Lectures
glFrustum Case Studies

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

How does glFrustum work?

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

- glFrustum computes a viewing frustum in the world coordinates.
  - left and right
    - Specify the coordinates for the left and right vertical clipping planes.
  - bottom and top
    - Specify the coordinates for the bottom and top horizontal clipping planes.
  - near and far
    - Specify the distances to the near and far depth clipping planes. Both distances must be positive.

Note: “near” and “far” are only clipping planes. Neither of them have anything to do with the projection plane.

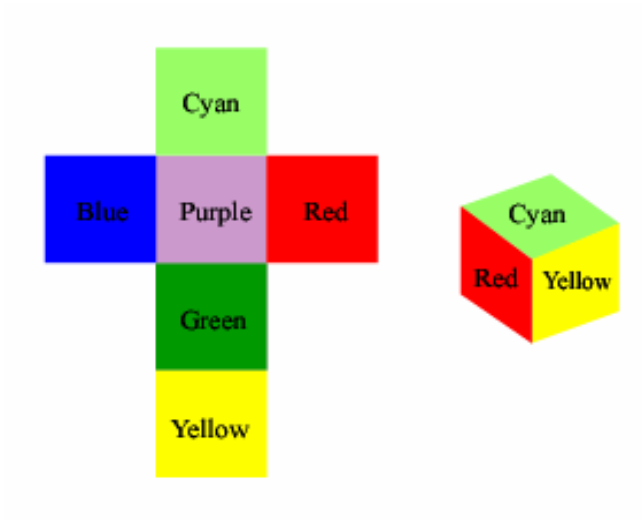
Case Study 1
left, right, top, bottom of glFrustum

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)

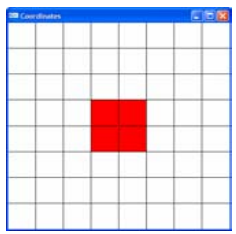


Goal:

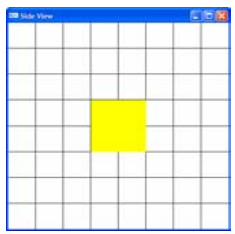
We will observe how varying left, right, top, bottom of gluFrustum will change the camera's view of the cube.

Orthogonal View

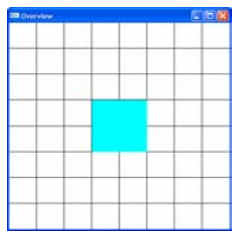
Front View



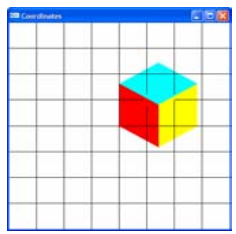
Side View



Top View



Diagonal View



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

Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 2, far = 9

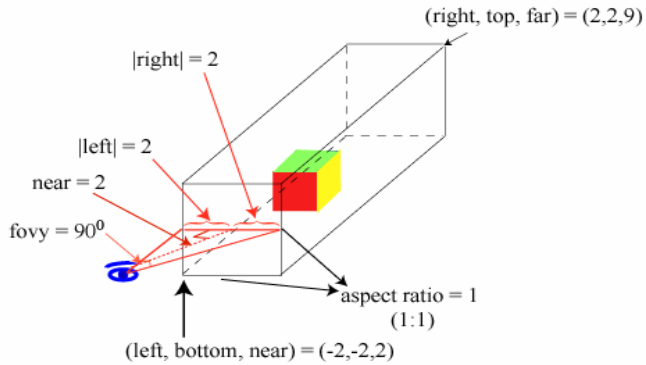
Set Up Viewing Angle (Fovy), Frustum and Camera

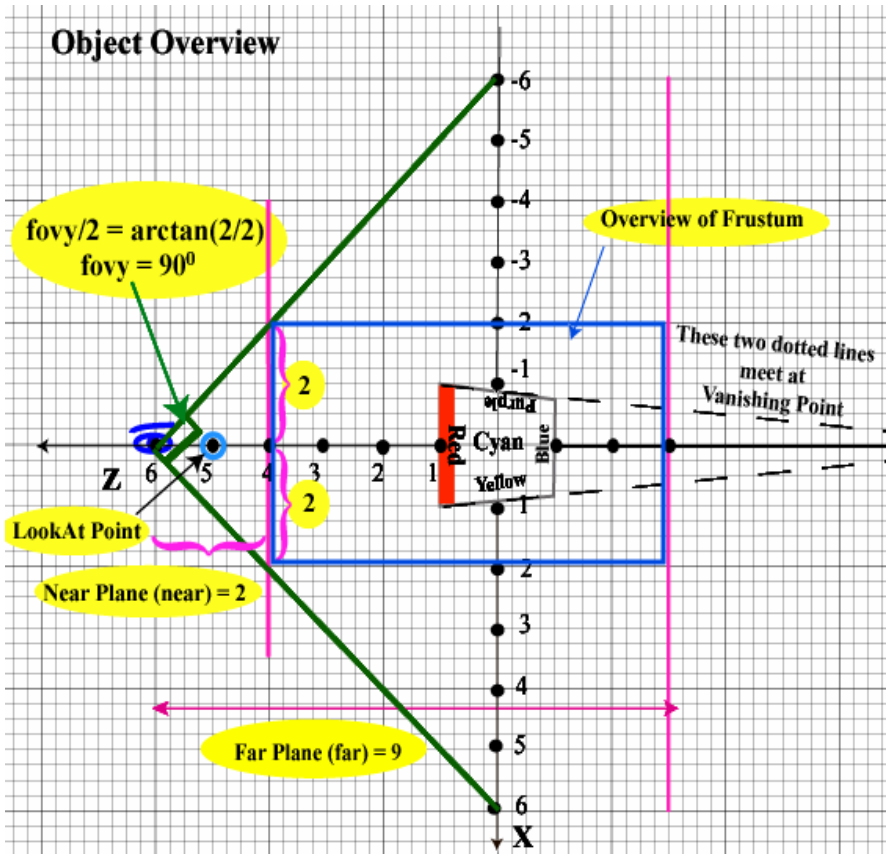
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,2,9);

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

Set Up 90^0 Viewing Angle

$$\text{fovy} = \arctan\left(\frac{|\text{left}|}{\text{near}}\right) + \arctan\left(\frac{|\text{right}|}{\text{near}}\right)$$





Note:

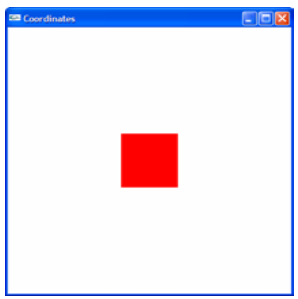
- **Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.**
- **The viewing angle indirectly defined by the glFrustum is equivalent to the Fovy angle defined by gluPerspective.**

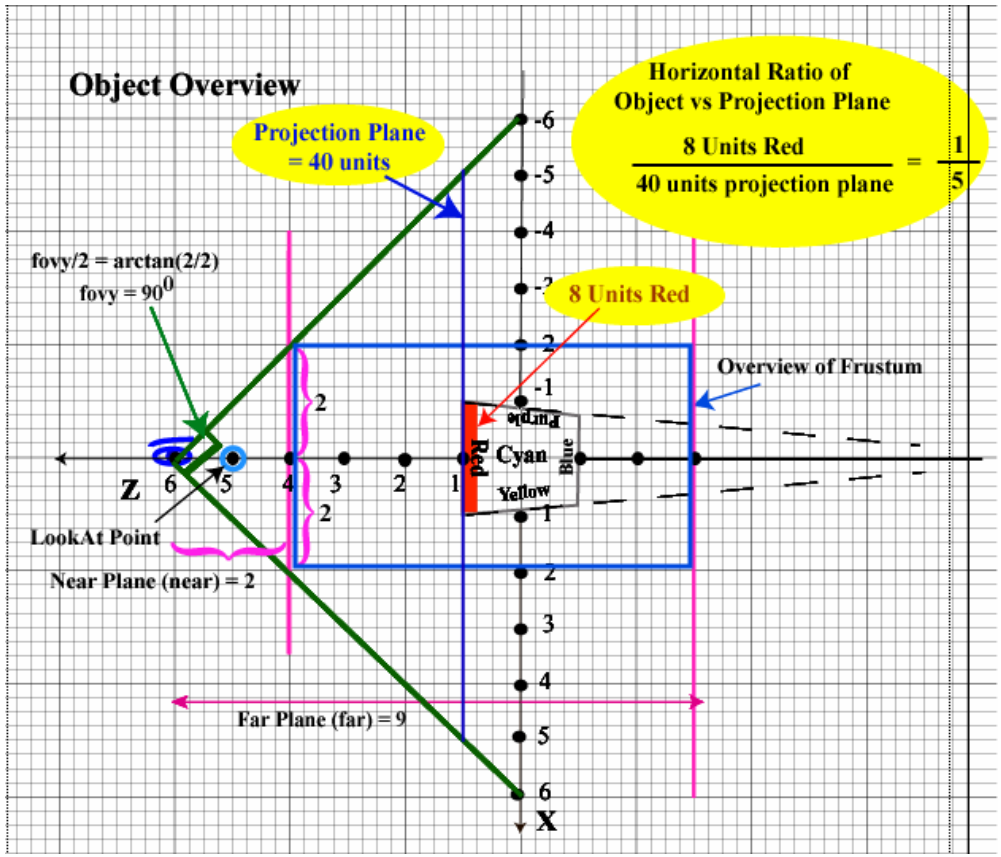
Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 2, far = 9
Calculate Object vs Projection Plane Ratio

Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,2,9);

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

Perspective View, Front View





Example 2: left = -2, right = 2, bottom = -2, top = 2, near = 5, far = 9

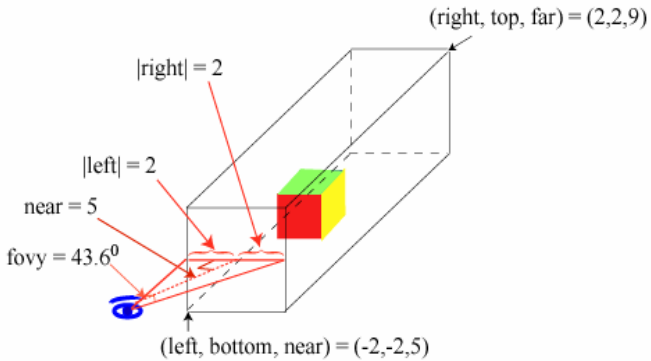
Set Up Viewing Angle (Fovy), Frustum and Camera

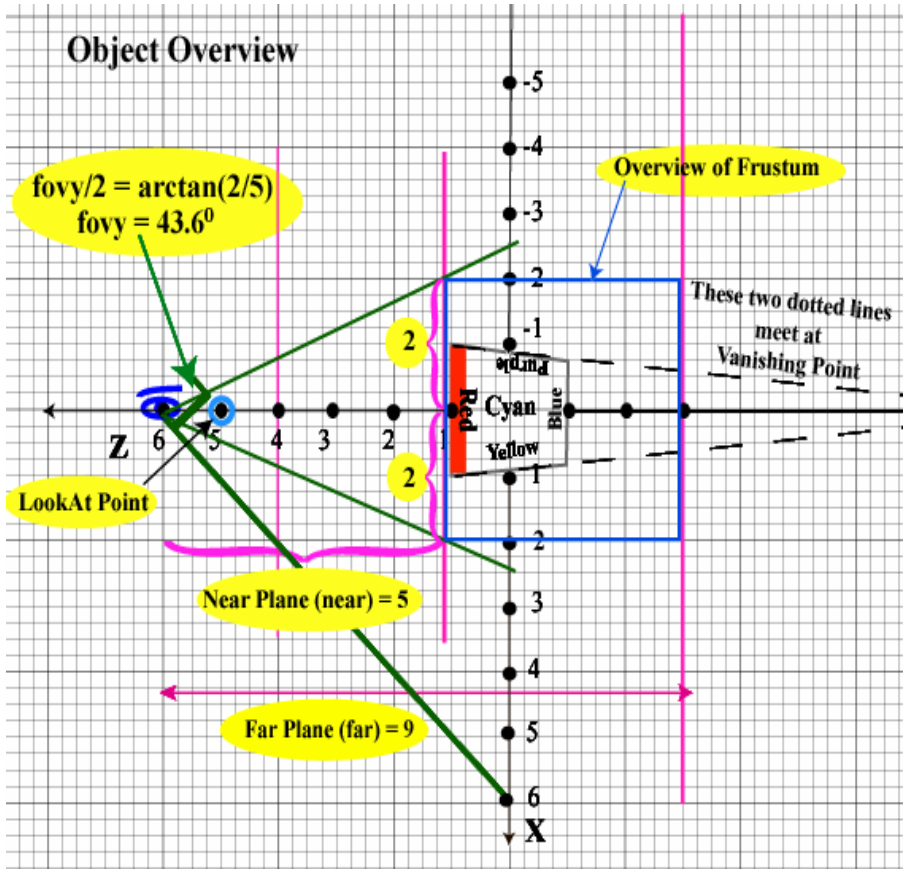
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,5,9);

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

Set Up 43.6^0 Viewing Angle

$$\text{fovy} = \arctan\left(\frac{|\text{left}|}{\text{near}}\right) + \arctan\left(\frac{|\text{right}|}{\text{near}}\right)$$





Note:

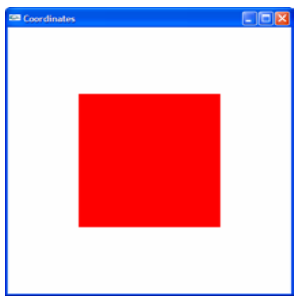
- **Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.**
- **The viewing angle indirectly defined by the glFrustum is equivalent to the Fovy angle defined by gluPerspective.**

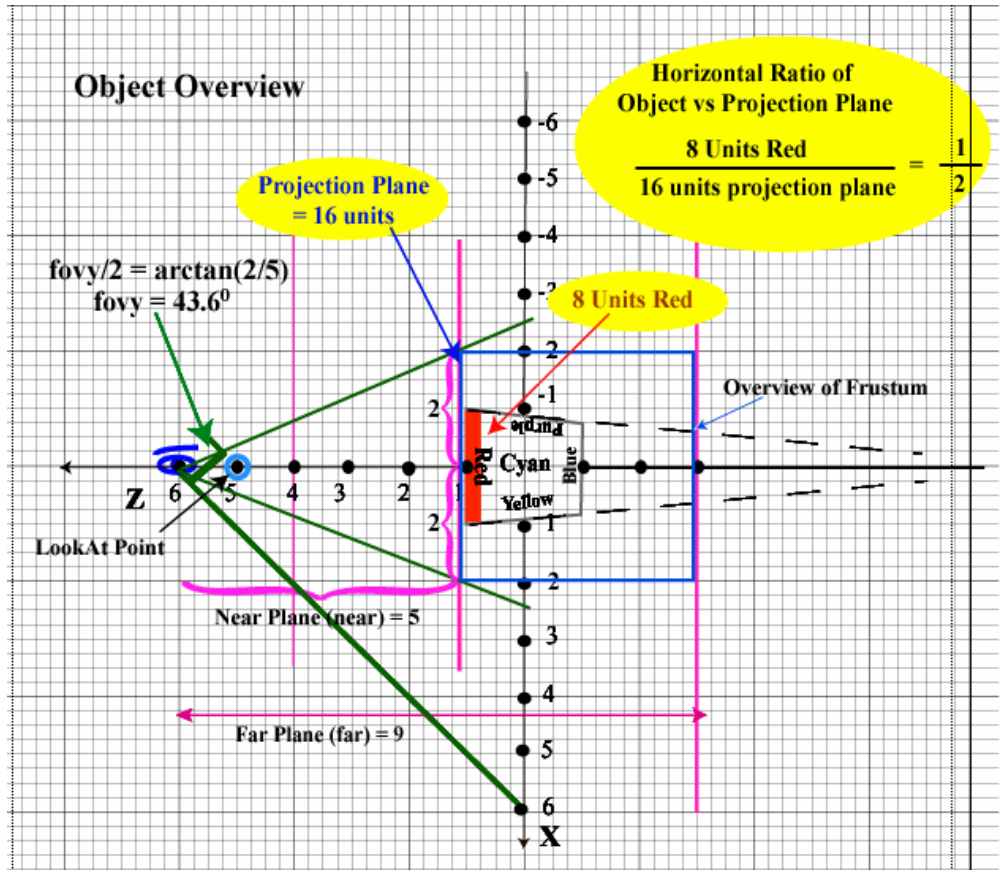
Example 2: left = -2, right = 2, bottom = -2, top = 2, near = 5, far = 9
Calculate Object vs Projection Plane Ratio

Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,5,9);

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

Perspective View, Front View





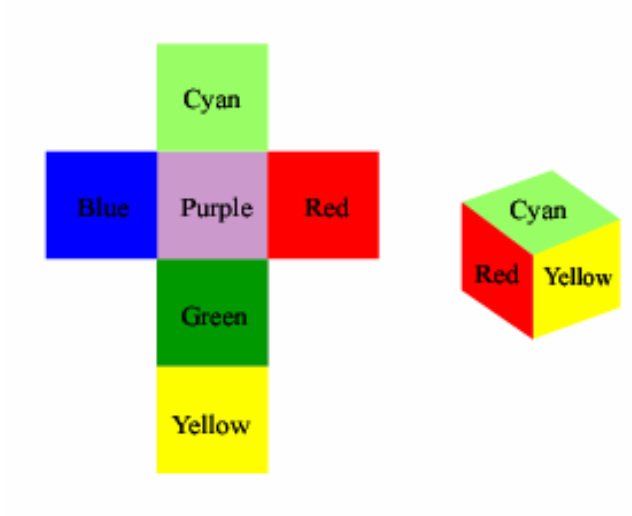
Case Study 2
Fovy Angle of gluPerspective

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)

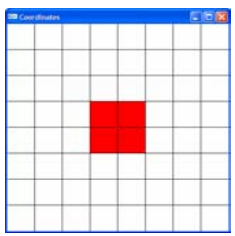


Goal:

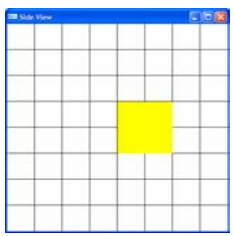
We will observe how varying fovy angle of gluPerspective will change the camera's view of the cube.

Orthogonal View

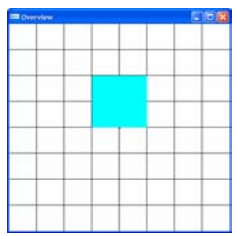
Front View



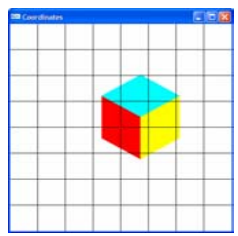
Side View



Top View



Diagonal View



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

Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 2, far = 9

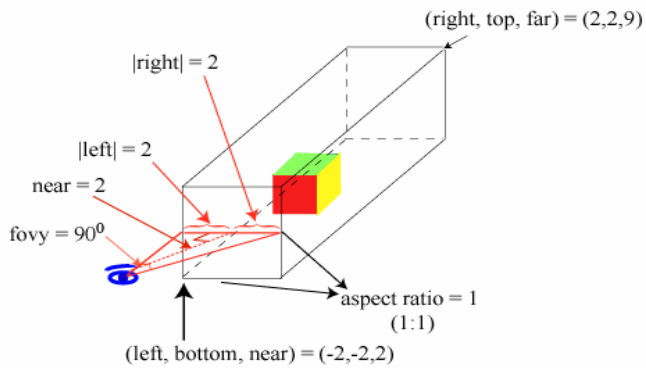
Set Up Viewing Angle (Fovy), Frustum and Camera

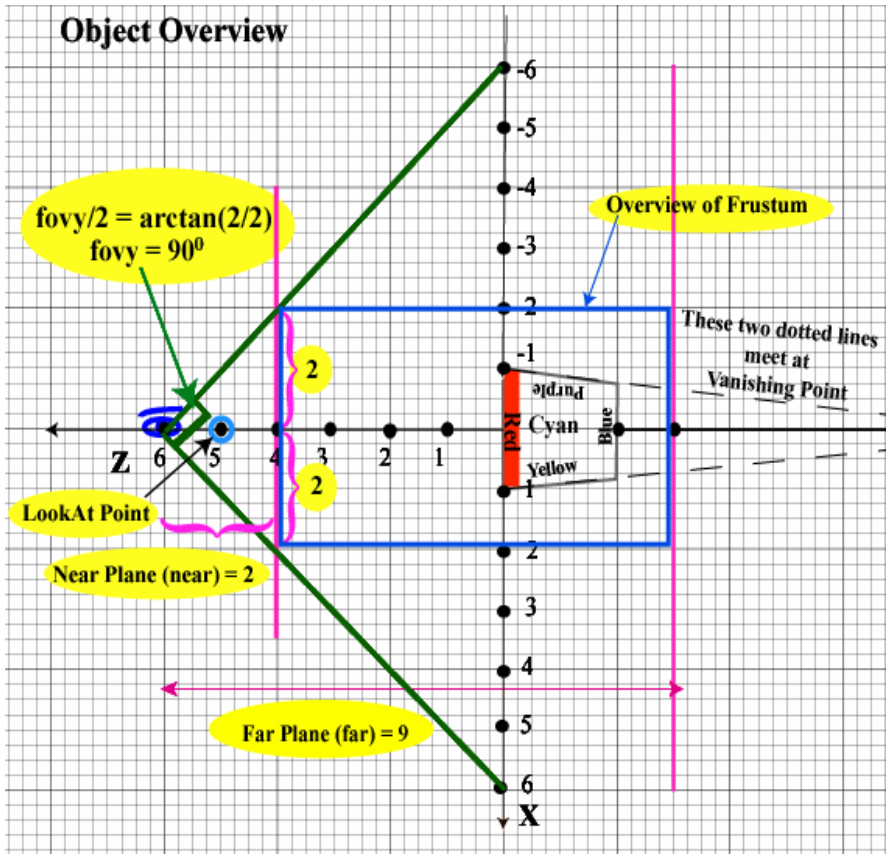
Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,2,9);

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

Set Up 90^0 Viewing Angle

$$\text{fovy} = \arctan\left(\frac{|\text{left}|}{\text{near}}\right) + \arctan\left(\frac{|\text{right}|}{\text{near}}\right)$$





Note:

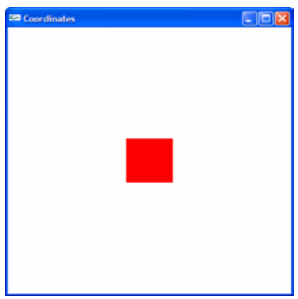
- **Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.**
- **The viewing angle indirectly defined by the glFrustum is equivalent to the Fovy angle defined by gluPerspective.**

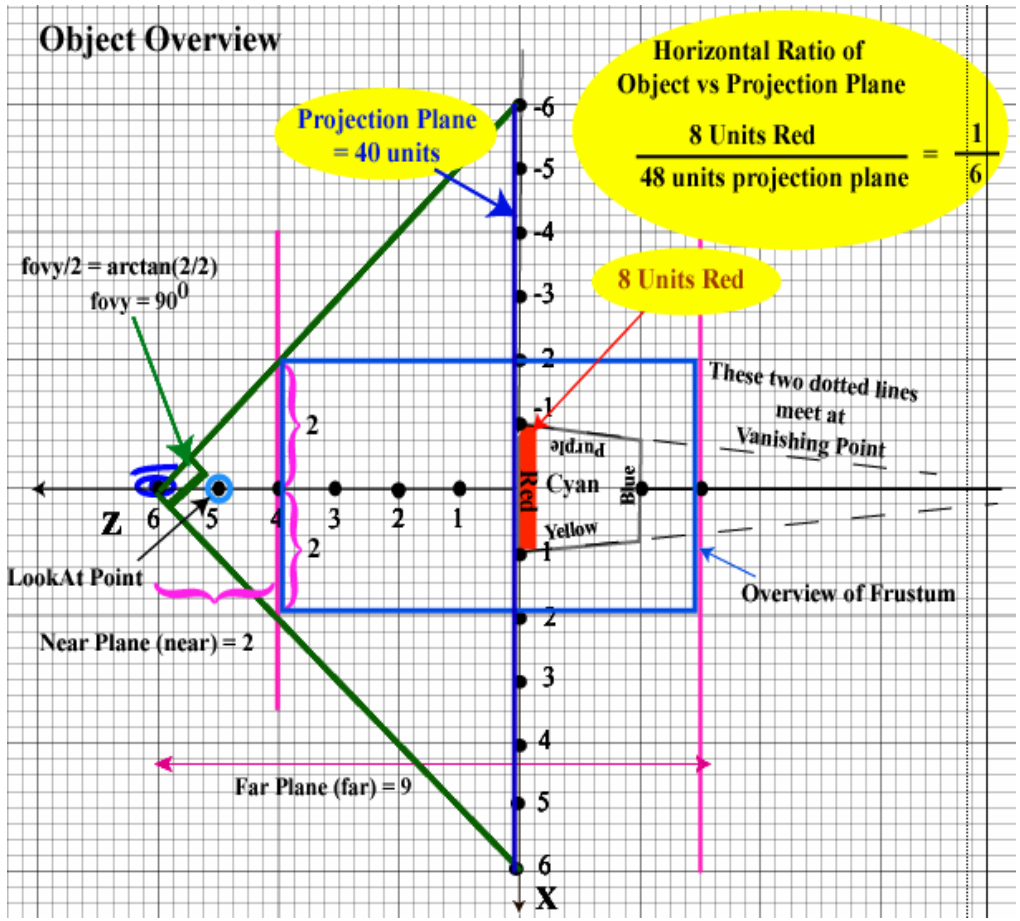
Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 2, far = 9
Calculate Object vs Projection Plane Ratio

Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,2,9);

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

Perspective View, Front View





Example 2: left = -2, right = 2, bottom = -2, top = 2, near = 6, far = 9

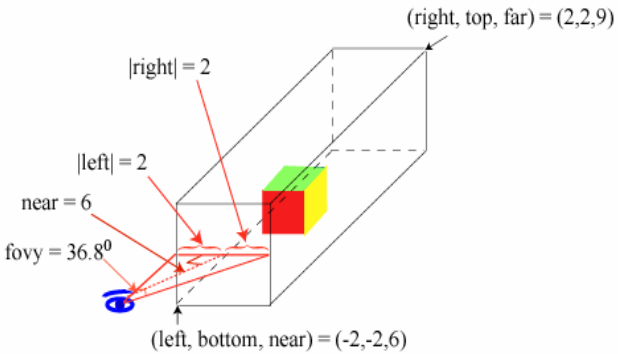
Set Up Viewing Angle (Fovy), Frustum and Camera

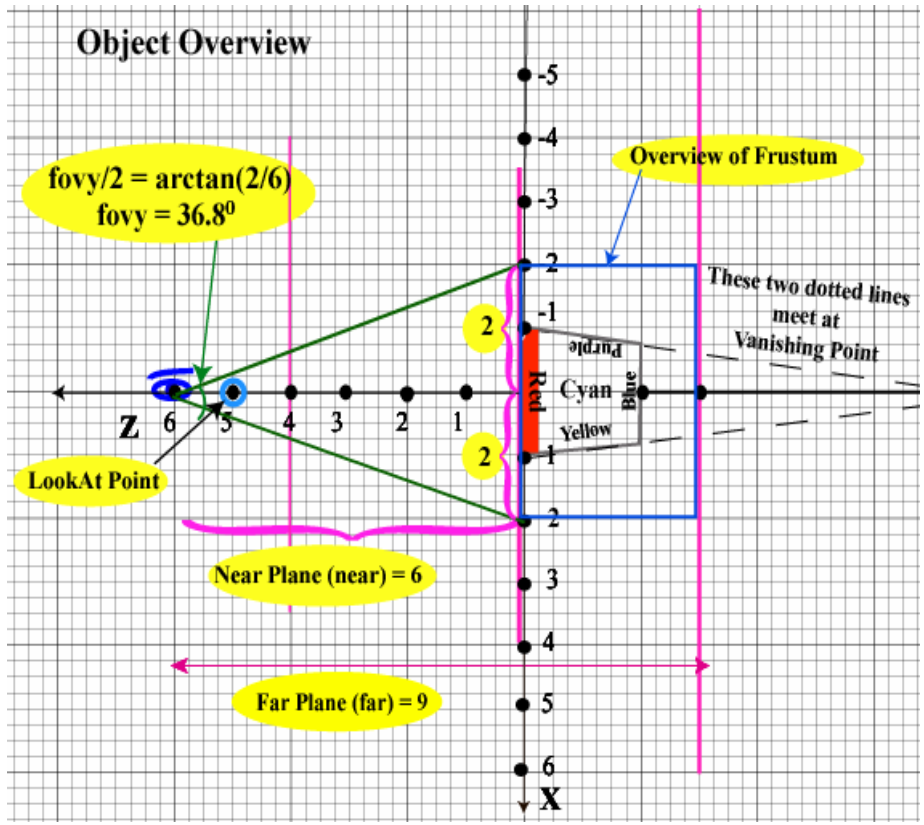
Code :

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,6,9);

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

Set Up 36.8^0 Viewing Angle

$$fovy = \arctan\left(\frac{|left|}{near}\right) + \arctan\left(\frac{|right|}{near}\right)$$





Note:

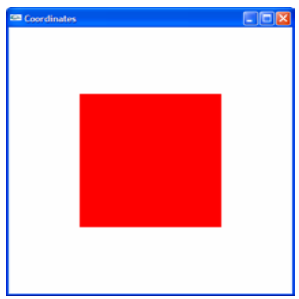
- **Projection plane is a plane perpendicular to the line passing through the camera and its reference point. The center of the projection plane lies on the line passing through camera and its reference point. Projection plane is what the camera sees.**
- **The viewing angle indirectly defined by the glFrustum is equivalent to the Fovy angle defined by gluPerspective.**

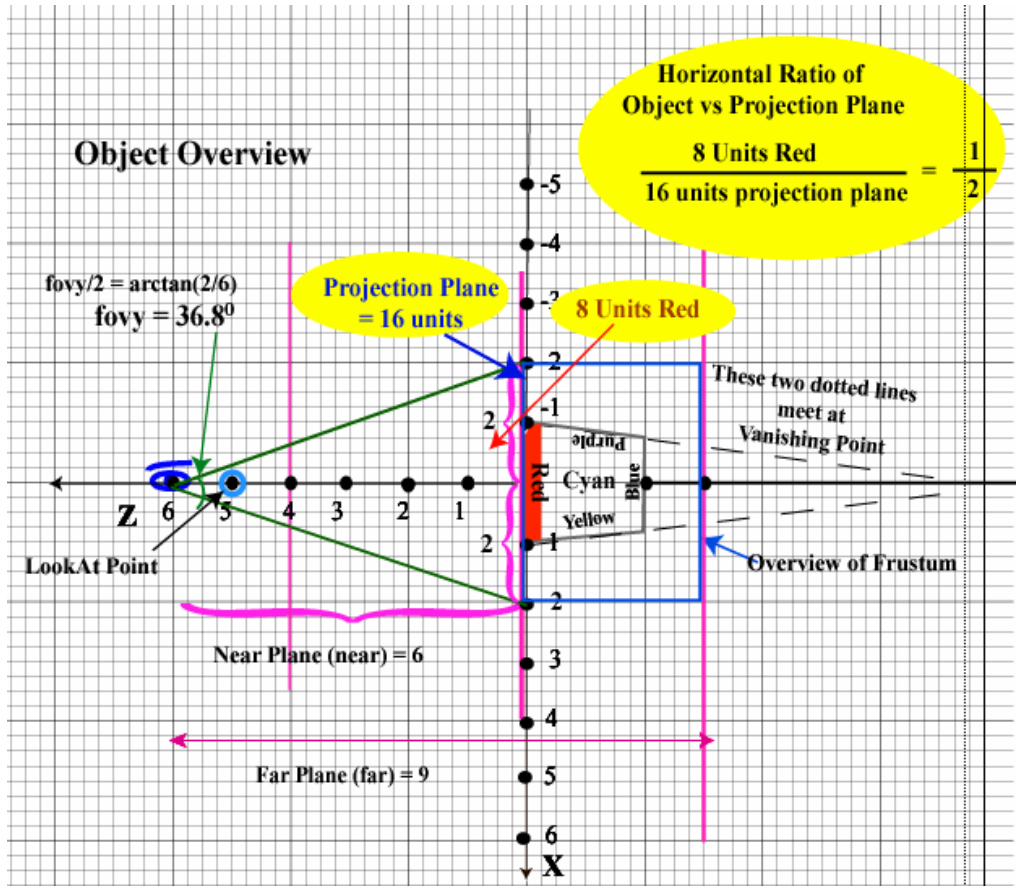
Example 2: left = -2, right = 2, bottom = -2, top = 2, near = 6, far = 9
Calculate Object vs Projection Plane Ratio

Code:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-2,2,-2,2,6,9);

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

Perspective View, Front View





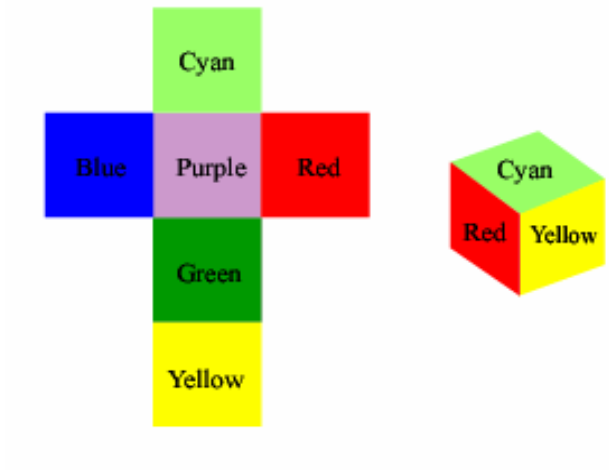
Case Study 3
Aspect Ratio of glFrustum

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)

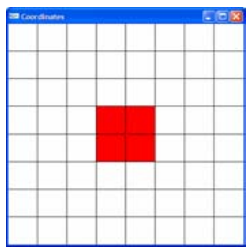


Goal:

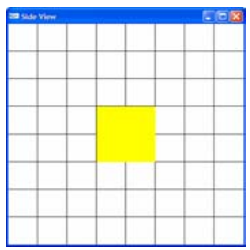
We will observe how varying Aspect Ratio of gluPerspective while fixing the window's Aspect Ratio will change the camera's view of the cube.

Orthogonal View

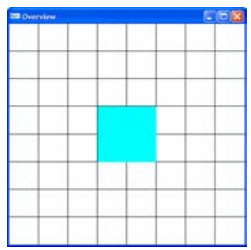
Front View



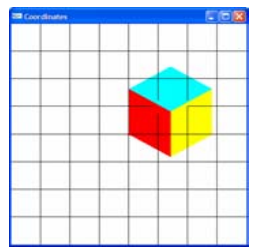
Side View



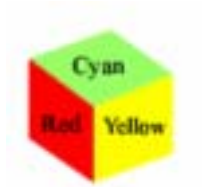
Top View



Diagonal View



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

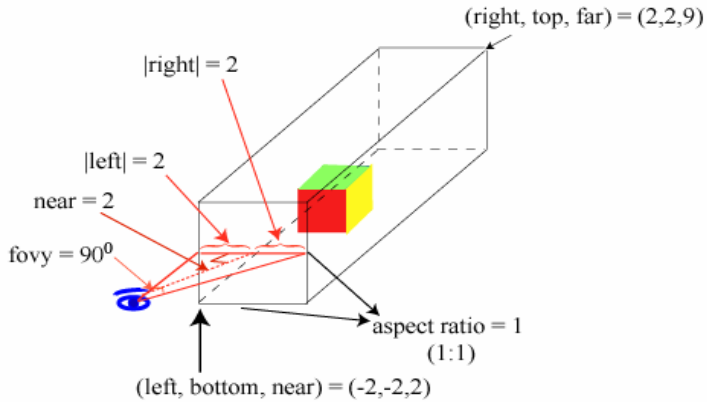


Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 5, far = 9
glFrustum's Aspect Ratio matches Window's Aspect Ratio

Code :

```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
      glFrustum(-2,2,-2,2,5,9); //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);
```

$$fovy = \arctan\left(\frac{|left|}{near}\right) + \arctan\left(\frac{|right|}{near}\right)$$

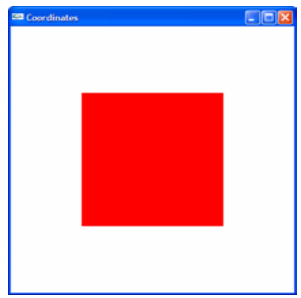


Note:

- **Aspect Ratio: width/height ratio**
  - **In this case: (2-(-2))/(2-(-2)) = 1**

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

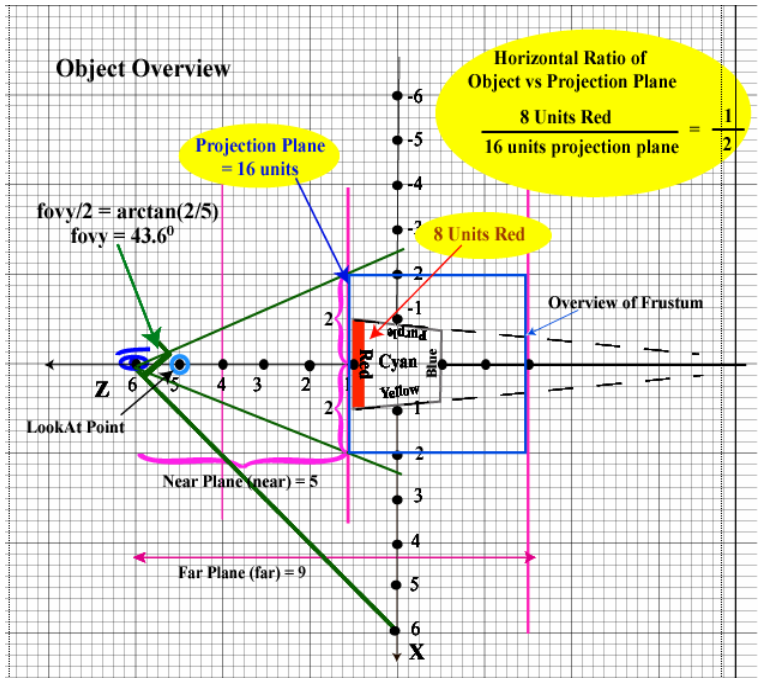
Perspective View, Front View



Example 1: left = -2, right = 2, bottom = -2, top = 2, near = 5, far = 9
glFrustum's Aspect Ratio matches Window's Aspect Ratio

Code :

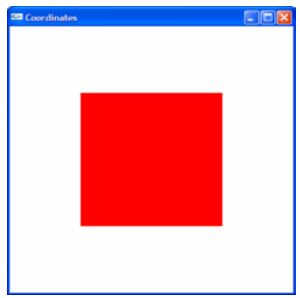
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
      glFrustum(-2,2,-2,2,5,9); //aspect ratio 1

     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:

- **glFrustum's Aspect Ratio: 1**
- **Window's Aspect Ratio: 400px/400px = 1**
- **Object is not distorted.**

Perspective View, Front View



Example 2: left = -4, right = 4, bottom = -2, top = 2, near = 5, far = 9
glFrustum's Aspect Ratio is Twice Window's Aspect Ratio

Code :

```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-4,4,-2,2,5,9); //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:

- **glFrustum's Aspect Ratio: (width/height ratio)**
  (4-(-4))/(2-(-2)) = 2

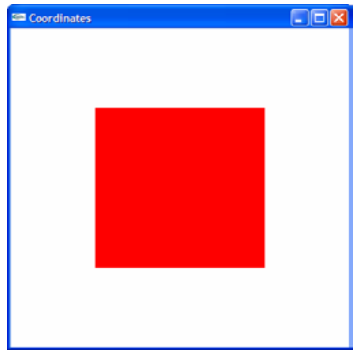
- **Window's Aspect Ratio: 400px/400px = 1**

- **When glFrustum's Aspect Ratio is twice of the Window', object's height remains the same, but its width shrinks by half.**

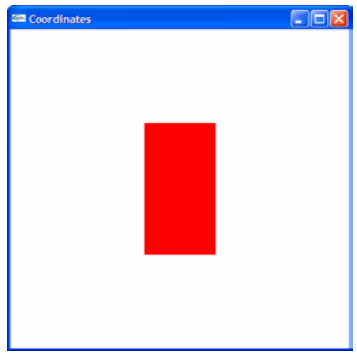
- **The effect is like: glScale(2,1,1)**

Perspective View, Front View

glFrustum's Aspect Ratio = 1



glFrustum's Aspect Ratio = 2



Example 3: left = -2, right = 2, bottom = -4, top = 4, near = 5, far = 9
glFrustum's Aspect Ratio is half of Window's Aspect Ratio

Code :

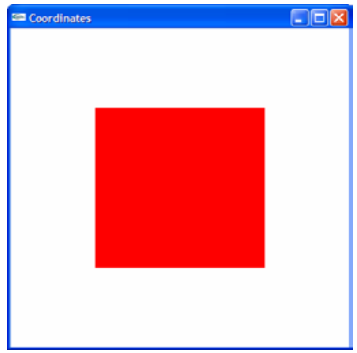
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-2,2,-4,4,5,9); //aspect ratio 0.5

     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);
```

Note:

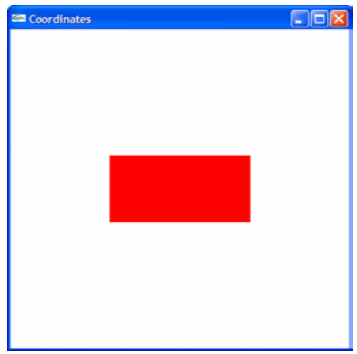
- **Aspect Ratio(width/height ratio)**
  (2-(-2)) / (4-(-4)) = 0.5
- **Window's Aspect Ratio:**
  400px/400px = 1
- **When gluFrustum's Aspect Ratio is half of the Window's, object's width remains the same, but its height shrinks by half.**
- **The effect is like: glScale(1,0.5,1);**

Perspective View, Front View

glFrustum's Aspect Ratio = 1



glFrustum's Aspect Ratio = 0.5



Difference between gluPerspective and glFrustum when 0< Aspect Ratio < 1

Code A:

```
…
glFrustum(-2,2,-4,4,5,9); //aspect ratio 0.5
…
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0(); // a 2x2x2 cube, centered at (0,0,0)
glutInitWindowSize(400,400);
```

Code B:

```
…
glPerspective(43.6, 0.5, 5,9); //aspect ratio 0.5
…
gluLookAt(0,0,6,0,0,5,0,1,0);
colorcube0(); // a 2x2x2 cube, centered at (0,0,0)
glutInitWindowSize(400,400);
```

Note:

- **Code A and B's Similarity:**
  - **Aspect Ratio: width/height ratio = 0.5**
  - **Window's Aspect Ratio: 400px/400px = 1**

- **However the output of two codes are different:**

  - **When gluFrustum's Aspect Ratio is half of the Window's, object's width remains the same, but its height shrinks by half.**

  - **When gluPerspective's Aspect Ratio is half of Window's, object's height remains the same, but the width shrinks by half.**

Perspective View, Front View

Aspect Ratio = 1

glFrustum's Aspect Ratio = 0.5

gluPerspective's Aspect Ratio = 0.5

Example 4: left = -1, right = 3, bottom = -4, top = 4, near = 5, far = 9
glFrustum's Aspect Ratio is half of Window's Aspect Ratio

Code :

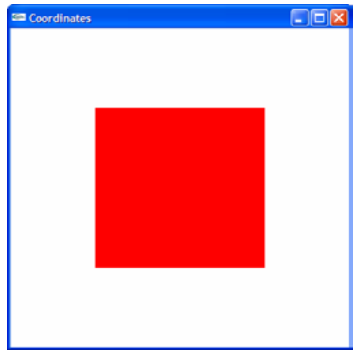
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-1,3,-2,2,5,9); //aspect ratio 0.5

     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);
```

Note:

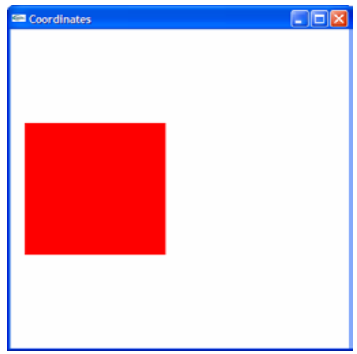
- Aspect Ratio (width/height ratio)
  (3-(-1)) / (2-(-2)) = 1
  - Window's Aspect Ratio: 400px/400px = 1
- When aspect ratio of gluFrustum matches the window's Aspect Ratio, object will not be distorted.
- When |left| < |right|, object shifts to the left.
  - In this case: |left| = 1, |right| = 3, object shifts left.

Perspective View, Front View

glFrustum's Aspect Ratio = 1



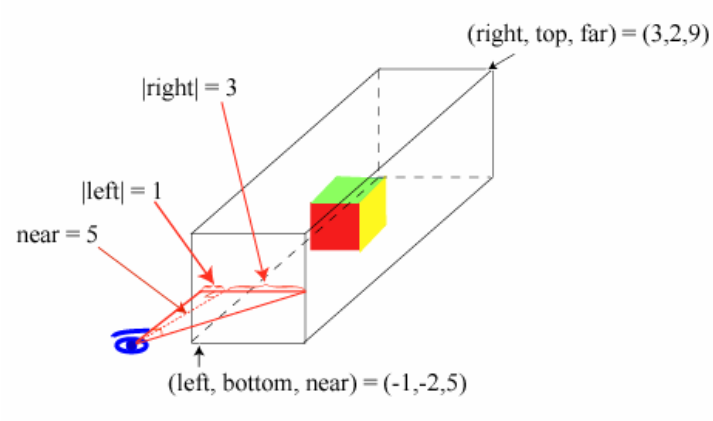
glFrustum's Aspect Ratio = 1
|left| < |right|



Example 4: left = -1, right = 3, bottom = -4, top = 4, near = 5, far = 9
glFrustum's Aspect Ratio is half of Window's Aspect Ratio

Code :

```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-1,3,-2,2,5,9); //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);
```



Perspective View, Front View

Note:

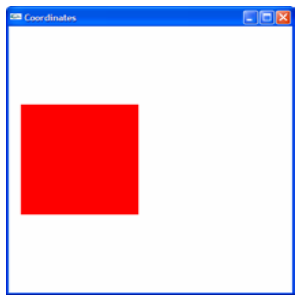
- **Aspect Ratio (width/height ratio)**
  (3-(-1)) / (2-(-2)) = 1

  • Window's Aspect Ratio: 400px/400px = 1

- **When aspect ratio of gluFrustum matches the window's Aspect Ratio, object will not be distorted.**

- **When |left| < |right|, object shifts to the left.**
  •In this case: |left| = 1, |right| = 3, object shifts left.

glFrustum's Aspect Ratio = 1
|left| < |right|



Example 5: left = -3, right = 1, bottom = -4, top = 4, near = 5, far = 9
glFrustum's Aspect Ratio is half of Window's Aspect Ratio

Code :

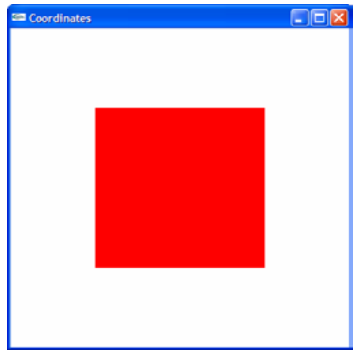
```
// in reshape() function
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glFrustum(-3,1,-2,2,5,9); //aspect ratio 1

     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:

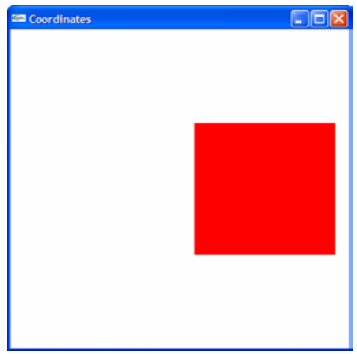
- **Aspect Ratio (width/height ratio):**
  (1-(-3)) / (2-(-2)) = 1
- **Window's Aspect Ratio: 400px/400px = 1**
- **When gluFrustum's Aspect Ratio is half of the Window's, object's width remains the same, but its height shrinks by half.**
- **When |left| > |right|, object shifts to the right.**
  - **In this case: |left| = 3, |right| = 1, object shifts right.**

Perspective View, Front View

glFrustum's Aspect Ratio = 1



glFrustum's Aspect Ratio = 1
|left| > |right|



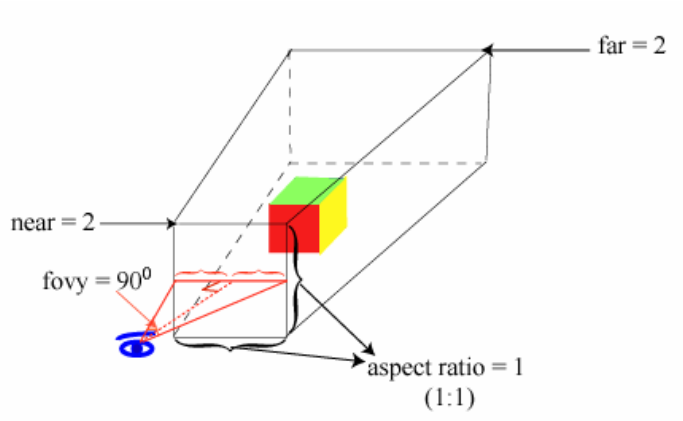
glFrustum vs gluPerspective's Similarities

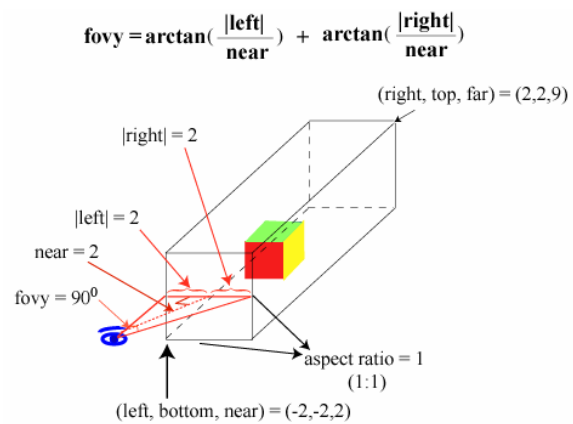
- The gluFrustum can simulate fovy in gluPerspective by adjusting near plane and the distance between left and right of the glFrustum.

The following two commands display the same output.

```
gluPerspective(90,          // fovy
               1,           // aspect ratio
               2,           // near
               9);          // far
```

```
(b) glFrustum(-2,           // left
               2,           // right
              -2,           // bottom
               2,           // top
               2,           // near
               9);          // far
```



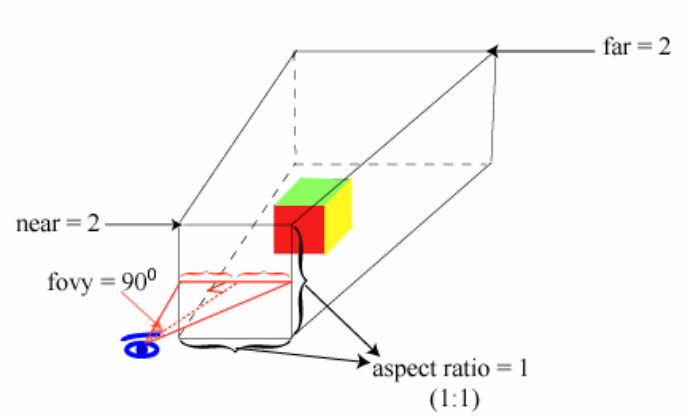


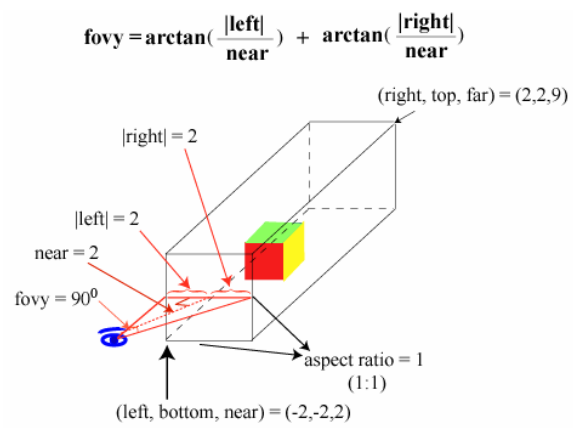
glFrustum vs gluPerspective's Similarities, continued

- The near plane in gluPerspective is only near clipping plane, and has nothing to do with the viewing angle. Yet glFrustum's near plane in also both as near clipping plane, and one important parameter which contributes to the construction of viewing angle.

```
gluPerspective(90,      // fovy
               1,       // aspect ratio
               2,       // near
               9);      // far
```

```
(b) glFrustum(-2,       // left
               2,       // right
              -2,       // bottom
               2,       // top
               2,       // near
               9);      // far
```



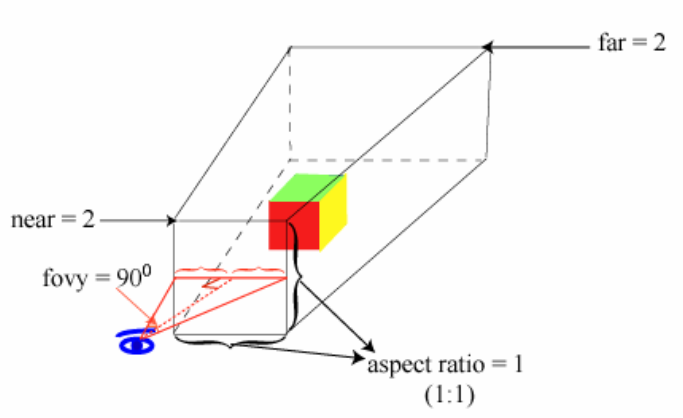


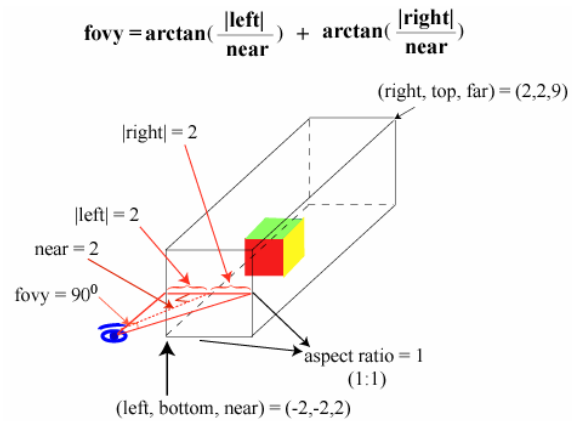
glFrustum vs gluPerspective's Similarities, continued

- Both far plane in gluPerspective and gluFrustum are clipping planes only and have nothing to do with the viewing angle.

```
gluPerspective(90,      // fovy
               1,       // aspect ratio
               2,       // near
               9);      // far
```

```
(b) glFrustum(-2,       // left
               2,       // right
              -2,       // bottom
               2,       // top
               2,       // near
               9);      // far
```



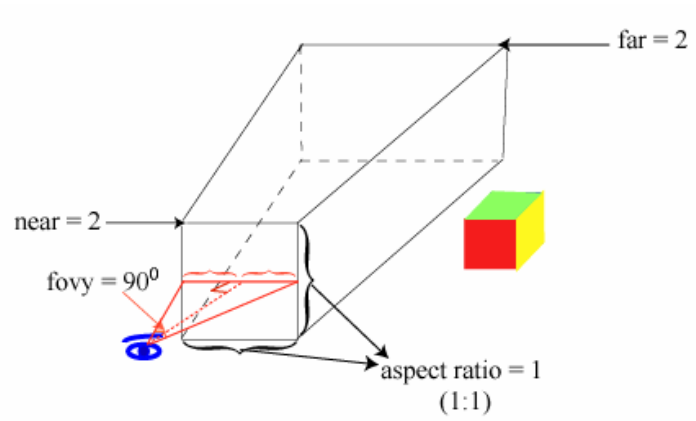


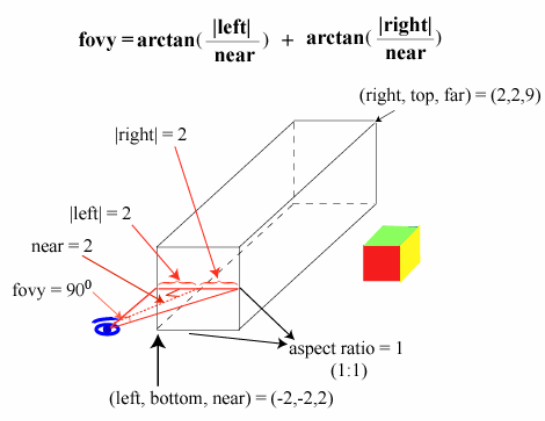
glFrustum vs gluPerspective's Similarities, continued

- For both gluPerspective and glFrustum, when object is outside the area between the near and far planes, object cannot be viewed.

```
gluPerspective(90,      // fovy
               1,       // aspect ratio
               2,       // near
               9);      // far
```

```
(b) glFrustum(-2,       // left
               2,       // right
              -2,       // bottom
               2,       // top
               2,       // near
               9);      // far
```





glFrustum vs gluPerspective's Similarities, continued

- Changing the Aspect Ratio will result the reverse changing the of width and height ratio.

gluPerspective's Aspect Ratio = 1

glFrustum's Aspect Ratio = 1

gluPerspective's Aspect Ratio = 2

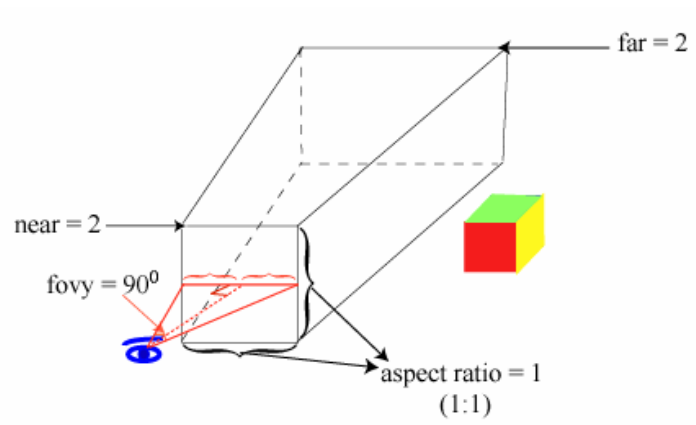
glFrustum's Aspect Ratio = 2

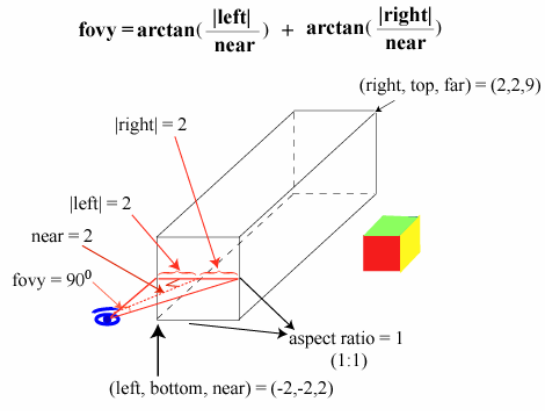
glFrustum vs gluPerspective's Differences

- gluPerspective is more intuitive and easy to use than Frustum.
  - It's easy to think of "changing the fovy" of gluPerspective to change the view angle than think of "changing the left, right, top, bottom" parameters of glFrustum to change the view angle.

```
gluPerspective(90,      // fovy
               1,       // aspect ratio
               2,       // near
               9);      // far
```

```
(b) glFrustum(-2,       // left
               2,       // right
              -2,       // bottom
               2,       // top
               2,       // near
               9);      // far
```



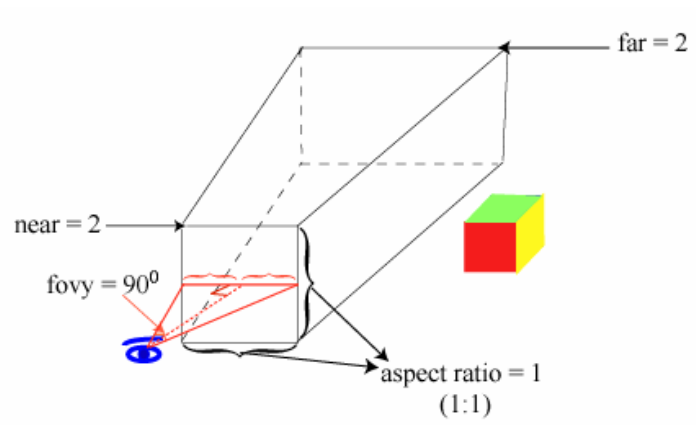


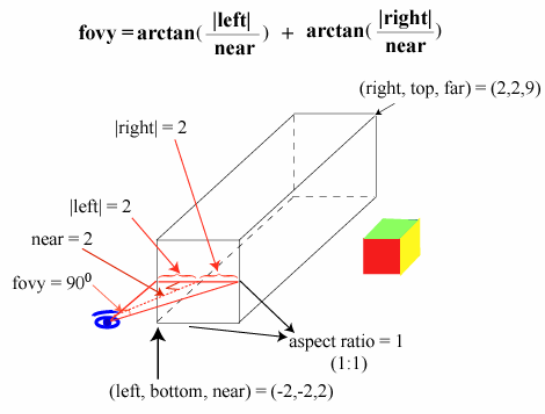
glFrustum vs gluPerspective's Differences, continued

- The sideview profile of gluPerspective's frustum is trapezoidal, whereas the sideview profile of glFrustum's viewing frustum is always rectangular. Therefore, glFrustum's viewing frustum is more restricted than gluPerspect.
  - (This is another reason to use gluPerspective than rather than glFrustum.)

```
gluPerspective(90,          // fovy
               1,           // aspect ratio
               2,           // near
               9);          // far
```

(b)
```
glFrustum(-2,          // left
           2,          // right
          -2,          // bottom
           2,          // top
           2,          // near
           9);         // far
```





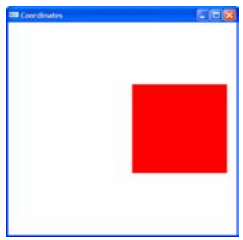
glFrustum vs gluPerspective's Differences, continued

- glFrustum can have input where left <> right, and therefore, shift the object's position on the projection plane; whereas, gluPerspective will not shift the object's position.
  - Example: `glFrustum(-1,3,-2,2,5,9); //aspect ratio 1`

Perspective View, Front View

glFrustum's Aspect Ratio = 1

glFrustum's Aspect Ratio = 1
|left| > |right|

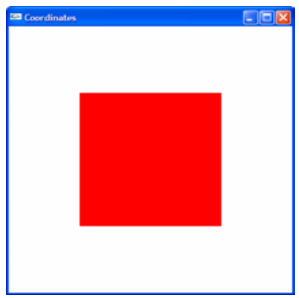


glFrustum vs gluPerspective's Differences, continued

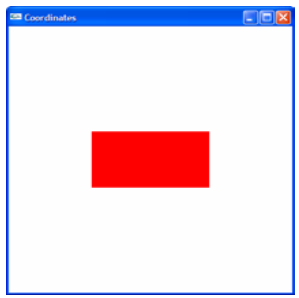
- When the Aspect Ratio is less than one, gluPerspective and glFrustum work slightly differently. gluPerspective fix the height, lengthens the width, whereas glFrustum fixes the width, shrinks the height.

Perspective View, Front View

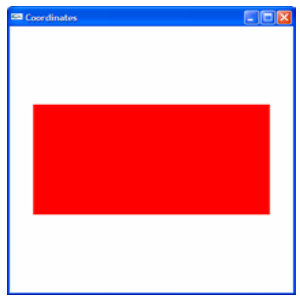
Aspect Ratio = 1



glFrustum's Aspect Ratio = 0.5



gluPerspective's Aspect Ratio = 0.5



Summary
glFrustum vs gluPerspective

- Similarities
  - Fovy in gluPerspective can be simulated by adjusting near plane and the distance between left and right of the glFrustum.
  - The near plane in gluPerspective is only near clipping plane, and has nothing to do with the viewing angle. Yet glFrustum's near plane in also both as near clipping plane, and one important parameter which contributes to the construction of viewing angle.
  - Both far plane in gluPerspective and gluFrustum are clipping planes only and have nothing to do with the viewing angle.
  - For both gluPerspective and glFrustum, when object is outside the area between the near and far planes, object cannot be viewed.
  - Changing the Aspect Ratio will result the reverse changing the of width and height ratio.

- Differences
  - gluPerspective is more intuitive and easy to use than Frustum.
    - It's easy to think of "changing the fovy" of gluPerspective to change the view angle than think of "changing the left, right, top, bottom" parameters of glFrustum to change the view angle.
  - The sideview profile of gluPerspective's frustum is trapezoidal, whereas the sideview profile of glFrustum's viewing frustum is always rectangular.
    - (This is another reason to use gluPerspective than rather than glFrustum.)
  - glFrustum can have input where left <> right, and therefore, shift the object's position on the projection plane; whereas, gluPerspective will not shift the object's position.
  - When the Aspect Ratio is less than one, gluPerspective and glFrustum work slightly differently. gluPerspective fix the height, lengthens the width, whereas glFrustum fixrs the width, shrinks the height.