```
# Eric Liu
# Graphics

import pygame as pg
from math import sin, cos

WINDOW_SIZE=400
window = pg.display.set_mode((WINDOW_SIZE, WINDOW_SIZE))
clock = pg.time.Clock()

# projection matrix used to convert 3D to 2D (do not change!)
projection_matrix = [[1,0,0],
                     [0,1,0],
                     [0,0,0]]

# makes a list of size 8 for the 8 points in a cube
# enter all the vertices of our cube into each index(do not change!)
cube_points = [n for n in range(8)]
cube_points[0] = [[-1],[-1],[1]]
cube_points[1] = [[1],[-1],[1]]
cube_points[2] = [[1],[1],[1]]
cube_points[3] = [[-1],[1],[1]]
cube_points[4] = [[-1],[-1],[-1]]
cube_points[5] = [[1],[-1],[-1]]
cube_points[6] = [[1],[1],[-1]]
cube_points[7] = [[-1],[1],[-1]]

# Let's do matrix multiplication! This allows us to convert our 3d points into a 2d
array
def multiply_m(a, b):
  a_rows = len(a)
  a_cols = len(a[0])

  b_rows= len(b)
  b_cols= len(b[0])
  results=[[0 for _ in range(b_cols)] for _ in range(a_rows)]

  if a_cols == b_rows:
    for i in range(a_rows):
      for j in range(b_cols):
        for k in range(b_rows):
          results[i][j] += a[i][k] * b[k][j]
  else:
    print("incomplete matrix size")
  return results
```

```
def connect_points(i, j, points):
  pg.draw.line(window, (255, 255, 255), (points[i][0], points[i][1]),(points[j][0],
points[j][1]))



# to view our work
scale = 100
angle_x = angle_y = angle_z = 0
while True:
  clock.tick(60)
  window.fill((0,0,0))
  rotation_x = [[1, 0, 0],
               [0, cos(angle_x), -sin(angle_x)],
               [0, sin(angle_x), cos(angle_x)]]

  rotation_y = [[cos(angle_y), 0, sin(angle_y)],
               [0, 1, 0],
               [-sin(angle_y), 0, cos(angle_y)]]

  rotation_z = [[cos(angle_z), -sin(angle_z), 0],
               [sin(angle_z), cos(angle_z), 0],
               [0, 0, 1]]

  angle_x+= 0.01
  angle_y+= 0.01
  angle_z+= 0.03



  points = [0 for _ in range(len(cube_points))]
  i=0

  for point in cube_points:
    rotate_x = multiply_m(rotation_x, point)
    rotate_y = multiply_m(rotation_y, rotate_x)
    rotate_z = multiply_m(rotation_z, rotate_y)
    point_2d = multiply_m(projection_matrix, rotate_z)
    # point_2d = multiply_m(projection_matrix, point)

    x= (point_2d[0][0] * scale) + WINDOW_SIZE/2
    y= (point_2d[1][0] * scale) + WINDOW_SIZE/2

    points[i]=(x,y)
    i += 1
    pg.draw.circle(window, (255, 0, 0), (x,y), 5)

# HW- Complete connectind the lines
```

```
connect_points(0, 1, points)
connect_points(1, 2, points)
connect_points(1, 3, points)
connect_points(2, 1, points)
connect_points(2, 2, points)
connect_points(0, 4, points)
connect_points(5, 2, points)
connect_points(4, 7, points)
connect_points(5, 4, points)
connect_points(7, 6, points)
connect_points(6, 2, points)
connect_points(5, 3, points)
connect_points(3, 4, points)
connect_points(2, 6, points)
connect_points(3, 2, points)
connect_points(5, 6, points)
connect_points(4, 6, points)
connect_points(7, 5, points)
connect_points(7, 4, points)








# don't change, allows you to quit preview
for event in pg.event.get():
  if event.type==pg.QUIT:
    pg.quit()
pg.display.update()
```