**Завдання**

**Завдання**:

1. Ознайомитися з теоретичними відомостями реалізації алгоритмів сегментації медичних зображень, що засновані на пошуку границь.
2. Розробити програмний додаток для завантаження медичного зображення в форматі DICOM та виконання складних двовимірних геометричних перетворень.
3. Розміри частини вінка програмного додатка для візуалізації графічних даних (без інтерфейсу користувача) мають бути вдвічі більше відповідно розмірів завантаженого зображення.
4. Задати параметри спостереження таким чином, щоб у вікні програмного додатку були відображені три координатні вісі та весь тривимірний об’єкт (початок координати має бути розташований в центрі частини вікна програмного додатка, що відведена для візуалізації графічних даних).
5. Із серії зображення плоско-паралельних томографічних зрізів створити об’єкт багато площинного перетворення та розмістити його в першому октанті тривимірного простору (***x***, ***y*** та ***z*** координати об’єкта мають додатні значення).
6. Створити події від клавіатури, при обробці яких можна окремо виконати інтерактивну модифікацію тривимірного об’єкта, а також задане у варіанті геометричне перетворення тривимірного об’єкта в просторі та відновити його оригінальне відображення.
7. Надати можливість інтерактивної модифікації тримірного об’єкта за допомогою генерації нового зображення зрізу в заданій площині та підстановки згенерованих даних у відповідне положення (площину використати відповідно до заданих у варіанті).
8. Виконати геометричне перетворення об’єкта в просторі, при цьому параметри перетворень (вектор трансляції, кут повороту та зсуву, коефіцієнти масштабування та рівняння ліній для віддзеркалення) задавати інтерактивно (тривимірне геометричне перетворення об’єкта використати відповідно до заданого в варіанті).
9. Скласти і захистити звіт по роботі.

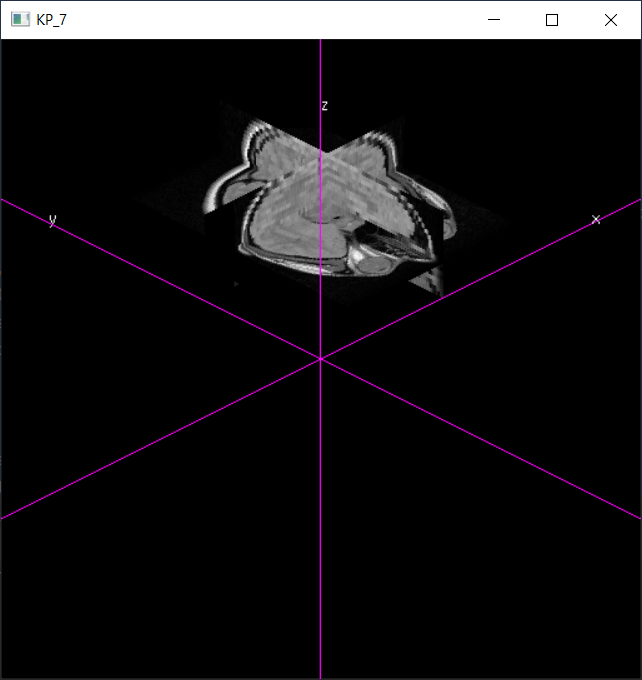
| **Номер варіанту** | **Тривимірне геометричне перетворення об’єкта** | **Площина модифікації** |
| --- | --- | --- |
| 8 | Тривимірне віддзеркалення відносно координатної площини ***zOx*** | корональна |

**Лістинг программи**

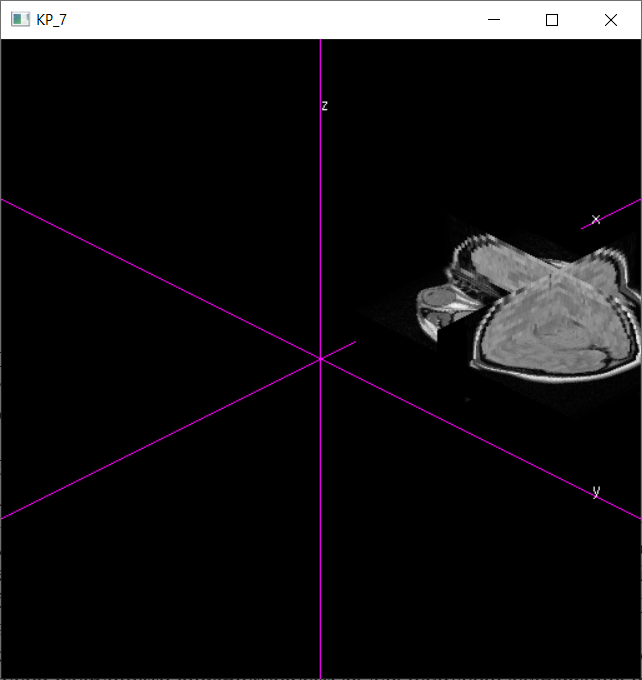
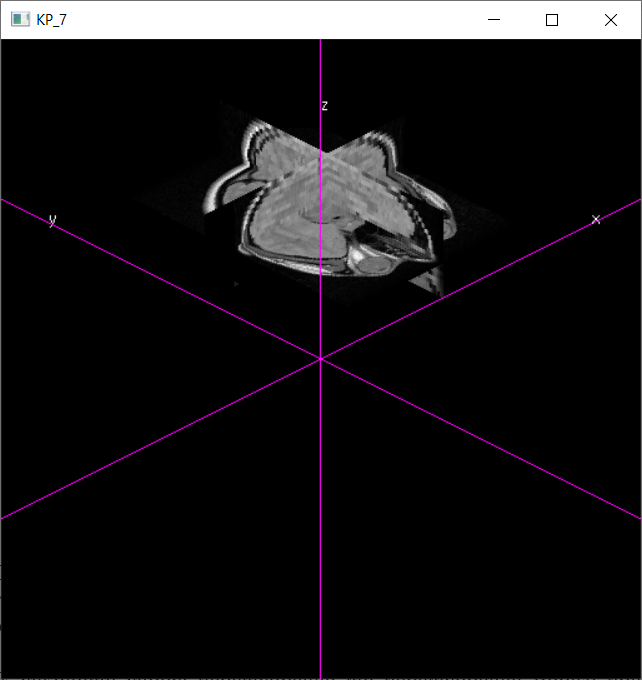
from OpenGL.GL import \*  
from OpenGL.GLUT import \*  
import pydicom  
import numpy as np  
  
path\_dir = "Lab#7"  
  
  
class Image:  
 def \_\_init\_\_(self, path):  
 self.num = len([file\_name for file\_name in os.listdir(path) if os.path.isfile(os.path.join(path, file\_name))])  
  
 self.width, self.height = 256, 256  
 self.image\_pixels = np.zeros((self.num, self.height, self.width))  
 self.front = np.zeros((self.height, self.num + 12, self.width))  
 self.side = np.zeros((self.width, self.num + 12, self.height))  
  
 self.image\_pixels = []  
  
 for file\_name in os.listdir(path):  
 self.ds = pydicom.read\_file('Lab#7/' + file\_name)  
 self.image\_pixels.append(self.normalization\_func(self.ds.pixel\_array, 0, 255))  
  
 for i in range(self.height):  
 for j in range(self.num):  
 for n in range(self.width):  
 self.front[i][j][n] = self.image\_pixels[j][i][n]  
  
 for i in range(self.width):  
 for j in range(self.num):  
 for n in range(self.height):  
 self.side[i][j][n] = self.image\_pixels[j][n][i]  
  
 self.t, self.t\_front, self.t\_side = 0, 0, 0  
  
 def init(self):  
 glClearColor(0, 0, 0, 0.0)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 glOrtho(-1, 1, -1, 1, -1, 1)  
 glEnable(GL\_DEPTH\_TEST)  
 glMatrixMode(GL\_MODELVIEW)  
 glRotatef(-60, 1, 0, 0)  
 glRotatef(45, 0, 0, 1)  
  
 def display(self):  
 glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)  
  
 self.draw\_coordinates()  
 self.draw\_texture(self.t, self.t\_front, self.t\_side)  
  
 glutSwapBuffers()  
   
 # малює координатну площину  
 def draw\_coordinates(self):  
 glColor3f(1, 0, 1)  
 glBegin(GL\_LINES)  
 glVertex3f(-2.0, 0.0, 0.0)  
 glVertex3f(2.0, 0.0, 0.0)  
 glVertex3f(0.0, -2.0, 0.0)  
 glVertex3f(0.0, 2.0, 0.0)  
 glVertex3f(0.0, 0.0, -2.0)  
 glVertex3f(0.0, 0.0, 2.0)  
 glEnd()  
 self.print\_text(1.2, 0, 0, GLUT\_BITMAP\_HELVETICA\_12, "x")  
 self.print\_text(0, 1.2, 0, GLUT\_BITMAP\_HELVETICA\_12, "y")  
 self.print\_text(0, 0, 0.9, GLUT\_BITMAP\_HELVETICA\_12, "z")  
   
 # малює три площини зображення  
 def draw\_texture(self, t, t\_front, t\_side):#data, internal\_format, type\_transform):  
 self.default(self.image\_pixels[t], self.height, self.width)  
 # основна(нижня)  
 glEnable(GL\_TEXTURE\_2D)  
 glBegin(GL\_QUADS)  
 glTexCoord2f(0, 0)  
 glVertex3f(0, 0, t \* 4 / self.height)  
 glTexCoord2f(1, 0)  
 glVertex3f(1, 0, t \* 4 / self.height)  
 glTexCoord2f(1, 1)  
 glVertex3f(1, 1, t \* 4 / self.height)  
 glTexCoord2f(0, 1)  
 glVertex3f(0, 1, t \* 4 / self.height)  
 glEnd()  
  
 self.default(self.side[t\_front], self.height, 32)  
 # передня  
 glBegin(GL\_QUADS)  
 glTexCoord2f(0, 0)  
 glVertex3f(t\_front / self.height, 0, 0)  
 glTexCoord2f(1, 0)  
 glVertex3f(t\_front / self.height, 1, 0)  
 glTexCoord2f(1, self.num / 32)  
 glVertex3f(t\_front / self.height, 1, self.num \* 4 / self.height)  
 glTexCoord2f(0, self.num / 32)  
 glVertex3f(t\_front / self.height, 0, self.num \* 4 / self.height)  
 glEnd()  
  
 self.default(self.front[t\_side], self.width, 32)  
 # бокова  
 glBegin(GL\_QUADS)  
 glTexCoord2f(0, 0)  
 glVertex3f(0, t\_side / self.height, 0)  
 glTexCoord2f(1, 0)  
 glVertex3f(1, t\_side / self.height, 0)  
 glTexCoord2f(1, self.num / 32)  
 glVertex3f(1, t\_side / self.height, self.num \* 4 / self.height)  
 glTexCoord2f(0, self.num / 32)  
 glVertex3f(0, t\_side / self.height, self.num \* 4 / self.height)  
 glEnd()  
  
 glDisable(GL\_TEXTURE\_2D)  
 glFlush()  
  
 def default(self, data, height, width):  
 glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_LUMINANCE, height, width, 0, GL\_LUMINANCE, GL\_UNSIGNED\_BYTE, data)  
 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)  
  
 def print\_text(self, x, y, z, font, line):  
 glColor3f(1, 1, 1)  
 glPushAttrib(GL\_DEPTH\_TEST)  
 glRasterPos3f(x, y, z)  
 for i in line:  
 glutBitmapCharacter(font, ord(i))  
 glPopAttrib()  
  
 # функція виконує нормальзацію  
 def normalization\_func(self, pixels, p\_min, p\_max):  
 min\_pixel = np.min(pixels)  
 max\_pixel = np.max(pixels)  
 normalization = (pixels - min\_pixel) / (max\_pixel - min\_pixel) \* (p\_max - p\_min) + p\_min  
 return normalization.astype(int)  
   
 # матриця перетворення для віддзеркалення відносно осі zOx  
 @staticmethod  
 def get\_transform\_matrix():  
 matrix = np.array([  
 1, 0, 0, 0,  
 0, -1, 0, 0,  
 0, 0, 1, 0,  
 0, 0, 0, 1,  
 ])  
  
 return matrix  
  
 def keyboard\_func(self, my\_key, x, y):  
 key = unicode(my\_key, errors='ignore')  
 if key == "t": # виконує віддзеркалення  
 transform\_matrix = self.get\_transform\_matrix()  
 glMultMatrixf(transform\_matrix)  
 elif key == "w" and self.t < self.num - 1: # підіймає основну площину  
 self.t += 1  
 elif key == "s" and self.t > 0: # опускає основну площину  
 self.t -= 1  
 elif key == "d" and self.t\_front < self.width - 1: # рухає в назад передню площину  
 self.t\_front += 1  
 elif key == "a" and self.t\_front > 0: # рухає вперед передню площину  
 self.t\_front -= 1  
 elif key == "," and self.t\_side < self.height - 1: # рухає вправо бокову площину  
 self.t\_side += 1  
 elif key == "." and self.t\_side > 0: # рухає вліво бокову площину  
 self.t\_side -= 1  
 self.display()  
  
def init\_window(width, height):  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - width) // 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - height) // 2)  
 glutCreateWindow('KP\_7')  
  
  
def main():  
 glutInit()  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
 file = Image(path\_dir)  
 init\_window(file.width \* 2, file.height \* 2)  
 file.init()  
 glutDisplayFunc(file.display)  
 glutKeyboardFunc(file.keyboard\_func)  
 glutMainLoop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат

*Початкове зображення*



*Віддезракелля і повернення у початковий стан*

**Контрольні запитання**

1. *Що є основною метою методів візуалізації об’єму в медицині?*

Створення точних та реалістичних візуальних представлених об’єктів за медичними даними.

1. *Як виконати видові перетворення та перетворення моделі при роботі в тривимірному просторі?*

Спочатку відбувається перетворення світових координат на видові (без зміни положення зображення). Далі виконується перспективне перетворення, яке створює ефект перспективи засновуючить на відстані від об’єкта до екрана та від точки спостереження до екрана. Далі тривимірні видові координати перетворюються на двовимірні координати. При побудові паралельної проекції перспективне перетворення не виконується і видові координати використовуються для побудови зображення на екрані.

А видові перетворення базуються на матричних перетвореннях.

1. *Що таке багато площинне перетворення, яка його основна ідея?*

Багатоплощинне петерворення – це простий підхід до візуалізації оцінки об’єму за допомогою візуалізації серії зрізів, паралельних одній з поверхонь об’єму, або в напрямку лід якимось кутом.

1. *Визначити математичний апарат модифікації багатоплощинного перетворення для косих зрізів.*

Матриці перетворень.

1. *Перелічити три основних різновидів візуальної оцінки об’єму з використанням вигнутого перетворення площини.*

* Project CPR
* Stretched CPR
* Straightened CPR

1. *В чому полягає основна відмінність методів поверхневого та воксельного рендерінгу?*

При рендерінгу на основі вокселів зображення формується безпосередньо з об’ємних даних, а ключовою ідеєю поверхневого рендерінгу є побудова проміжного опису поверхні необхідних об’єктів по об’ємним даним, де виділяється лише поверхня об’єкта за допомогою сегментації.  
Перевага воксельного підходу полягає у збережені інформації про інтенсивність зображення протягом процесу рендерінга.

1. *Що таке теселяція?*

Теселяці – це процес розбиття об’єктів на полігоні (частіше за все розбиваються на трикутники.

1. *Які недоліки використання алгоритму маршируючих кубів?*

Він може виконувати дуже велику кількість поділів на досить простих ділянках поверхні, також він генерує значно меншу кількість трикутників, ніж інші алгоритми. Під гострими кутами цей алгоритм може утворювати артефакти, які можуть виглядати непослідовними і мати низьку якість.

1. *Перелічити чотири основні алгоритми для воксельного рендерінгу об’єму, їх переваги та недоліки.*

Алгоритм відстеження променів: створює зображення кидаючи промінь на кожен піксель об’єму і об’єднує в єдине зображення світло, яке відбито.

Алгоритм сплатінга: вокселі «кидаються» на зображення в прямій проекції формуючи слід, а результат накопичується в площині зображення.

Алгоритм зрушення-деформації: досягає прискорення за допомогою вирівнювання об’єму та розгляду такого перетину, щоб лінію вокселів можна було спроектувати в пікселі, та компенсує перше перетворення деформацією.

Алгоритм об’ємної текстурної візуалізації: об’єм виступає у вигляді тривимірної текстури, а частини зображення проектуються і компанються, використовуючи графічні апаратні засоби.

1. *В чому особливості роботи* ***OpenGL*** *з матричними формами тривимірних геометричних перетворень?*

Усі перетворення над координатами бібліотека OpenGL робить із використанням відповідних матриць перетворення (розмірністю 4´4). Передбачено 3 типи матриць – видова, проекцій і текстури. Для початку роботи з кожною з них, її необхідно зробити поточною.

Команда **void glMatrixMode(Glenum mode)** робить поточною задану параметром mode матрицю. При цьому даний параметр може приймати одне з наступних значень:

**GL\_MODELVIEW** поточною буде видова матриця;

**GL\_PROJECTION** поточною буде матриця проекцій;

**GL\_TEXTURE** поточною буде матриця текстури.

Команда **glMatrixMode**(GL\_MODELVIEW) робить поточною видову матрицю перетворення. Матриця моделі - glMatrixMode (GL\_MODELVIEW) пов'язана з координатами об'єктів. Це матриця в базисі видових координат, вона використовується для побудови картинки в тому вигляді як її бачить спостерігач.

Матриця проектування - **glMatrixMode** (GL\_PROJECTION). Матриця в системі координат пристрою. Обчислює нормалізовані координати, які перетворюються в екранні після трансформацій, пов'язаних з областю виводу.

Перш, ніж використовувати матрицю, її потрібно ініціалізувати. Команда:**void glLoadIdentity(void)** замінює поточну матрицю перетворення на одиничну (яку називають матрицею ідентичності).

При виконанні різних перетворень часто виникає необхідність збереження значень поточної матриці з можливим наступним їхнім відновленням. Команди:

**void glPushMatrix(void);**

**void glPopMatrix(void);**

запам'ятовують у стеці і відновлюють значення поточної матриці (відповідно).

Бібліотека OpenGL дозволяє виконувати наступні видові перетворення:

обертання (glRotate\*);

перенос (glTranslate\*);

масштабування (glScale\*).