Lectures
glScale Case Studies

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

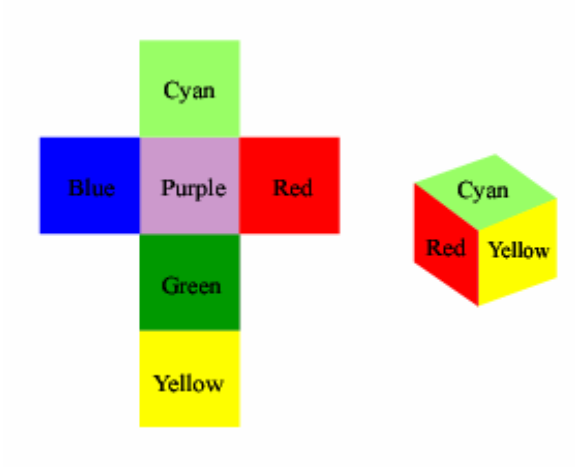
Case Study 1

Case Study Setup:

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

colorcube4 ():

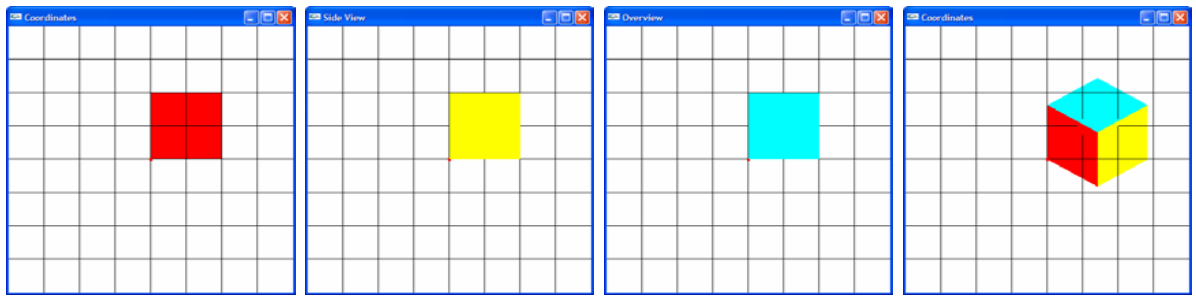
The front left right vertex is at (0,0,0)



Goal:

We will observe how to scale an object using glScale{f,d}.

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



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

First, Set up the Projection View

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

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

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

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

- Under the model view, we can decide how camera views the objects, which angle, how far away. Also, we can decide the size and shape of the objects.
- **What will be scaled, Camera's view volume or Objects' size?**

  This depends on the order of specification of gluLookAt and glScale{f,d}.

  – If we specify gluLookAt before glScale{f,d}, the objects drawn after glScale{f,d} will be scaled.

  – If we specify gluLookAt after glScale{f,d}, we will be scaling the Camera view volume instead of the objects.
- If you don't specify gluLookAt, the camera will take its default position and aim, which is camera located at (0,0,0), looking at (0,0,-1). Then the Scaling applies only to the objects.

How glScale{f,d} works?

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

- **glScale{f,d}** produces a general scaling along x, y, z axis.
  - *x*, *y*, *z*
    Specify the scaling factor along x, y, z axes.

- Under glScale{d,f}(x,y,z), we can also scale camera's view volume.
  - For example, if gluLookAt is defined after glScale, camera's view volume will be scaled and its scaling factor will be (1/x, 1/y, 1/z) in x, y, and z direction.

    If gluLookAt is defined before glScale, object will be scaled, and its scaling factor will be (x,y,z) in x, y, and z direction.

- Just like rotation, scaling also has a fixed point. This fixed point is located at the origin and is unchanged by the scaling.
  - If you need the fixed point to be other than origin, you need to use translation.

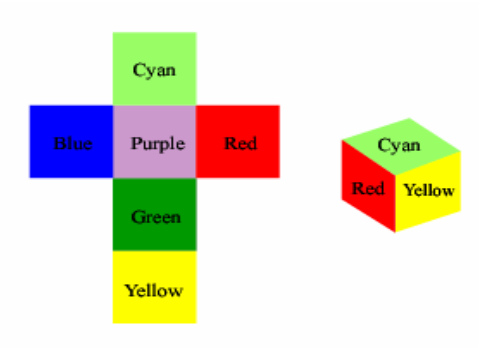
Question

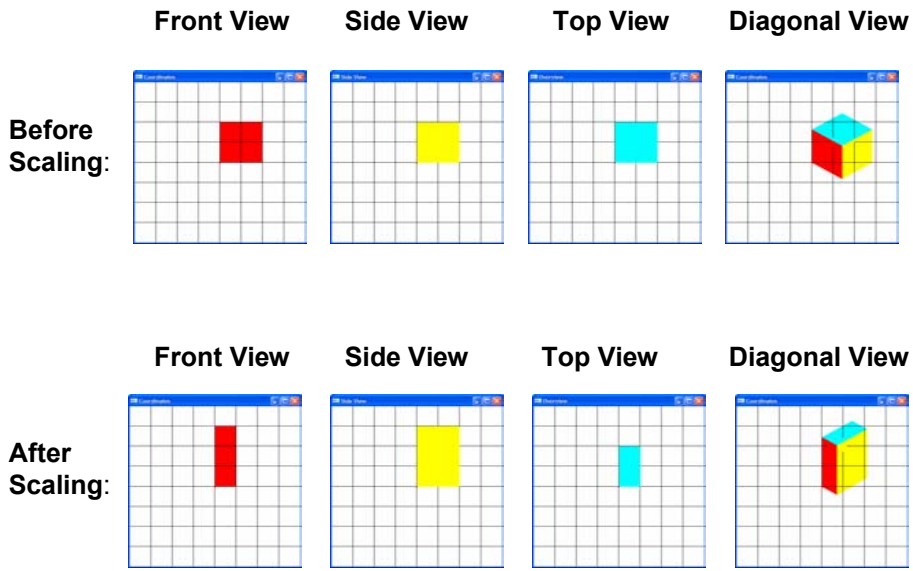
1. **What is being Scaled? Camera's view volume or Object's size?**

   - **If the Camera's view volume is scaled, then what is the scaling factor in x, y, z direction? Where is the fixed point?**
   - **If the Object's size is scaled, then what is the scaling factor in x, y, z direction? Where is the fixed point?**

- **Code A:**

```
glLoadIdentity();
glScalef(0.5,1.5,1);
colorcube4();
```





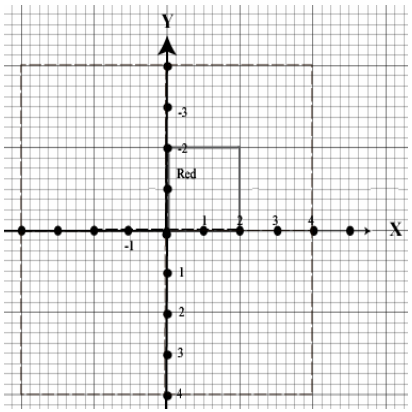
Note: The diagonal view above uses: gluLookAt(1,1,1, 0,0,0, 0,1,0);

Answers

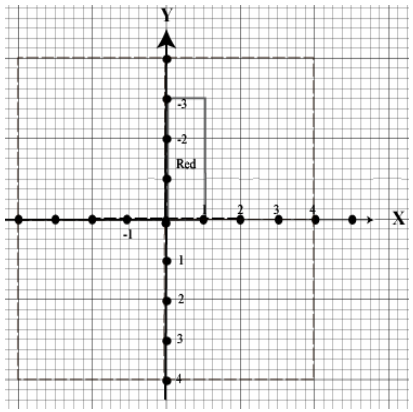
- **Code A:**

```
glLoadIdentity();
glScalef(0.5,1.5,1);
colorcube4();
```

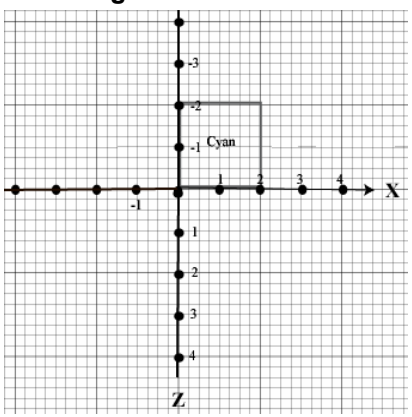
Original Position



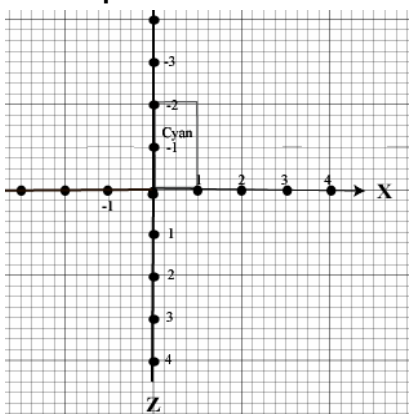
Scale down X by ½
Scale up Y by 1.5



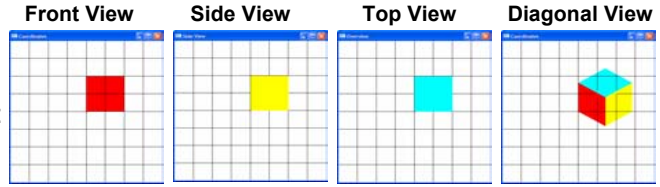
Original Position



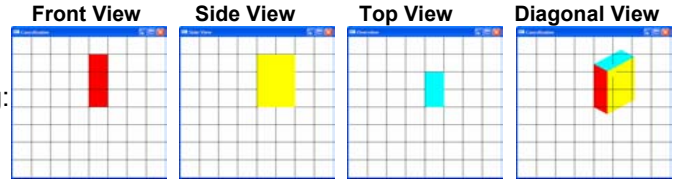
Scale down X by ½
Keep Z the same scale



Before Scaling:



After Scaling:



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

4. **The scaling factor in X,Y,Z direction:**
   - The object is scaled ½ in x direction, 1.5 in y direction, and there is no scaling in z direction since the scaling factor is 1.

Case Study 2

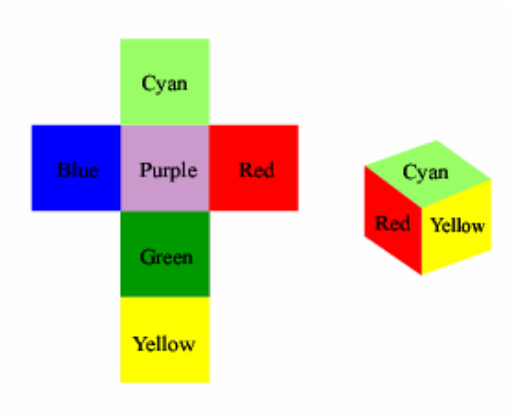
What is being translated? Camera or Object?

1. If the Camera's view volume is scaled, then what is the scaling factor in x, y, z direction? Where is the fixed point?
2. If the Object's size is scaled, then what is the scaling factor in x, y, z direction?

   Where is the fixed point?

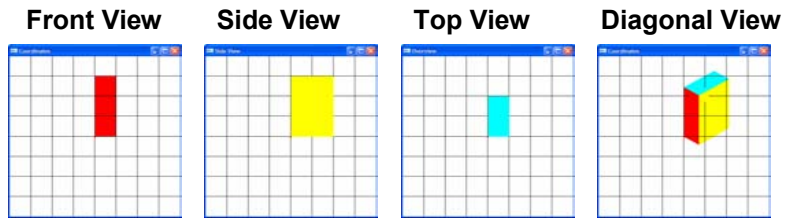
- **Code B:**

```
glLoadIdentity();
glScalef(0.5,1.5,1,0);
glLookAt(0,0,0,
         0,0,-1,
         0,1,0);
colorcube4();
```



Before Scaling:

After Scaling:



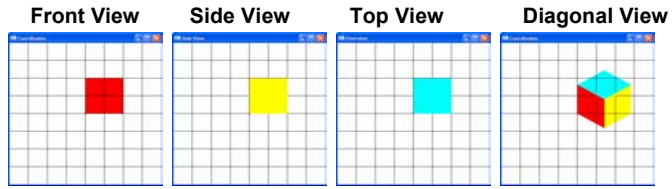
Files Used: glTranslate.c, DrawCubes.c, DrawCubes.h, MyMatrix.c, MyMatrix.h

Answers

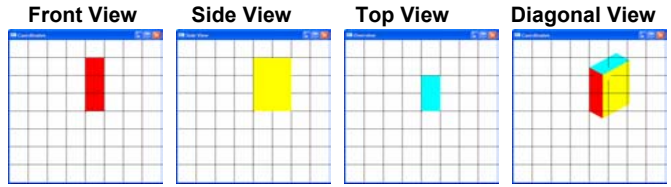
- **Code A:**

```
glLoadIdentity();
glScalef(0.5,1.5,1);
colorcube4();
```

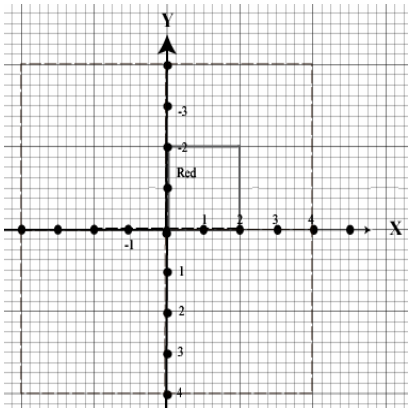
Before Scaling:



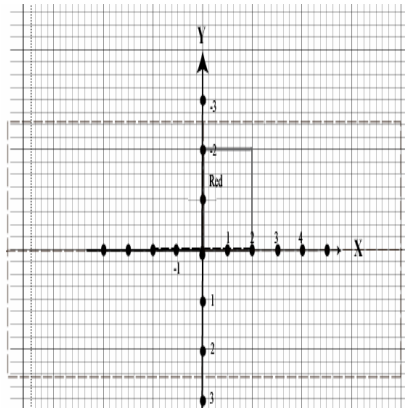
After Scaling:



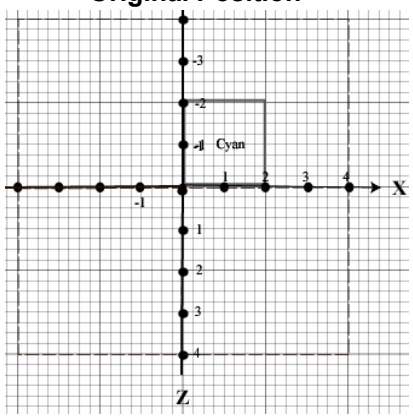
Original Position



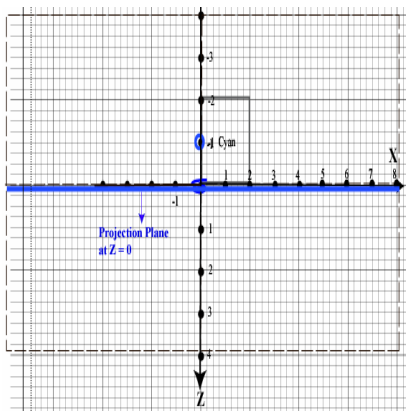
Scale up X by 2, Scale down Y by 1.5
Fit the camera view to 1x1 view port



Original Position



Scale up X by 2, Keep Z the same scale
Fit the camera view into the 1x1 viewport



1. **The Camera is being Scaled:**
   - Since gluLookAt is not set, camera takes its default position, (0,0,0), aiming at (0,0,-1)
   - Since after glScale and before the drawing of objects, there is no gluLookAt specification, therefore prior to glScale's execution, there is no camera specification. Therefore, glScale will scale the objects.
2. **The scaling factor in X,Y,Z direction:**
   - The object is scaled ½ in x direction, 1.5 in y direction, and there is no scaling in z direction since the scaling factor is 1.

Case Study 3
Why the scaling cause the object to move in the Screen?

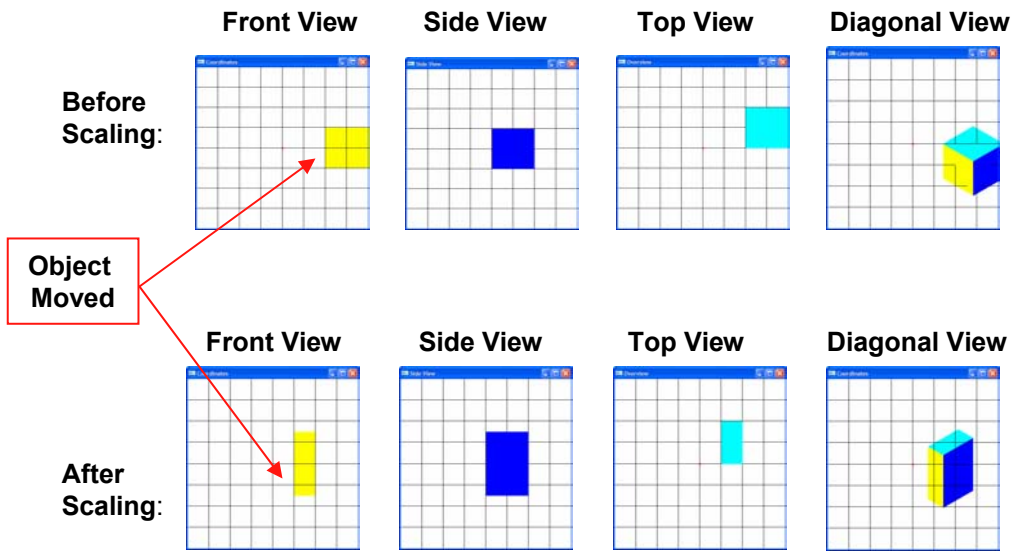
Case Study Setup:

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

colorcube3 ():
centered at (3,0,-1)

Code:

```
glLoadIdentity();
glScalef(0.5,1.5,1);
colorcube3();
```

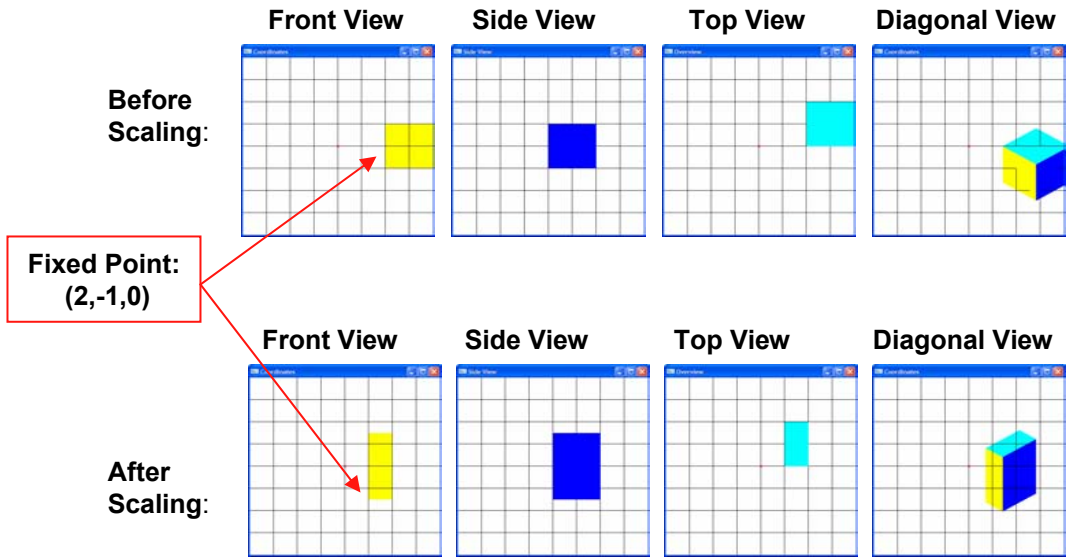


Case Study 3

Answer: The fit point is at the origin (0,0,0). The scaling is with respect to the origin.

Code:

```
glLoadIdentity();
glScalef(0.5,1.5,1);
```



Scaling Effect Observed:

Before Scaling:

Front lower left: (-2,-1,0)

Front lower right: (4,-1,0)

After Scaling:

Front lower left: (-2,-1,0) X (0.5,1.5,1) = (-1,-1.5,0)

Front lower right: (4,-1,0) X (0.5,1.5,1) = (2,-1.5,0)

Case Study 4
How to scale with any fixed point?

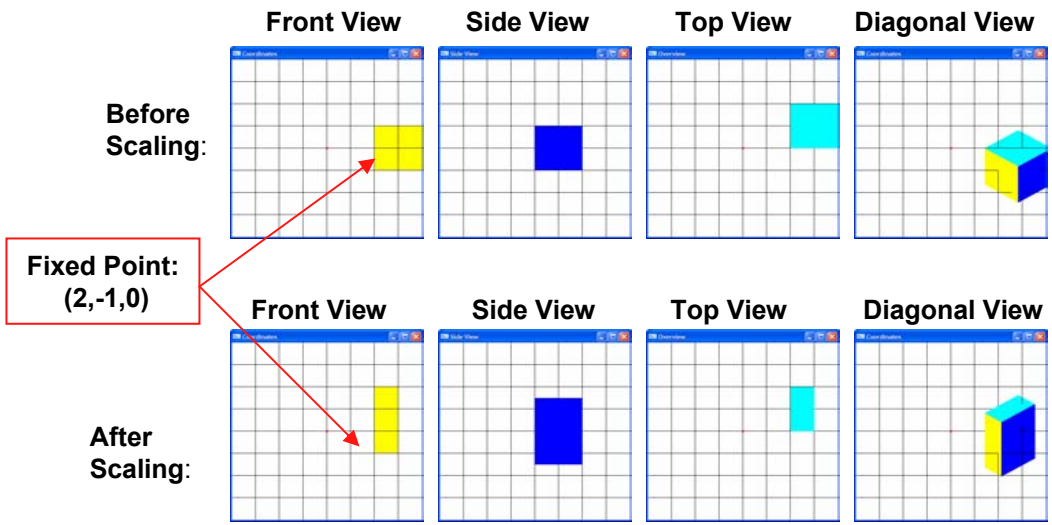
Case Study Setup:

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

colorcube3 (): centered at (3,0,-1)

Goal:

- Scale the cube.
- Scaling factors: (0.5,1.5,1)
- Fix point desired:
  Front lower left (2,-1,0)

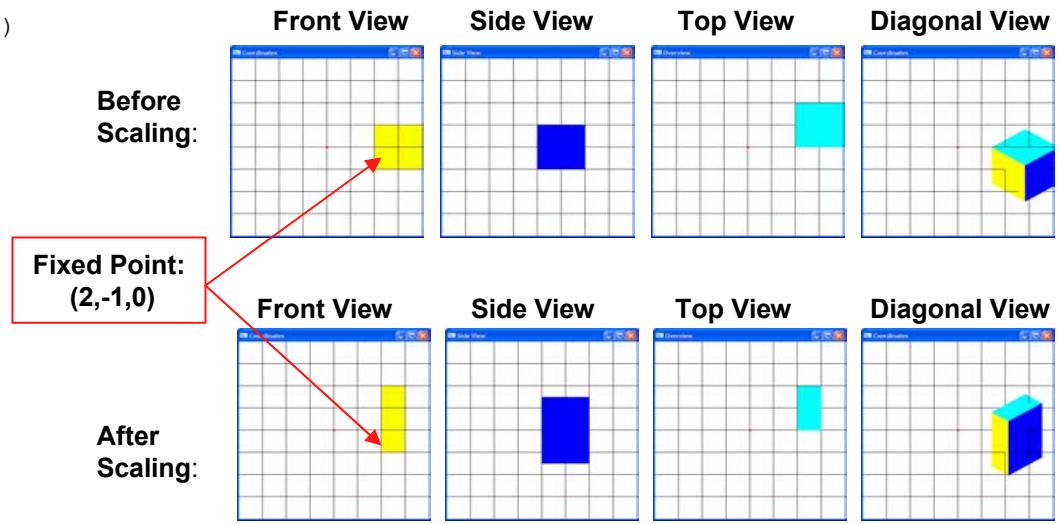


Case Study 4

Answer: Translate the fix point of scaling you want to the origin, scale the object, then translate the fix point back. The Actual implementation is in the reverse order of logical order.

Code:

```
glLoadIdentity();
// fix point desired is (2,-1,0)
glTranslatef(2,-1,0);
glScalef(0.5,1.5,1);
glTranslatef(-2,1,0);
colorcube3();
```



Scaling Effect Observed:

Before Scaling:
- Front lower left: (2,-1,0)
- Front lower right: (4,-1,0)

After Scaling:
- Front lower left: (2,-1,0)
- Front lower right: (3,1,0)

Case Study 4, continued

Generic Code and Matrices for Scaling with respect to any fix point

Summery:

Assume the following Scaling Constrains:
Fix Point (Fx, Fy, Fz)
Scaling Factor: (Sx, Sy, Sz)

Code for scaling:

```
glLoadIdentity();
glTranslatef(Fx,Fy,Fz);          // T1(Fx,Fy,Fz)
glScalef(Sx,Sy,Sz);              // S(Sx,Sy,Sz)
glTranslatef(-Fx,-Fy,-Fz);       // T2(-Fx,-Fy,-Fz)
// draw objects afterwards
```

Matrix Representations:

```
T1(Fx,Fy,Fz) =  1  0  0  Fx
                0  1  0  Fy
                0  0  1  Fz
                0  0  0  1

S(Sx,Sy,Sz) =   Sx 0  0  0
                0  Sy 0  0
                0  0  Sz 0
                0  0  0  1

T2(-Fx,-Fy,-Fz)=  1  0  0  -Fx
                  0  1  0  -Fy
                  0  0  1  -Fz
                  0  0  0   1

T1(Fx,Fy,Fz)* S(Sx,Sy,Sz)*T2(-Fx,-Fy,-Fz) =
                Sx   0    0    (1-Sz)Fx
                0    Sy   0    (1-Sy)Fy
                0    0    Sz   (1-Sz)Fz
                0    0    0      1
```

(Note: Matrix Multiplication is associative. Therefore both premutiplication (T1*S)*T2 and post multiplication T1*(S * T2) produce the same results.

Case Study 4, continued

Generic Code and Matrices for Scaling with respect to any fix point

Example

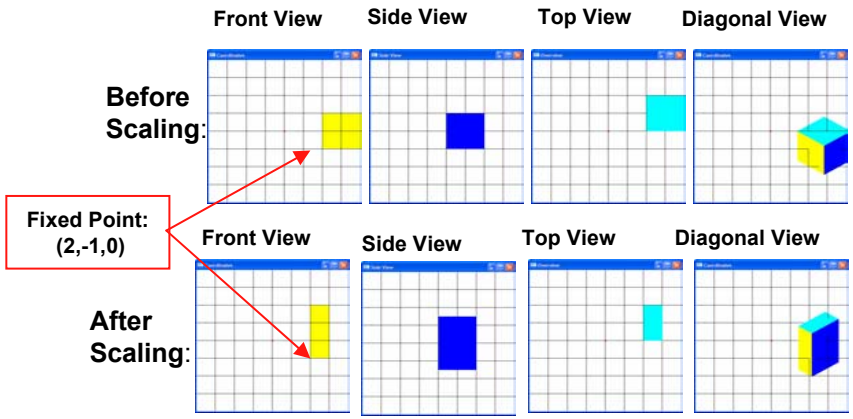
Code for scaling using Translation:

```
glLoadIdentity();
// fix point desired is (2,-1,0)
glTranslatef(2,-1,0);          // T(Fx,Fy,Fz)
glScalef(0.5,1.5,1);           // S(Sx,Sy,Sz)
glTranslatef(-2,1,0);          // T(-Fx,-Fy,-Fz)
// draw objects afterwards
```

Code for Sclaing using Matrix only:

```
float *MyScalingWithFixPoint(float Sx, float Sy, float Sz,
                             float Fx, float Fy, float Fz)
        {
        float *m0;
        float *m1;

        m0 = malloc(16 * sizeof(float));
        m1 = malloc(16 * sizeof(float));

        m0[0] = Sx;   m0[4] = 0;    m0[8] = 0;     m0[12]=(1-Sz)*Fx;
        m0[1] = 0;    m0[5] = Sy;   m0[9] = 0;     m0[13]=(1-Sy)*Fy;
        m0[2] = 0;    m0[6] = 0;    m0[10]= Sz;    m0[14]=(1-Sz)*Fz;
        m0[3] = 0;    m0[7] = 0;    m0[11] =0;     m0[15]=1;

        glMultMatrixf(m0);
        glGetFloatv(GL_MODELVIEW_MATRIX,m1);

        return m1;
        }
```

Both two codes produce the same results



Matrix Used for Scaling:

$$T(Fx,Fy,Fz) * S(Sx,Sy,Sz) * T(-Fx,-Fy,-Fz) =$$

$$\begin{bmatrix} Sx & 0 & 0 & (1-Sz)Fx \\ 0 & Sy & 0 & (1-Sy)Fy \\ 0 & 0 & Sz & (1-Sz)Fz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

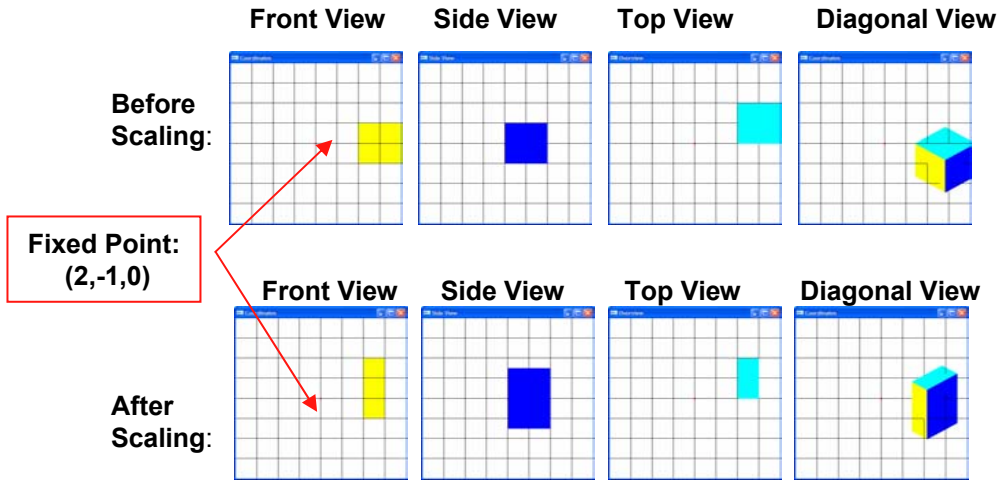
Case Study 4

What happen to points on the objects during scaling with respect to a fix point?

Summery:
Assume the following Scaling Constrains:
Fix Point (Fx, Fy, Fz)
Scaling Factor: (Sx, Sy, Sz)

Before Scaling:
A point in the object (X,Y,Z)

After Scaling:
Fixed point (Fx,Fy,Fz) will not change.
A point in the object (X,Y,Z) that is not a fixed point: will become (X', Y', Z')
(X',Y',Z')
= (X,Y,Z) X (Sx,Sy,Sz) + (Fx,Fy,Fz) X (Sx,Sy,Sz)



Example:
Before Scaling:
Front lower left: (2,-1,0)
Front lower right: (4,-1,0)

After Scaling:
Front lower left: (2,-1,0)
Front lower right: (3,1,0)

Note:
(2,-1,0) is the fixed point, therefore it will not be scaled.

(4,-1,0) X (0.5,1.5,1) + (2,-1,0) X (0.5,1.5,1)
= (3,-1,0)

Code:
```
glLoadIdentity();
// fix point desired is (2,-1,0)
glTranslatef(2,-1,0);
glScalef(0.5,1.5,1);
glTranslatef(-2,1,0);
colorcube3(); // cube of size 2,
              // centered at (-3,0,-1)
```

Case Study 5
What's the accumulative effect of several glScale?

General Formula

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

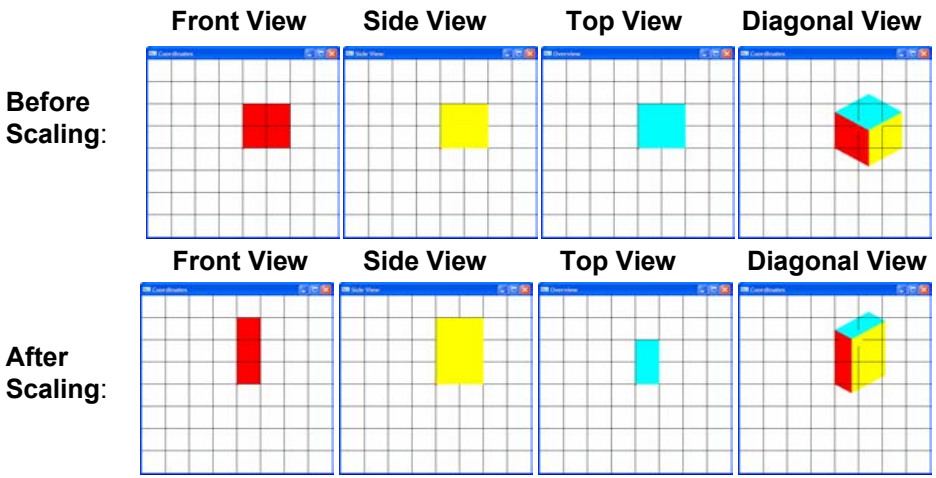
```
glScale{f,d} (x2*x1,y2*y1,z2*z1);
```

Example

```
Code C:
    glLoadIdentity();
    glScalef(0.25,3,2);
    glScalef(2,0.5,0.5);
    colorcube0();
```

```
Code D:
    glLoadIdentity();
    glScale(0.5,1.5,1);
    colorcube0();
```

This above two pieces of code produces the same result:



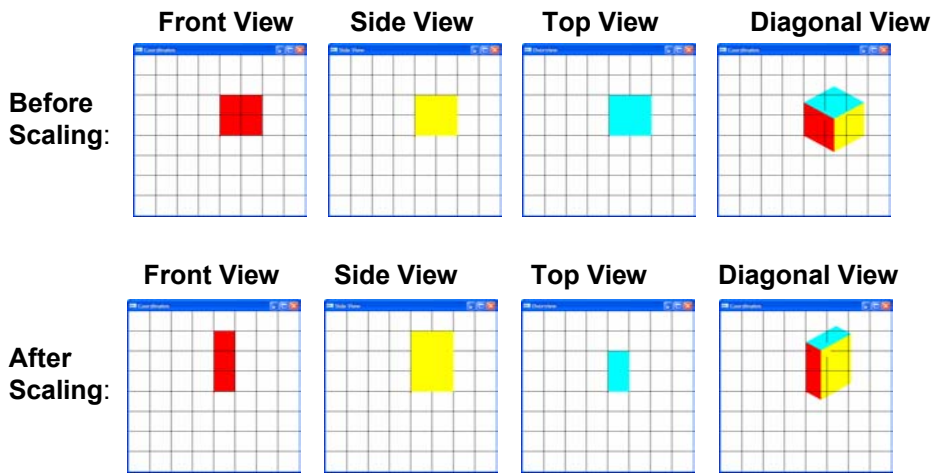
The Actual Matrix Glut used in its Implementation

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

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

glScalef(0.5,1.5,0.25);
glGetFloatv(GL_MODELVIEW_MATRIX,CT);
PrintMToFile(CT,
    "After glScalef(0.5,1.5,0.25);");

colorcube4();
```

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

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

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

```
The Output of printed Current Matrices

Current Matrix, After glScalef(0.5,1.5,1);
0.25 0.00    0.00        0.00
0.00 3.00    0.00        0.00
0.00 0.00    2.00        0.00
0.00 0.00    0.00        1.00

Current Matrix, After glScalef(2,0.5,0.5);
0.50 0.00    0.00        0.00
0.00 1.50    0.00        0.00
0.00 0.00    1.00        0.00
0.00 0.00    0.00        1.00
```



Case Study 5, continued
Generic Matrix for translation

Recall glLoadIdentity() is defined as follows:

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

Scaling Matrix for
glScale{f,d}(x,y,z)is as follows:

```
Sx,0, 0, 0
0, Sy,0, 0
0, 0, Sz,0
0, 0, 0, 1
```

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

```
glLoadIdentity();
glScalef(0.25,3,2);
glScalef(2,0.5,0.5);
colorcube4();                    // Draw a cube centered at (0,0,0)
```

Case Study 5, continued

Using Matrix Multiplication:

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

Using Glut Functions:

```
glLoadIdentity();
glScalef(0.25,3,2);
glScalef(2,0.5,0.5);

colorcube4();
```

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

```
m1:
0.25, 0, 0, 2
0,    3, 0, -1
0,    0, 2, 1
0,    0, 0, 1
```

```
m2:
2, 0,   0,   -4
0, 0.5, 0,   3
0, 0,   0.5, -2
0, 0,   0,   1
```

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

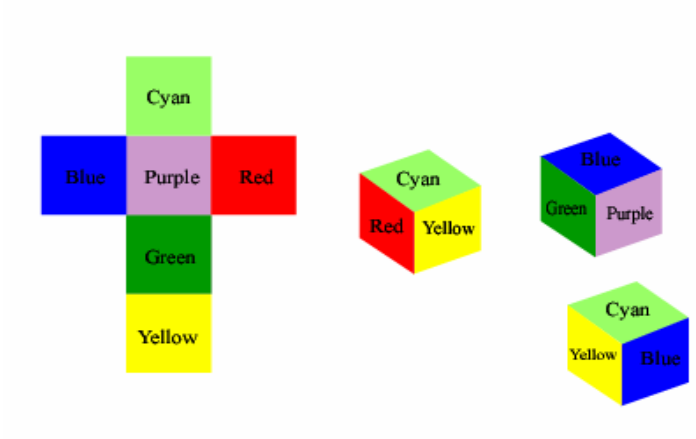
Note: The m0,m1,m3 were displayed in row major. However, in the actual implementation, we need to construct our matrix array in column major.

Case Study 6
Scaling of Three Cubes
- glScale.c, DrawCubes.c, DrawCubes.h

Case Study Setup:

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

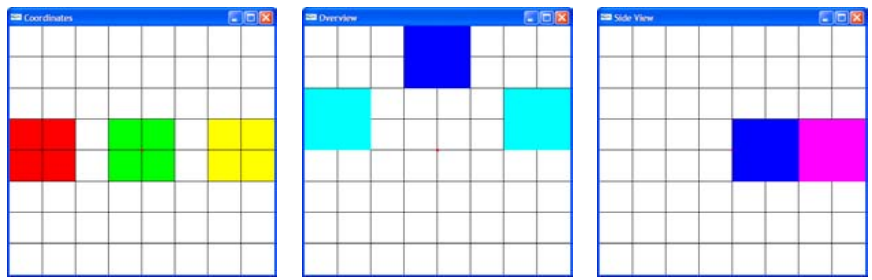
Cube (1): centered at (-3,0,-1)
Cube (2): centered at (3,0,-1)
Cube (3): centered at (0,0,-3)



Goal:

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

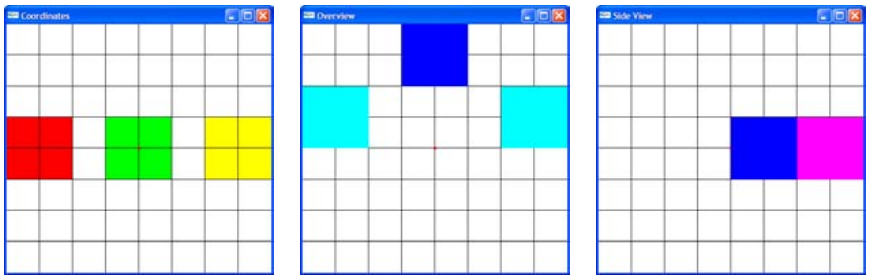
Front View **Top View** **Side View**



Case Study 6, continued
- Scaling of 3 cubes

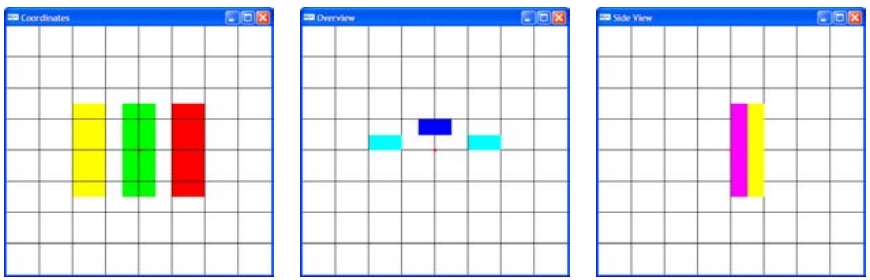
Before Scaling

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



After Scaling

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



Code:

```
glLoadIdentity();
// camera default: located at (0,0,0)
// looking at (0,0,-1)
glScalef(-0.5,1.5,0.25);  // Scaling three objects

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

Observation:

1. The fixed point is (0,0,0), and is unaffected by the scaling.

2. On the X direction: Objects are scaled down by half, and reflected, because of scaling factor “-0.5”.

3. On the Y direction: Objects are scaled up by 1.5.

4. On the Z direction: Objects are scaled down by 0.25.