A glass of water by

18101716 Kashfia Hasan
19101145 Tasnia Zarin Shailee
19101147 Ahmad Zubair
19101011 Saik Rahman
19101316 Md. Wasif Ferdous

Code

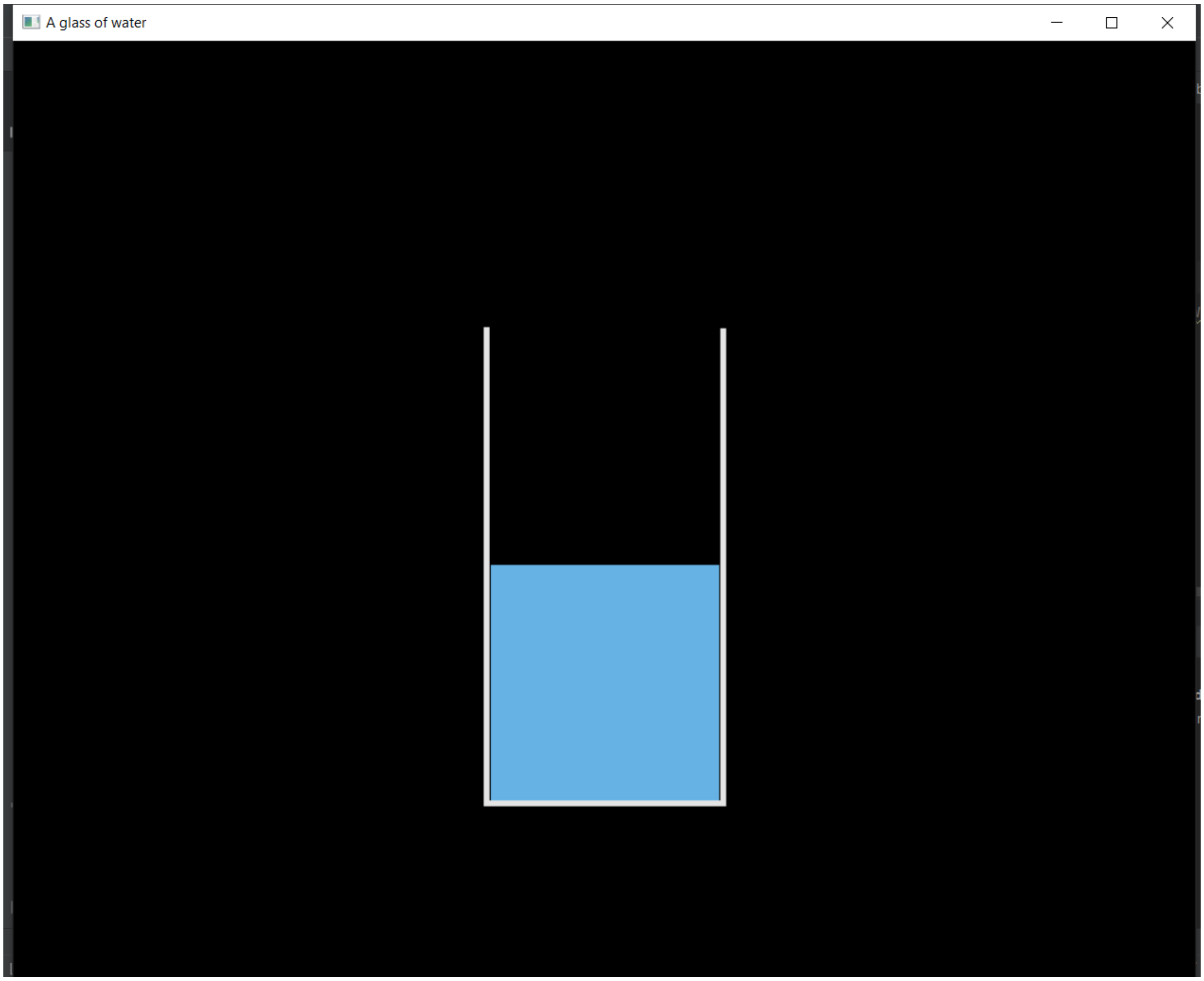
```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import numpy as np
import math
import time

found_zone = None


def find_zone(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    zone = None
    if abs(dx) >= abs(dy):
        if dx >= 0 and dy >= 0:
            zone = 0
        elif dx <= 0 and dy >= 0:
            zone = 3
        elif dx <= 0 and dy <= 0:
            zone = 4
        elif dx >= 0 and dy <= 0:
            zone = 7
    else:
        if dx >= 0 and dy >= 0:
            zone = 1
        elif dx <= 0 and dy >= 0:
            zone = 2
        elif dx <= 0 and dy <= 0:
            zone = 5
        else: # dx= > 0 and dy <= 0
            zone = 6
    return zone


def convert_zone(x1, y1, x2, y2):
    global found_zone
    found_zone = find_zone(x1, y1, x2, y2)

    if found_zone == 0:
        return x1, y1, x2, y2
    if found_zone == 1:
        a1 = y1
        b1 = x1
        a2 = y2
        b2 = x2
    elif found_zone == 2:
        a1 = y1
        b1 = -x1
        a2 = y2
        b2 = -x2
    elif found_zone == 3:
        a1 = -x1
        b1 = y1
        a2 = -x2
        b2 = y2
    elif found_zone == 4:
        a1 = -x1
        b1 = -y1
        a2 = -x2
        b2 = -y2
    elif found_zone == 5:
        a1 = -y1
        b1 = -x1
        a2 = -y2
        b2 = -x2
    elif found_zone == 6:
        a1 = -y1
        b1 = x1
        a2 = -y2
        b2 = x2
    else:
        a1 = x1
        b1 = -y1
        a2 = x2
        b2 = -y2

    return a1, b1, a2, b2


def convert_to_origin(x1, y1):
    if found_zone == 0:
        return x1, y1

    if found_zone == 1:
        a1 = y1
        b1 = x1
    elif found_zone == 2:
        a1 = -y1
        b1 = x1
    elif found_zone == 3:
        a1 = -x1
        b1 = y1
    elif found_zone == 4:
        a1 = -x1
        b1 = -y1
    elif found_zone == 5:
        a1 = -y1
        b1 = -x1
    elif found_zone == 6:
        a1 = y1
        b1 = -x1
    else:
        a1 = x1
        b1 = -y1

    return a1, b1


def mp_l(a1, b1, a2, b2):
    x1, y1, x2, y2 = convert_zone(a1, b1, a2, b2)
    dx = x2 - x1
    dy = y2 - y1
    d = 2 * dy - dx
    dNE = 2 * (dy - dx)
    dE = 2 * dy

    x = x1
    y = y1
    while x < x2:
        a, b = convert_to_origin(x, y)
        draw_points(a, b)
        if d <= 0:
            d = d + dE
            x += 1
            y += 0
        else:
            d = d + dNE
            x += 1
            y += 1


def circle_points(x, y, X, Y):
    draw_points(X+x, Y+y)
    draw_points(Y+y, X+x)
    draw_points(Y+y, X-x)
    draw_points(X+x, Y-y)
    draw_points(X-x, Y-y)
    draw_points(Y-y, X-x)
    draw_points(Y-y, X+x)
    draw_points(X-x, Y+y)


def midpoint_circle(rad, X, Y):
    d = 1 - rad
    x = 0
    y = rad

    while x <= y:
        circle_points(x, y, X, Y)
        if d < 0:
            d = d + 2 * x + 3
            x += 1
        else:
            d = d + 2 * (x - y) + 5
            x += 1
            y -= 1


def draw_points(x, y):
    glPointSize(5) #pixel size. by default 1 thake
    glBegin(GL_POINTS)
    glVertex2f(x,y) #jekhane show korbe pixel
    glEnd()


def draw_glass():
    glColor3f(0.9, 0.9, 0.9)
    mp_l(-100, 100, -100, -300)
    mp_l(-100, -300, 100, -300)
    mp_l(100, -300, 100, 100)


def draw_water():
    glColor3f(0.4, 0.7, 0.9)
    for i in range(-300+5, -100):  # y
        mp_l(-100+6, i, 100-5, i)    # x and 100-4 is exclusive


def draw_drops():
    glColor3f(1, 1, 1)
    midpoint_circle(4, -90, -90)
    midpoint_circle(9, -50, -50)
    midpoint_circle(7, 2, 2)


def scale(sc):
    v1 = np.array([[-300+5],
                   [100+5],
                   [1]])
    s = np.array([[1, 0, 0],
                  [0, sc, 0],
                  [0, 0, 1]])

    v11 = np.matmul(s, v1)
    v11_int = int(v11[1][0])

    for i in range(-300+5, v11_int):    # y
        mp_l(-100+6, i, 100-5, i)  # x and 100-4 is exclusive


def rotate_glass(angle):
    a = math.cos(math.radians(angle))
    b = math.sin(math.radians(angle))

    r = np.array([[a, -b, 0],
                  [b, a, 0],
                  [0, 0, 1]])

    g1 = np.array([[-100],
                   [100],
                   [1]])
    g2 = np.array([[-100],
                   [-300],
                   [1]])
    g3 = np.array([[100],
                   [-300],
                   [1]])
    g4 = np.array([[100],
                   [100],
                   [1]])

    g11 = np.matmul(r, g1)
    g22 = np.matmul(r, g2)
    g33 = np.matmul(r, g3)
    g44 = np.matmul(r, g4)

    glColor3f(0.9, 0.9, 0.9)
    mp_l(g11[0][0], g11[1][0], g22[0][0], g22[1][0])
    mp_l(g22[0][0], g22[1][0], g33[0][0], g33[1][0])
    mp_l(g33[0][0], g33[1][0], g44[0][0], g44[1][0])


def rem_water():
    glColor3f(0.4, 0.7, 0.9)
    mp_l(-250, -280, 300, -280)
    mp_l(-260, -285, 270, -285)
    mp_l(-270, -290, 300, -290)
    mp_l(-250, -295, 305, -295)
    mp_l(-300, -300, 330, -300)
    mp_l(-270, -305, 310, -305)
    mp_l(-290, -310, 350, -310)
    mp_l(-260, -315, 290, -315)


def iterate():
    glViewport(0, 0, 1000, 1000)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(-500, 500, -500, 500, 0.0, 1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()


def showScreen():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()
    iterate()
    #call the draw methods here
    draw_glass()
    draw_water()
    glutSwapBuffers()

    print("Press 1 to raise the water level\nPress 2 to raise the water level more
and\nPress 0 to emtpy the glass")
    user_input = input()
    if user_input == '1':
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glLoadIdentity()
        iterate()
        draw_glass()
        draw_water()
        scale(0.08)
        draw_drops()
        glutSwapBuffers()

    if user_input == '2':
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glLoadIdentity()
        iterate()

        draw_glass()
        draw_water()
        scale(0.9)
        draw_drops()
        glutSwapBuffers()

    if user_input == '0':
        angle = 0
        while angle >= -180:
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
            glLoadIdentity()
            iterate()
            rotate_glass(angle)

            time.sleep(0.06)
            glutSwapBuffers()
            angle -= 10
        rem_water()
    glutSwapBuffers()


glutInit()
glutInitDisplayMode(GLUT_RGBA)
glutInitWindowSize(1000, 1000) #window size
glutInitWindowPosition(0, 0)
wind = glutCreateWindow(b"A glass of water") #window name
glutDisplayFunc(showScreen)
```

```
glutMainLoop()
```

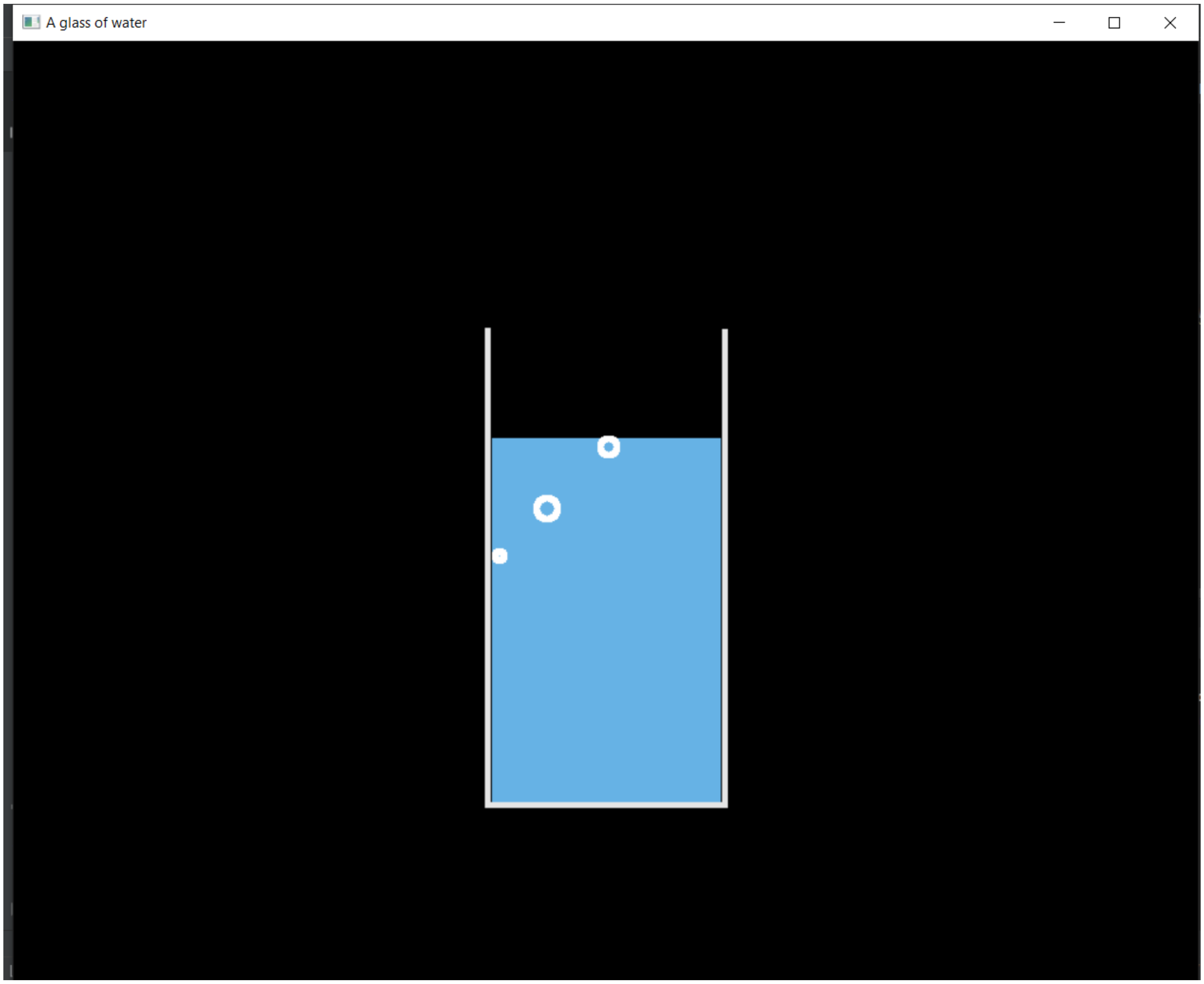
Output

a half-filled glass of water. The water level slightly rises when 1 is entered as input. The water level is increased even further when the input value of 2 is used. Last but not least, if input 0 is given, the glass will be spun 180 degrees and all of the water that has fallen from it will be on the floor.

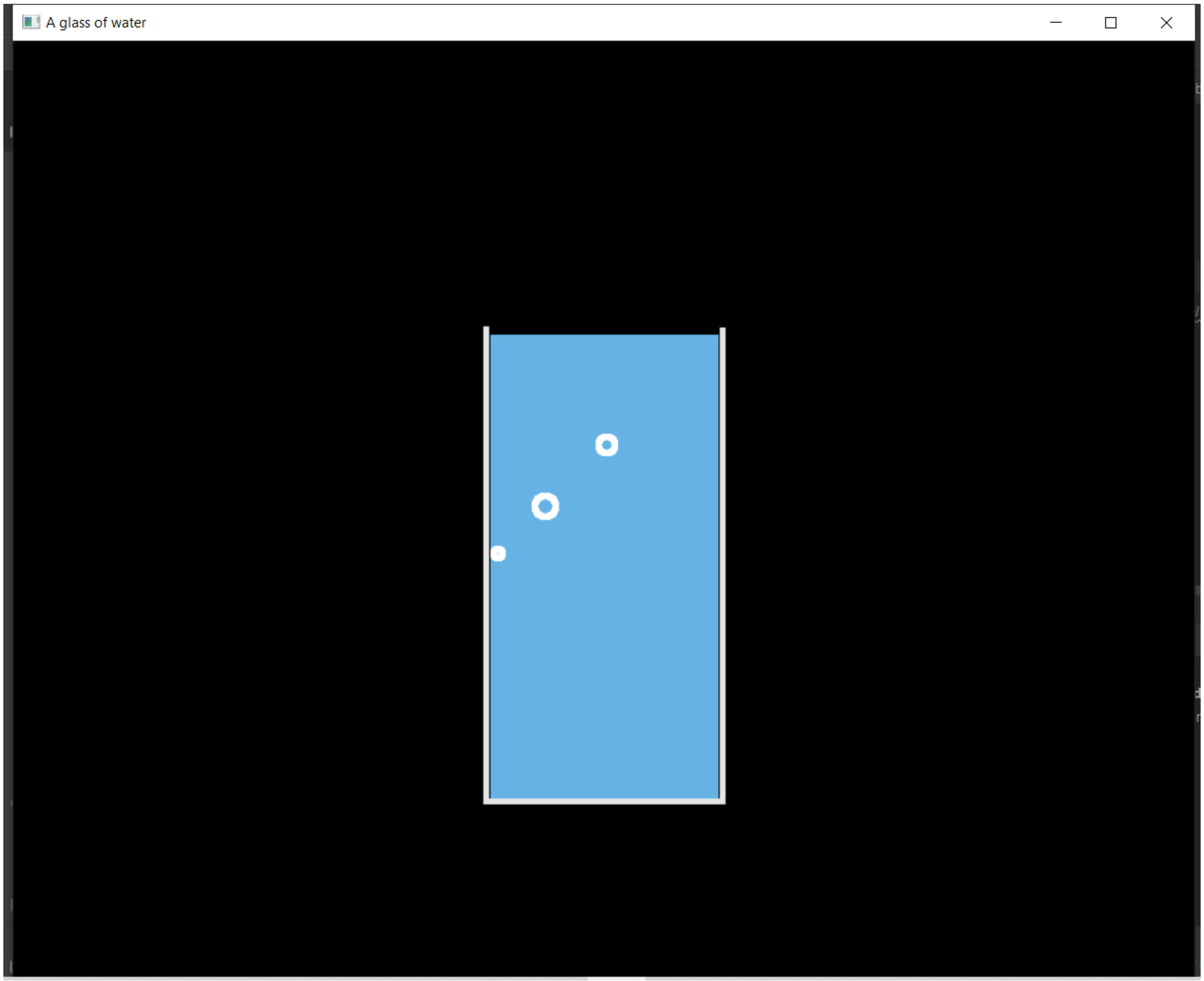
- Initial output



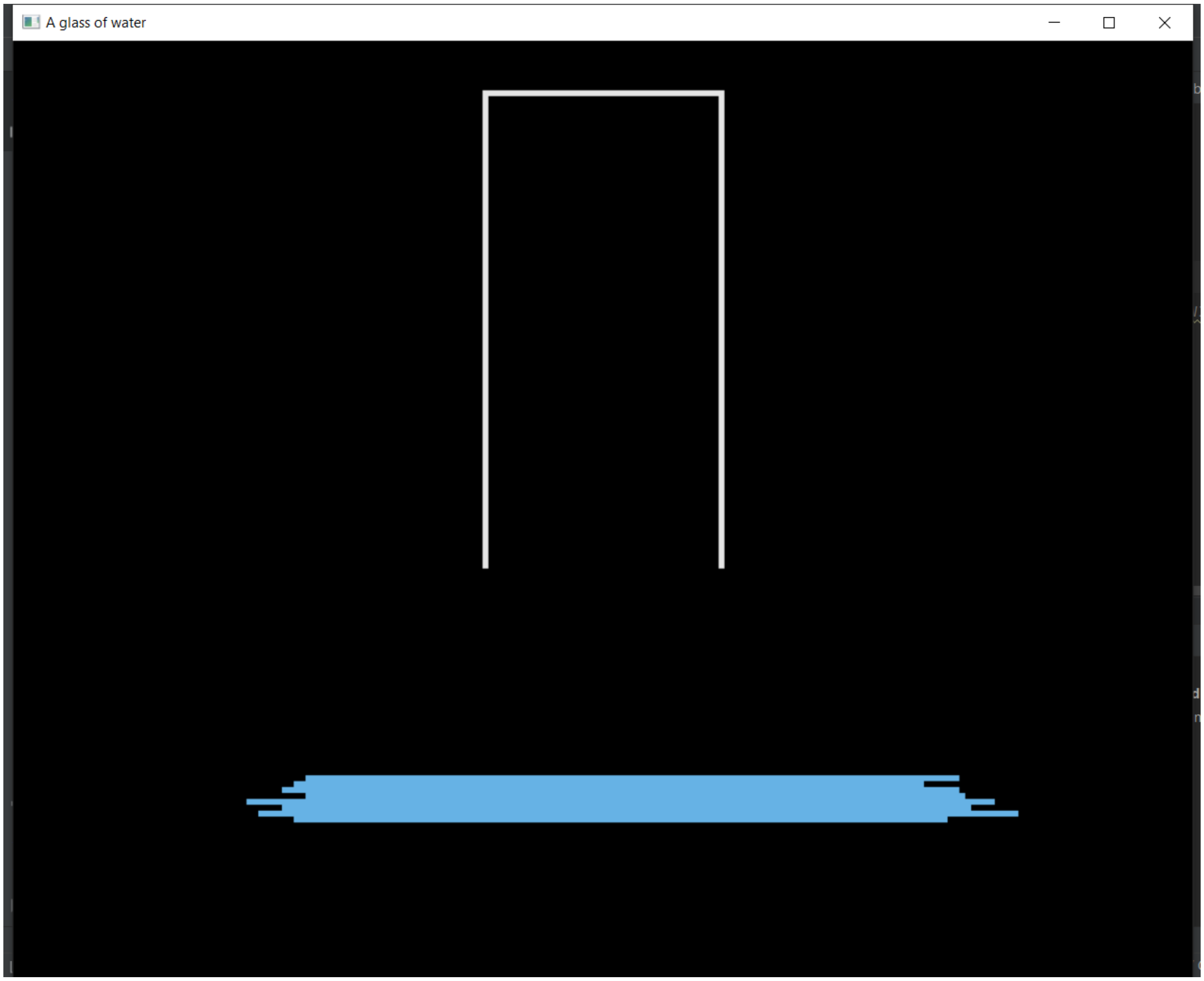
- When input is 1



- When input is 2



- When input is 0


A glass of water