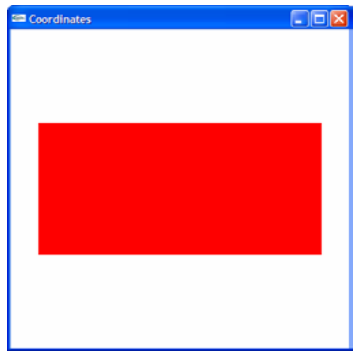
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(43.6, 0.5,2,9); //aspect ratio 0.5

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     gluLookAt(0,0,6,0,0,5,0,1,0);
     colorcube0(); // a 2x2x2 cube,
                   // centered at (0,0,0)

// in main
glutInitWindowSize(400,400);
glutReshapeFunc(reshape);
```

Perspective View, Front View

gluPerspective Aspect Ratio = 1



gluPerspective Aspect Ratio = 0.5



Note:

- **Aspect Ratio: width/height ratio**
- **When aspect ratio of gluPerspective matches the window's Aspect Ratio, object will not be distorted.**

  - **In this case:**
    - **Window's Aspect Ratio: 400px/400px = 1**
    - **gluPerspective's Aspect Ratio: 0.5**

  - **When gluPerspective's Aspect Ratio is half of Window', object's height remains the same, but its width is twice of its height. (This behavior is different from glFrustum. When glFrustum's Aspect Ratio is half of Window's, object's width remains the same, but the height shrinks by half.)**