Discussion Session 1

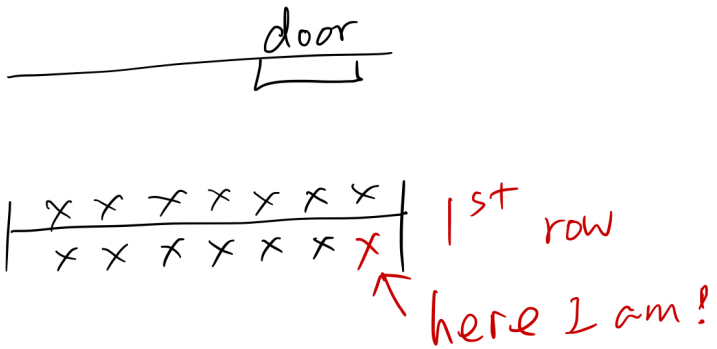
Sikun LIN
sikun@ucsb.edu

Role of the discussion sessions

- Coding examples
- Explain assignments
- Sample exam questions
- Q&A

Office Hours

- MWF 10am-noon
  - may change for a particular week, will have make up hours
- Location: CSIL lab, HFH 1140



Today’s Topics

- How to create objects for assignment 1
- How to program with OpenGL

Assignment 1

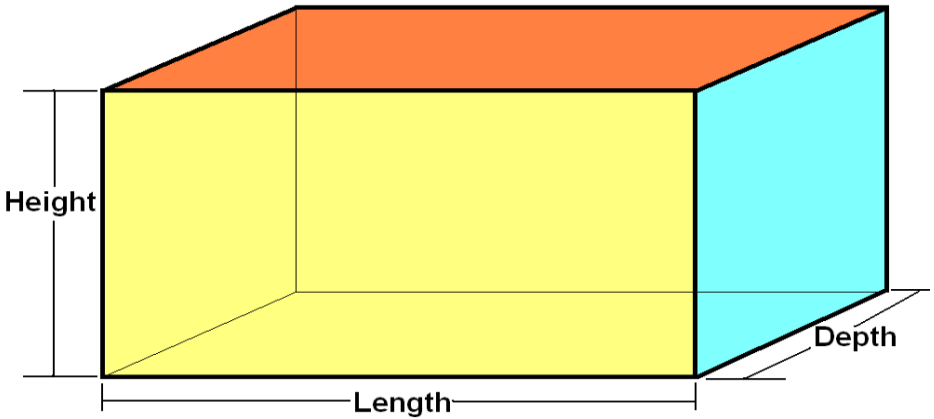
- Create static objects
  - No texture
  - No light
  - No animation
  - **DON’T use modeling softwares**
- Use **GLUT**
- Program should work on CSIL machines
  - Should have a Makefile
- Please read the assignment description carefully
- **Deadline: Midnight, Wednesday, October 18th**

Ways to Create Objects

- Two most basic primitives
  - Cube
  - Curved piece
- Composed Objects
  - One object can be modeled using other objects

Primitives

- **Cube**
  - **drawCube(length, height, depth, ...)**



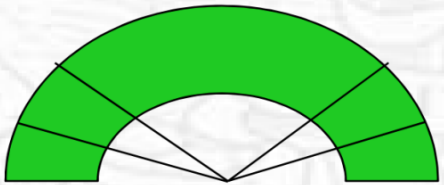

```
glBegin(GL_POLYGON);
// or glBegin(GL_QUADS);
    glVertex3f(...);
    glVertex3f(...);
    glVertex3f(...);
    glVertex3f(...);
    glEnd();
```

Primitives

- **Curved Piece**
  - **GL_TRIANGLE_STRIP**
  - **GL_QUAD_STRIP**

STRIPs are simpler! (write fewer coordinates)

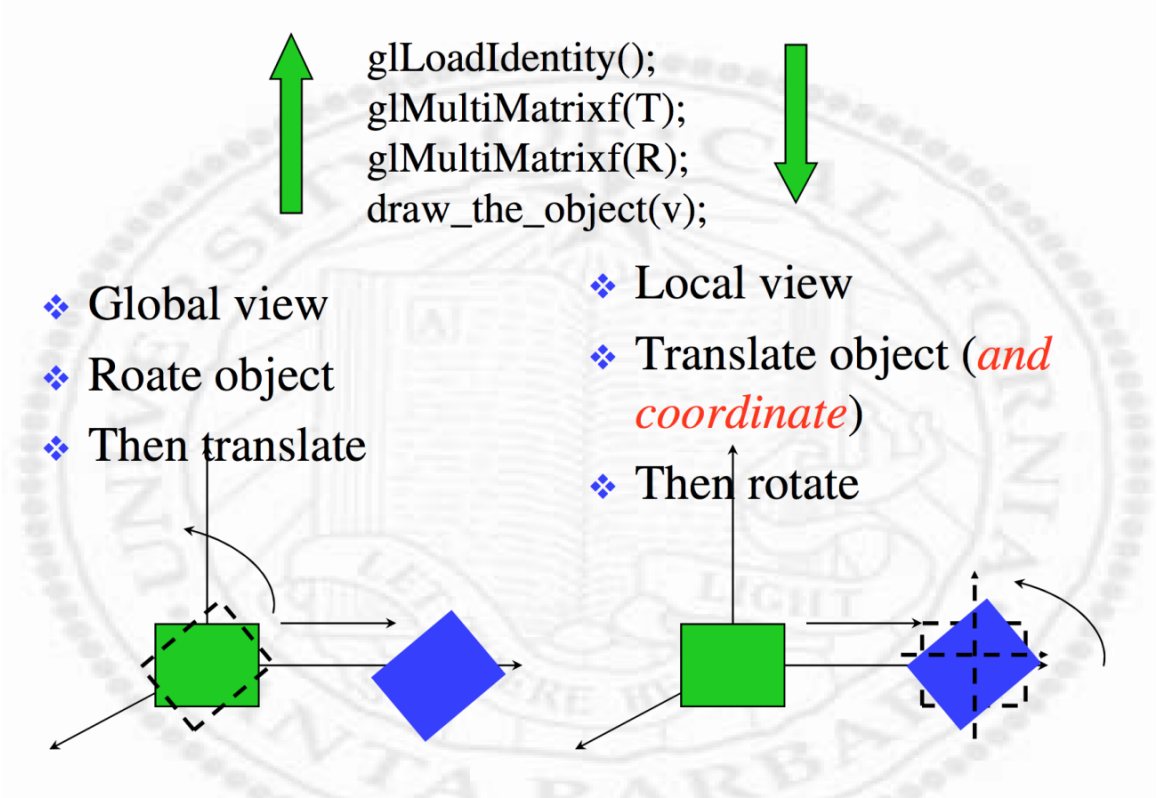
- A flexible quad or triangular strip, allowing:
  - Any angular extent
  - Different width
  - Sampling rate
  - Etc.



Composed Objects

- Use created objects
- 3 modeling transformations
  - glTranslate*()
  - glRotate*()
  - glScale*()
- glPushMatrix() & glPopMatrix()

Important: Local Coordinate System



Ex. Draw a composed block

1. This block is composed of:
    - Cubes
    - Numbers (curved pieces)
    - Letters (curved pieces)
2. Supposed we already have two general Functions:
    - drawNumber(...)
    - drawLetter(...)

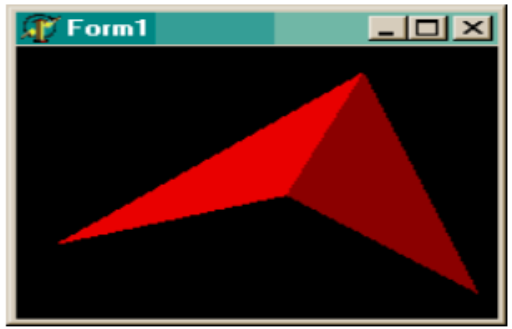
```
drawBlock(...) {
    drawCube(...); // draw cube 1: inner cube

    // draw edge cube 1
    glPushMatrix();
    glTranslate3f(...);
    glScale3f(...); // optional
    drawCube(...);
    glPopMatrix();

    ...

    // draw shapes on the surfaces
    glPushMatrix();
    glTranslate3f(...);
    glRotate3f(...);
    glScale3f(); // optional
    drawNumber(...); // or drawLetter()
    glPopMatrix();

    ...
}
```

Programming with OpenGL

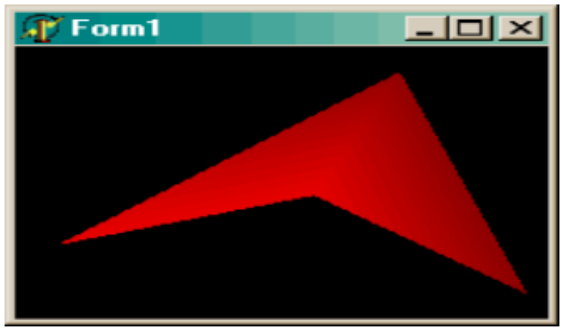
```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("demo");
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

```
void init(void)
{
     glClearColor (0.0, 0.0, 0.0, 0.0);
     glShadeModel (GL_FLAT);
     glEnable(GL_DEPTH_TEST);
}
```

GL_FLAT



GL_SMOOTH



No
glEnable(GL_DEPTH_TEST);

glEnable(GL_DEPTH_TEST);

```
void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
    glMatrixMode (GL_MODELVIEW);
}
```

GL_MODELVIEW -
working with object
in real view

GL_PROJECTION -
translate 3D to 2D

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glLoadIdentity (); /* clear the matrix */
    /* viewing transformation */
    gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glScalef (1.0, 2.0, 1.0); /* modeling transformation */
    glutWireCube (1.0);
    glFlush ();
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
    case 'q':
        exit(0);
        break;
    }
}
```

Q & A