CT4201/EC4215: Computer Graphics

OpenGL: User Interface

BOCHANG MOON

Keyboard Input

- Define a specific action when users press a key
  - glfwSetKeyCallback(window, key_callback);
    - When a user types for your OpenGL program, your can program a specific action

```
int main()
{
    // Init GLFW
    ...
    // Set the required callback functions
    glfwSetKeyCallback(window, key_callback);

    // Render loop
    ...
}
```

Keyboard Input

- Define a specific action when users press a key
  - glfwSetKeyCallback(window, key_callback);
    - When a user types for your OpenGL program, your can program a specific action

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode) {
    ...
}
```

window	The window that received the event.
key	The keyboard key that was pressed or released.
scancode	The system-specific scancode of the key.
action	GLFW_PRESS, GLFW_RELEASE or GLFW_REPEAT.
mods	Bit field describing which modifier keys were held down. (Alt, Control , Shift ...)

Keyboard Input

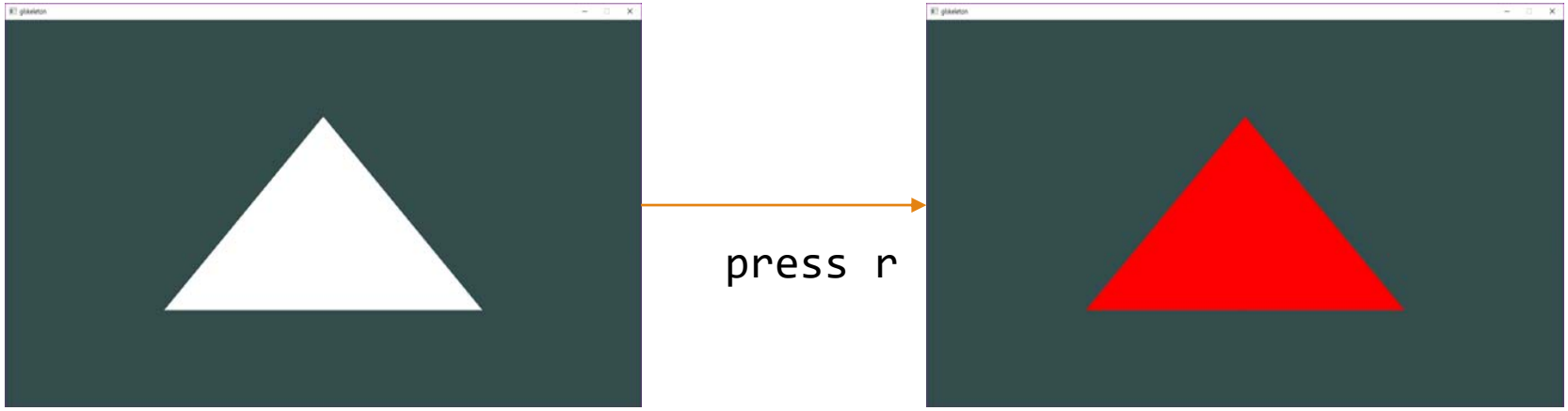
- Define a specific action when users press a key
  - glfwSetKeyCallback(“your window variable”, “your func name”);
    - When a user types for your OpenGL program, your can program a specific action

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

void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    if (key == GLFW_KEY_R && action == GLFW_PRESS) {
        red = 1.0, green = 0.0, blue = 0.0;
    }

}
```

```
// Render loop
while (!glfwWindowShouldClose(window))
{
    glfwPollEvents();

    // Render
    // Clear the colorbuffer
    glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_TRIANGLES);
    glColor3d(red, green, blue);
    glVertex3f(-0.5f, -0.5f, 0);
    glVertex3f(0.5, -0.5f, 0);
    glVertex3f(0, 0.5f, 0);
    glEnd();

    // Swap the screen buffers
    glfwSwapBuffers(window);
}
```

Keyboard Input

- Define a specific action when users press a key
  - glfwSetKeyCallback(“your window variable”, “your func name”);
    - When a user types for your OpenGL program, your can program a specific action



Keyboard Input

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
     if (key == GLFW_KEY_R && action == GLFW_PRESS) {
     red = 1.0, green = 0.0, blue = 0.0;
     }
     else if (key == GLFW_KEY_G && action == GLFW_PRESS) {
     red = 0.0, green = 1.0, blue = 0.0;
     }
     else if (key == GLFW_KEY_B && action == GLFW_PRESS) {
     red = 0.0, green = 0.0, blue = 1.0;
    }
}
```

Keyboard key table
https://www.glfw.org/docs/3.0/group__keys.html

Mouse Input

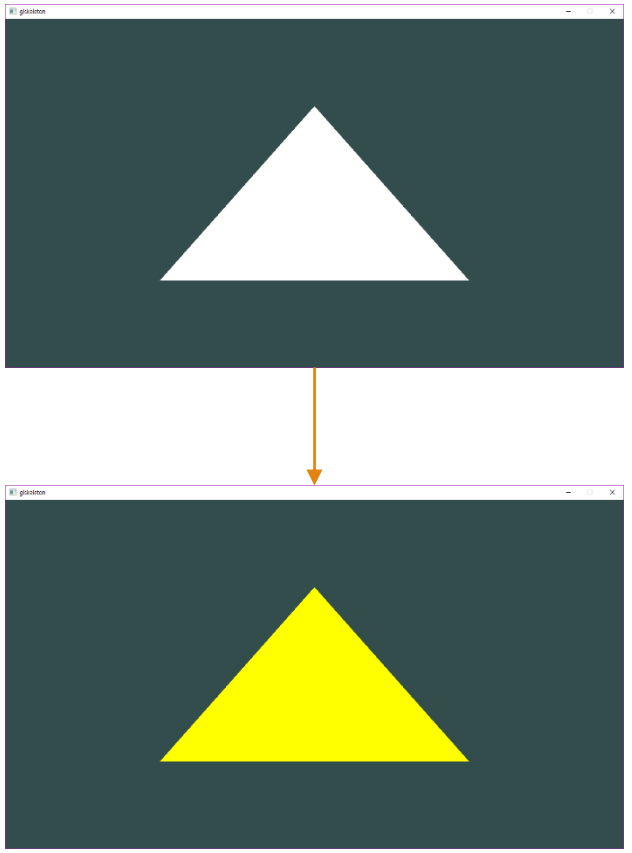
- Define a specific action when users give a mouse action
  - glfwSetCursorPosCallback(“your window variable”, “your func name”);
    - This function will be called when the mouse cursor position is changed.
  - glfwSetMouseButtonCallback(“your window variable”, “your func name”);
    - This function will be called when the mouse button is pressed, released.

```
void cursor_position_callback(GLFWwindow* window, double xpos, double ypos)
{
    ...
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (button == GLFW_MOUSE_BUTTON_RIGHT &&
     action == GLFW_PRESS)
            popup_menu();

}
```

Mouse Input

```
void mouse_button_callback(GLFWwindow* window,
int button, int action, int mods)
{
      if (button == GLFW_MOUSE_BUTTON_LEFT &&
          action == GLFW_PRESS)
      {
          red = 1.0, green = 1.0, blue = 0.0;
      }
}
```

```
// Render loop
while (!glfwWindowShouldClose(window))
{
    glfwPollEvents();

    // Render
    // Clear the colorbuffer
    glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_TRIANGLES);
    glColor3d(red, green, blue);
    glVertex3f(-0.5f, -0.5f, 0);
    glVertex3f(0.5, -0.5f, 0);
    glVertex3f(0, 0.5f, 0);
    glEnd();

    // Swap the screen buffers
    glfwSwapBuffers(window);
}
```



Other Events

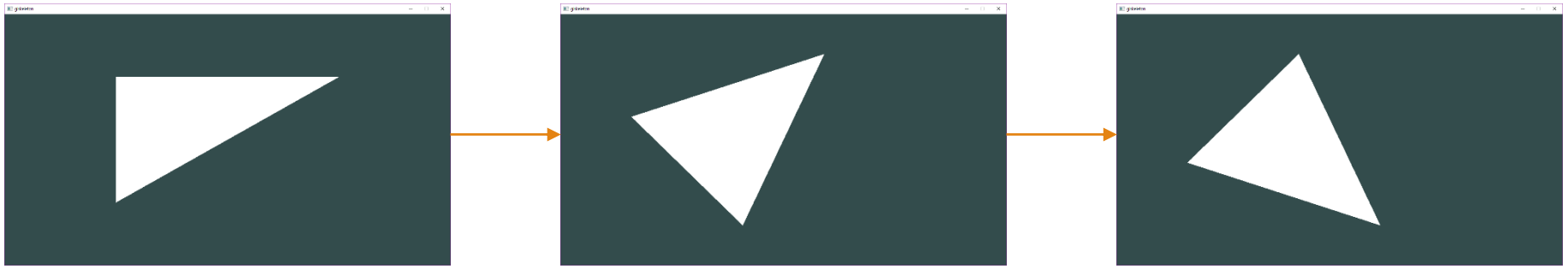
- Window resize callback functions
  - glfwSetWindowSizeCallback(window, “your func”);

```
void window_reshape_callback(GLFWwindow* window, int width, int height)
{
        // do something
}
```

- See more input handling guides in this link!
  - https://www.glfw.org/docs/latest/input_guide.html

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

```
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

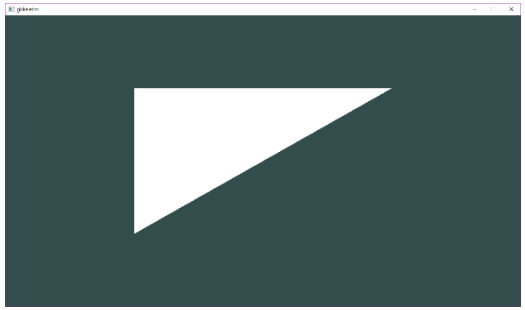
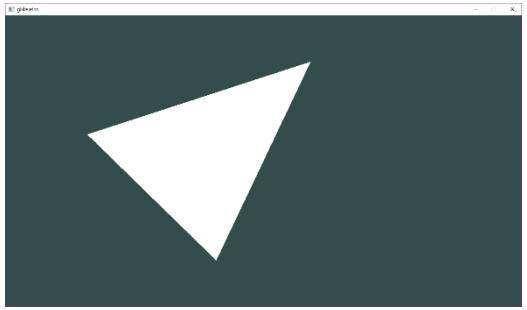
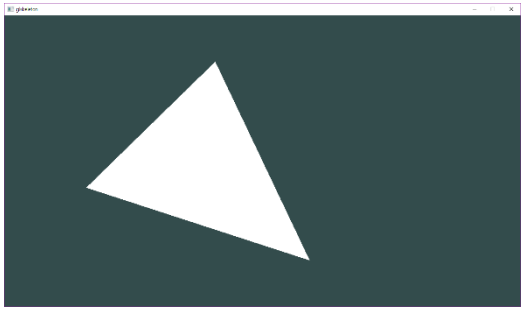
```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS) {
        degree = degree + 30;
    }
}
```

Examples – Rotate Your Triangle

```
// Render loop
while (!glfwWindowShouldClose(window))
{
    glfwPollEvents();

    // Render
    // Clear the colorbuffer
    glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    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));

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

    // Swap the screen buffers
    glfwSwapBuffers(window);
}
```

Examples – Rotate Your Triangle

- Display function

Callback Functions

```
int main()
{
    // Init GLFW

    // Set the required callback functions
    glfwSetKeyCallback(window, key_callback);
    glfwSetCursorPosCallback(window,  cursor_position_callback);
    glfwSetMouseButtonCallback(window, mouse_button_callback);

    // Define the viewport dimensions
    glViewport(0, 0, WIDTH, HEIGHT);

    // Render loop
    ...
}
```

Registration

Callback Functions

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    ...
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    ...
}

void cursor_position_callback(GLFWwindow* window, double xpos, double ypos)
{
    ...
}
```

Callback functions

More References

- Tutorial for GLFW
  - https://www.glfw.org/docs/latest/quick_guide.html