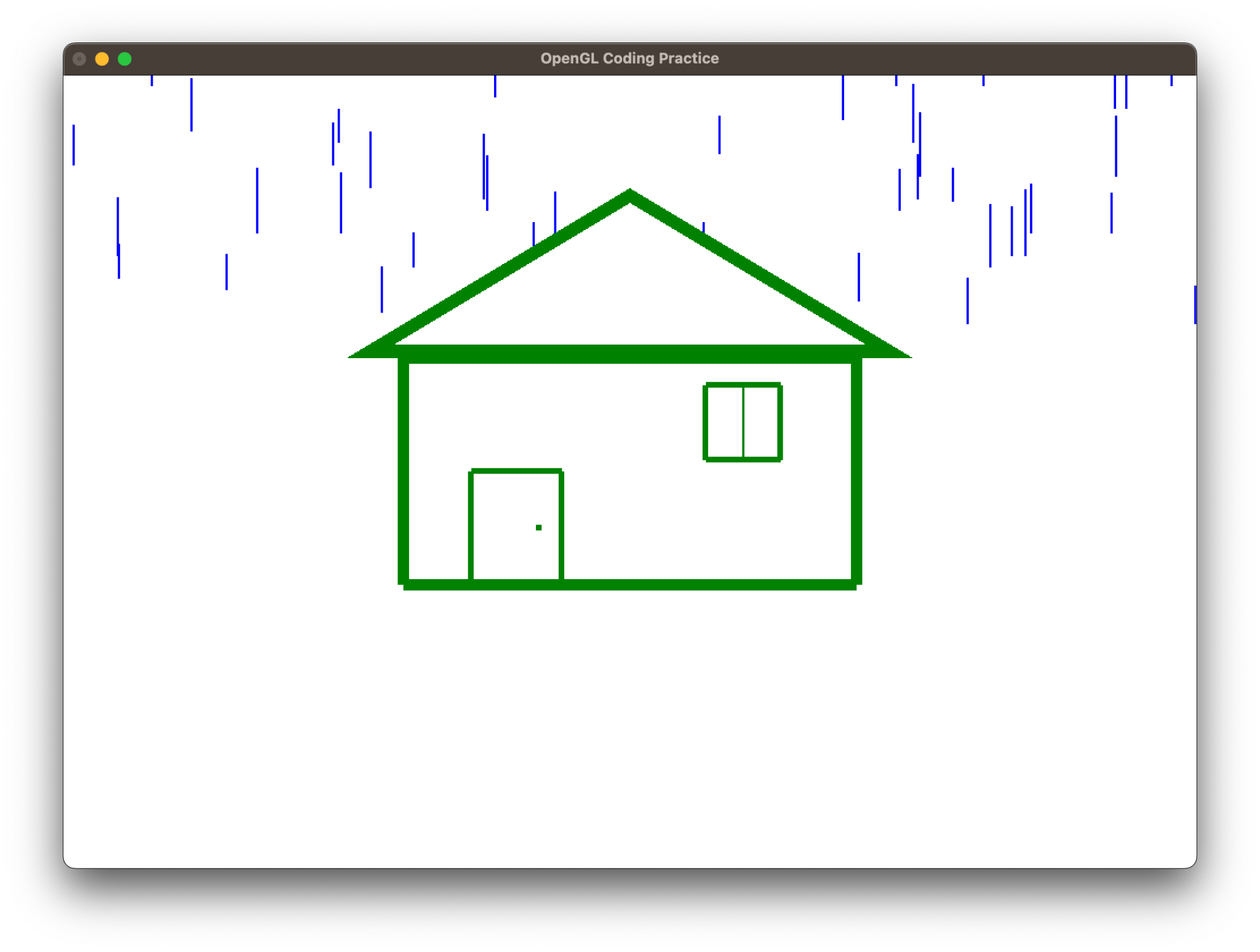
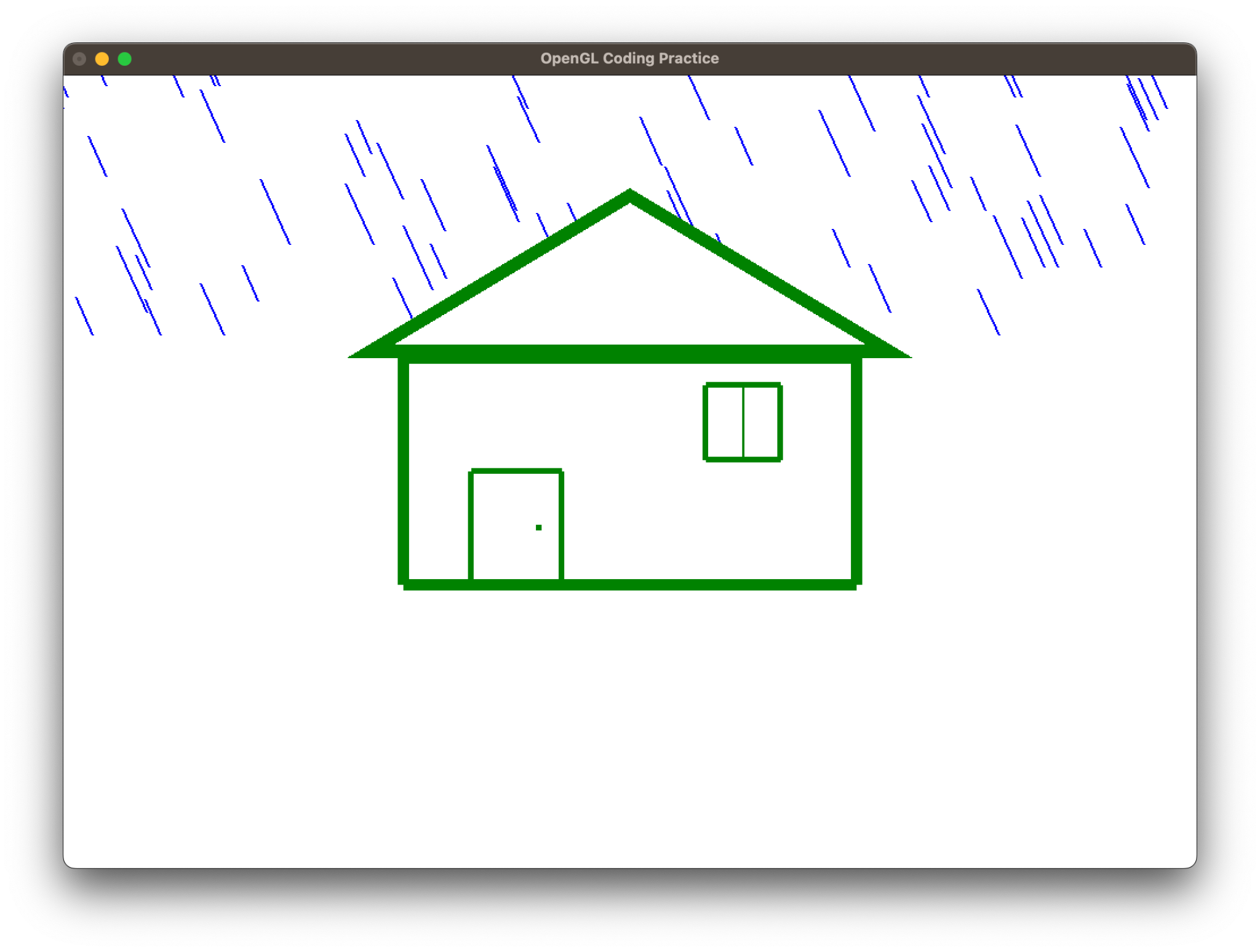
Mosheur Rahman Wolied

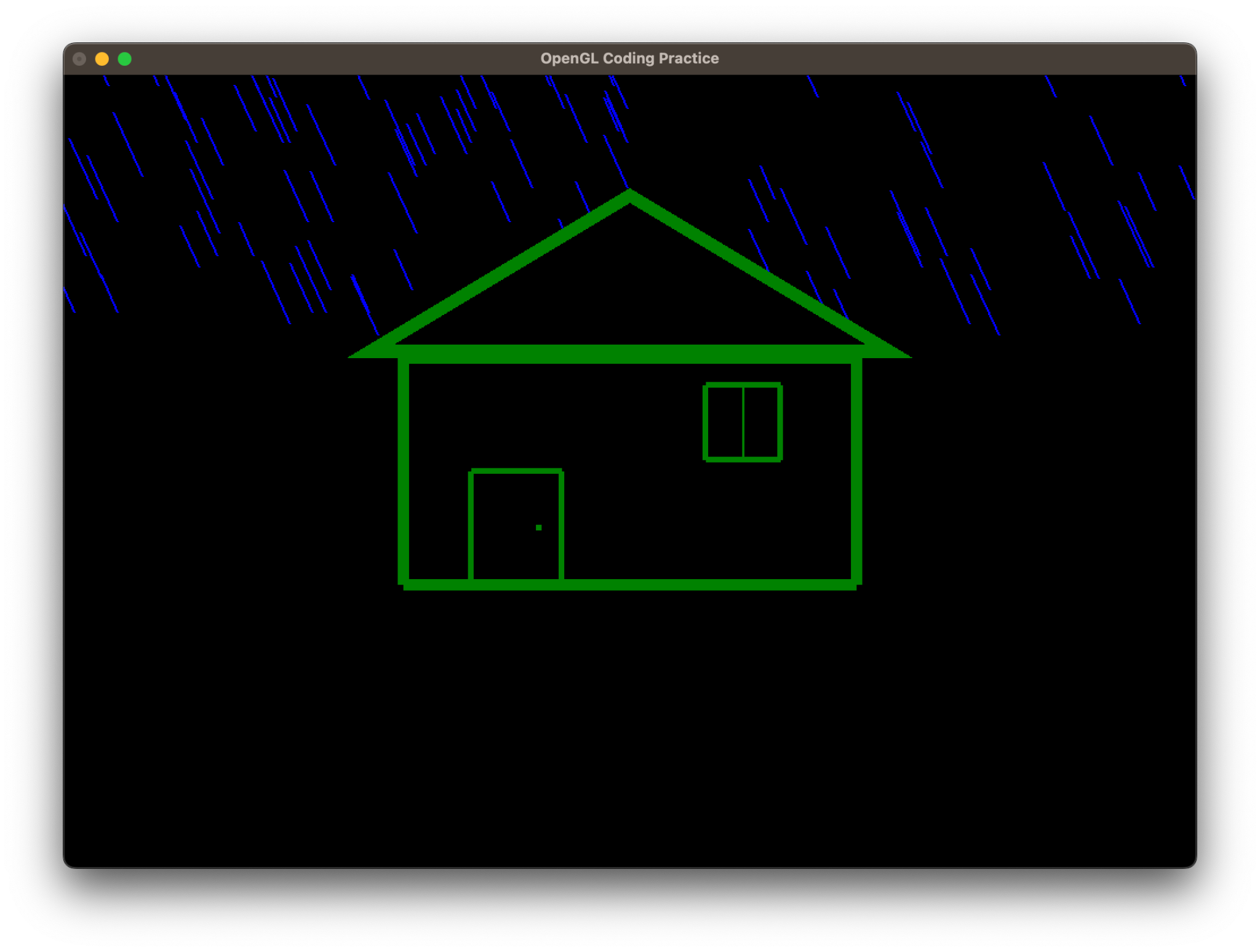
CSE423 Lab Assignment 1

Task 1:

| from OpenGL.GL import \* from OpenGL.GLUT import \* from OpenGL.GLU import \*  import random import time import math  SCREEN\_H = 700 SCREEN\_W = 1000  rain\_drop\_count = 100 rain\_points = [(random.randint(0,SCREEN\_W),random.randint(SCREEN\_H,SCREEN\_H+200), random.randint(30,60))] rain\_direction\_vertical = 'DOWN' rain\_skew = 0 rain\_speed\_horizontal = 2 rain\_speed = [10,rain\_speed\_horizontal] time\_interval = 1 last\_rain\_time = time.time() last\_time\_day = time.time() day = True color = 1  def change\_time():  global day  day = not day  def set\_day():  global day,color, last\_time\_day  change = False  if day:  if color < 1 and time.time() > last\_time\_day + 0.05:  last\_time\_day = time.time()  color += 0.01  change = True  else:  if color > 0 and time.time() > last\_time\_day + 0.05 :  last\_time\_day = time.time()  color -= 0.01  change = True  if color < 0:  color = 0  change = True    if change:  glClearColor(color,color,color,color)  glutPostRedisplay()   def keyboardListener(key,x,y):  if key == b'w':  print('time changed to', day)  change\_time()  def specialKeyListener(key, x, y):  global rain\_direction\_vertical,rain\_skew, day    if key==GLUT\_KEY\_UP:  rain\_direction\_vertical = 'UP'  print("Rain Direction Changed to UP")  if key== GLUT\_KEY\_DOWN:   rain\_direction\_vertical = 'DOWN'  print("Rain Direction Changed to DOWN")  if key==GLUT\_KEY\_LEFT and rain\_skew > -50:  rain\_skew -= 0.1  if key== GLUT\_KEY\_RIGHT and rain\_skew < 50:   rain\_skew += 0.1   glutPostRedisplay()  def draw\_house(starting\_x, starting\_y,base,height,roof\_height,roof\_base,scale=15):  global color  # Draw the roof  glColor3f(0.1, 0.5, 0.0)  draw\_triangle(starting\_x, starting\_y, base=roof\_base,height=roof\_height, hollow=True ,scale=scale, color=color)    glColor3f(0.1, 0.5, 0.0)  # Draw the house  x1,y1 = int(starting\_x - (roof\_base/2)) + int((roof\_base-base)/2) , starting\_y-roof\_height  x2,y2 = x1,y1-height  x3,y3 = x2+base, y2  x4,y4 = x3, y1    draw\_box(x1,y1,x2,y2,x3,y3,x4,y4,10)    # draw the door  door\_height = int(height/2)  door\_width = int(base/5)  door\_x1 = int(x1 + (base/4) - (door\_width/2))  draw\_box(door\_x1, y2, door\_x1, y2+door\_height, door\_x1+door\_width, y2+door\_height, door\_x1+door\_width, y2)  draw\_points(door\_x1+door\_width-20, y2+int(door\_height/2))    #draw Window  window\_height = int(base/6)  window\_width = int(base/6)    window\_x1 = int(x3 - (base/4) - (window\_width/2))  window\_y1 = y2+door\_height+10  draw\_box(window\_x1, window\_y1, window\_x1, window\_y1+window\_height, window\_x1+window\_width, window\_y1+window\_height, window\_x1+window\_width, window\_y1)  draw\_line(window\_x1+window\_width/2, window\_y1, window\_x1+window\_width/2, window\_y1+window\_height,2)  def add\_rain\_point():  global rain\_drop\_count,rain\_points,time\_interval,last\_rain\_time  now = time.time()  if now > last\_rain\_time + time\_interval and len(rain\_points) < rain\_drop\_count:  print(len(rain\_points))  last\_rain\_time = now  rain\_points.append((random.randint(0,SCREEN\_W),random.randint(SCREEN\_H,SCREEN\_H+200), random.randint(30,60)))  def rain(speed):  global x,rain\_direction\_vertical, time,rain\_skew, time\_interval  if rain\_direction\_vertical == 'UP': speed[0] = -speed[0]  last\_time = time.time()  if time.time() > last\_time + 0.01 and time\_interval > 0.3:  time\_interval -= 0.5   for x,y,h in rain\_points:  new\_y = (y - speed[0])  new\_x = x + speed[1] \* (rain\_skew/abs(rain\_skew)) if rain\_skew != 0 else x   if rain\_direction\_vertical == "DOWN" and new\_y < SCREEN\_H/1.5:  new\_y = SCREEN\_H   elif rain\_direction\_vertical == "UP" and new\_y > SCREEN\_H:  new\_y = SCREEN\_H/1.5   if new\_x < 0:  new\_x = SCREEN\_W   elif new\_x > SCREEN\_W:  new\_x = 0   rain\_points[rain\_points.index((x,y,h))] = (new\_x,new\_y,h)  glutPostRedisplay()     def iterate():  glViewport(0, 0, SCREEN\_W, SCREEN\_H)  glMatrixMode(GL\_PROJECTION)  glLoadIdentity()  glOrtho(0.0, 1000, 0.0, 700, 0.0, 1.0)  glMatrixMode (GL\_MODELVIEW)  glLoadIdentity()  def draw\_rain(x,y,h):  global rain\_skew  if rain\_skew == 0:  draw\_line(x,y,x,y+h,2)  else:  x2 = int(math.atan(rain\_skew) \* h)  draw\_line(x,y,x-x2,y+h,2)  def showScreen():  global x,rain\_skew, rain\_drop\_count,day  add\_rain\_point()  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)   glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glMatrixMode(GL\_MODELVIEW)   glLoadIdentity()  iterate()  glColor3f(0.0, 0.0, 0.0) #konokichur color set (RGB)    glMatrixMode(GL\_MODELVIEW)   #call the draw methods here  glColor3f(0.0, 0.0, 1.0)  for x,y,h in rain\_points:  draw\_rain(x,y,h)    set\_day()  draw\_house(starting\_x=SCREEN\_W/2, starting\_y=600,base=400,height=200,roof\_base=500,roof\_height=150, scale=10)    glutSwapBuffers()  def init():  #//clear the screen  glClearColor(1,1,1,1);  #//load the PROJECTION matrix  glMatrixMode(GL\_PROJECTION)  #//initialize the matrix  glLoadIdentity()  #//give PERSPECTIVE parameters  gluPerspective(504, 1, 1, 1000.0)  # \*\*(important)\*\*aspect ratio that determines the field of view in the X direction (horizontally). The bigger this angle is, the more you can see of the world - but at the same time, the objects you can see will become smaller.  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\_line(x1,y1,x2,y2, w=5):  glLineWidth(w)  glBegin(GL\_LINES)  glVertex2f(x1,y1)  glVertex2f(x2,y2)  glEnd()  def draw\_box(x1,y1,x2,y2,x3,y3,x4,y4,w=5):  glLineWidth(w)  glBegin(GL\_LINES)    # left  glVertex2f(x1,y1)  glVertex2f(x2,y2)    #bottom  glVertex2f(x2,y2)  glVertex2f(x3,y3)    #right  glVertex2f(x3,y3)  glVertex2f(x4,y4)    #top  glVertex2f(x4,y4)  glVertex2f(x1,y1)    glEnd()  def draw\_quad(x1,y1,x2,y2,x3,y3,x4,y4):  glBegin(GL\_QUADS)  glVertex2f(x1,y1)  glVertex2f(x2,y2)  glVertex2f(x3,y3)  glVertex2f(x4,y4)  glEnd()   def draw\_triangle(starting\_x, starting\_y, base,height, hollow=False,scale=5, color=1.0):  x1,y1 = starting\_x, starting\_y  x2,y2 = int(x1 - (base/2)), y1 - height  x3,y3 = int(x1 + (base/2)), y1 - height  glBegin(GL\_TRIANGLE\_FAN)  glVertex2f(x1,y1) #left  glVertex2f(x2,y2) #top  glVertex2f(x3,y3) #right  glEnd()   if hollow:  center\_y = y1 - int(height/2)  scale = 1.2  base = int(base \* (1/scale) )  height = int(height \* (1/scale) )  y1 = center\_y + int(height/2)  x2,y2 = int(x1 - (base/2)), y1 - height  x3,y3 = int(x1 + (base/2)), y1 - height    # draw a smaller triangle inside the larger one  glColor3f(color, color, color)  glBegin(GL\_TRIANGLE\_FAN)    glVertex2f(x1,y1)   glVertex2f(x2,y2)   glVertex2f(x3,y3)   glEnd()   glColor3f(0.0, 0.0, 0.0)    # draw\_line(x1,y1,x2,y2,scale)  # draw\_line(x2,y2,x3,y3,scale)  # draw\_line(x1,y1,x3,y3,scale)  def main():  glutInit()  glutInitWindowSize(1000, 700) #window size  glutInitWindowPosition(0, 0)  glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGB)   wind = glutCreateWindow(b"OpenGL Coding Practice") #window name  init()   glutDisplayFunc(showScreen)  glutSpecialFunc(specialKeyListener)  glutKeyboardFunc(keyboardListener)  glutIdleFunc(lambda: rain(rain\_speed))   glutMainLoop()  if \_\_name\_\_ == '\_\_main\_\_':  main() |
| --- |





Task 2:

| from OpenGL.GL import \* from OpenGL.GLUT import \* from OpenGL.GLU import \* import random import time  # Global Variables SCREEN\_W = 500 SCREEN\_H = 500  points = [] colors = []  ballx = bally = 0 ball\_size = 5 speed= 0.01  create\_new= False bg\_color = (0,0,0)  blink\_duration = 1000 blinked = False last\_blinked = 0  frozen = False  def convert\_coordinate(x, y):  global SCREEN\_W,SCREEN\_H  a = x - (SCREEN\_W / 2)  b = (SCREEN\_H / 2) - y  return a,b  def draw\_points(points,s):  glPointSize(s)  for x, y, color in points:  glBegin(GL\_POINTS)  glColor3f(color[0],color[1],color[2])  glVertex2f(x,y)  glEnd()  def keyboardListener(key, x,y):  global ball\_size,frozen  if key == b' ':  frozen = not frozen  glutPostRedisplay()  def specialKeyListener(key,x, y):  global speed,frozen  if key == GLUT\_KEY\_UP:  speed \*= 2  if key == GLUT\_KEY\_DOWN:  speed = speed / 2  if speed < 0.05:  speed = 0.05  glutPostRedisplay()  def mouseListener(key, direction, x, y):  global points, frozen, blinked, last\_blinked, colors  if key == GLUT\_RIGHT\_BUTTON and direction == GLUT\_DOWN and frozen == False:  color = (random.uniform(0,1), random.uniform(0,1), random.uniform(0,1))  c\_X,c\_y = convert\_coordinate(x,y)  points.append((c\_X, c\_y, color))   if key == GLUT\_LEFT\_BUTTON and direction == GLUT\_DOWN and frozen == False:  blinked = not blinked  last\_blinked = time.time()  for i in range(len(points)):  color = points[i][2]  colors.append(color)  points[i]= (points[i][0], points[i][1], bg\_color)  def check\_blink():  global points, colors, blinked, last\_blinked  if time.time() > last\_blinked + 1 and blinked:  blinked = not blinked  for i in range(len(points)):  color = colors[i]  points[i] =(points[i][0], points[i][1], color)  glutPostRedisplay()  def showScreen():  global ball\_size  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glClearColor(0, 0, 0, 0)  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glMatrixMode(GL\_MODELVIEW)  glLoadIdentity()  gluLookAt(0, 0, 200, 0, 0, 0, 0, 1, 0)  glMatrixMode(GL\_MODELVIEW)    draw\_points(points,ball\_size)  check\_blink()  glutSwapBuffers()  def animate():  global frozen, points, speed  if not frozen:  for ballx, bally, color in points:  new\_ballx = ballx + random.choice([-10,10])\*speed  new\_bally = bally + random.choice([-10,10])\*speed  points[points.index((ballx,bally,color))] = (new\_ballx,new\_bally,color)  glutPostRedisplay()  def init():  glClearColor(0, 0, 0, 0)  glMatrixMode(GL\_PROJECTION)  glLoadIdentity()  gluPerspective(104, 1, 1, 1000.0)  def main():  glutInit()  glutInitWindowSize(SCREEN\_W, SCREEN\_H)  glutInitWindowPosition(0, 0)  glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGB)   wind = glutCreateWindow(b"OpenGL Coding")  init()  glutDisplayFunc(showScreen)  glutIdleFunc(animate)  glutKeyboardFunc(keyboardListener)  glutSpecialFunc(specialKeyListener)  glutMouseFunc(mouseListener)  glutMainLoop()  if \_\_name\_\_ == '\_\_main\_\_':  main() |
| --- |

