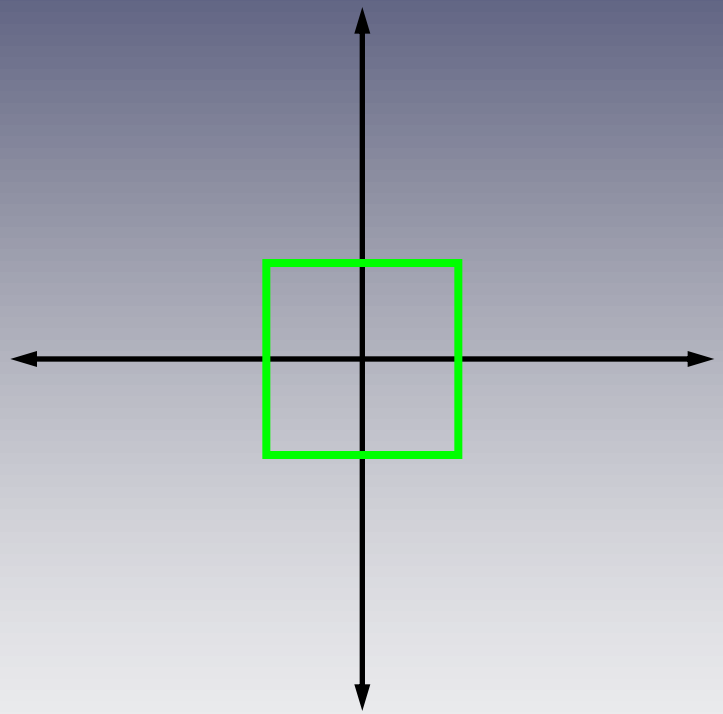
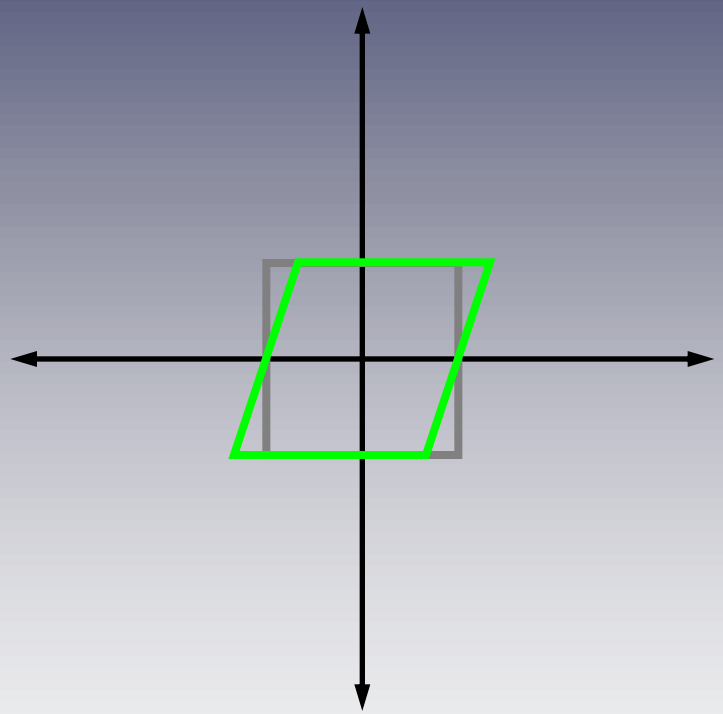
Reflections and shearing

Unit 2- Lecture 3

2-D Shear (horizontal)

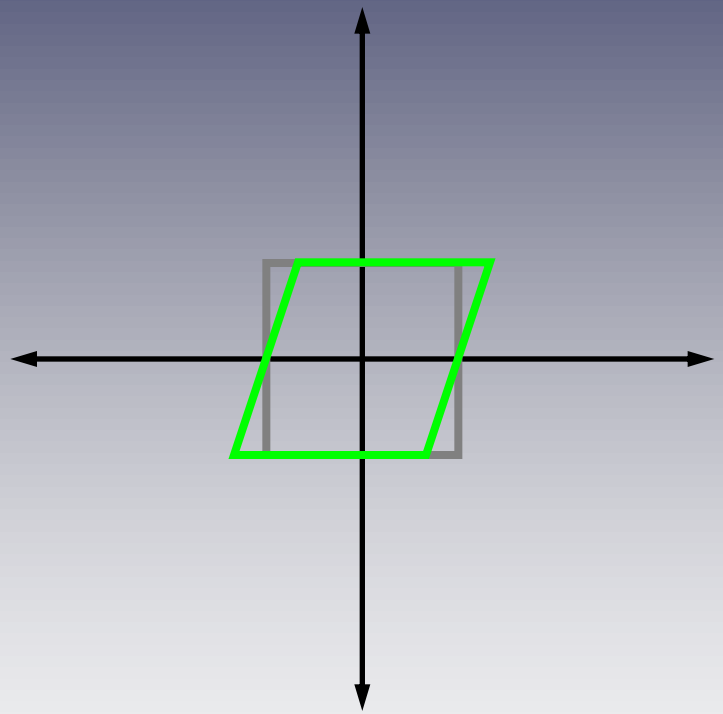


2-D Shear (horizontal)



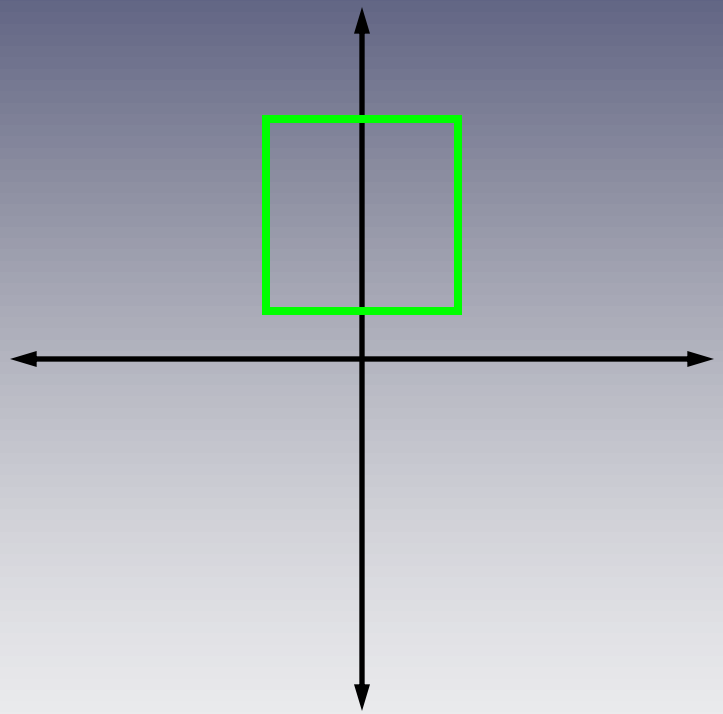
Horizontal displacement proportional to vertical position

2-D Shear (horizontal)

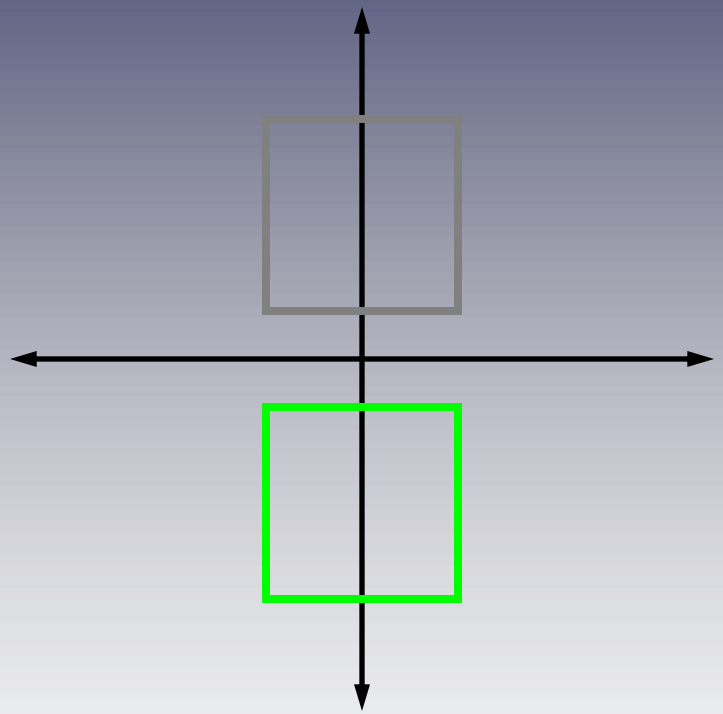


$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Reflection (vertical)

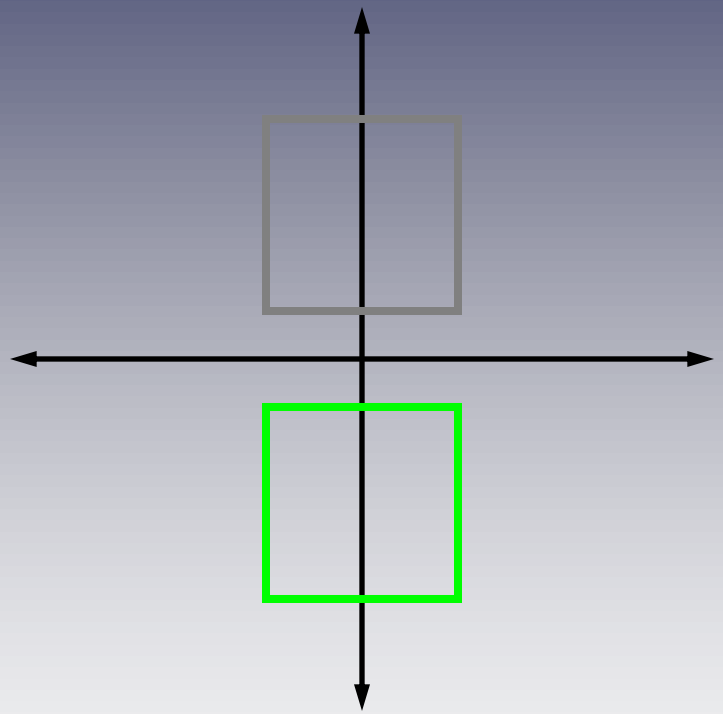


2-D Reflection (vertical)



Just a special case of scaling—”negative” scaling

2-D Reflection (vertical)



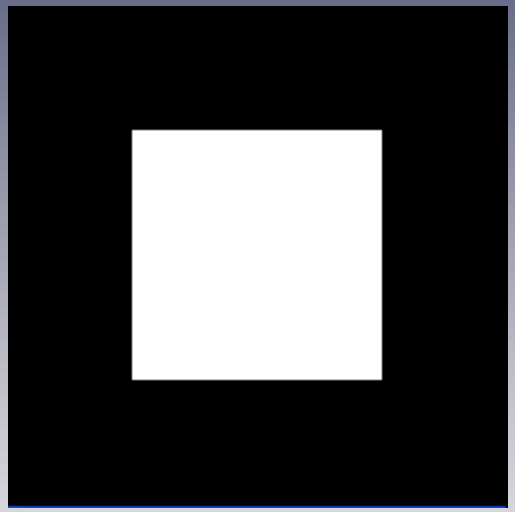
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Transformations: OpenGL

- 2-D transformation functions*
  - **`glTranslate(x, y, 0)`**
  - **`glScale(sx, sy, 0)`**
  - **`glRotate(theta, 0, 0, 1)`** (angle in degrees; direction is counterclockwise)
- Notes
  - Set **`glMatrixMode(GL_MODELVIEW)`** first
  - Transformations should be specified **before** drawing commands to be affected
  - Multiple transformations are applied in **reverse** order

*Technically, these are 3-D

Example: 2-D Translation in OpenGL

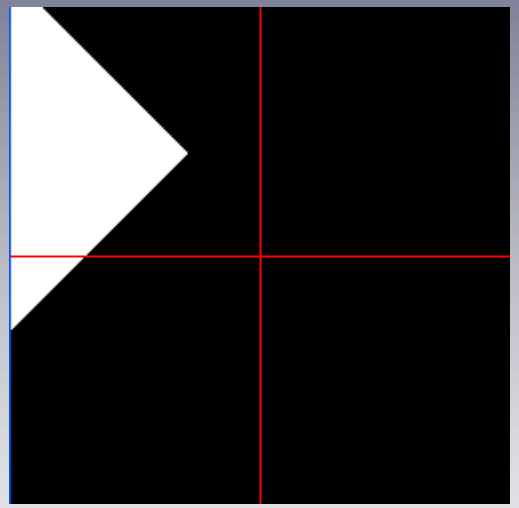


Two ways to do this:

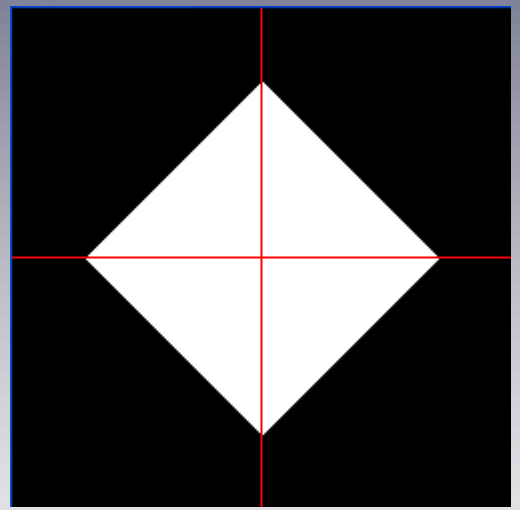
```
glRectf(.25,.25,.75,.75);
```

```
glTranslatef(.5,.5,0);
glRectf(-.25,-.25,.25,.25);
```

Example: Order of transformations



```
glRotatef(45,0,0,1);
glTranslatef(.5,.5,0);
glRectf(-.25,-.25,.25,.25);
```



```
glTranslatef(.5,.5,0);
glRotatef(45,0,0,1);
glRectf(-.25,-.25,.25,.25);
```

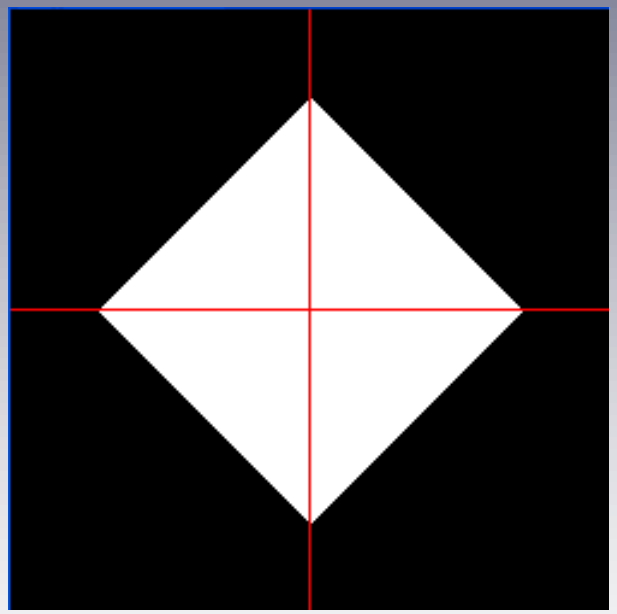
Remember: Order of application is **backwards** from drawing commands

Limiting "Scope" of Transformations

- Transformations are ordinarily applied to **all** subsequent draw commands
- To limit effects, use push/pop functions:

```
glPushMatrix();
// transform
// draw affected by transform
glPopMatrix();
// draw unaffected by transform
```

Example: Pushing, popping transformations



```
glPushMatrix();
glTranslatef(.5,.5,0);
glRotatef(45,0,0,1);
glRectf(-.25,-.25,.25,.25);
glPopMatrix();

glPushMatrix();
// draw axis lines
glPopMatrix();
```