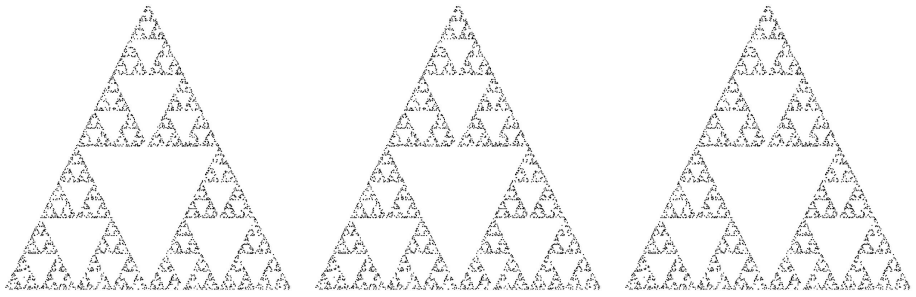
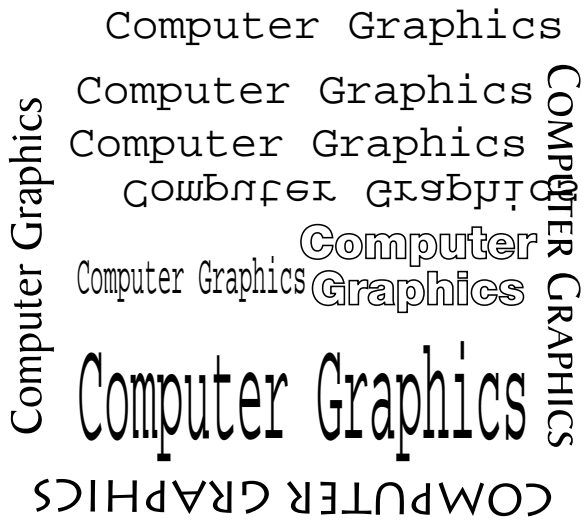


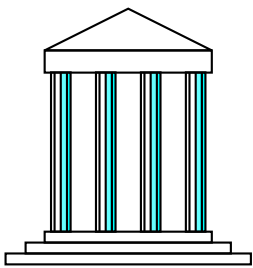
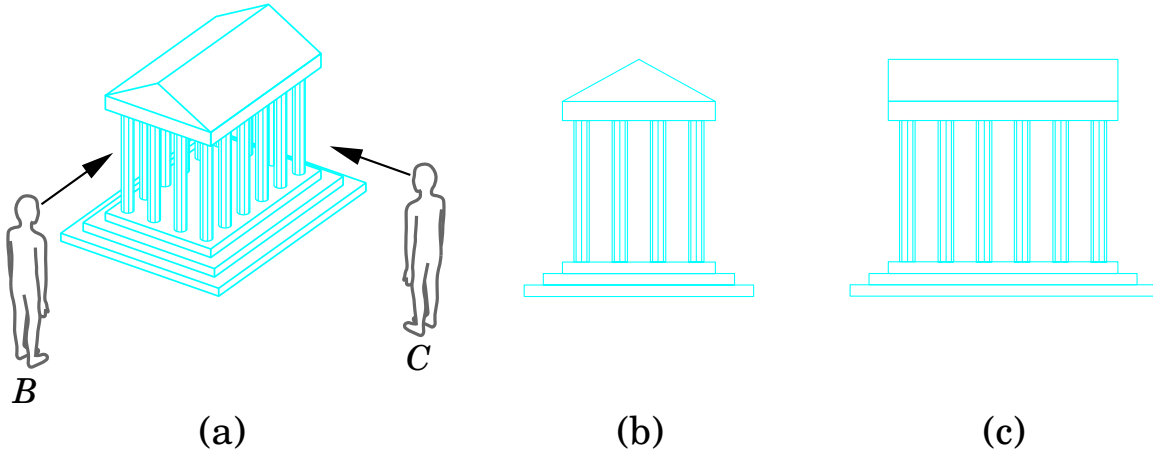
Uses of transformations:

- modeling: build complex models by positioning and transforming simple components

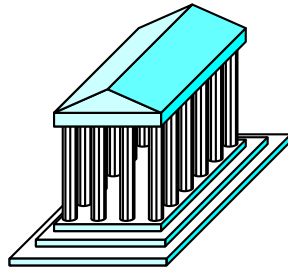


Uses of transformations, cont.:

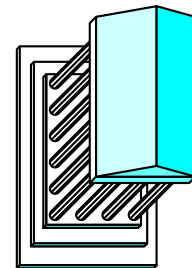
- viewing: place the virtual camera in the world:



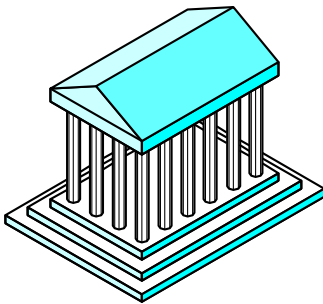
Front elevation



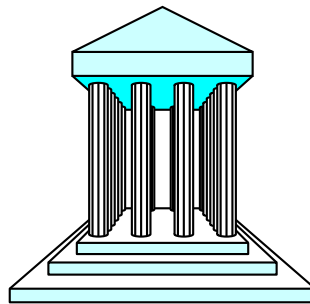
Elevation oblique



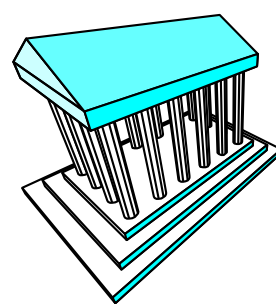
Plan oblique



Isometric



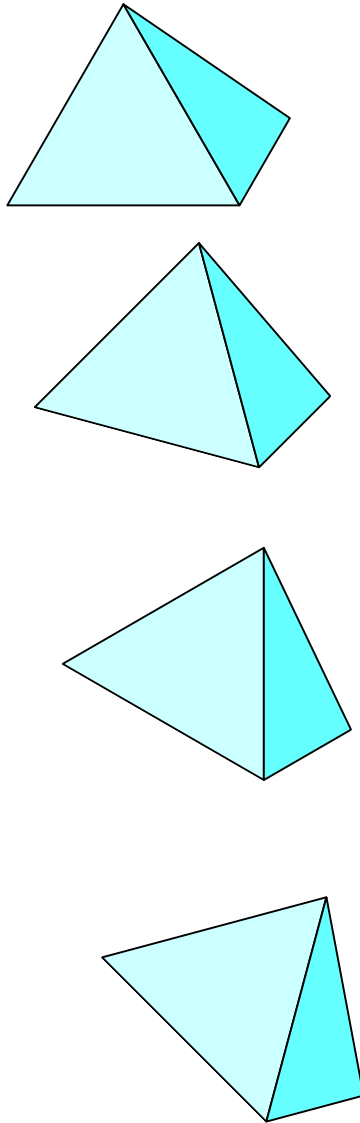
One-point perspective



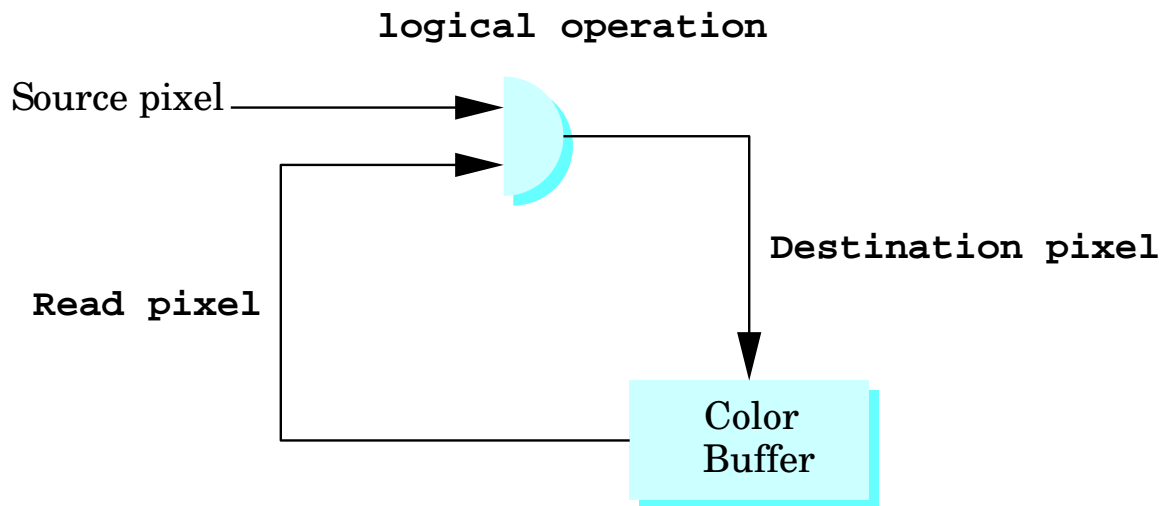
Three-point perspective

Uses of transformations, cont.:

- animation: vary transformations over time to create appearance of motion



In-class problem: pixelwise logic operations:

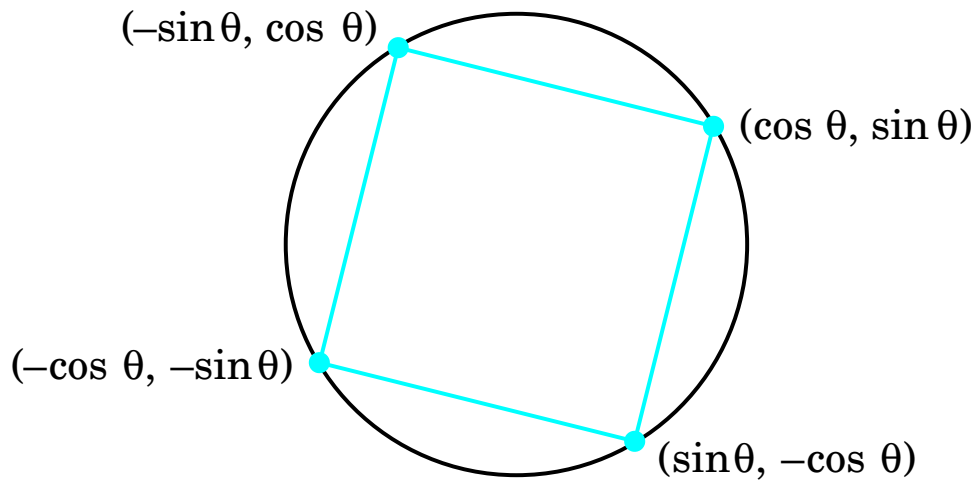


- enable: `glEnable(GL_COLOR_LOGIC_OP);`
- select mode: `glLogicOp([mode]);`
- XOR drawing mode: `glLogicOp(GL_XOR);`

Using logic operations:

```
if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
{
    glLogicOp(GL_COPY);
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
        glVertex2f(xm, ym);
        glVertex2f(xmm, ymm);
    glEnd();
    glFlush();
    glLogicOp(GL_XOR);
    glColor3f(0.0, 1.0, 0.0);
}
```

In-class problem: the mathematics of rotation:



```
/* assumes a global var theta */  
void display(void)  
{  
    glClear (GL_COLOR_BUFFER_BIT);  
    glBegin(gl_polygon);  
        thetar = theta / ((2 * M_PI)/360.0);  
        glVertex2f(cos(thetar), sin(thetar));  
        glVertex2f(-sin(thetar), cos(thetar));  
        glVertex2f(-cos(thetar), -sin(thetar));  
        glVertex2f(sin(thetar), -cos(thetar));  
    glEnd();  
}
```

Animating the rotating square:

```
void idle(void)
{
    theta += 2;
    if (theta>=360.0 theta -= 360.0;
    glutPostRedisplay();
}
```

Turning the animation on and off:

```
void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        glutIdleFunc(idle);
    if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
        glutIdleFunc(NULL);
}
```