

CT5510: Computer Graphics

# OpenGL: User Interface

---

BOCHANG MOON



# Keyboard Input

---

- Define a specific action when users press a key
  - `glutKeyboardFunc("your function name");`
    - When a user types for your OpenGL program, you can program a specific action

```
void main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowSize(512, 512);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Computer Graphics");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
}
```

# Keyboard Input

---

- Define a specific action when users press a key
  - `glutKeyboardFunc("your function name");`
    - When a user types for your OpenGL program, you can program a specific action
  - `glutPostRedisplay();`
    - Explicitly let OpenGL redraw the window

```
void keyboard(unsigned char key, int x, int y) {  
    if (key == 'r') {  
        red = 1.0, green = 0.0, blue = 0.0;  
    }  
  
    glutPostRedisplay();  
}
```

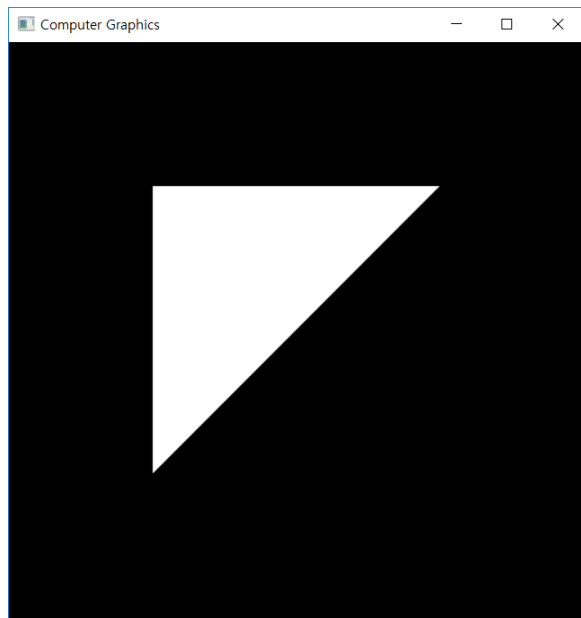
```
double red = 1.0;  
double green = 1.0;  
double blue = 1.0;
```

```
void display() {  
    glClearColor(0, 0, 0, 0); // Clear the screen  
  
    glColor3d(red, green, blue);  
    glBegin(GL_TRIANGLES);  
    glVertex2d(-0.5, -0.5);  
    glVertex2d(-0.5, 0.5);  
    glVertex2d(0.5, 0.5);  
    glEnd();  
  
    glFlush();  
}
```

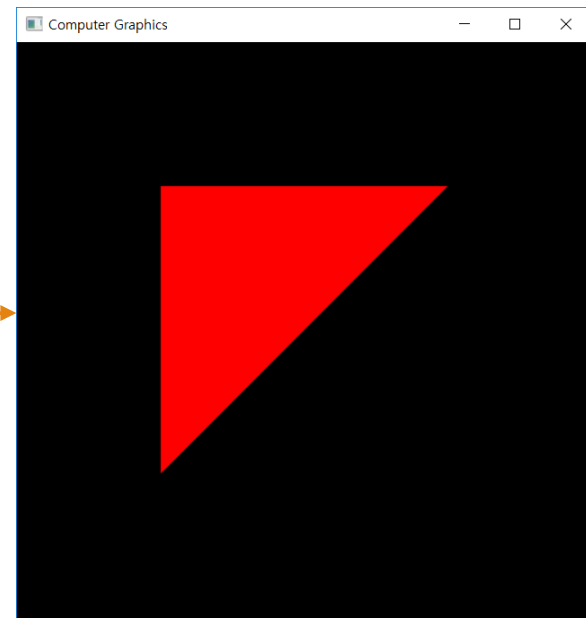
# Keyboard Input

---

- Define a specific action when users press a key
  - `glutKeyboardFunc("your function name");`
    - When a user types for your OpenGL program, you can program a specific action
  - `glutPostRedisplay();`
    - Explicitly let OpenGL redraw the window



press 'r'



# Keyboard Input

---

- Define a specific action when users press a key
  - `glutKeyboardFunc("your function name");`
    - When a user types for your OpenGL program, you can program a specific action
  - `glutPostRedisplay()`
    - Explicitly let OpenGL redraw the window
  - `void keyboard(unsigned char key, int x, int y)`
    - `key`: pressed keyboard character
    - `x, y`: location of the mouse when a key was pressed.

```
void keyboard(unsigned char key, int x, int y) {  
    if (key == 'r')  
        red = 1.0, green = 0.0, blue = 0.0;  
    else if (key == 'g')  
        red = 0.0, green = 1.0, blue = 0.0;  
    else if (key == 'b')  
        red = 0.0, green = 0.0, blue = 1.0;  
  
    glutPostRedisplay();  
}
```

# Mouse Input

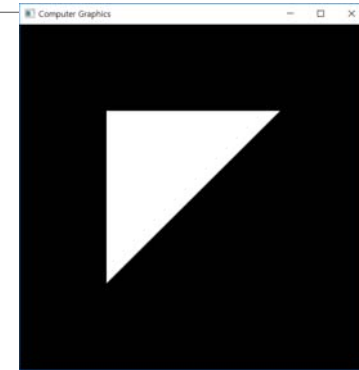
---

- Define a specific action when users give a mouse action
  - `glutMouseFunc("your function name")`
    - When a mouse button is pressed or released.
  - `glutMotionFunc("your function name")`
    - When a mouse pointer moves within your window while one or more mouse buttons are pressed.
- `void mouse(int button, int state, int x, int y)`
  - `button`: GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON, GLUT\_RIGHT\_BUTTON
  - `state`: GLUT\_UP, GLUT\_DOWN
  - `x, y`: pixel position when the mouse event occurred

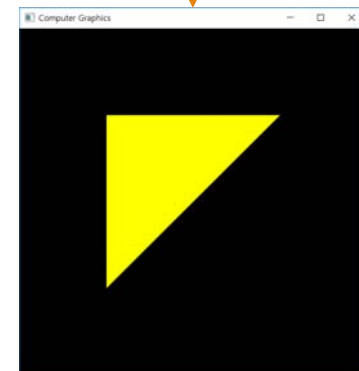
# Mouse Input

```
void main(int argc, char* argv[]) {  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);  
    glutInitWindowSize(512, 512);  
    glutInitWindowPosition(100, 100);  
    glutCreateWindow("Computer Graphics");  
    glutDisplayFunc(display);  
    glutKeyboardFunc(keyboard);  
    glutMouseFunc(mouse);  
    glutMainLoop();  
}
```

```
void mouse(int button, int state, int posX, int posY) {  
    if (button == GLUT_LEFT_BUTTON) {  
        red = 1.0, green = 1.0, blue = 0.0;  
    }  
}
```



press left  
button



# Other Events

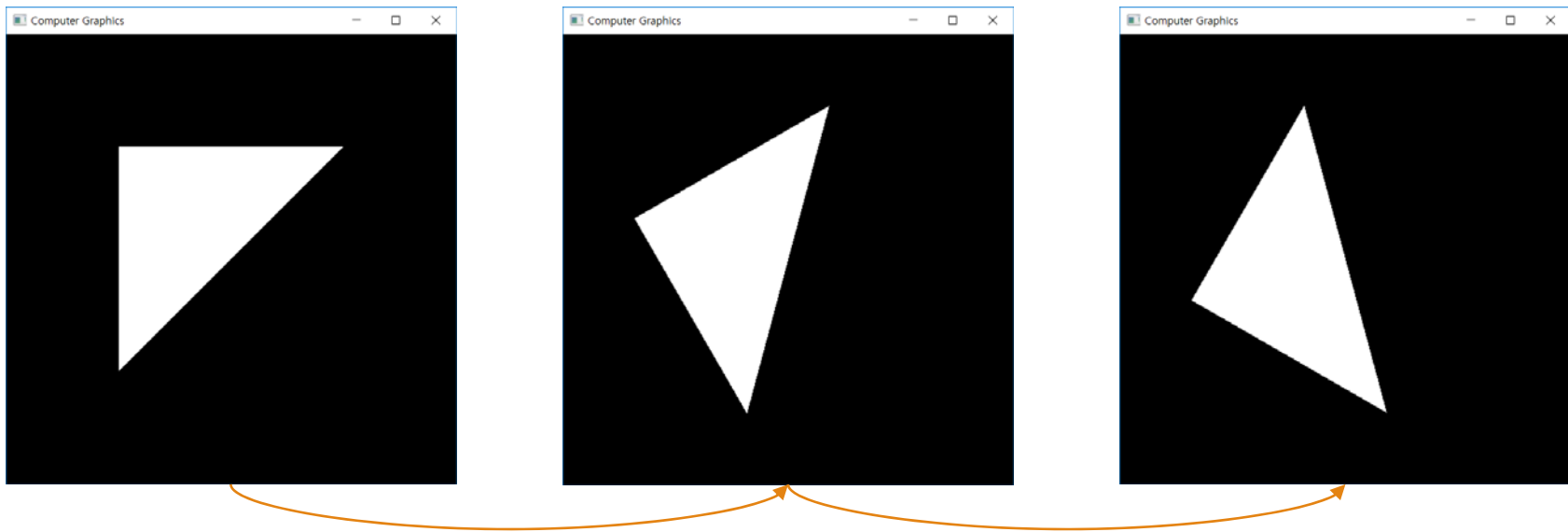
---

- `glutDisplayFunc("display")`
  - Be called whenever your window needs to be redrawn.
  - Can be explicitly called by `glutPostRedisplay()`
- `glutReshapeFunc("reshape")`
  - Be called whenever your window is resized or moved.

# Examples – Rotate Your Triangle

---

- Problem specification
  - Draw a triangle
  - Rotate your triangle by 30 degrees whenever the left button of your mouse is pressed.



# Examples – Rotate Your Triangle

---

- Recall the 2D transformation matrix
  - $rotate(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
  - $x_{new} = x_{old}\cos\theta - y_{old}\sin\theta$
  - $y_{new} = x_{old}\sin\theta + y_{old}\cos\theta$
  - Hint: utilize predefined c/c++ function *sin* and *cos*(an angle expressed in radians)
    - $radian = \frac{180}{\pi} \approx 57.2958$
    - *e.g.*, 30 degree =  $30 \times \frac{\pi}{180}$
- `double v1_new_x = v1_x * cos(degree * (PI / 180.0)) - v1_y * sin(degree * (PI / 180.0));`
- `double v1_new_y = v1_x * sin(degree * (PI / 180.0)) + v1_y * cos(degree * (PI / 180.0));`

# Examples – Rotate Your Triangle

---

- Initialize some variables

```
#include <math.h>

const double PI = 3.14159265359;

double v1_x = -0.5, v1_y = -0.5;
double v2_x = -0.5, v2_y = 0.5;
double v3_x = 0.5, v3_y = 0.5;
double degree = 0;
```

- Define your mouse event handler

```
void mouse(int button, int state, int posX, int posY) {
    if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN)) {
        degree = degree + 30.0;
    }
    glutPostRedisplay();
}
```

# Examples – Rotate Your Triangle

---

- Display function

```
void display() {
    glClearColor(0, 0, 0, 0);    // Specify the background color
    glClear(GL_COLOR_BUFFER_BIT); // Clear the screen with the background color

    double v1_new_x = v1_x * cos(degree * (PI / 180.0)) - v1_y * sin(degree * (PI / 180.0));
    double v1_new_y = v1_x * sin(degree * (PI / 180.0)) + v1_y * cos(degree * (PI / 180.0));

    double v2_new_x = v2_x * cos(degree * (PI / 180.0)) - v2_y * sin(degree * (PI / 180.0));
    double v2_new_y = v2_x * sin(degree * (PI / 180.0)) + v2_y * cos(degree * (PI / 180.0));

    double v3_new_x = v3_x * cos(degree * (PI / 180.0)) - v3_y * sin(degree * (PI / 180.0));
    double v3_new_y = v3_x * sin(degree * (PI / 180.0)) + v3_y * cos(degree * (PI / 180.0));

    glColor3d(red, green, blue);
    glBegin(GL_TRIANGLES);
    glVertex2d(v1_new_x, v1_new_y);
    glVertex2d(v2_new_x, v2_new_y);
    glVertex2d(v3_new_x, v3_new_y);
    glEnd();

    glFlush();
}
```

# Examples – Rotate Your Triangle

---

- Display function

