Daily Assignment 5

- Write down a Python program to..
- Draw a triangle using render() function in the next slide (DO NOT modify it!)

- If you press (not release) a key, the triangle should be transformed as this Table:

- Transformations should be **accumulated** (composed with previous one) unless you press '1'.
  - Use: accumulatedM = newM @ accumulated
  - You'll need to make 'accumulatedM' as a global variable

- **Set the window title to your student number.**
- Set the window size to (480,480).

Key	Transformation
W	Scale by 0.9 times in x direction
E	Scale by 1.1 times in x direction
S	Rotate by 10 degrees counterclockwise
D	Rotate by 10 degrees clockwise
X	Shear by a factor of -0.1 in x direction
C	Shear by a factor of 0.1 in x direction
R	Reflection across x axis
1	Reset the triangle with identity matrix

Daily Assignment 5

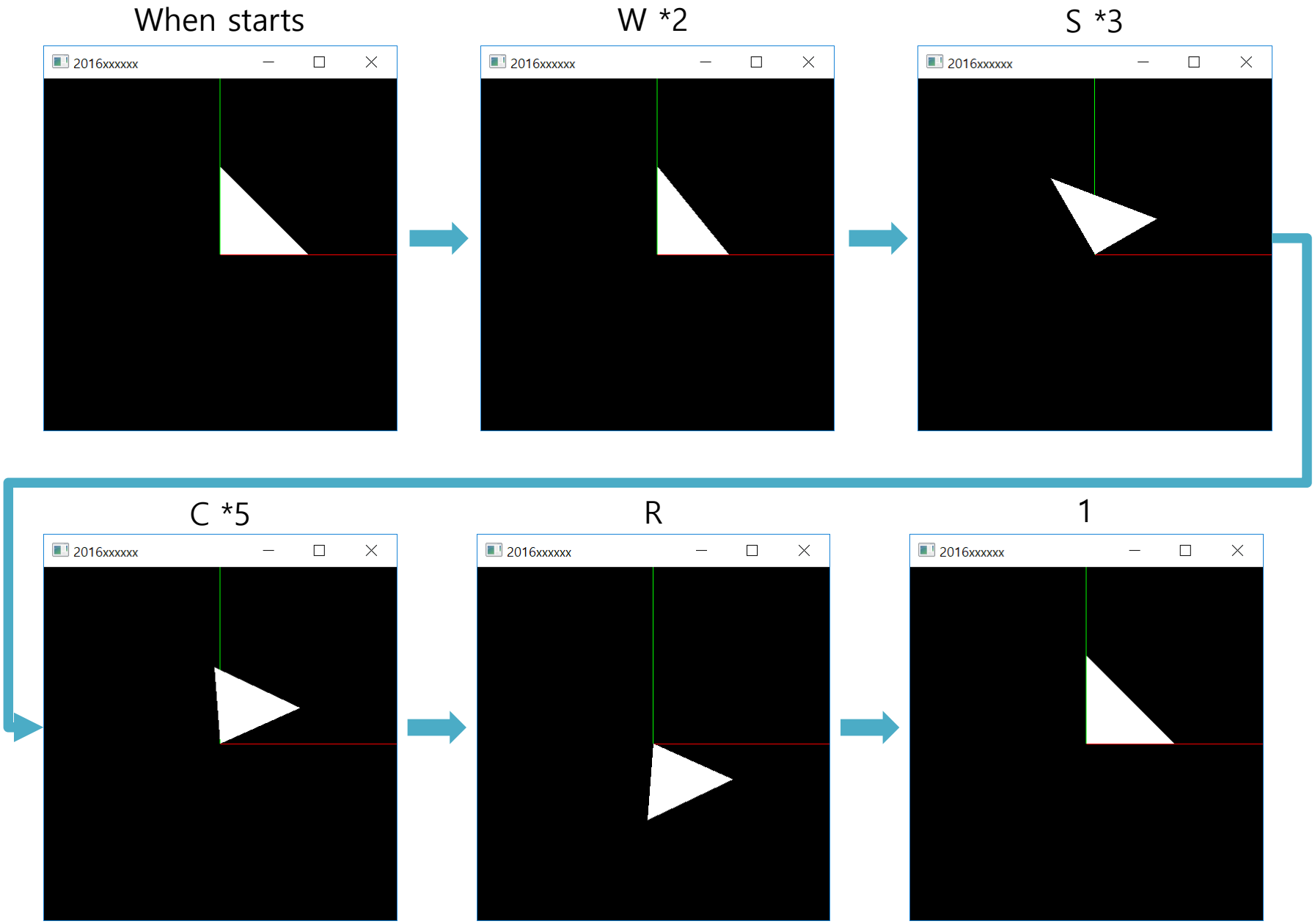
- render()

```
def render(T):
    glClear(GL_COLOR_BUFFER_BIT)
    glLoadIdentity()

    # draw cooridnate
    glBegin(GL_LINES)
    glColor3ub(255, 0, 0)
    glVertex2fv(np.array([0.,0.]))
    glVertex2fv(np.array([1.,0.]))
    glColor3ub(0, 255, 0)
    glVertex2fv(np.array([0.,0.]))
    glVertex2fv(np.array([0.,1.]))
    glEnd()

    # draw triangle
    glBegin(GL_TRIANGLES)
    glColor3ub(255, 255, 255)
    glVertex2fv(T @ np.array([0.0,0.5]))
    glVertex2fv(T @ np.array([0.0,0.0]))
    glVertex2fv(T @ np.array([0.5,0.0]))
    glEnd()
```

An example set of continuous transformation



```python
import glfw
from OpenGL.GL import *
import numpy as np

gComposedM = np.identity(2)

def render(T):
    #...

def key_callback(window, key, scancode, action, mods):
    global gComposedM
    if action==glfw.PRESS:
        if key==glfw.KEY_1:
            gComposedM = np.identity(2)
        elif key==glfw.KEY_W:
            M = np.array([[.9,0.],
                          [0.,1.]])
            gComposedM = M @ gComposedM
        elif key==glfw.KEY_E:
            M = np.array([[1.1,0.],
                          [0.,1.]])
            gComposedM = M @ gComposedM
        elif key==glfw.KEY_S:
            th = np.radians(10)
            M = np.array([[np.cos(th), -np.sin(th)],
                          [np.sin(th), np.cos(th)]])
            gComposedM = M @ gComposedM
        elif key==glfw.KEY_D:
            th = np.radians(-10)
            M = np.array([[np.cos(th), -np.sin(th)],
                          [np.sin(th), np.cos(th)]])
            gComposedM = M @ gComposedM
        elif key==glfw.KEY_X:
            M = np.array([[1.,-.1],
                          [0.,1.]])
            gComposedM = M @ gComposedM
        elif key==glfw.KEY_C:
            M = np.array([[1.,.1],
                          [0.,1.]])
            gComposedM = M @ gComposedM

        elif key==glfw.KEY_R:
            M = np.array([[1.,0.],
                          [0.,-1.]])
            gComposedM = M @ gComposedM


def main():
    global gComposedM
    if not glfw.init():
        return
    window =
glfw.create_window(480,480,"2016xxxxxx",
None,None)
    if not window:
        glfw.terminate()
        return
    glfw.make_context_current(window)
    glfw.set_key_callback(window, key_callback)

    while not glfw.window_should_close(window):
        glfw.poll_events()

        render(gComposedM)

        glfw.swap_buffers(window)

    glfw.terminate()

if __name__ == "__main__":
    main()
```