**Завдання**

**Завдання**:

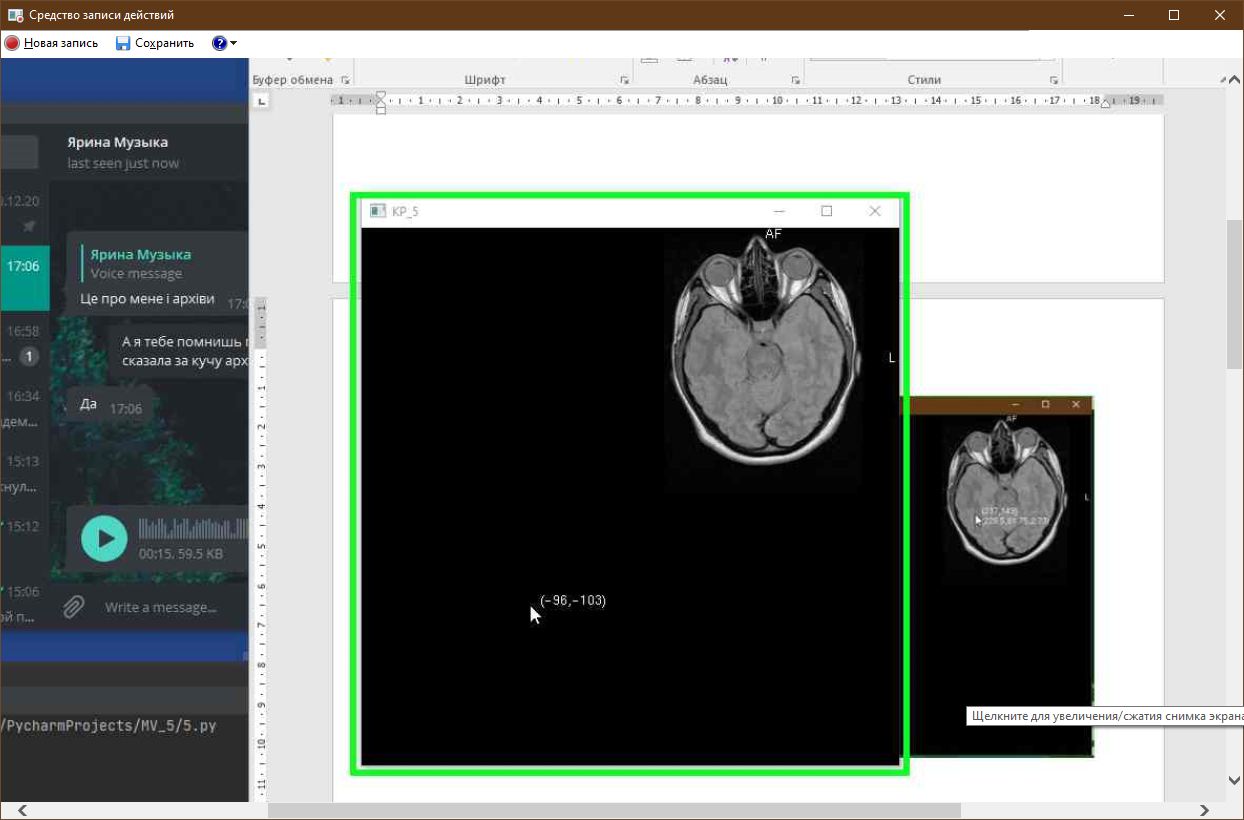
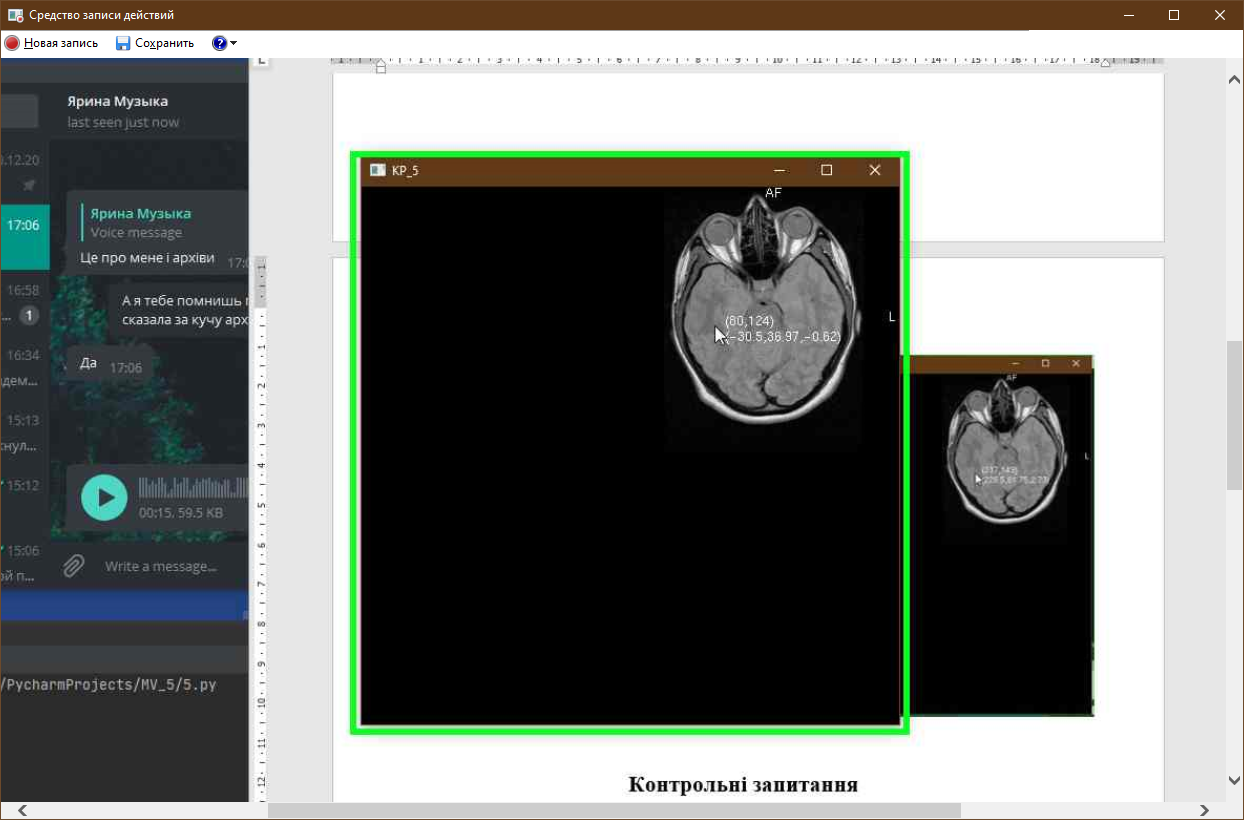
1. Вивчити теоретичні основи щодо визначення фізичних розмірів та просторовго положення медичних томографічних зображень.
2. Розробити програмний застосунок для завантаження медичного зображення (томографічного зрізу) в форматі DICOM та визначення тривимірних координат пікселів та просторового положення площини зображення.
3. Завантажити зображення томографічного зрізу в форматі DICOM в оперативну пам’ять та відобразити дані зображення у вікні програми: одному пікселу зображення відповідає один піксел екрана: розміри програмного вікна застосунку (графічна частина) мають бути вдвічі більше відповідних розмірів завантаженого медичного зображення; відображення даних пікселів зображення має бути виконане відповідно до схеми, що зазначена в варіанті завдання; початок координат має бути розташований в центрі.
4. Реалізувати інтерактивне текстове відображення у вікні програмного застосунку значень координат пікселів, що розташовані під курсором миші: під час переміщення миші на даних графічної частини мають відображатися координати в заданій в програмі системі відображення (початок координат – центр графічної частини програмного застосунку), під час переміщення миші на даних завантаженого зображення мають додатково відображатись тривимірні координати в заданій під час сканування системі (початок координат – ізоцентр сканера).
5. Реалізувати відображення просторового положення зображення в формі абревіатур необхідних літер (або їх комбінації) на зображенні під час перегляду.
6. Скласти і захистити звіт по роботі.

| **Номер варіанту** | **Схема відображення зображення** | **Схема розташування зображення** |
| --- | --- | --- |
| 2 |  |  |

**Лістинг программи**

from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
import pydicom  
import numpy as np  
import math  
from pydicom.uid import generate\_uid  
  
path\_file = "copy of DICOM\_Image\_for\_Lab\_2.dcm"  
  
  
class Image:  
 def \_\_init\_\_(self, path):  
 self.ds = pydicom.read\_file(path)  
 self.image\_pixels = np.array(self.ds.pixel\_array)  
  
 self.image\_position = np.array(self.ds[0x0200032].value)  
 self.X\_orientation = np.array(self.ds[0x0200037].value[:3])  
 self.Y\_orientation = np.array(self.ds[0x0200037].value[3:6])  
 self.letter\_X = self.check\_letters(self.X\_orientation[0], self.X\_orientation[1], self.X\_orientation[2])  
 self.letter\_Y = self.check\_letters(self.Y\_orientation[0], self.Y\_orientation[1], self.Y\_orientation[2])  
 self.patient\_orientation = self.ds[0x0200020].value  
  
 self.width, self.height = self.ds[0x280010].value, self.ds[0x280011].value  
 self.isTextVisible = False  
 self.mouse\_on\_image = False  
 self.x\_coord, self.y\_coord = 0, 0  
  
 def init(self):  
 glClearColor(0, 0, 0, 0.0)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 gluOrtho2D(-self.width, self.width, -self.height, self.height)  
  
 def check\_letters(self, cos\_1, cos\_2, cos\_3):  
 if cos\_1 == 1 and cos\_2 == cos\_3 == 0:  
 return 'L'  
 elif cos\_1 == cos\_3 == 0 and cos\_2 == 1:  
 return 'A'  
 elif cos\_1 == cos\_2 == 0 and cos\_3 == 1:  
 return 'H'  
 elif cos\_1 == 0 and cos\_2 > math.cos(45) > cos\_3:  
 return 'AF'  
 elif cos\_1 == 0 and cos\_2 > math.cos(45) and cos\_3 > math.cos(45):  
 return 'AH'  
 elif cos\_1 == 0 and cos\_2 < math.cos(45) < cos\_3:  
 return 'PH'  
 elif cos\_1 == 0 and cos\_2 < math.cos(45) and cos\_3 < math.cos(45):  
 return 'PF'  
  
 def display(self):  
 glClear(GL\_COLOR\_BUFFER\_BIT)  
 glColor3f(1, 1, 1)  
  
 self.draw\_texture()  
  
 self.print\_text(self.width - 10, self.height / 2, GLUT\_BITMAP\_HELVETICA\_12, self.letter\_X)  
 self.print\_text(self.width / 2, self.height - 10, GLUT\_BITMAP\_HELVETICA\_12, self.letter\_Y)  
 if self.isTextVisible == True:  
 self.print\_text(self.x\_coord - self.width + 10, -self.y\_coord + self.height, GLUT\_BITMAP\_HELVETICA\_12, "(" + str(self.x\_coord - self.width) + "," + str(self.height - self.y\_coord) + ")")  
 if self.mouse\_on\_image:  
 pixel\_coord = self.coord\_3D()  
 self.print\_text(self.x\_coord - self.width + 10, -self.y\_coord + self.height - 15, GLUT\_BITMAP\_HELVETICA\_12, "(" + str(round(pixel\_coord[0], 2)) + "," + str(round(pixel\_coord[1], 2)) + "," + str(round(pixel\_coord[2], 2)) + ")")  
  
 glutSwapBuffers()  
  
 def print\_text(self, x, y, font, line):  
 glColor3f(1, 1, 1)  
 glPushAttrib(GL\_DEPTH\_TEST)  
 glRasterPos2d(x, y)  
 for i in line:  
 glutBitmapCharacter(font, ord(i))  
 glPopAttrib()  
  
 def draw\_texture(self):  
 glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_LUMINANCE, self.width, self.height, 0, GL\_LUMINANCE, GL\_UNSIGNED\_BYTE, self.image\_pixels)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)  
 glEnable(GL\_TEXTURE\_2D)  
  
 glBegin(GL\_QUADS)  
 glTexCoord2d(0.0, 0.0)  
 glVertex2d(self.width, self.height)  
 glTexCoord2d(1.0, 0.0)  
 glVertex2d(0.0, self.height)  
 glTexCoord2d(1.0, 1.0)  
 glVertex2d(0.0, 0.0)  
 glTexCoord2d(0.0, 1.0)  
 glVertex2d(self.width, 0.0)  
 glEnd()  
 glDisable(GL\_TEXTURE\_2D)  
  
 def motion\_func(self, x, y):  
 self.isTextVisible = True  
 if x > self.width and y < self.height:  
 self.mouse\_on\_image = True  
 else:  
 self.mouse\_on\_image = False  
 self.x\_coord, self.y\_coord = x, y  
 self.display()  
  
 def coord\_3D(self):  
 c, r = self.x\_coord - self.width, self.height - self.y\_coord  
 pixel\_coord = self.image\_position + c \* self.X\_orientation + r \* self.Y\_orientation  
 return pixel\_coord  
  
  
def reshape(width, height):  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - width) // 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - height) // 2)  
  
  
def init\_window(width, height):  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - width) // 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - height) // 2)  
 glutCreateWindow('KP\_5')  
 glutReshapeWindow(width, height)  
  
  
def main():  
 glutInit()  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
 file = Image(path\_file)  
 init\_window(file.width \* 2, file.height \* 2)  
 file.init()  
 glutDisplayFunc(file.display)  
 glutPassiveMotionFunc(file.motion\_func)  
 glutMainLoop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат

**Контрольні запитання**

1. *Що таке ізоцентр сканера, в який чин він визначається?*

Точка яка знаходиться в середині магніту, з координатами . У цій точці магнітне поле має напруженість та резонансну частоту .

1. *Як розрахувати тривимірні координати певного піксела зображення, який розташований в r-му рядку та в c-му стовпчику?*

Для розрахунку тривимірної координати пікселя зображення *P*, який розташований в r-му рядку та в c-му стовпчику, можна використати формулу:

1. *Як розрахувати відстань до ізоцентра сканера від певного піксела зображення, який розташований в r-му рядку та в c-му стовпчику?*

В більшості випадків початок координат визначається лазерними пристроями для зовнішнього вирівнювання положення пацієнта на столі перед початком роведення дослідження. Такий центр знаходиться у початку координат «відносної системи координат» (*RCS* – *Reference Coordinates System*), яка визначає просторове положення зображення щодо тіла людини під час роведення сканування.

Ми знаемо що ізоцеентр сканера знаходиться у точці , а для визначення тривимірної координати будь якого пікселя зображення використовується формула . Отже розрахувавши за формулою тривимірні координати необхідного піксемя, ми можемо знайти його відстань від центра (PC):

1. *За якими елементами DICOM визначається інформація щодо фізичних розмірів відсканованого об’єму?*

Інформація стосовно фізичних розмірів зображення в міліметрах зберігаєтсья у елементі *0028,0030 “Pixel Spacing”*. Якщо у елемнті знаходяться значення 0.5мм для та 0.5мм для , то для довжини в 10 пікселів відповідна фізична довжина рівна 5мм. Знаючи роздільність зображення в пікселях, можна визначити його фізичні розміри (область перегляду, *FoV*).

1. *Чому не можна визначати фізичний розмір відсканованого об'єму за третім виміром, тобто вздовж осі сканування, за інформацією, яка зберігається в елементі DICOM 0018.0050 "Slice Thickness”.*

В елементі *0018,0050 "Slice Thickness”* зберігається інформація щодо товщини томографічного зрізу в міліметрах. Проте на реальних данних відстань між зрізами може не дорівнювати товщині самих зрізів, вона може бути меньша або більша. Через це використовувати його для визначення розміру відсканованого об’єкту недоцільно.

1. *За якими елементами DICOM визначається інформація щодо просторового положення медичного томографічного зображення?*

Просторове положення зображення можна визначити за двома елементами:

* *0020,0032 “Image Position”* – зберігає інформацію щодо координат лівого верхнього кута (першого пікселя) зображення в міліматрах, що допомагає визначити положення у тривимірному просторі.
* *0020,0037 “Imae Orientation”* – зберігає напрямні косинуси тривіальних векторів напрямку рядків (перші три значення елемента) та стовпців (наступні три).

1. *Визначити напрямні косинуси тривимірних векторів напрямку рядків vr та стовпців v*с *зображення для косої корональної проекції у випадку її нахилу до трансверсальної площини (окремо для кожного випадку SA, SP, IA, IP).*

*SA* (0, 1, 0.5)

*SP* (0, 1, -0.5)

*IA* (0, 1, 0.5)

*IP* (0, 1, -0.5)

1. *Схематично зобразити необхідні літери (або їх комбінацію) для відображення просторової орієнтації зображення, для якого напрямні косинуси, що зберігаються в значенні елементу 0020,0037 “Image Orientation", мають значення 0.5\0\0.8660254\0\1\0.*



1. *За яким елементом D1COM визначається інформація щодо положення пацієнта на столі під час проведення дослідження?*

Елемент *0018,5100 “Pation Position”* зберігає інформацію щодо положення пацієнта відносно сканера.

1. *Визначити напрямні косинуси тривимірних векторів напрямку рядків v*с *та стовпців vr зображення для восьми можливих положень пацієнта під час сканування: HFP (Head First Prone), HFS (Head First - Supine), HFDR (Head First — Decubitus Right), HFDL (Head First - Decubitus Left), FFDR (Feet First - Decubitus Right), FFDL (Feet First Decubitus Left), FFP (Feet First Prone), FFS (Feet First Supine).*

| *HFP* | *vr=*(-1, 0, 0) | *vc=*(0, -1, 0) |
| --- | --- | --- |
| *HFS* | *vr=*(1, 0, 0) | *vc=*(0, 1, 0) |
| *HFDR* | *vr=*(0, 1, 0) | *vc=*(-1, 0, 0) |
| *HFDL* | *vr=*(0, -1, 0) | *vc=*(1, 0, 0) |
| *FFDR* | *vr=*(0, -1, 0) | *vc=*(-1, 0, 0) |
| *FFDL* | *vr=*(0, 1, 0) | *vc=*(1, 0, 0) |
| *FFP* | *vr=*(1, 0, 0) | *vc=*(0, -1, 0) |
| *FFS* | *vr=*(-1, 0,0) | *vc=*(0, 1, 0) |