**Завдання**

1. Обрати середовище розробки.

Мною була обрана мова програмування Python, тому що:

* Це доступна мова, якак набирає свою популярність;
* Вона підходить для розробки веб-додтків, мобільних додатків, descktop додатків та інтегрува цого у системи;
* Велика кулькість бібліотек для роботи з графічними данними та файлами.

1. Обрати середовище розробки.

Мною болі обрано середовище розробки PyCharm:

* Це середовище призначенно для роботи с Python;
* Також вона підтримує можливість видкривати файли будь-яких інших мов програмування;
* Існує підтримка проетів та система управління версіями.

1. Обрати бібліотеки для роботи з данними.

Мною були обрані такі бібліотеки для роботи з данними:

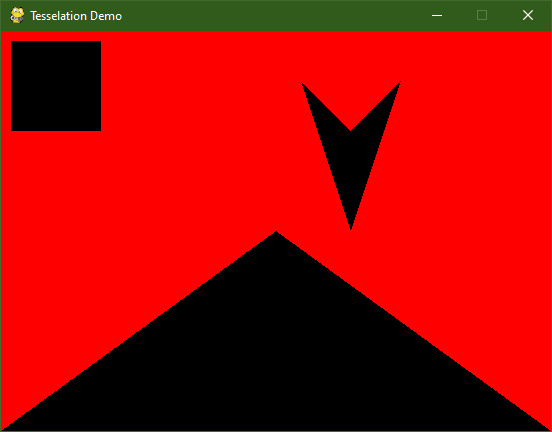
* pydicom
* бібліотека що використовується для обробки медичних зображень DICOM;
* дозволяє читати, змінбвати та записувати данні DICOM;
* працює будь-де де працює Python та не має спецільних вимог.
* PyOpenGL
* дозволяє працювати з функціми OpenGL, CLU та CLUT, а також з розширенням OpenGL;
* використовується для роботи з 3D зображеннями;
* широкий функціонал для роботи з шейдерами та буферними об’єктами.

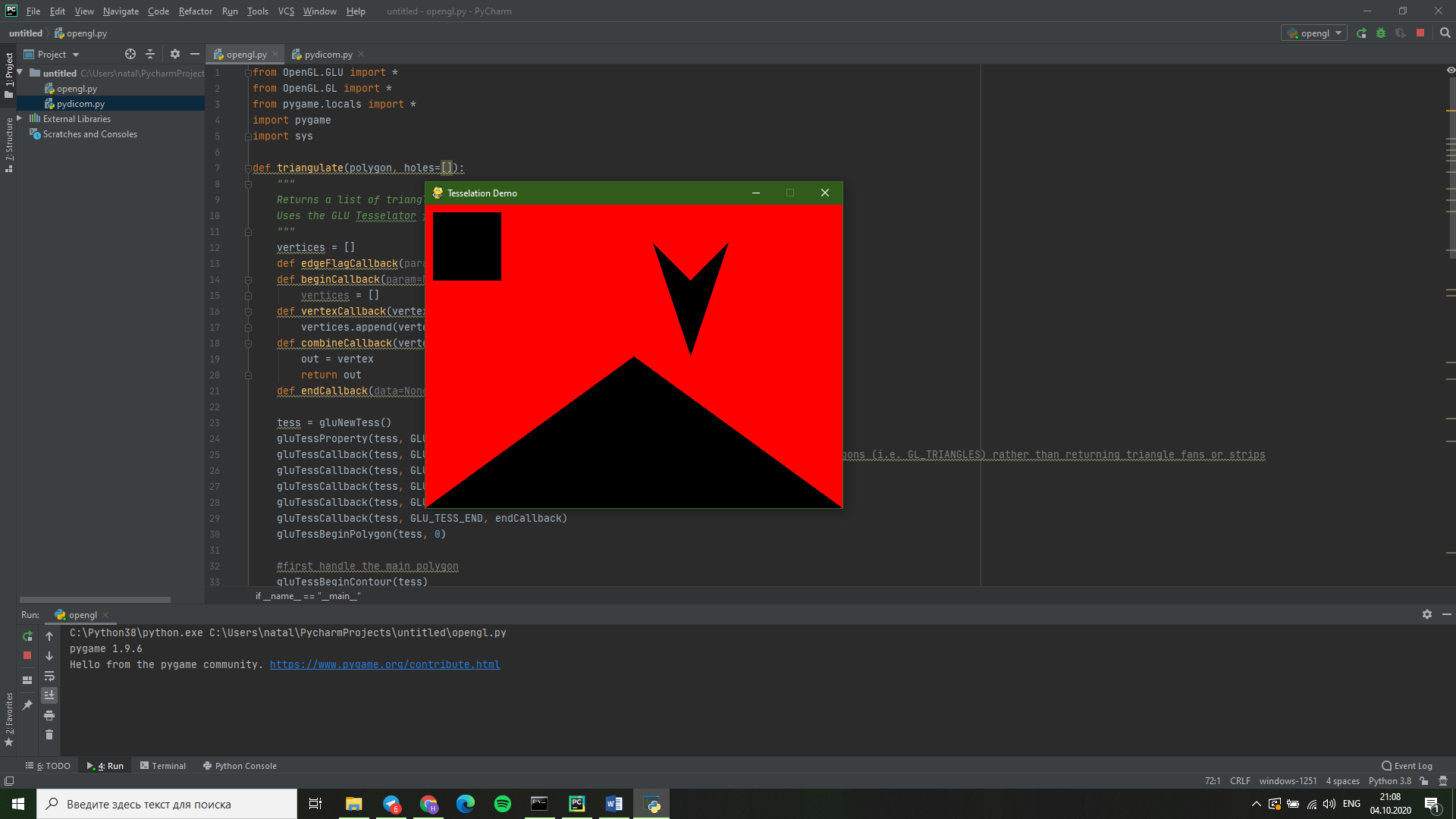
**Лістинг программи**

***Використання OpenGL***

from OpenGL.GLU import \*  
from OpenGL.GL import \*  
from pygame.locals import \*  
import pygame  
import sys  
  
def triangulate(polygon, holes=[]):  
 *"""  
 Returns a list of triangles.  
 Uses the GLU Tesselator functions!  
 """* vertices = []  
 def edgeFlagCallback(param1, param2): pass  
 def beginCallback(param=None):  
 vertices = []  
 def vertexCallback(vertex, otherData=None):  
 vertices.append(vertex[:2])  
 def combineCallback(vertex, neighbors, neighborWeights, out=None):  
 out = vertex  
 return out  
 def endCallback(data=None): pass  
  
 tess = gluNewTess()  
 gluTessProperty(tess, GLU\_TESS\_WINDING\_RULE, GLU\_TESS\_WINDING\_ODD)  
 gluTessCallback(tess, GLU\_TESS\_EDGE\_FLAG\_DATA, edgeFlagCallback)#forces triangulation of polygons (i.e. GL\_TRIANGLES) rather than returning triangle fans or strips  
 gluTessCallback(tess, GLU\_TESS\_BEGIN, beginCallback)  
 gluTessCallback(tess, GLU\_TESS\_VERTEX, vertexCallback)  
 gluTessCallback(tess, GLU\_TESS\_COMBINE, combineCallback)  
 gluTessCallback(tess, GLU\_TESS\_END, endCallback)  
 gluTessBeginPolygon(tess, 0)  
  
 #first handle the main polygon  
 gluTessBeginContour(tess)  
 for point in polygon:  
 point3d = (point[0], point[1], 0)  
 gluTessVertex(tess, point3d, point3d)  
 gluTessEndContour(tess)  
  
 #then handle each of the holes, if applicable  
 if holes != []:  
 for hole in holes:  
 gluTessBeginContour(tess)  
 for point in hole:  
 point3d = (point[0], point[1], 0)  
 gluTessVertex(tess, point3d, point3d)  
 gluTessEndContour(tess)  
  
 gluTessEndPolygon(tess)  
 gluDeleteTess(tess)  
 return vertices  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 width, height = 550, 400  
 pygame.init()  
 pygame.display.set\_mode((width, height), DOUBLEBUF|OPENGL)  
 pygame.display.set\_caption("Tesselation Demo")  
 clock = pygame.time.Clock()  
 glClear(GL\_COLOR\_BUFFER\_BIT)  
 glClear(GL\_DEPTH\_BUFFER\_BIT)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 glOrtho(0, width, height, 0, -1, 1)#flipped so top-left = (0, 0)!  
 glMatrixMode(GL\_MODELVIEW)  
 glLoadIdentity()  
  
 #define the polygon and some holes  
 polygon = [(0, 0), (550, 0), (550, 400), (275, 200), (0, 400)]  
 hole1 = [(10, 10), (10, 100), (100, 100), (100, 10)]  
 hole2 = [(300, 50), (350, 100), (400, 50), (350, 200)]  
 holes = [hole1, hole2]  
 vertices = triangulate(polygon, holes=holes)  
  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 glColor(1, 0, 0)  
 glBegin(GL\_TRIANGLES)  
 for vertex in vertices:  
 glVertex(\*vertex)  
 glEnd()  
  
 pygame.display.flip()  
 clock.tick\_busy\_loop(60)

Результат

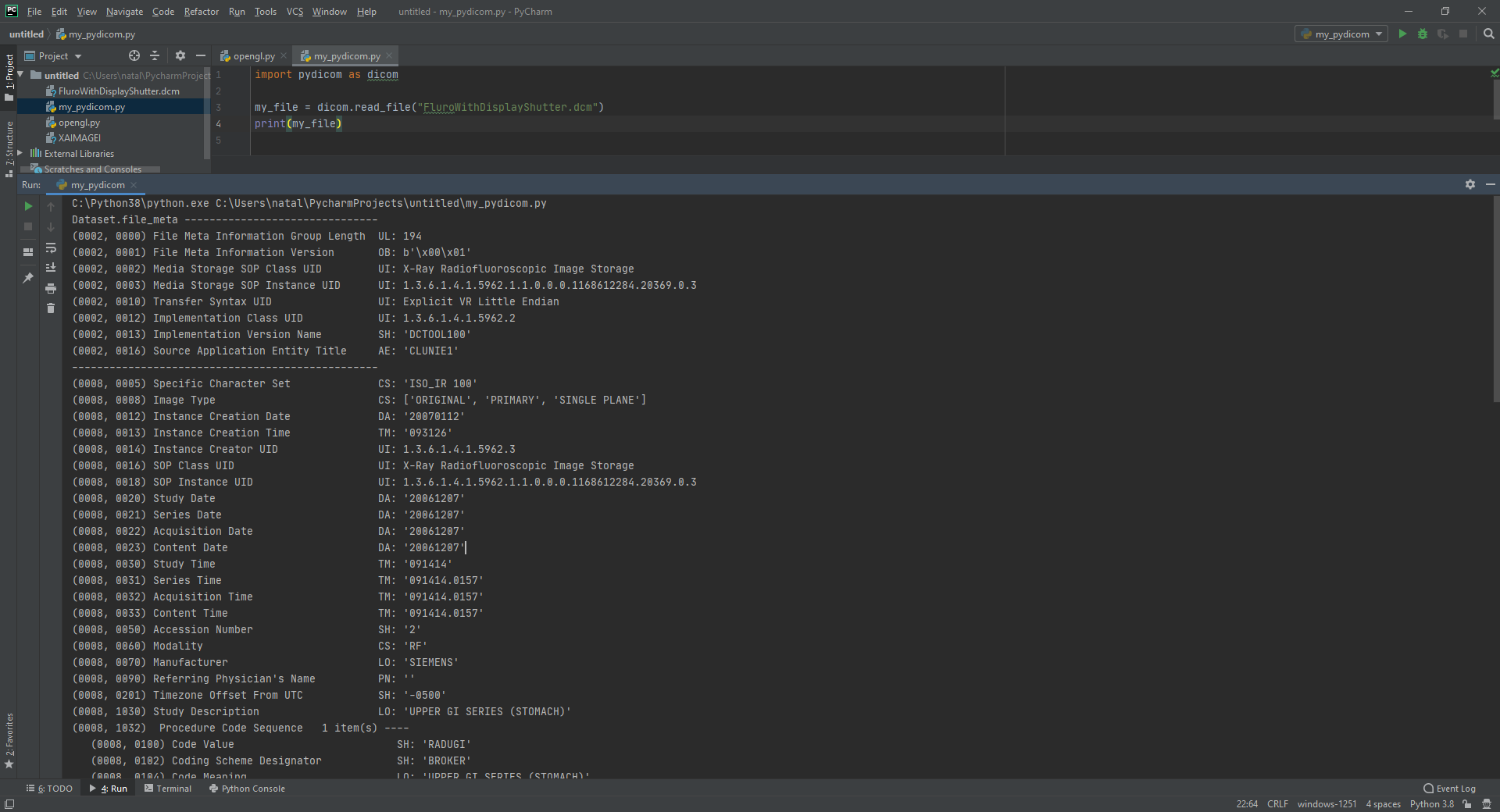




***Використання pydicom***

import pydicom as dicom  
  
my\_file = dicom.read\_file("FluroWithDisplayShutter.dcm")  
print(my\_file)

Результат



**Контрольні запитання**

1. Коротко опишіть архітектуру бібліотеки OpenGL та організацію конвеєра графічних перетворень.

Архітектура данної бібліотеки реалізує схему конвеєра, що має кілька єтапів обробки графічних данних, які виконуються послідовно.

Апроксимація кривих і поверхонь → Обробка вершин і складання примітивів (атрибути вершин та джерела світка) → Растеризація і обробка фрагментів (текстури) → Операції над пікселями → Передача даних в буфер кадру.

Команди виконуються в порядку їх надходження.

1. Назвіть категорії базових команд (функцій) бібліотеки.

Базові команди (функції) можна розділити на п’ять категорій:

* *Фукнції опису примітивів* (визначають об’єкти нижнього рівня ієрархії (примітивів), які здатні відображати графічна система. Для OpenGL примітивними є точки, лінії, багатокутники и т.д. Примітиви визначають що з’явиться на екрані)
* *Функції опису джерела* (використовуються для опису положення та параметрів джерела світла, розташованих у тривімірній сцені)
* *Функції завдання атрибутів* (атрибути відповідають за спосіб виведення об’єкта на екран)
* *Функції візуалізації* (дозволяються задати положення спостерігачу у віртувальному просторі)
* *Функції геометричних перетворень* (дозволяють виконувати різноманітні перетворення об’єктів)

1. Навіщо потрібні різні варіанти команд OpenGL, що відрізняються тільки типами параметрів?

Через те що інтерфейс OpenGL не розрахований на конкретну мову програмування і він повинен бути максимально універсальним. Звичайно можна було б цього позбути використовуючи перевантаження операцій мови С++.

1. Чому організація OpenGL часто поріснюється із скінченним автоматом?

Тому що, уся інформаія про вигляд фігури (значення спеціальних змінних і значення поточної нормалі, кольору, коортинати текстури й інших атрибутів) буди виконистана під час надходження координат вершини для побудованої фігури у графічну систему. Зміна станів також буде відбуватися за допомогою команд, що оформлюються як виклик функцій.