НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Наталія АУШЕВА

«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Візуалізація тривимірних політочкових перетворень кулі»

Виконав:
студент IV курсу, групи ТР-82
Водопшин Володимир Володимирович _____

Керівник:
Доцент
Сидоренко Юлія Всеволодівна _____

Рецензент:
Доцент
Рачинський Артур Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Водопшину Володимиру Володимировичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи «Візуалізація тривимірних політочкових перетворень кулі»

керівник роботи___ _____ Сидоренко Юлія Всеволодівна, к.т.н._____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”08” червня 2022р. № _____

2. Строк подання студентом роботи _10.06.2022_____

3. Вихідні дані до роботи:
мова програмування Python, середа розробки PyCharm_____

4. Зміст розрахунково-пояснювальної записки:
аналіз існуючих мов, обґрунтування вибору засобу реалізації, шляхи розробки програмних додатків, розробка програмного забезпечення, висновки та аналіз результатів роботи;_____

5. Перелік ілюстративного матеріалу: постановка завдання, математичний апарат програми, функціональна схема системи, результати роботи програми_____

6. Дата видачі завдання ”10 ” вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Реалізація методу політочкових перетворень	9.02.2022-5.03.2022	
2.	Розробка структури майбутньої програми	9.03.2022-26.03.2022	
3.	Реалізація розрахункового модулю з використанням NumPy	29.03.2022-23.04.2022	
4.	Реалізація графічного модулю, який включає в себе інтерфейс для роботи з користувачем, з використанням PyGame	26.04.2022-14.05.2022	
5.	Оформлення пояснювальної записки	17.05.2022-28.05.2022	
6.	Захист програмного продукту	23.05.2022-27.05.2022	
7.	Предзахист	06.06.2022-09.06.2022	
8.	Захист	20.06.2022-30.06.2022	

Студент _____ _____Водопшин В.В._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Сидоренко Ю.В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Робота складається з 48 сторінок, містить 19 рисунків та 3 таблиці. Робота має 1 додаток, 30 бібліографічних найменувань за списком використаних джерел.

Метою даної бакалаврської роботи є створення програмного забезпечення для візуалізації тривимірних перетворень кулі політочковим методом. Система дозволяє проводити тривимірну деформацію кулі шляхом зміни точок базису. Користувач має змогу досліджувати та вивчати метод політочкових перетворень та аналізувати результат проведених деформацій над об’єктом.

Ключові слова: політочкові перетворення, геометричне моделювання, деформаційні перетворення, тривимірні перетворення.

THE SUMMARY

The work consists of 48 pages, contains 19 figures and 3 tables. The work has 1 appendix, 30 bibliographic titles according to the "List of used sources".

The purpose of this bachelor's thesis is to create software for visualization of three-dimensional transformations of the sphere by the polipoints method. The system allows to carry out three-dimensional deformation of the ball by changing the points of the base. The user has the opportunity to research and study the method of polipoints transformations and analyze the result of deformations over the object.

Keywords: polipoints transformations, geometric modeling, deformation transformations, three-dimensional transformations.

ЗМІСТ

ВСТУП

З незворотнім розвитком комп'ютеризації дедалі більше сфер діяльності починають використовувати можливості, які їм готові надати інформаційні технології. Чи не найбільш важливою є тема моделювання деформацій об’єктів, яка є однією з найбільш розповсюджених у використанні. Наразі моделювання об’єктів широко використовується в багатьох сферах, починаючи від дизайну і закінчуючи машинобудуванням та хірургією.

Одночасно із широким використанням моделювання, виникає питання про вибір відповідного йому методу. Найбільш підходящими є методи деформаційного моделювання, адже вони застосовують деформаційні зміні до простору, що в свою чергу сприяє адекватному перетворенню об’єкта. Яскравим представником даного класу моделей є метод політочкових перетворень. Даний метод дозволяє реалізувати моделювання тривимірних об’єктів. Для цього обхідно представити об’єкт у вигляді множини площин та послідовно провести політочкові перетворення для кожної з них. Найяскравіше всі нюанси методу можливо продемонструвати при моделюванні кулі, яка насправді є многокутником, побудованим за своїми правилами.

Наразі у вільному доступі не існує програмного продукту, що дозволяє проводити політочкові перетворення кулі в режимі реального часу.

Таким чином, описана задача є актуальною, адже розроблений програмний продукт дозволить проводити політочкові перетворення об’єкту в просторі, з майбутнім використанням результатів та вивченням та розвитком методу.

1 ЗАДАЧА ВІЗУАЛІЗАЦІЇ ТРИВИМІРНИХ ПОЛІТОЧКОВИХ ПЕРЕТВОРЕНЬ КУЛІ

У зв’язку з широкою областю використання методу деформаційного моделювання існує реальна потреба в програмному забезпеченні, що дозволяє візуалізовувати перетворення в режимі реального часу. Тому було поставлено задачу створити систему візуалізації тривимірних політочкових перетворень кулі.

Для реалізації поставленої задачі необхідно створити програмне забезпечення, що відображає на екрані процеси перетворення кулі методом політочкових перетворень, яка була деформована діями користувача, з можливістю обертати кулю в будь-якому напрямку, та змінювати кількість граней кулі.

Основною задачею, яку має виконувати програма, є надання користувачу можливості деформувати кулю шляхом зміни положення точок базису, не впливаючи на сам об’єкт, а тільки на простір, у якому він знаходиться. Необхідно отримати координати перетвореного об’єкта та провести візуалізацію результату перетворень у реальному часі.

Вхідна інформація: кількість граней кулі, положення точок базису.

Вихідна інформація: візуальне зображення об’єкту, оновлене положення точок базису.

Потенційними користувачами системи можуть бути дослідники, інженери, співробітники відділу моделювання або будь-які фахівці, що безпосередньо працюють у сфері деформаційного моделювання.

2 АНАЛІЗ СХОЖИХ СИСТЕМ

При проектуванні системи було знайдено та проаналізовано декілька схожих систем, та виділені певні переваги та недоліки:
PolipointsDeformations – система тривимірного моделювання з використанням політочкових перетворень.

Таблиця 2.1 - Переваги та недоліки PolipointsDeformations

Переваги	Недоліки
– Легка в освоєнні – Низькі системні вимоги до комп'ютера – Знаходиться у відкритому доступі	– Для перетворень використовується метод вагових політочкових перетворень – Працює лише на Windows системах – Реалізована з використанням застарілих алгоритмів

Blender 3D – opensource система, що використовується для тривимірного моделювання.

Таблиця 2.2 - Переваги та недоліки Blender 3D

Переваги	Недоліки
– Має обширну документацію – Має великий інструментарій – Підтримує усі існуючі операційні системи – Знаходиться у відкритому доступі	– Важкий у освоєнні – Високі системні вимоги – Потребує великого об'єму пам'яті для інсталяції

Unreal Engine 4 – система, що знаходиться у відкритому доступі на порталі Epic Games Store. Хоча основною функцією є проектування ігор, також підтримує можливості проводити деформаційні перетворення .

Таблиця 2.3 - Переваги та недоліки Unreal Engine 4

Переваги	Недоліки
– Знаходиться у відкритому доступі – Має обширну документацію – Підтримує велику кількість об’єктів для перетворень	– Для використання усіх функцій необхідна платна підписка – Потребує великого об’єму пам’яті для інсталяції – Високі системні вимоги

Проаналізувавши усі перераховані недоліки, можна виділити основні задачі, які необхідно реалізувати у власній системі:

- актуальність алгоритмів системи;
- доступний інтерфейс;
- мінімальні вимоги до системи користувача;
- висока швидкість розрахунків;

3 МАТЕМАТИЧНИЙ АПАРАТ СИСТЕМИ

3.1. Політочкові перетворення площини

Використовуючи спосіб політочкових перетворень площинних об’єктів, перейдемо до тривимірного простору. У випадку політочкових перетворень прямої на площині базисом перетворень була множина точок площини, а об’єктом перетворень була пряма, або множина прямих, якщо потрібно було проводити деформацію об’єкта, заданого певним набором прямих [15,19.22]. У випадку тривимірних політочкових перетворень базисом перетворень буде множина точок тривимірного простору, а об’єктом перетворень буде площина, або площини, у випадку, коли тривимірний об’єкт може бути представлений множиною площин [14,18,26].

Розглянемо розрахункові залежності, що реалізують політочкові перетворення однієї заданої площини у тривимірному просторі.

Принцип двоїстості у формульному вигляді може бути застосований до однорідного простору .

Однорідний простір визначається введенням невизначеного множника як четвертої координати :

$$X_{однор.} = WX$$
$$Y_{однор.} = WY$$
$$Z_{однор.} = WZ$$
$$h_{однор.} = W \qquad (3.1)$$

В однорідному просторі рівняння площини буде мати вигляд:

$$a\,X_{однор.} + b\,Y_{однор.} + c\,Z_{однор.} + d\,h_{однор.} = 0 \qquad (3.2)$$

Це рівняння є симетричним відносно до координат **Ходнор., Yоднор., Zоднор., h однор.** та коефіцієнтів **a, b, c, d.** Таким чином, можна вважати, що площина визначається «координатами» **a, b, c**, **d,** а будь-яка точка простору - **Ходнор., Yоднор., Zоднор., h однор.**

Політочкові перетворення у тривимірному просторі будуть визначатись як перетворення площин, а базисом таких перетворень буде множина точок, яку ми будемо змінювати в залежності від умов, накладених на розв'язок задачі деформування заданого об'єкта[20,21].

Геометрично проілюструємо задачу тривимірних політочкових перетворень.

Як показано на рисунку 3.1, точки початкового каркасу **1, 2, 3, 4, 5** задаються декартовими координатами x_i , y_i , z_i . Проводимо висхідну площину (площину, яка підлягає перетворенню), і визначаємо відстань від точок початкового базису до заданої площини. Для цього її рівняння (за умови **h однор.** =1) приводимо до нормалізованого вигляду і після підстановки у рівняння координат базисних точок отримуємо координати γ_i , які є політочковими координатами площини.

Змінюємо каркас, тобто міняємо декартові координати точок початкового базису, як це зображує рисунок 3.2. Нові точки **1′, 2′, 3′, 4′, 5′** утворюють новий базис, у якому потрібно знайти положення нової (перетвореної) площини.

Розглянемо докладніше політочкові перетворення за умови багатоточкового каркасу[19,22].

Площина-прообраз визначиться у висхідному базисі системою рівнянь:

$$\gamma_i = ax_i + by_i + cz_i + dh_i\,,\ i = 1 \ldots n\ , \qquad (3.3)$$

де **a, в, c, d** – коефіцієнти заданої площини,

x_i, y_i, z_i, h_i – координати точок висхідного базису.

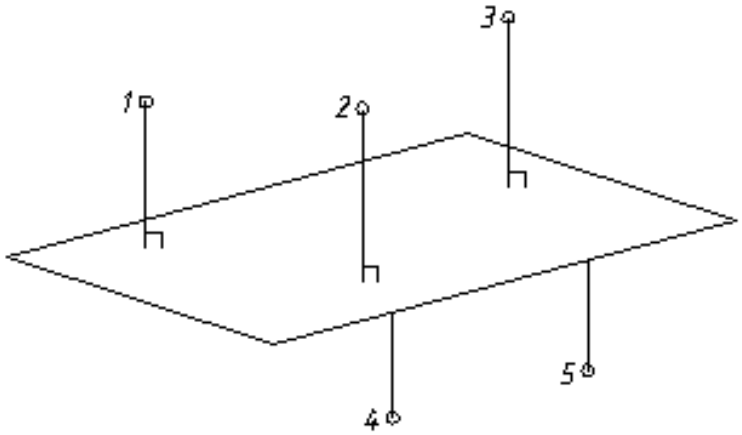


Рисунок 3.1 - Площина у початковому базисі

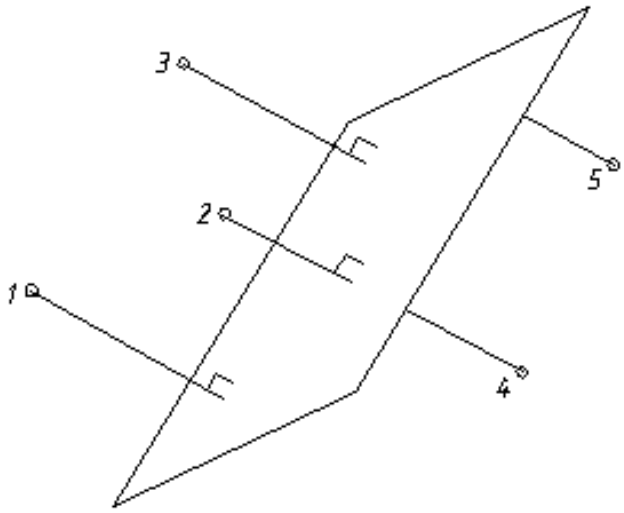


Рисунок 3.2 - Перетворена площина у новому базисі

Потрібно знайти нове положення площини у зміненому базисі. Перетворена площина буде визначатись системою рівнянь вигляду:

$$\phi_i = AX_i + BY_i + CZ_i + DH_i, i = 1..n, \quad (3.4)$$

де **A, B, C, D** – невідомі коефіцієнти шуканої площини,

X_i, Y_i, Z_i , H_i – координати нового базису,

φ_i – відстань від точок висхідного базису до площини-прообразу.

Якщо розглянути конкретну площину у багатоточковому каркасі, то її політканинні координати до перетворення і після нього у загальному випадку не співпадатимуть, тобто $\gamma_i \neq \varphi_i$, **i = 1, 2, …, p**. Якщо ж прийняти протилежне, тобто спробувати побудувати площину у перетвореному базисі за умови $\gamma_i = \varphi_i$, **i=1,2,…p**, то замість неї ми отримаємо деякий простір [1,4,13]. Таким чином, постає задача отримати однозначний розв'язок задачі політочкового перетворення простору $\mathbf{R^3}$, тобто отримання функціонального взаємозв'язку в базисах, між координатами γ_i та φ_i, **i=1,2,…** .

За аналогією з площинними політочковими перетвореннями, враховуючи, що координати площини у політочковому базисі є аналогом її віддаленості від відповідних базисних точок, політочкове перетворення у тривимірному просторі можно записати у вигляді:

$$\phi_i = \omega_i \gamma_i, \text{ i = 1, 2, …, n.} \quad (3.5)$$

Підставимо φ_i в ліву частину рівняння (3.5) :

$$\omega_i \gamma_i = Ax_i + By_i + Cz_i + Dh_i, \quad \text{i = 1, 2, …, n.} \quad (3.6)$$

Ця система містить **n** рівнянь та **n+4** невідомих (ω_i **, i = 1, 2, …, n, A, B, C, D**). Таким чином, для отримання однозначного розв'язку системи необхідно ввести чотири додаткові умови.

Для отримання додаткових умов пропонується знайти ω_i, **i = 1,2,…,n** при відхилені мінімальному від 1, тоді φ_i мінімально відхилятиметься від γ_i [15,16]. Таким чином, пропонується мінімізувати функціонал такого вигляду:

$$\sum(\phi_i - \gamma_i)^2 \to 0,\ i = 1 \ldots n. \qquad (3.7)$$

Задача зводиться до пошуку екстремуму цього функціоналу, для чого необхідно знайти частинні похідні за змінними **A**, **B**, **C**, а четверте рівняння системи має на меті зберегти нормалізований вигляд перетвореної площини[7]. Отримаємо нелінійну систему рівнянь, яку потрібно розв'язати.

$$\begin{cases} Ax_1 + By_1 + Cz_1 + D = \beta_1, \\ Ax_2 + By_2 + Cz_2 + D = \beta_2, \\ Ax_3 + By_3 + Cz_3 + D = \beta_3, \\ A^2 + B^2 + C^2 = 1. \end{cases}$$

$$\begin{cases} A(x_1 - x_2) + B(y_1 - y_2) + C(z_1 - z_2) = \beta_1 - \beta_2, \\ A(x_2 - x_3) + B(y_2 - y_3) + C(z_2 - z_3) = \beta \ \ - \beta_3, \\ A^2 + B^2 + C^2 = 1. \end{cases}$$

$$\begin{cases} A\dfrac{(x_1 - x_2)}{(z_1 - z_2)} + B\dfrac{(y_1 - y_2)}{(z_1 - z_2)} + C = \dfrac{\beta_1 - \beta_2}{z_1 - z_2}, \\ A\dfrac{(x_2 - x_3)}{(z_2 - z_3)} + B\dfrac{(y_2 - y_3)}{(z_2 - z_3)} + C = \dfrac{\beta_2 - \beta_3}{z_2 - z_3}, \\ A^2 + B^2 + C^2 = 1. \end{cases}$$

$$\begin{cases} A\left(\dfrac{x_1 - x_2}{z_1 - z_2} - \dfrac{x_2 - x_3}{z_2 - z_3}\right) + B\left(\dfrac{y_1 - y_2}{z_1 - z_2} - \dfrac{y_2 - y_3}{z_2 - z_3}\right) = \dfrac{\beta_1 - \beta_2}{z_1 - z_2} - \dfrac{\beta_2 - \beta_3}{z_2 - z_3}, \\ A^2 + B^2 + C^2 = 1. \end{cases}$$

$$\begin{cases} A\left(\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}\right)+B\left(\frac{y_1-y_2}{z_1-z_2}-\frac{y_2-y_3}{z_2-z_3}\right)=\frac{\beta_1-\beta_2}{z_1-z_2}-\frac{\beta_2-\beta_3}{z_2-z_3}, \\ A^2+B^2+\left(\frac{\beta_2-\beta_3}{z_2-z_3}-A\frac{x_2-x_3}{z_2-z_3}-B\frac{y_2-y_3}{z_2-z_3}\right)^2=1. \end{cases}$$

Введемо позначення:

$$t_1=\frac{x_1-x_2}{z_1-z_2};\ t_2=\frac{x_2-x_3}{z_2-z_3};\ t_3=\frac{y_1-y_2}{z_1-z_2};$$

$$t_4=\frac{y_2-y_3}{z_2-z_3};\ t_5=\frac{\beta_1-\beta_2}{z_1-z_2};\ t_6=\frac{\beta_2-\beta_3}{z_2-z_3}.$$

$$\begin{cases} A(t_1-t_2)+B(t_3-t_4)=t_5-t_6, \\ A^2+B^2+(t_6-At_2-Bt_4)^2=1. \end{cases}$$

$$\begin{cases} A(t_1-t_2)+B(t_3-t_4)=t_5-t_6, \\ A^2+B^2+t_6^2-2t_6(At_2+Bt_4)+(At_2+Bt_4)^2=1. \end{cases}$$

$$\begin{cases} A(t_1-t_2)+B(t_3-t_4)=t_5-t_6, \\ A^2+B^2+t_6^2-2At_2t_6-2Bt_4t_6+A^2t_2^2+2ABt_2t_4+B^2t_4^2=1. \end{cases}$$

$$\begin{cases} A(t_1-t_2)+B(t_3-t_4)=t_5-t_6, \\ A^2(1+t_2^2)+2ABt_2t_4-2At_2t_6-2Bt_4t_6+B^2(1+t_4^2)-1t_6^2=0. \end{cases}$$

Розглянемо перше рівняння:

$$A=\frac{t_5-t_6-B(t_3-t_4)}{t_1-t_2}.$$

$$A=\frac{t_5-t_6}{t_1-t_2}-B\frac{t_3-t_4}{t_1-t_2}.$$

$$\left(\frac{t_5 - t_6}{t_1 - t_2} - B\frac{t_3 - t_4}{t_1 - t_2}\right)(1 + t_2^2) + 2\left(\frac{t_5 - t_6}{t_1 - t_2} - B\frac{t_3 - t_4}{t_1 - t_2}\right)(Bt_2t_4 - t_2t_6) -$$

$$-2Bt_4t_6 + B^2(1 + t_4^2) - 1 + t_2^6 = 0.$$

Введемо позначення:

$$a\_1 = \frac{t\_5 - t\_6}{t\_1 - t\_2};\ \ a\_2 = \frac{t\_3 - t\_4}{t\_1 - t\_2};\ \ a\_3 = 1 + t\_2^2; a4 = t\_2t\_4;$$

$$a_5 = t_2t_6;\ a_6 = t_4t_6;\ a_7 = 1 + t_4^2;\ a_8 = -1 + t_6^2.$$

$$(a_1 - Ba_2)^2a_3 + 2(a_1 - Ba_2)(Ba_4 - a_5) - 2Ba_6 + B^2a_7 + a_8 = 0.$$

$$a_3a_1^2 - 2Ba_1a_2a_3 + B^2a_2^2a_3 + 2(Ba_1a_4 - B^2a_2a_4 - a_1a_5 + Ba_2a_5) -$$

$$-2Ba_6 + B^2a_7 + a_8 = 0.$$

$$a_3a_1^2 - 2Ba_1a_2a_3 + B^2a_2^2a_3 + 2Ba_1a_4 - 2B^2a_2a_4 - 2a_1a_5 + 2Ba_2a_5 -$$
$$-2Ba_6 + B^2a_7 + a_8 = 0.$$

$$B^2(a_2^2a_3 - 2a_2a_4 + a_7) + B(-2a_1a_2a_3 + 2a_1a_4 + 2a_2a_5 - 2a_6) +$$

$$+a_3a_1^2 = 2a_1a_5 + a_8 = 0.$$

Введемо позначення.

$$K_1 = a_2^2a_3 - 2a_2a_4 + a_7;\ K_2 = -2a_1a_2a_3 + 2a_1a_4 + 2a_2a_5 - 2a_6;$$

$$K_3 = a_3a_1^2 - 2a_1a_5 + a_8.$$

Отримаємо квадратне рівняння.

$$B_2 = \frac{-K_2 + \sqrt{K_2^2 - 4K_1K_3}}{2K_1}.\ B^2K_1 + BK_2 + K_3 = 0.$$

$$D = K_2^2 - 4K_1K_3.$$

$$B_1 = \frac{-K_2 - \sqrt{K_2^2 - 4K_1K_3}}{2K_1}.$$

де $K_1 = a_2^2a_3 - 2a_2a_4 + a_3 =$

$$= \left(\frac{t_3 - t_4}{t_1 - t_2}\right)^2 (1 + t_2^2) - 2\left(\frac{t_3 - t_4}{t_1 - t_2}\right) t_2t_4 + 1 + t_4^2 =$$

$$\left(\frac{t_3 - t_4}{t_1 - t_2}\right)\left[\frac{(t_3 - t_4)(1 + t_2^2)}{(t_1 - t_2)} - 2t_2t_4\right] + 1 + t_4^2 =$$

$$= \frac{(t_3 - t_4)}{t_1 - t_2} \cdot \frac{\left((t_3 - t_4)(1 + t_2^2) - 2(t_1 - t_2)t_2t_4\right)}{t_1 - t_2} + 1 + t_4^2 =$$

$$= \frac{\frac{y_1 - y_2}{z_1 - z_2} - \frac{y_2 - y_3}{z_2 - z_3}}{\frac{x_1 - x_2}{z_1 - z_2} - \frac{x_2 - x_3}{z_2 - z_3}} \cdot \left[\frac{\left(\frac{y_1 - y_2}{z_1 - z_2} - \frac{y_2 - y_3}{z_2 - z_3}\right)\left(1 + \left(\frac{x_2 - x_3}{z_2 - z_3}\right)^2\right) - }{\frac{x_1 - x_2}{z_1 - z_2} - \frac{x_2 - x_3}{z_2 - z_3}}\right.$$

$$\left.\frac{-2\left(\frac{x_1 - x_2}{z_1 - z_2} - \frac{x_2 - x_3}{z_2 - z_3}\right) \cdot \frac{x_2 - x_3}{z_2 - z_3} \cdot \frac{y_2 - y_3}{z_2 - z_3}}{}\right] + 1 + \left(\frac{y_2 - y_3}{z_2 - z_3}\right)^2.$$

$$K_2 = -2a_1a_2a_3 + 2a_1a_4 + 2a_2a_5 - 2a_6 =$$

$$= -2\frac{t_5 - t_6}{t_1 - t_2} \cdot \frac{t_3 - t_4}{t_1 - t_2} \cdot (1 + t_2^2) + 2\frac{t_5 - t_6}{t_1 - t_2} t_2t_4 +$$

$$+2\frac{t_3 - t_4}{t_1 - t_2} t_2t_6 - 2t_4t_6 =$$

$$= 2\left[\frac{t_5 - t_6}{t_1 - t_2}\left(\frac{(t_3 - t_4)(1 + t_2^2)}{t_1 - t_2} + t_2t_4\right) + t_6\left(\frac{(t_3 - t_4)t_2}{t_1 - t_2} - t_4\right)\right] =$$

$$= 2\left[\frac{t_5 - t_6}{t_1 - t_2}\left(\frac{(t_3 - t_4)(1 + t_2^2) + (t_1 - t_2)t_2t_4}{t_1 - t_2}\right) +\right.$$

$$+t_6\left(\frac{(t_3-t_4)t_2-t_4(t_1-t_2)}{t_1-t_2}\right)\Bigg]=$$

$$=\frac{2}{t_1-t_2}\left[\frac{t_5-t_6}{t_1-t_2}\Big((t_3-t_4)(1+t_2^2)(t_1-t_2)t_2t_2\Big)+\right.$$
$$\left.+t_6\big((t_3-t_4)t_2-t_4(t_1-t_2)\big)\right]=$$

$$=\frac{2}{\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}}\left[\frac{\frac{\beta_1-\beta_2}{z_1-z_2}-\frac{\beta_2-\beta_3}{z_2-z_3}}{\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}}\left(\left(\frac{y_1-y_2}{z_1-z_2}-\frac{y_2-y_3}{z_2-z_3}\right)\times\right.\right.$$

$$\left.\times\left(1+\left(\frac{x_2-x_3}{z_2-z_3}\right)^2\right)+\left(\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}\right)\frac{x_2-x_3}{z_2-z_3}\cdot\frac{y_2-y_3}{z_2-z_3}\right)$$

$$+\frac{\beta_2-\beta_3}{z_2-z_3}\left(\left(\frac{y_1-y_2}{z_1-z_2}-\frac{y_2-y_3}{z_2-z_3}\right)\cdot\frac{x_2-x_3}{z_2-z_3}-\right.$$

$$\left.\left.-\frac{y_2-y_3}{z_2-z_3}\left(\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}\right)\right)\right].$$

$$K_3=a_3a_1^2-2a_1a_5+a_8=$$

$$(1+t_2^2)\cdot\left(\frac{t_5-t_6}{t_1-t_2}\right)^2-2\frac{t_5-t_6}{t_1-t_2}t_2t_6+t_6^2-1=$$

$$=\left(1+\left(\frac{x_2-x_3}{z_2-z_3}\right)^2\right)\cdot\left(\frac{\frac{\beta_1-\beta_2}{z_1-z_2}-\frac{\beta_2-\beta_3}{z_2-z_3}}{\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}}\right)^2-$$

$$-2\left(\frac{\frac{\beta_1-\beta_2}{z_1-z_2}-\frac{\beta_2-\beta_3}{z_2-z_3}}{\frac{x_1-x_2}{z_1-z_2}-\frac{x_2-x_3}{z_2-z_3}}\right)\cdot\frac{x_2-x_3}{z_2-z_3}\cdot\frac{\beta_2-\beta_3}{z_2-z_3}+\left(\frac{\beta_2-\beta_3}{z_2-z_3}\right)^2-1.$$

Коефіцієнт **B** використовується для знаходження інших коефіцієнтів **A, C**, та **D,** але формули для знаходження є досить громіздкими, тому розглянемо інший підхід[8].

Знайдемо частинні похідні за змінними **A**, **B**, **C** та **D**

$$\frac{\partial \sum(\phi_i - \gamma_i)^2}{\partial A} = 2\sum X_i(AX_i + BY_i + CZ_i + DH_i - \gamma_i) = 0.$$
$$\frac{\partial \sum(\phi_i - \gamma_i)^2}{\partial B} = 2\sum Y_i(AX_i + BY_i + CZ_i + DH_i - \gamma_i) = 0.$$
$$\frac{\partial \sum(\phi_i - \gamma_i)^2}{\partial C} = 2\sum Z_i(AX_i + BY_i + CZ_i + DH_i - \gamma_i) = 0.$$
$$\frac{\partial \sum(\phi_i - \gamma_i)^2}{\partial D} = 2\sum H_i(AX_i + BY_i + CZ_i + DH - \gamma_i) = 0.$$

Ці чотири рівняння утворюють лінійну систему рівнянь. У матричному вигляді будемо мати:

$$A \cdot V = B,$$

де

$$A = \begin{bmatrix} \sum X_i^2 & \sum X_i Y_i & \sum X_i Z_i & \sum X_i H_i \\ \sum Y_i X_i & \sum Y_i^2 & \sum Y_i Z_i & \sum Y_i H_i \\ \sum Z_i X_i & \sum Z_i Y_i & \sum Z_i^2 & \sum Z_i H_i \\ \sum H_i X_i & \sum H_i Y_i & \sum H_i Z_i & \sum H_i^2 \end{bmatrix}.$$

$$V = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}. \qquad B = \begin{bmatrix} \sum X_i \gamma_i \\ \sum Y_i \gamma_i \\ \sum Z_i \gamma_i \\ \sum H_i \gamma_i \end{bmatrix}. \tag{3.8}$$

Розв’язком цієї лінійної системи рівнянь будуть значення **A, B, C, D,** тобто коефіцієнти шуканої перетвореної площини.

Політочкові перетворення площини є основою, для подальшого моделювання об’єктів у тривимірному просторі, які зазнали дефформативних змін за допомогою тривимірних політочкових перетворень [2] – [6]. Так, як майже усі тривимірні об’єкти можна розбити на множини площин, з яких об’єкт складається, то при застосування політочкових перетворень до кожної з площин, ми отримаємо перетворений тривимірний об’єкт.

3.2. Політочкові перетворення гранованих тіл

При політочкових перетвореннях у тривимірному просторі об’єкт представляється набором плоских трикутників. Кожний такий трикутник можна представити чотирма площинами, які перетинаються у його вершинах , як на рисунку 3.3.

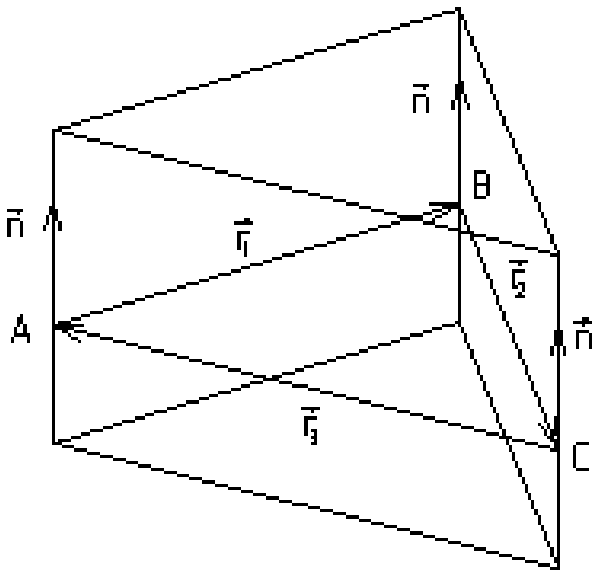


Рисунок 3.3 - Представлення об’єкта перетворень трикутниками

До кожної з площин застосовуються політочкові перетворення, тобто у новому базисі визначаються нові положення утворюючих площин, які в свою чергу визначають положення трикутника перетвореного об’єкта в новому базисі. Перетворені трикутники утворюють грановане тіло, яке є образом заданого [9]-[12].

Алгоритм перетворення тривимірного тіла можна описати так [26,29]:

1. Тривимірне тіло занурюється у точковий базис. Це робить користувач, обираючи точки базису з урахуванням умов, накладених на задачу, наприклад, на кінцеву форму об’єкта.
2. Тіло, яке підлягає деформації, представляється множиною площин (трикутників). Це можна зробити, наприклад, за допомогою тріангуляції.
3. Виконуються послідовні політочкові перетворення кожної з площин та визначається їх перетин, таким чином отримуємо трикутники об’єкта у новому базисі.
4. Застосовуючи існуючі методи просторової інтерполяції (або можливості сучасних графічних пакетів), згладжуємо отриману грановану поверхню.

Результатом роботи алгоритму перетворення буде деформований за усіма правилами об’єкт перетворень. Даний алгоритм є найефективнішим при завданні деформації тривимірного тіла, він потребує мінімальної кількості дій та при цьому надає результати високої точності.

4 ОПИС СИСТЕМИ ТРИВИМІРНИХ ПОЛІТЧКОВИХ ПЕРЕТВОРЕНЬ КУЛІ

Після аналізу предметної області і застосування теорії політочкових перетворень було створено систему тривимірних перетворень кулі.

4.1 Структура системи

Створена система складається з трьох окремих структурних частин [17,25]. Кожна з яких реалізована в вигляді одного виконуваного файлу. На рисунку 4.1 представлена структурна схема програми.

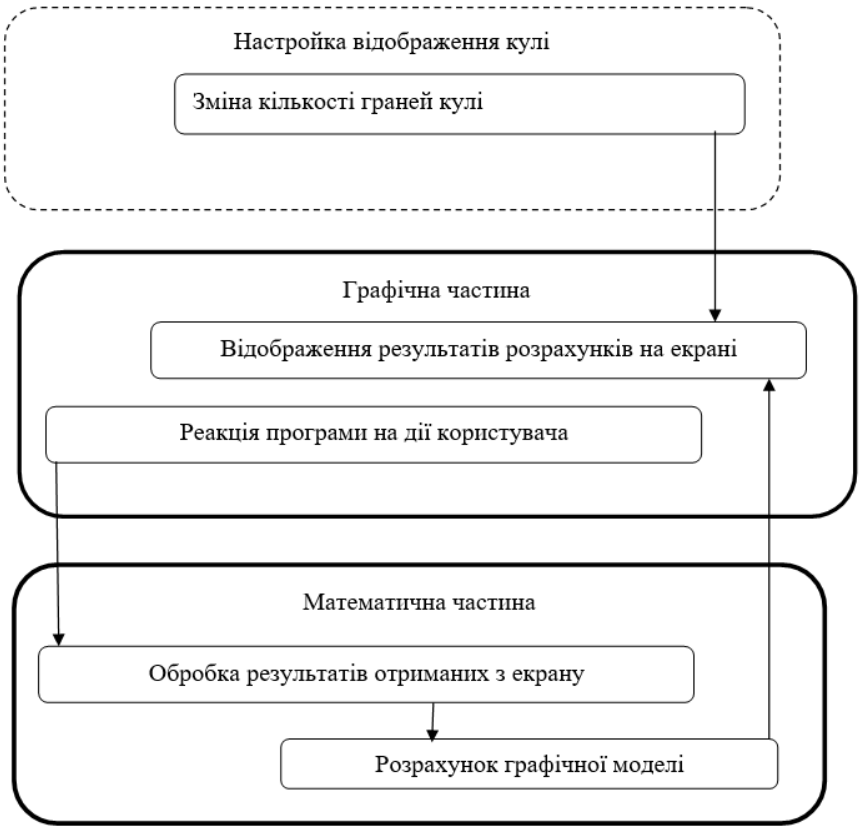


Рисунок 4.1 - Структурна схема програми

Як видно зі схеми, робота кожного модулю відбувається тільки після того, як попередній модуль завершив свою роботу. Після того, як програма виконала побудову кулі, вона знов готова приймати дані від користувача, і так, доки робота програми не буде завершена [24].

4.2 Графічний модуль

Графічний модуль виконує дві основні функції: обробка дій, виконаних користувачем, і побудова видозміненої кулі на екрані, кожна з яких буде описана нижче окремо, зі своїми вхідними та вихідними даними.

4.2.1 Функція обробки даних, введених користувачем

Для можливості проводити перетворення в режимі реального часу конче важливо швидко й якісно приймати дані, введені користувачем, й перетворювати їх в дані, що будуть використані розрахунковим модулем для обчислення перетворень методом політочкових перетворень. Для цього необхідно відслідковувати переміщення курсору по екрані, які потім переводяться в відносні координати та передаються в модуль розрахунків для подальших обчислень[28].

Для реалізації такої функції найкраще підійде фреймворк для мови Python, що називається PyGame. Він ньому вже реалізована більшість необхідних механізмів, тож достатньо лише якісно інтегрувати їх в систему.

Вхідні дані – переміщення курсору миші, якщо була зажата ліва клавіша, вихідні дані - оновлені координати точок базису.

4.2.2 Функція побудови кулі на екрані

На основі набору точок кулі, що були обчислені модулем розрахунків, відбувається побудова фігури. Так як перетворення відбуваються в просторі, необхідно відобразити фігуру так, щоб передати її об’єм, використовуючи при

цьому плоский екран[29]. Якнайліпше для цього підходить бібліотека PyGame, в якій вже реалізована велика кількість корисних функцій. Після побудови фігура передається на екран користувачу.

Вхідні дані – набір точок кулі, вихідні дані – побудована фігура.

4.3 Математичний модуль

Після дій користувача оброблені дані передаються розрахунковому модулю, який в свою чергу проводить перерахунок координат усіх точок, на основі яких буде будуватися куля[30]. Всі розрахунки проводяться за правилами методу політочкових перетворень. Після проведених розрахунків оновлені координати точок кулі передаються до графічного модулю.

Для коректної роботи необхідно оброблювати велику кількість даних за короткий проміжок часу, аби уникнути затримок між діями користувача та візуалізацією проведених перетворень. Для цього було обрано фреймворк NumPy.

Вхідні дані — оновлені координати, вихідні — оновлений набір точок кулі.

4.4 Модуль налаштування кулі

Найперше, що побачить користувач запустивши програму, буде меню налаштування множини граней, їх кількості, на яких побудована куля. Даний модуль має одну єдину функцію – зчитати число, яке користувач ввів у відповідне поле та передати його до розрахункового модулю.

Наявність даного модулю надає користувачу можливість налаштування вигляду об’єкту для кращого сприйняття та передачі об’єму кулі[23,27]. В той же час ця функція надає можливість програмі працювати на слабкіших пристроях, аби зменшити вимоги до комп’ютеру, достатньо просто будувати кулю з меншого числа точок.

5 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

5.1 Операційна система

Найпоширенішою операційною системою у світі є Windows, за сьогоднішніми даними нею користується близько 78% від усіх користувачів персональним комп'ютером, тому було вирішено обрати саме її. Не менш важливим фактором є й те, що програма буде доступна для використання незалежно від версії, тобто на будь-якій ОС сімейства Windows. Комфортна робота додатку потребує однієї з систем: Windows7, Windows8, Windows10, Windows11, у випадку з ОС старішого покоління можуть виникати помітні затримки в роботі.

Таким чином, у користувачів існує достатньо широкий вибір ПК для роботи з програмою.

Також окрім усього вищезгаданого необхідно зазначити й інші переваги ОС сімейства Windows:

- єдиний зручний інтерфейс користувача;
- широке розповсюдження цього сімейства ОС;
- доступність безкоштовного встановлення ОС на ПК;
- 64-розрядні системи, що є достатньою основою для функціонування різноманітних програм на необхідному рівні;
- широкий вибір доступних функцій та методів роботи з графікою, що дозволяє обрати підходящий до кожної з задач;
- спеціальні алгоритми оптимізації, що дозволяє економити ресурси ПК, що в свою чергу пришвидшує та оптимізує роботу на системах з малопотужними комп'ютерами;

5.2 Мова програмування та середовище реалізації

Для обрання потрібної мови реалізації було вирішено розробити вимоги. Так, як з технічної точки зору підходять майже усі мови програмування, було вирішено звернути увагу на можливості мови, що дозволять спростити виконання задачі та зменшити вимоги додатку до апаратного та програмного забезпечення, а саме:

- підтримка великої кількості безкоштовних бібліотек для роботи з графікою;
- велика кількість літератури та ком'юніті для можливого вирішення проблем та помилок під час розробки;
- підтримка безкоштовного середовища розробки, що має широкий функціонал та задовольняє усі поставлені задачі;

За визначеними вимогами будуть підходити досить багато найпопулярніших мов програмування, наприклад: C++, Python, Java, Visual Basic та інші, так само відповідних ним середовищ. Тому було вирішено обрати мову програмування Python та відповідну йому середу розробки PyCharm, ґрунтуючись на найбільшому досвіді роботи з ними.

Були виділені основні переваги обраної мови програмування, такі як:

- портативність, можливість запустити додаток на будь якій ОС без зміни коду;
- дуже легка читабельність коду;
- наявність модулів та бібліотек для роботи з графікою, що полегшить та роботу на графічним модулем програми;
- велика кількість матеріалів та прикладів;

5.3 Системні вимоги до комп’ютера

Для стабільної роботи програми, необхідно щоб виконувалися наступні вимоги до комп’ютера:

- операційна система Windows7 та новіші, 64-розрядна;
- відеокарта: з 1 ГБ пам'яті;
- оперативна пам'ять: 2 ГБ;
- процесор: із тактовою частотою 2.5–2.9 ГГц;
- додаткових вимог не існує;

У випадку, якщо комп'ютер не відповідає вимогам, описаним вище, користувач може спостерігати помітні затримки під час обрахунків та побудови фігури.

6 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

6.1 Модуль розрахунків перетворень

Розрахунковий модуль відповідає за усі обчислення, що виконуються в додатку, він же оброблює й дані, що надійшли йому від користувача, для подальшої її використання в розрахунках перетворень кулі. На рисунку 6.1 представлена блок-схема розрахункового модулю.

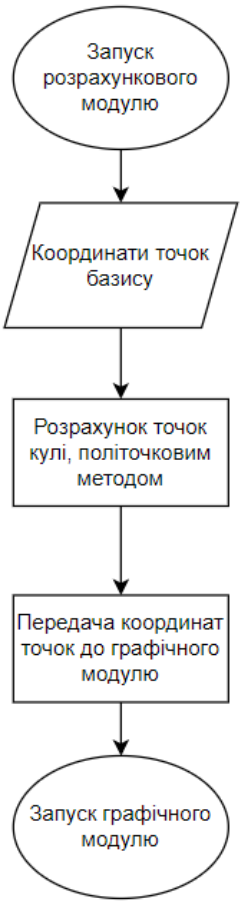


Рисунок 6.1 - Блок-схема розрахункового модулю

Після того, як розрахунковий модуль виконав розрахунки, оновлені дані з координатами точок передаються до графічного модулю, який на їх основі будує об’єкт, який буде відображений на екрані.

6.2 Графічний модуль

Основна задача графічного модулю – обробити дані, які йому були передані від розрахункового модулю, та на їх основі виконати побудову кулі, яка буде відображена на екрані користувача. На рисунку 6.2 представлена блок-схема графічного модулю.

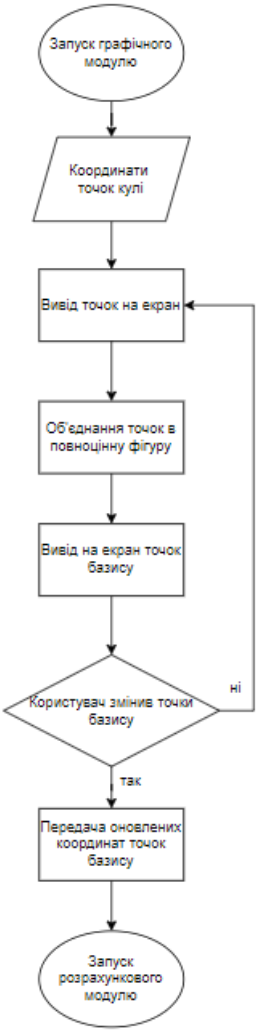


Рисунок 6.2 - Блок-схема графічного модулю

Після того, як користувач вносить зміни до точок базису, вони передаються на обробку, до розрахункового модулю, для подальшої можливості розрахунку та оновлення зображення на екрані.

6.3 Модуль налаштування кількості граней кулі

Даний модуль необхідний для діалогу користувача з програмою, метою якого є можливість налаштування відображення кулі, тобто користувач може обрати з якої кількості граней буде складатися об’єкт, що надає можливість налаштування кулі та покращує вигляд об’єкту. На рисунку 6.3 представлена блок-схема модулю налаштування кулі

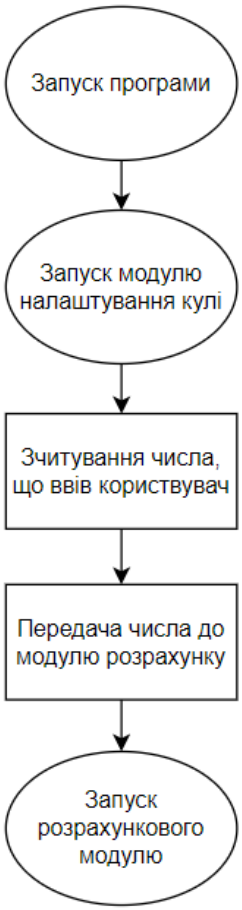
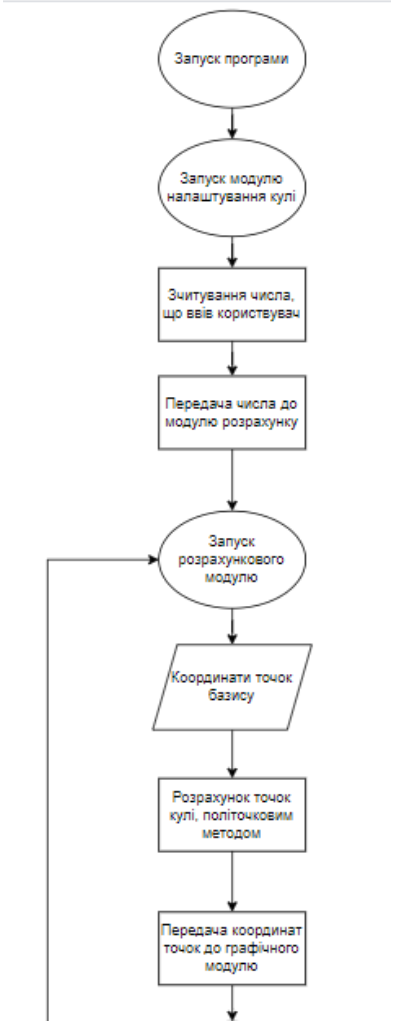


Рисунок 6.3 - Блок-схема модулю налаштування кулі

Після того як модуль зчитає дані, що були введені користувачем, він передає їх до розрахункового модулю

6.4 Загальний вигляд роботи програми

Робота програми від самого початку була поділена на різні модулі, кожен з яких виконує функцію відповідну йому, це дозволяє простіше розібратися в роботі програми та побудувати лінійну структуру програми, де кожен модуль після виконання відведеної йому задачі, запускає наступний модуль і так доки користувач не завершить роботу з програмою. Блок-схема програми зображена на рисунку 6.5.



Продовження рисунку 6.4:

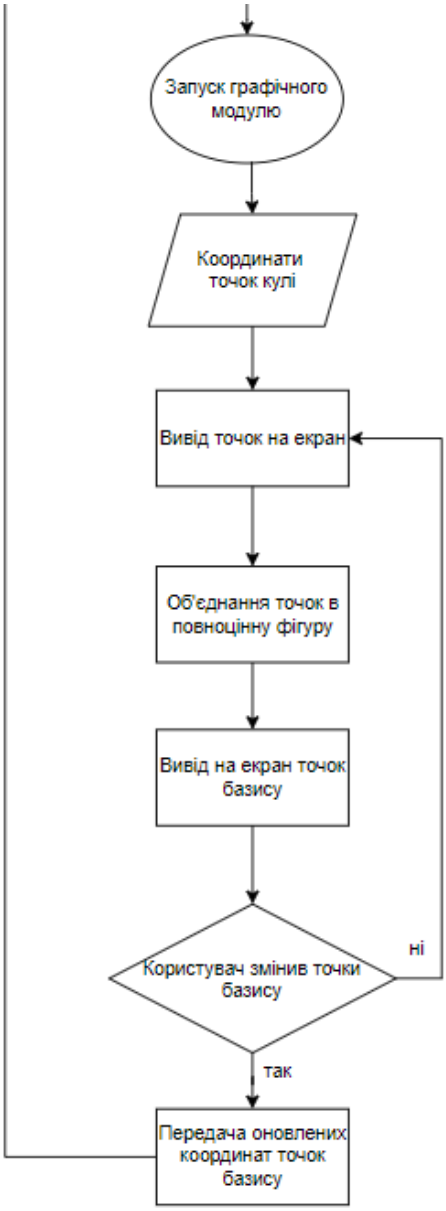


Рисунок 6.4 - Блок-схема роботи програми

На даній блок-схемі, зображено принцип роботи усієї програми, на ній можна спостерігати, як різні модулі пов’язані та співпрацюють один з одним.

7. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

Система візуалізації проводить розрахунки перетворень політочковим методом на основі даних, що були введені користувачем. Результатом таких обчислень є зображення деформованої кулі та зміщених точок базису. Побудова оновленого об’єкту виконується в режимі реального часу, тож користувач має змогу видозмінювати об’єкт стільки разів, скільки йому необхідно, без жодних затримок у часі та очікувань.

7.1 Встановлення додатку на комп’ютер

Для того, щоб почати використовувати додаток, його спершу необхідно встановити на комп’ютер. Для зручності користувачів усі файли програми були скомпільовані в один виконуваний файл – *polipoints_deformations.exe*. Для більш зручного доступу до файлу, було вирішено завантажити його на гугл-диск та надати доступ за посиланням, що зображено на рисунку 7.1.

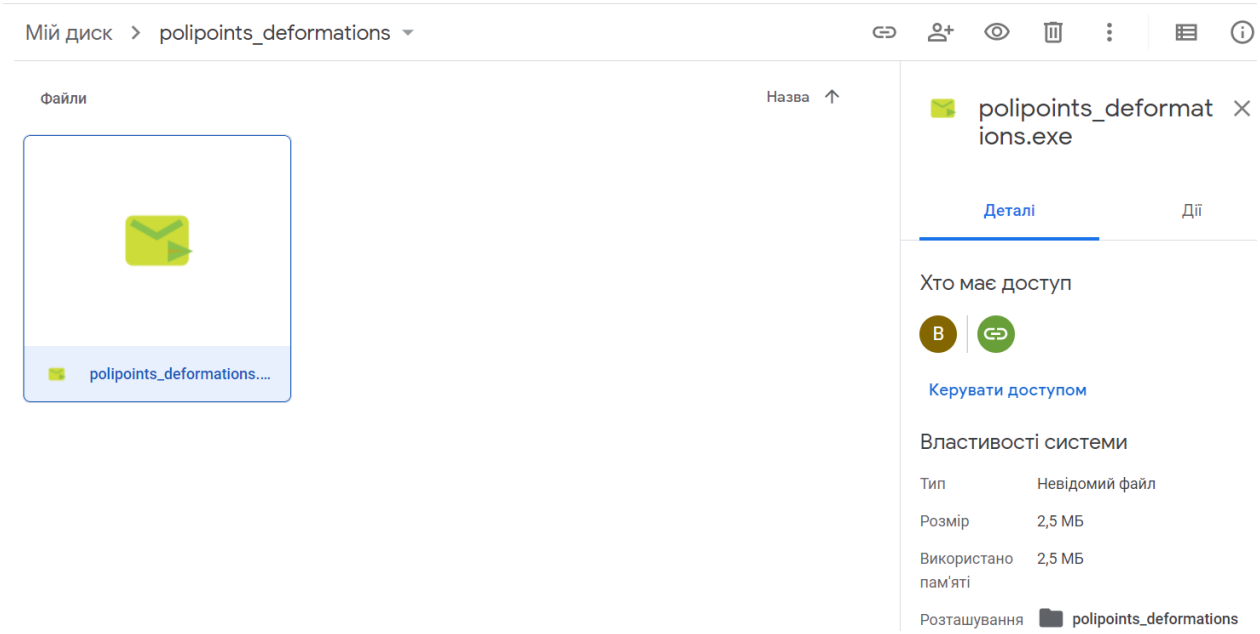


Рисунок 7.1 – Виконуваний файл на гугл-диску

Після того, як користувач перейшов за посиланням на гугл-диск, необхідно натиснути правою кнопкою миші на файл *polipoints_deformations.exe* та у відкритому меню обрати “Завантажити, як це зображено на рисунку 7.2.

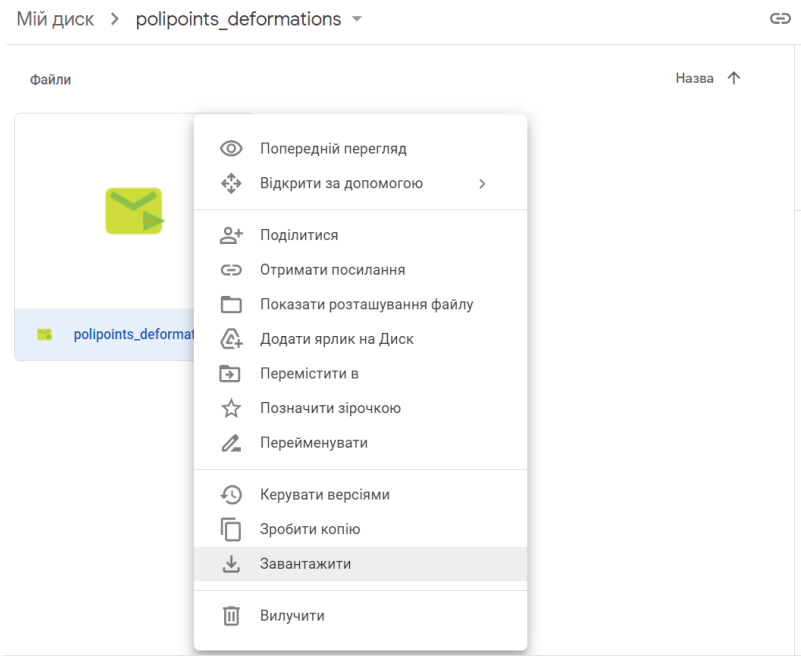


Рисунок 7.2 – Завантаження програми на комп’ютер

Коли файл буде завантажено на комп’ютер, необхідно знайти його розташування та подвійним кліком мишки на файл запустити додаток. На рисунку 7.3 зображено приклад.

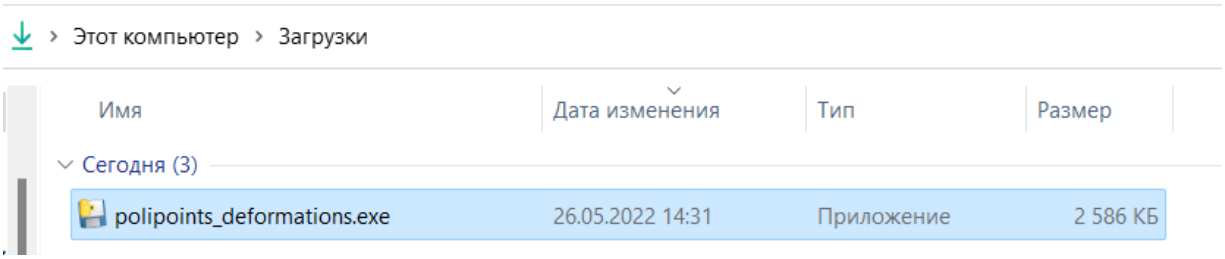


Рисунок 7.3 – Додаток скачаний на комп’ютер

Після запуску додатку на екрані комп’ютера відобразиться початкове вікно програми. Дане вікно представляє собою меню налаштування кулі. Його користувач повинен використовувати для майбутньої зміни вигляду кулі.

7.2 Меню налаштування кількості граней кулі

Одразу після запуску програми нам необхідно ввести натуральне число, як це показано на рисунку 7.4, що буде дорівнювати половині кількості граней кулі та кількості прямих з яких складається одна грань, чим більше введене число, тип сильніше навантажується комп’ютер, в наслідок чого можуть відбуватися помітні затримки в процесі. Оптимальним варіантом є 10-20, цього достатньо для детального зображення перетворень, які в свою чергу проходять досить плавно.

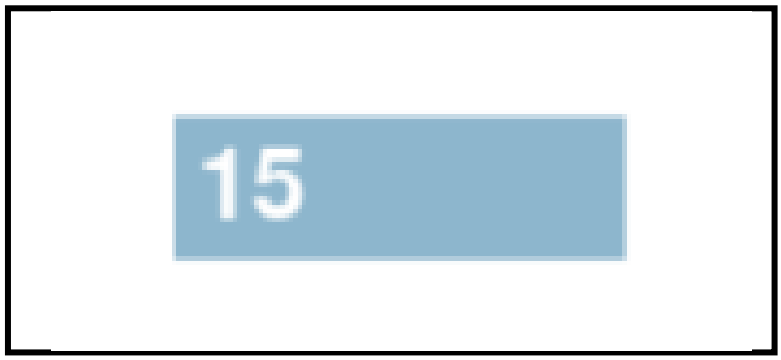


Рисунок 7.4 - Вигляд вікна з налаштуванням відображення кулі

Після чого необхідно натиснути клавішу “Enter” на клавіатурі для побудови кулі.

Якщо дані були введені неправильно, замість побудови кулі програма очистить вікно вводу та буде очікувати на введення даних користувачем. Так буде відбуватися, доки користувач не введе коректні дані.

7.3 Перетворення кулі діями користувача

Після налаштування відображення кулі, який було описано у пункті 6.2, на екрані комп’ютера відобразиться побудована сфера, як це зображено на рисунку 7.5, яку вже своїми діями користувач може деформувати та перетворити у еліпсоїд будь-якої форми.

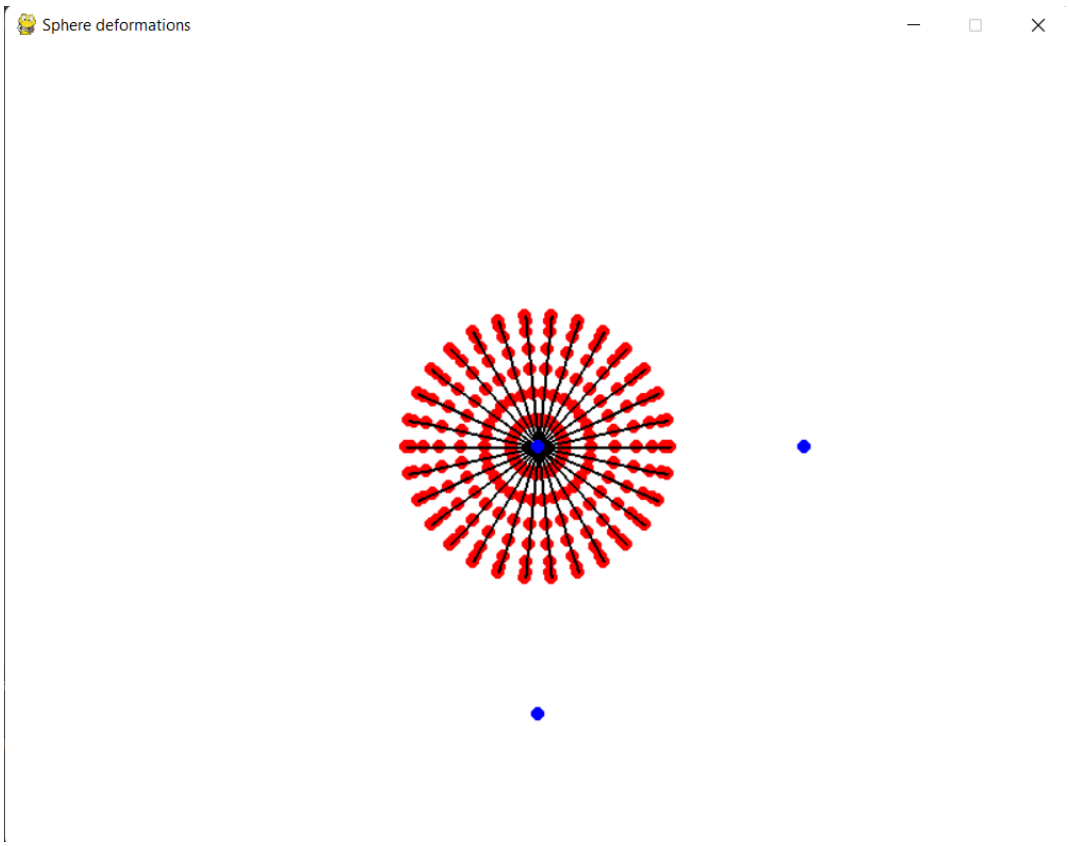


Рисунок 7.5 – Побудова кулі одразу після запуску програми

Користувач має можливість повертати кулю в будь-якому напрямку, що відображає рисунок 7.6, аби усвідомитися в об’ємності фігури та мати можливість більш детально розібратися в роботі методу політочкових перетворень. Для цього необхідно затиснути ліву кнопку миші у будь-якому місці вікна програми, окрім контрольних точок, які відображаються синім кольором, та потягнути у тому напрямі, в якому користувач бажає обернути кулю.

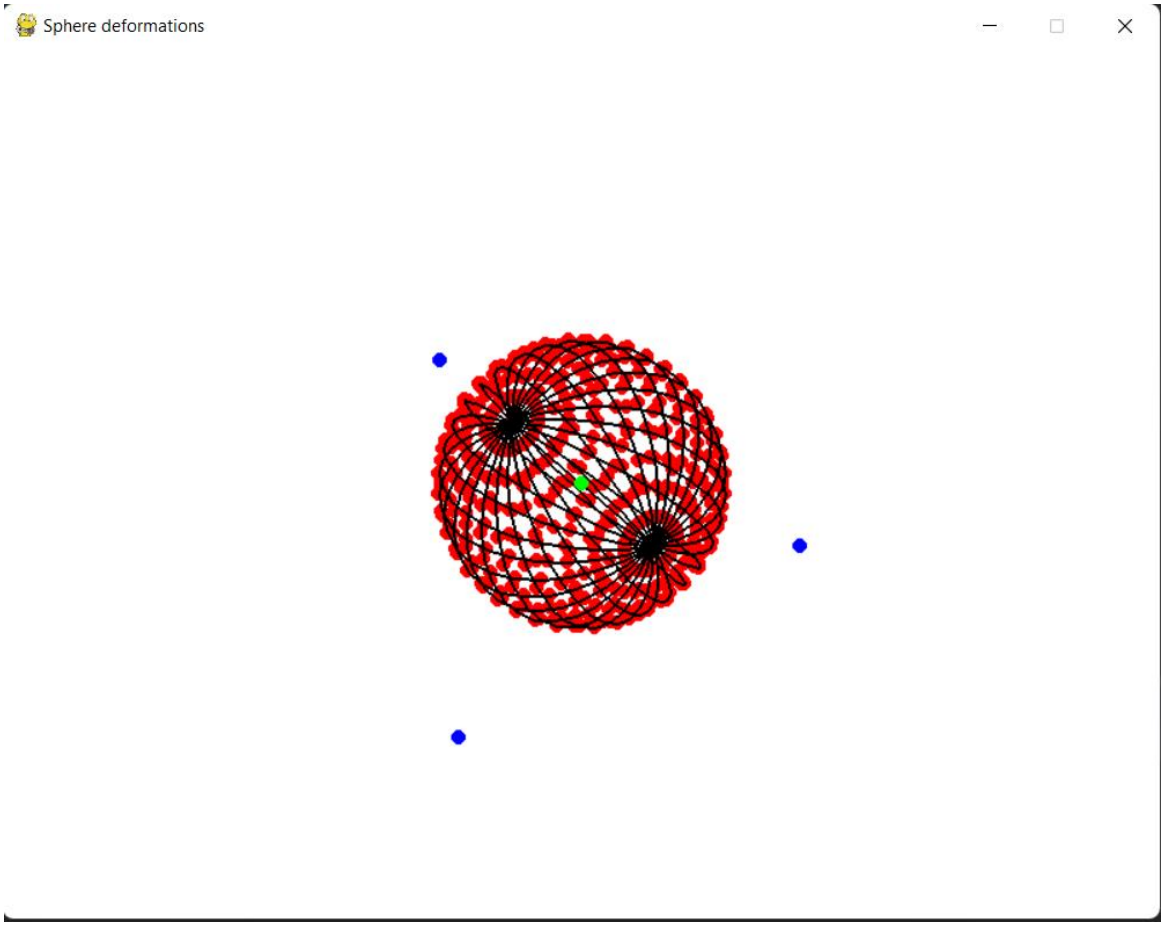


Рисунок 7.6 - Вигляд кулі після її повороту

На екрані, окрім сфери, знаходяться також 4 точки, одна з яких зафарбована зеленим кольором. Ця точка є центром фігури і попри всі деформації, що будуть відбуватися з кулею вона буде знаходитися точно в її центрі. Вона відображається для ліпшого сприйняття та несе в собі ціль допомогти користувачу в навігації на екрані. Інші три точки – синього кольори, це контрольні точки, які користувач використовує саме для видозміни фігури на екрані.

Для перетворення еліпсоїду, приклад на рисунку 7.7 та рисунку 7.8, користувачу необхідно навести курсором на будь-яку контрольну точку, після чого затиснути ліву кнопку миші та перемістити її. В цей час фігура буда деформуватися відповідно до правил методу.

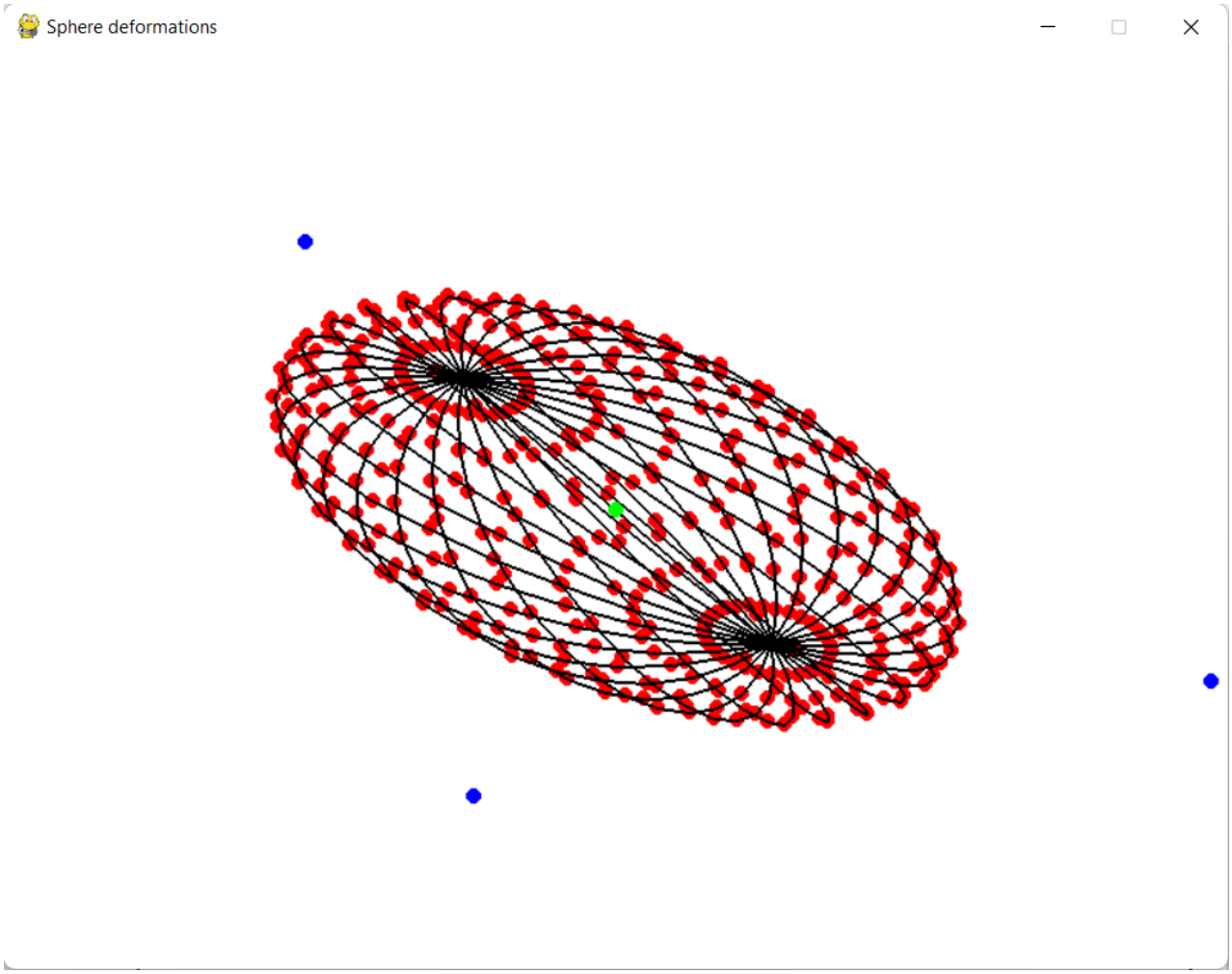


Рисунок 7.7 - Вигляд кулі після перетворень

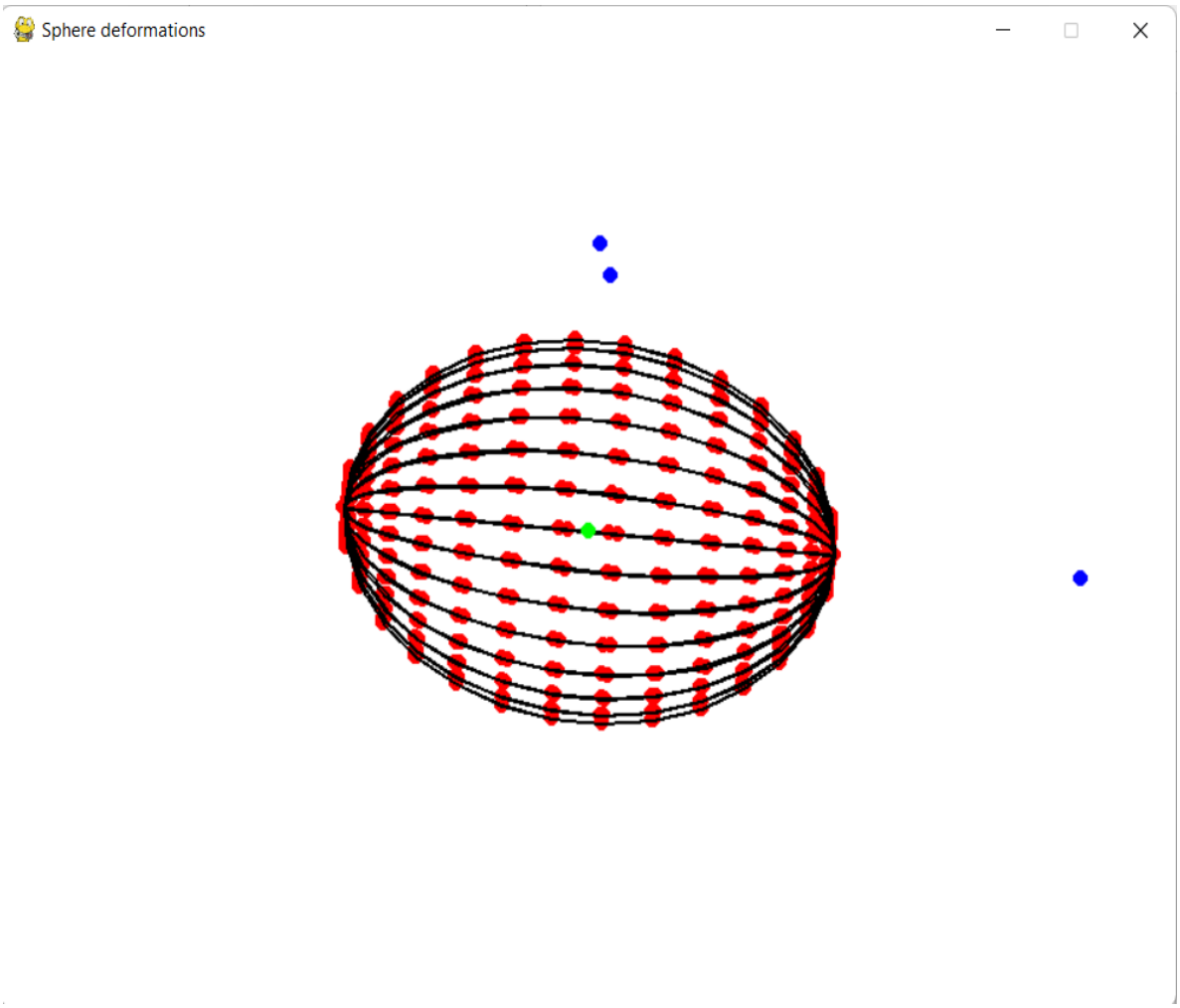


Рисунок 7.8 - Вигляд кулі після перетворень та поворотів

Після того як користувач виконає усю необхідну роботу, для виходу з програми достатньо натиснути клавішу escape на клавіатурі, або закрити її, натиснувши на хрестик у правому верхньому куті.

7.4 Зміна кількості граней кулі

Система надає можливість користувачу змінювати кількість граней, з яких буде побудована фігура. Завдяки чому з’являється можливість підвищити точність побудови об’єкту, але при цьому має бути виконана більша кількість розрахунків. Кількість граней може варіюватися від 2 і більше.

При збільшенні кількості граней об’єкту, як не зображено на рисунку 7.9, збільшується і точність розрахунків перетворень, але при цьому підвищується навантаження на комп’ютер, тому при високому значенні граней можуть виникати помітні затримки у часі.

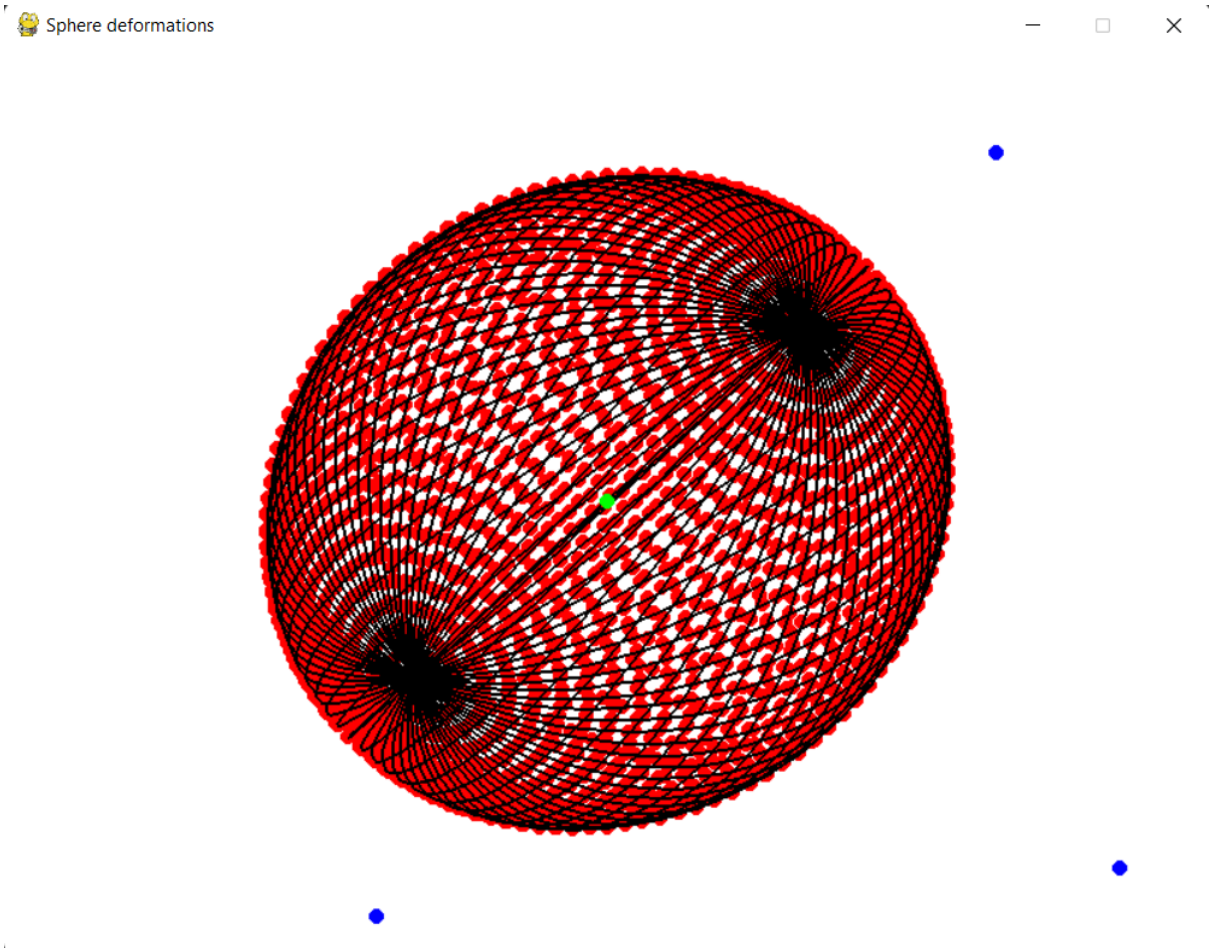


Рисунок 7.9 – Видозмінена куля побудована з 40 граней

При побудові кулі з невеликої кількості граней, як це зображено на рисунку 7.10, дуже помітними є ламані, з яких побудові грані кулі та відповідно сильно знижується точність розрахунків, але при цьому кількість обчислень помітно зменшується, що в свою чергу призводить до малого навантаження на комп’ютер та миттєвого відображення перетворень кулі на екран.

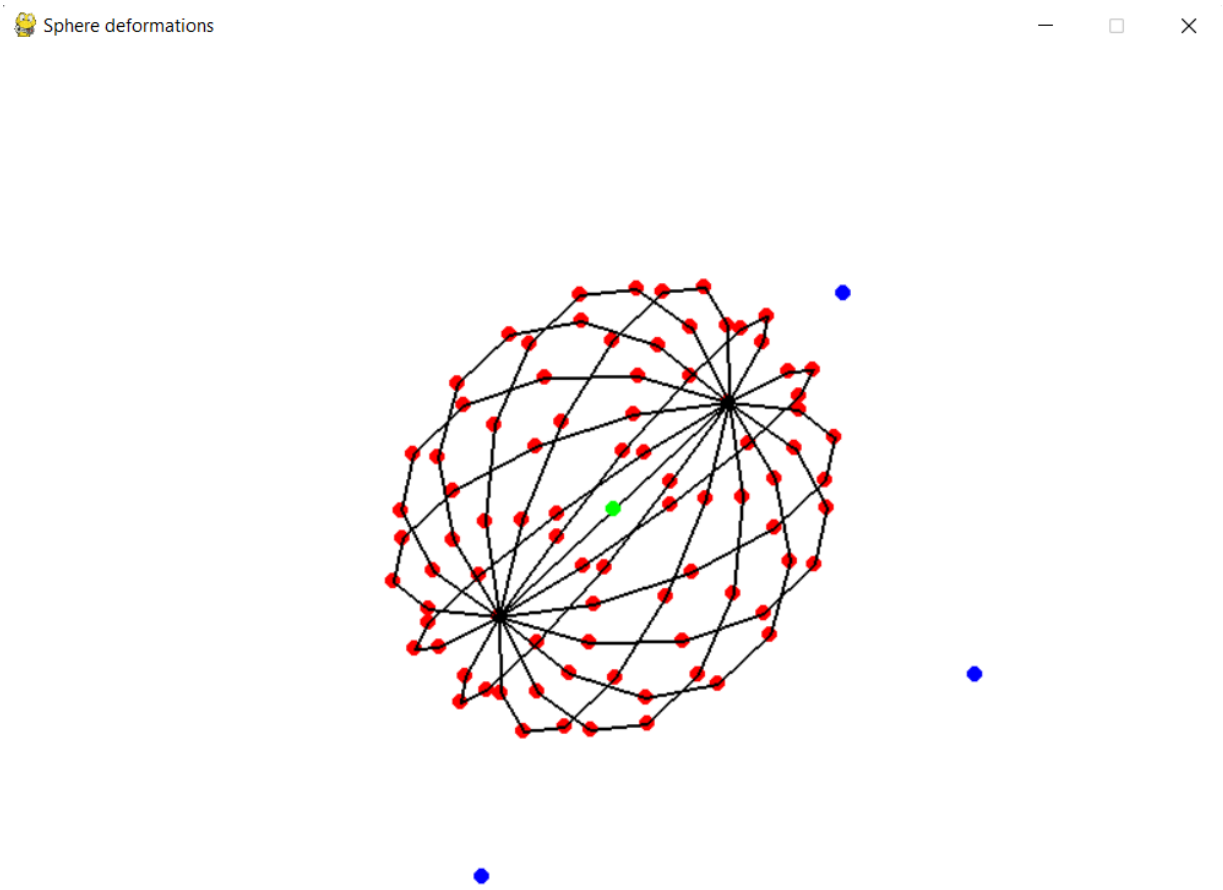


Рисунок 7.10 - Видозмінена куля побудована з 7 граней

Важливим є коректний вибір кількості граней, що зображує рисунок 7.11, аби зберегти достатній рівень розрахунків і при цьому не чекати занадто довго між зміною точки базису та відображенням перетвореної кулі на екран.

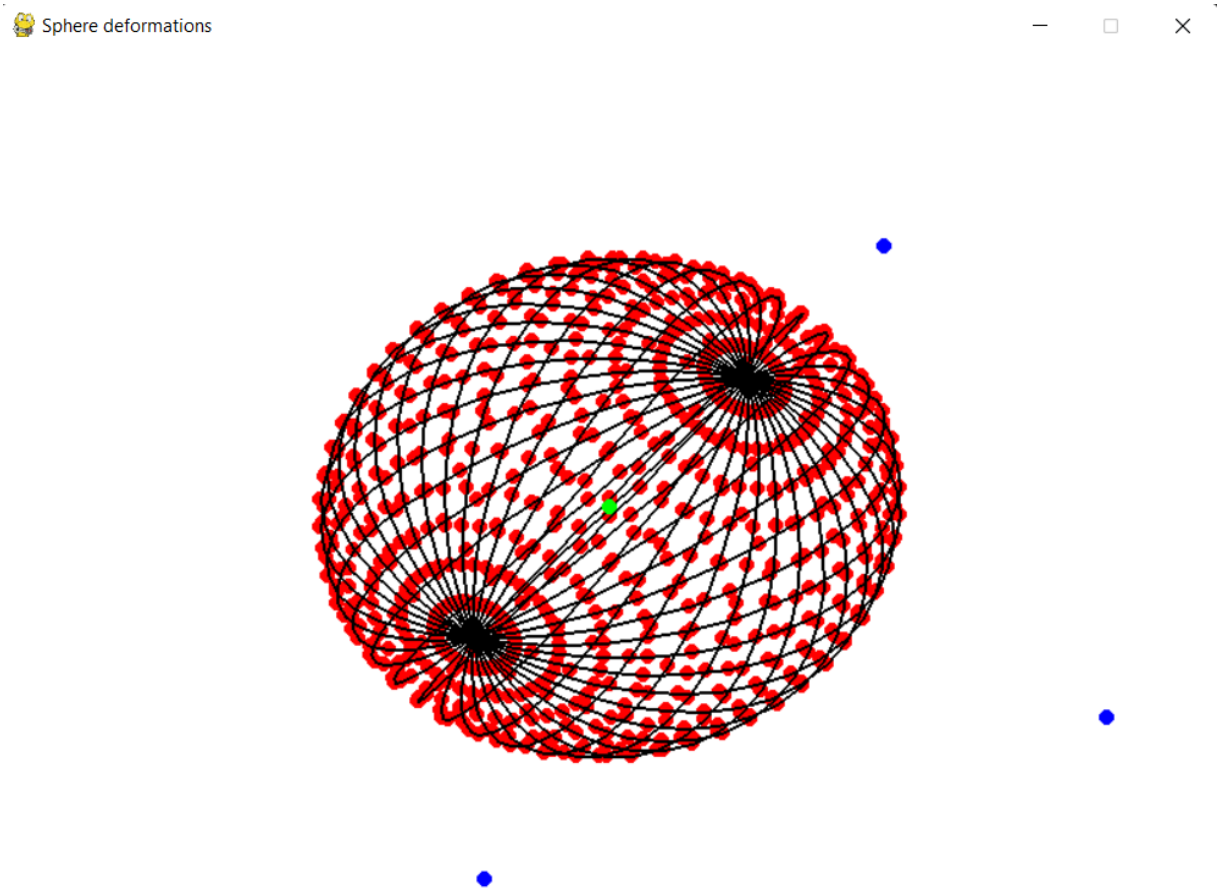


Рисунок 7.11 - Видозмінена куля побудована з 20 граней

Для звичайного користувача оптимальним значенням кількості граней є 15-25, в заданому діапазону комп’ютер миттєво оброблює дані та проводить розрахунки, а результат на екрані достатньо точний.

ВИСНОВКИ

Під час роботи над системою було проаналізовано предметну область, декомпозовано задачу, а також спроектовано та реалізовано систему, що виконує усі поставлені задачі. Система надає можливість відображати перетворення політочковим методом, що були виконані користувачем над кулею, змінювати загальний вигляд кулі шляхом зміни кількості граней з яких вона будується. В майбутньому, з метою покращення системи можуть бути додані наступні можливості:

- підтримка перетворень над іншими геометричними фігурами;
- впровадження інших методів деформаційного перетворення;
- надання можливості додавати точки базису під час роботи;

Даний проект було реалізовано мовою програмування Python з використанням фреймворку NumPy для проведення розрахунків. Інтерфейс було реалізовано з використанням графічної бібліотеки PyGame.

Було обґрунтовано технології розробки системи, які були обрані для реалізації поставленої задачі, а саме для розробки системи зміни кулі у тривимірному просторі, були описані можливості обраних інформаційних технологій, визначені необхідні бібліотеки та ресурси.

Порівнюючи розроблену систему з існуючими аналогами, можна виділити наступні переваги:

- доступний інтерфейс;
- мінімальні вимоги до системи користувача;
- висока швидкість розрахунків;
- актуальність алгоритмів системи;

Користувачами розробленого додатку можуть бути дослідники, інженери, співробітники відділу моделювання або будь-які фахівці, що безпосередньо працюють у сфері деформаційного моделювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Залевська О.В., Яблонський П.Н., Фіногенов О.Д., Сидоренко Ю.В, Ситник А.Ю. Реалізація алгоритму фрактального стиснення графічного зображення. *Сучасні проблеми моделювання*. 2021. Вип.22. С.8-11.
2. Залевська О.В., Яблонський П.М., Сидоренко Ю.В., Мірошниченко І.В., Ситник А.Ю. Удосконалення методу та алгоритму фрактального стиснення графічного зображення. *Прикладна геометрія та інженерна графіка*. 2021. Вип.100. С.118-125.
3. Сидоренко Ю.В. Геометричне моделювання складних об'єктів на основі політочкових відображень відрізків прямих/ Ю.В.Сидоренко, Ю. І. Бадаєв // Сучасні проблеми моделювання: зб. наук. праць.- Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2019. -Вип.16.-С.17-24.
4. Сидоренко, Ю., Залевська, О., Городецький, М., & Спірінцев, Д. Аналіз поведінки похибки обчислень при Гаус-інтерполяції функції Рунге. *Сучасні проблеми моделювання*, 2022, 159-167.
5. Сидоренко Ю.В. Підвищення точності алгоритму політочкових перетворень / Ю.В. Сидоренко, О.В. Залевська // Прикладна геометрія та інженерна графіка — К.:КДТУБА, 2020, вип.97 — С.129-135.
6. Сидоренко Ю.В. Аналіз роботи алгоритму інтерполяційної функції Гауса на елементарних алгебричних функціях/ Ю.В.Сидоренко, М. В. Городецький // Сучасні проблеми моделювання: зб. наук. праць.- Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2020.- Вип.19.-С.138-145.
7. Al Sweigart. Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners. No Starch Press; 2nd edition, 2019.
8. Al Sweigart. Making Games with Python & Pygame. Createspace

Independent Pub, 2012.

9. Allen Downey. Think Python: How to think like a computer scientist. O'Reilly Media; 2st edition, 2016.
10. Dan Bader. Python Tricks: A Buffet of Awesome Python Features. Dan Bader; 1st edition 2017.
11. David Beazley, Guido Van Rossum. Python: Essential Reference. New Riders Publishing, 1999.
12. David Beazley. Python Essential Reference. Addison-Wesley Professional; 4th edition, 2009.
13. Eli Bressert. Scipy And NumPy: An Overwiev for Developers Paperback. O'Reilly Media; 1st edition, 2012.
14. Eric Matthes. Python Crash Course, 2nd Edition: A Hands-On, Project- Based Introduction to Programming. No Starch Press; 2nd edition, 2019.
15. Florian Dedov. The Python Bible 7 in 1: Volumes One To Seven. Independently published, 2020.
16. Francois Chollet. Deep Learning with Python, Second Edition. Manning; 2nd edition, 2021.
17. Ivan Idris. Instant Pygame for Python Game Development How-to. Packt, 2013.
18. Ivan Idris. NumPy CookBook. Packt, 2012.
19. Ivan Van Laningham. Teach Yourself Python in 24 Hours. Sams, 2000.
20. Justin Seitz, Tim Arnold. Black Hat Python, 2nd Edition: Python Programming for Hackers and Pentesters. No Starch Press; 2nd edition, 2021.
21. Matt Harrison. Effective PyCharm: Learn the PyCharm IDE with a Hands-on Approach (Treading on Python). Independently published, 2019.
22. Mark Lutz. Learning Python, 5th Edition. O'Reilly Media; Fifth edition, 2013.
23. Pedro Kroger. Modern Python Development With PyCharm By. Independently published, 2015.

24. Roberto Ulloa. Kivy: Interactive Applications in Python / Roberto Ulloa. – Birmingham: Packt Publishing, 2013. – 122 c.
25. Sean McManus. Mission Python: Code a Space Adventure Game. No Starch Press, 2018.
26. Sloan Kelly. Python, PyGame and Raspberry Pi Game Development. Apress; 1st edition, 2002.
27. Tony Gaddis. Starting Out with Python. Pearson; 4th edition, 2017.
28. Will McGugan. Beginning Game Development with Python and Pygame: From Novice to Professional. Apress; 1st edition, 2007.
29. Will McGugan, Harrison Kinsley. Beginning Python Games Development, Second Edition: With PyGame. Apress; 2nd ed. edition, 2015.
30. Zed Shaw. Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code. Addison-Wesley Professional; 1st edition, 2017.

ДОДАТОК А

Візуалізація тривимірних політочкових перетворень кулі

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР-82359_22Б 12-1

Листів 6

2022

Позначення	Назва	Примітки
Документація		
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_81-1	Записка -Transformations.doc – Пояснювальна записка	
Компоненти		
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_12-1	Sphere.py—текст програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_13-1	Sphere.doc—описання програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_12-2	Points.py—текст програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_13-2	Points.doc—описання програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_12-3	main.py—текст програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_13-3	main.doc—описання програми	
УКР.НТУУ”КПИ”.ТЭФ.КН0122_04Б_14-1	ПоліточковіПеретворення.doc – описання програми	

Текст програми „main.py”

```
---------------------------------------------- налаштування вигляду вікна програми
def intro():
    pygame.init()
    clock = pygame.time.Clock()

    screen = pygame.display.set_mode([800, 600])
    base_font = pygame.font.Font(None, 32)
    user_text = ''
    input_rect = pygame.Rect(330, 268, 140, 32)
    color_active = pygame.Color('lightskyblue3')

    color_passive = pygame.Color('chartreuse4')
    color = color_passive

    active = False
------------------------------------------------------------------------ завершення програми
while True:
    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            if input_rect.collidepoint(event.pos):
                active = True
            else:
                active = False

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K_BACKSPACE:
                user_text = user_text[:-1]
            elif event.key == 13:
                return user_text
            else:
                user_text += event.unicode
    screen.fill((255, 255, 255))



-----------------------------------------------------налаштування початкового екрану
if active:
    color = color_active
else:
    color = color_passive

pygame.draw.rect(screen, color, input_rect)
text_surface = base_font.render(user_text, True, (255, 255, 255))
screen.blit(text_surface, (input_rect.x + 5, input_rect.y + 5))
input_rect.w = max(100, text_surface.get_width() + 10)
pygame.display.flip()
-----------------------------------------------------налаштування початкового
екрану

if __name__ == "__main__":
    while True:
        try:
            n = intro()
            main(int(n))
        except TypeError:
            continue
        except ValueError:
            continue
```

---------- початкова фігура і точки базису

```
def main(num):
    WHITE = (255, 255, 255)
    RED = (255, 0, 0)
    BLUE = (0, 0, 255)
    GREEN = (0, 255, 0)
    BLACK = (0, 0, 0)

    WIDTH, HEIGHT = 800, 600
    pygame.display.set_caption("Sphere deformations")
    screen = pygame.display.set_mode((WIDTH, HEIGHT))

    screen.fill(WHITE)


    projection_matrix = np.matrix([
        [1, 0, 0],
        [0, 1, 0]
    ])

    scale = 100
    circle_pos = [WIDTH/2, HEIGHT/2]
    angle = 0

    cp0 = ControlPoint(Point(0, 0, 0))
    cpx = ControlPoint(Point(2, 0, 0))
    cpy = ControlPoint(Point(0, 2, 0))
    cpz = ControlPoint(Point(0, 0, 2))

    cp0touch = cpxtouch = cpytouch = cpztouch = False
    cp0check = cpxcheck = cpycheck = cpzcheck = False
    deforming = False

    s = Sphere(ptsPerPi=num)
    pts = s.getPoints()
    spherePoints = []
    for p in pts:
        spherePoints.append(p.toMatrix())
        cpx.deforms(p)
        cpy.deforms(p)
        cpz.deforms(p)


    projected_spherePoints = [
        [n, n] for n in range(len(spherePoints))
    ]

    projected_controlPoints = [
    [n, n] for n in range(4)
    ]


    def connect_points(i, j, spherePoints, width=2):
        pygame.draw.line(
            screen, BLACK, (spherePoints[i][0], spherePoints[i][1]),
(spherePoints[j][0], spherePoints[j][1]), width=width)


    def distance2(x1, y1, x2, y2):
```

---------- обробка повороту фігури

```
clock = pygame.time.Clock()
while True:
    clock.tick(60)

    spherePoints = []
```

```
    for p in pts:
        spherePoints.append(p.toMatrix())

    rotation_z = np.matrix([
        [cos(angle), -sin(angle), 0],
        [sin(angle), cos(angle), 0],
        [0, 0, 1],
    ])

    rotation_y = np.matrix([
        [cos(angle), 0, sin(angle)],
        [0, 1, 0],
        [-sin(angle), 0, cos(angle)],
    ])

    rotation_x = np.matrix([
        [1, 0, 0],
        [0, cos(angle), -sin(angle)],
        [0, sin(angle), cos(angle)],
    ])

    rotated2d = np.dot(rotation_z, cp0.toMatrix().reshape((3, 1)))
    rotated2d = np.dot(rotation_y, rotated2d)
    rotated2d = np.dot(rotation_x, rotated2d)
    projected2d = np.dot(projection_matrix, rotated2d)
    x = int(projected2d[0][0] * scale) + circle_pos[0]
    y = int(projected2d[1][0] * scale) + circle_pos[1]
    cp0_coords = (x, y)

    rotated2d = np.dot(rotation_z, cpx.toMatrix().reshape((3, 1)))
    rotated2d = np.dot(rotation_y, rotated2d)
    rotated2d = np.dot(rotation_x, rotated2d)
    projected2d = np.dot(projection_matrix, rotated2d)
    x = int(projected2d[0][0] * scale) + circle_pos[0]
    y = int(projected2d[1][0] * scale) + circle_pos[1]
    cpx_coords = (x, y)

    rotated2d = np.dot(rotation_z, cpy.toMatrix().reshape((3, 1)))
    rotated2d = np.dot(rotation_y, rotated2d)
    rotated2d = np.dot(rotation_x, rotated2d)
    projected2d = np.dot(projection_matrix, rotated2d)
    x = int(projected2d[0][0] * scale) + circle_pos[0]
    y = int(projected2d[1][0] * scale) + circle_pos[1]
    cpy_coords = (x, y)

    rotated2d = np.dot(rotation_z, cpz.toMatrix().reshape((3, 1)))
    rotated2d = np.dot(rotation_y, rotated2d)
    rotated2d = np.dot(rotation_x, rotated2d)
    projected2d = np.dot(projection_matrix, rotated2d)
    x = int(projected2d[0][0] * scale) + circle_pos[0]
    y = int(projected2d[1][0] * scale) + circle_pos[1]
    cpz_coords = (x, y)



------------------------------------------------------------------------ обробка івентів
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        exit()
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
            pygame.quit()
            exit()
    if event.type == pygame.MOUSEMOTION:
        tempx, tempy = pygame.mouse.get_pos()
        if distance2(cp0_coords[0], cp0_coords[1], tempx, tempy)<10:
            cp0touch = True
```

```
            else:
                cp0touch = False
            if distance2(cpx_coords[0], cpx_coords[1], tempx, tempy) < 10:
                cpxtouch = True
            else:
                cpxtouch = False
            if distance2(cpy_coords[0], cpy_coords[1], tempx, tempy) < 10:
                cpytouch = True
            else:
                cpytouch = False
            if distance2(cpz_coords[0], cpz_coords[1], tempx, tempy) < 10:
                cpztouch = True
            else:
                cpztouch = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            mx, my = pygame.mouse.get_pos()
            cp0check = cp0touch
            cpxcheck = cpxtouch
            cpycheck = cpytouch
            cpzcheck = cpztouch
            if cp0check or cpxcheck or cpycheck or cpzcheck:
                deforming = True
            else:
                deforming = False
        if event.type == pygame.MOUSEBUTTONUP:
            mx1, my1 = pygame.mouse.get_pos()
            distance = sqrt((mx1-mx)**2+(my1-my)**2)
            if (mx1-cp0_coords[0])**2 + (my1-cp0_coords[1])**2 < (mx-cp0_coords[0])**2
    + (my-cp0_coords[1])**2:
                distance *= -1
            if not deforming:
                angle += 0.001*distance
            else:
                if cp0check:
                    pass
                elif cpxcheck:
                    cpx.move(distance/100, 0, 0)
                    cpx.deform()
                elif cpycheck:
                    cpy.move(0, distance/100, 0)
                    cpy.deform()
                elif cpzcheck:
                    cpz.move(0, 0, distance/100)
                    cpz.deform()

    screen.fill(WHITE)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                pygame.quit()
                exit()
        if event.type == pygame.MOUSEMOTION:
            tempx, tempy = pygame.mouse.get_pos()
            if distance2(cp0_coords[0], cp0_coords[1], tempx, tempy)<10:
                cp0touch = True
            else:
                cp0touch = False
            if distance2(cpx_coords[0], cpx_coords[1], tempx, tempy) < 10:
                cpxtouch = True
            else:
                cpxtouch = False

            if distance2(cpy_coords[0], cpy_coords[1], tempx, tempy) < 10:
                cpytouch = True
            else:
                cpytouch = False
            if distance2(cpz_coords[0], cpz_coords[1], tempx, tempy) < 10:
```

```
            cpztouch = True
        else:
            cpztouch = False
    if event.type == pygame.MOUSEBUTTONDOWN:
        mx, my = pygame.mouse.get_pos()
        cp0check = cp0touch
        cpxcheck = cpxtouch
        cpycheck = cpytouch
        cpzcheck = cpztouch
        if cp0check or cpxcheck or cpycheck or cpzcheck:
            deforming = True
        else:
            deforming = False
    if event.type == pygame.MOUSEBUTTONUP:
        mx1, my1 = pygame.mouse.get_pos()
        distance = sqrt((mx1-mx)**2+(my1-my)**2)
        if (mx1-cp0_coords[0])**2 + (my1-cp0_coords[1])**2 < (mx-cp0_coords[0])**2
+ (my-cp0_coords[1])**2:
            distance *= -1
        if not deforming:
            angle += 0.001*distance
        else:
            if cp0check:
                pass
            elif cpxcheck:
                cpx.move(distance/100, 0, 0)
                cpx.deform()
            elif cpycheck:
                cpy.move(0, distance/100, 0)
                cpy.deform()
            elif cpzcheck:
                cpz.move(0, 0, distance/100)
                cpz.deform()


------------------------------------------ оновлюємо вигляд кулі після перетворень
  screen.fill(WHITE)
# drawining stuff

i = 0
for point in spherePoints:
    rotated2d = np.dot(rotation_z, point.reshape((3, 1)))
    rotated2d = np.dot(rotation_y, rotated2d)
    rotated2d = np.dot(rotation_x, rotated2d)

    projected2d = np.dot(projection_matrix, rotated2d)

    x = int(projected2d[0][0] * scale) + circle_pos[0]
    y = int(projected2d[1][0] * scale) + circle_pos[1]

    projected_spherePoints[i] = [x, y]
    pygame.draw.circle(screen, RED, (x, y), 5)
    i += 1

for p in range(len(projected_spherePoints)-1):
    connect_points(p, p + 1, projected_spherePoints)

pygame.draw.circle(screen, GREEN*cp0touch + GREEN*(1-cp0touch), cp0_coords, 5)
pygame.draw.circle(screen, GREEN*cpxtouch + BLUE*(1 - cpxtouch), cpx_coords, 5)
pygame.draw.circle(screen, GREEN*cpytouch + BLUE*(1 - cpytouch), cpy_coords, 5)
pygame.draw.circle(screen, GREEN*cpztouch + BLUE*(1 - cpztouch), cpz_coords, 5)


pygame.display.update()
```