Міністерство освіти і науки України Департамент освіти і науки облдержадміністрації Комунальний позашкільний навчальний заклад

Відділення: Інформатика та програмування

Секція: Програмна інженерія

БУДУВАННЯ ТА ТРАНСФОРМАЦІЯ 3D ПРИМІТИВІВ У WPF

Роботу виконав:

Найденко Владислав Павлович,

учень 10-А класу комунального закладу «Дніпровський науковий ліцей

інформаційних технологій Дніпровської міської ради»

Дніпропетровської області

Науковий керівник:

Лавренюк Iрина Валерiївна, учитель інформатики комунального закладу «Дніпровський науковий ліцей

інформаційних технологій Дніпровської міської ради»

Дніпропетровської області

Дніпро – 2022

# АНОТАЦІЯ

БУДУВАННЯ ТА ТРАНСФОРМАЦІЯ 3D ПРИМІТИВІВ У WPF

Найденко Владислав Павлович   
Департамент освіти і науки облдержадміністрації

Комунальний позашкільний навчальний заклад

Дніпровський ліцей інформаційних технологій при ДНУ ім. Олеся Гончара

10-А клас, м. Дніпро

Керівник: Лавренюк Iрина Валерiївна, вчитель інформатики ДЛІТ при ДНУ ім. Олеся Гончара, вчитель

Метою дослідження було демонстрація графічних можливостей WPF на прикладі зображення деяких тривимірних фігур а саме сфери, циліндру, конусу та тору, а також можливості їх подальшої трансформації.

Виконано аналіз інтернет-джерел щодо використання мови XAML та WPF у C# а також математичного апарату що дозволяє описувати тривимірні трансформації.

В науково-дослідній роботі проведені дослідження алгоритмів побудови тривимірних примітивів та їх трансформації, та створений відповідний додаток з можливістю демонстрацій відповідних графічних побудов засобами WPF на C#.

Ключові слова: тривимірний примітив, XAML, WPF, матриця трансформації, матриця переходу, однорідні координати, сферична система координат, циліндрична система координат.

# ЗМІСТ

[АНОТАЦІЯ 2](#_Toc124018626)

[ЗМІСТ 3](#_Toc124018627)

[ВСТУП 4](#_Toc124018628)

[РОЗДІЛ 1 5](#_Toc124018629)

[1.1 Механізм побудови 3D фігур у WPF(тріангуляція) та система координат 5](#_Toc124018630)

[1.1.1 Камера 5](#_Toc124018631)

[1.1.2 Освітлення 6](#_Toc124018632)

[1.1.3 Тріангуляція 7](#_Toc124018633)

[1.2 Алгоритми побудови геометричних фігур 7](#_Toc124018634)

[1.2.1 Сфера 7](#_Toc124018635)

[1.2.2 Циліндр 9](#_Toc124018636)

[1.2.3 Конус 12](#_Toc124018637)

[1.2.4 Тор 13](#_Toc124018638)

[РОЗДІЛ 2 16](#_Toc124018639)

[2.1 Матриця трансформації 16](#_Toc124018640)

[2.2 Однорідні координати 17](#_Toc124018641)

[2.3 Різні види матриць трансформації 19](#_Toc124018642)

[РОЗДІЛ 3 21](#_Toc124018643)

[3.1 Середовище розробки та мова програмування 21](#_Toc124018644)

[3.2 Бібліотеки 21](#_Toc124018645)

[3.3 Інструменти роботи з 3D графікою у WPF 21](#_Toc124018646)

[3.4 Структура програми та код 22](#_Toc124018647)

[3.5 Робота програми 24](#_Toc124018648)

[ВИСНОВОК 25](#_Toc124018649)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 26](#_Toc124018650)

[ДОДАТОК А 27](#_Toc124018651)

[ДОДАТОК Б 28](#_Toc124018652)

# ВСТУП

**Актуальність** використання WPF полягає в тому що він використовує більш розширенні інструменти для вимальовування графіки. Якщо програми на основі WinForms для відображення графічних елементів використовують GDI+ то за рендеринг графічних елементів в WPF відповідає DirectX. Який для рендерингу графіки використовує графічний процесор відеокарти що дозволяє швидко вимальовувати навіть складні 3D об’єкти.

Ідеєю цього проекту є створення програми, що показувала би можливості створення тривимірних примітивів та їх трансформацій у WPF.

***Мета*** курсової роботи є створення програми з таким функціоналом:

* + 1. Відображення тривимірних фігур на екрані користувача за введеними даними.
    2. Трансформація відповідних фігур ( див. пункт 1) за даними користувача та відображення цих змін у реальному часі.

***Задачі***:

* Дослідити математичні засоби для малювання тривимірних моделей.
* Реалізувати алгоритми побудови тривимірних зображень на мові C# за допомогою WPF.
* Дослідити математичні засоби та нюанси тривимірних трансформацій з точки зору математики та їх реалізації у WPF.
* Реалізація трансформацій у реальному часі за даними що задані користувачем.

# РОЗДІЛ 1

ОГЛЯД ВІДОМОСТЕЙ ПРО ПОБУДОВУ ТРИВИМІРНИХ ФІГУР

## 1.1 Механізм побудови 3D фігур у WPF(тріангуляція) та система координат

3D моделювання це процес при якому здійснюється створення тривимірної моделі об'єкта.

Для побудови 3D моделі потрібно хоча б визначитися з наступними параметрами:

1. Освітлення
2. Камера
3. Моделі

У WPF визначення позиції об’єкту на сцені відбувається за допомогою тривимірної правосторонньої декартової системи координат. Однак слід відрізняти стаціонарну систему відліку від локальної.

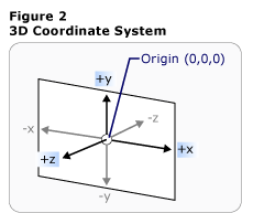


Рис. 1.1 Тривимірна стаціонарна система координат у WPF

### Камера

У WPF камери мають дві головні властивості, місце знаходження камери у тривимірній сцені та напрямком направлення камери що задається тривимірним вектором. Також можна задати вертикальний кут нахилу.

Існує три типи камер у WPF:

* *Камера з перспективною проекцією* – при такому виді камери об’єкти що знаходяться дещо далі від камери здаються меншими ніж об’єкти що до неї розташовані ближче.
* Камера з ортогональною проекцією – об’єкти що знаходяться далі від камери, та об’єкти що знаходяться ближче камери мають однаковий розмір.
* Матрична камера – дозволяє задавати своє відображення тривимірних об’єктів на площину екрану.

|  |  |
| --- | --- |
| Orthographic and perspective projection  а)Ортогональна проекція | Orthographic and perspective projectionб)Перспективна проекція |

Рис. 1.2. Приклад ортогональної й перспективної проекції

Рис. Приклд ортогональної й перспективної проекції

Ми будемо використовувати у проекті камеру з перспективною проекцією.

### Освітлення

Освітлення є важливою частиною тривимірної сцени, вона виконує ту ж роль що і світло у реальному житті. Якщо ж воно не встановлено то тривимірні моделі будуть просто невидимі, також деякі види освітлення дають ефект тіні. Усього у WPF існують чотири види освітлення:

* Розсіяне освітлення – всі об’єкти освітлюються однаково з усіх боків
* Напрямлене освітлення – створює напрямлене світло напрямок якого задається за допомогою вектору.
* Точкове освітлення – створює світло в певному діапазоні від заданих точок, також можна зробити ефект згасання освітлення в залежності від відстані до джерела світла.
* Точкове освітлення з напрямленням – задає напрямлене світло, від заданих точок.

У нашому випадку ми будемо використовувати напрямлене освітлення.

### Тріангуляція

Будь яка побудова у WPF здійснюється за допомогою трикутної полігональної сітки, яка математично є графом вершини якого є вершинами трикутників а ребра графу є ребрами трикутників. Тож для того щоб побудувати будь-яку тривимірну модель, потрібно вказати координати вершин трикутників з яких вона складається, а потім відповідно ребра між ними.

Також треба відмітити що фронтальне направлення задається порядком перерахування точок. Отже для того щоб трикутник “дивився” на нас потрібно перераховувати точки проти часової стрілки відносно осі Z, а від нас – по часовій стрільці відповідно.

## 1.2 Алгоритми побудови геометричних фігур

### 1.2.1 Сфера

Сфера – поверхня яка складається з точок що рівновіддалені від однієї точки яка називається центром сфери.

Сферична система координат – система координат на якій точка задається за допомогою трьох координат.

1. R-відстань від початку координат до точки
2. ϴ - зенітний кут, це кут між радіус-вектором та віссю Z
3. φ – азимутальний кут, це кут між проекцією радіус вектору на площину XY та віссю X.

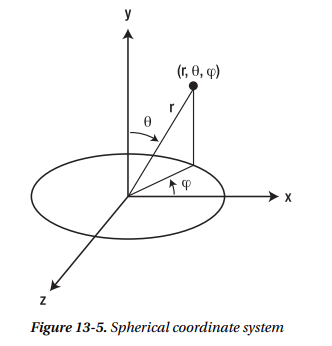


Рис. 1.3. Сферична система координат

Перехід до тривимірної декартової системи координат в цій роботі здійснюється за допомогою наступних формул:

; ; (1.2)

Для того щоб побудувати апроксимацію сфери потрібно знайти деякі ключові точки у просторі які будуть вершинами полігональної сітки. Для цього перейдемо до сферичної системи координат із центром у точці, що є центром сфери.

Радіус координат точок буде дорівнювати радіусу сфери, а кут φ кожної точки буде змінюватись від 0 включно до 2 включно на крок , кут ϴ кожної точки буде змінюватись від 0 до із кроком , n та m кількість точок що утворюють полігональну сітку по горизонталі та вертикалі відповідно. Координати цих точок ai,j можна для зручності розуміння занести до матриці A розміром n ✕ m де, стовпці будуть приймати значення пов’язані з кутом φ, а рядки – з кутом ϴ. Хоча при реалізації цього алгоритму на мові програмування двовимірний масив можна не створювати, а одразу рахувати і будувати, це дозволить пришвидшити алгоритм. За допомогою формул наведених вище перейдемо до декартової системи координат з центром у центрі сфері та потім перейдемо до світової тривимірної системи координат додавши відповідні координати центру сфери до координат точок. Таким чином ми отримали деяку множину точок сфери у основній декартовій системі координат.

Наступним кроком буде побудова множини чотирикутників на базі матриці A за наступним правилом: ai,j; ai+1,j; ai+1,j+1; ai,j+1, де i,j ∈ N та 1≤i≤n,1≤j≤m (рис. 1.4).

Ми вже говорили що у WPF побудови здійснюються за допомогою трикутників, через них ми можемо перейти до побудови чотирикутників, якщо обрати два трикутники зі спільним ребром.

Рис. Приклад чотирикутника

Рис. Приклад каркасу сфери

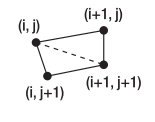


Рис. 1.4. Приклад прямокутного полігону

Відповідно, щоб побудувати чотирикутник потрібно знати координати чотирьох його вершин. Таким чином ми побудували площину що наближено є сферою(рис. 1.5.).

Зображення, що містить пристрій, вентилятор, світлий, силует

Description automatically generatedТакож треба зазначити, що при наближені n та m до нескінченності, побудована площина буде наближатися до сфери.

Рис. 1.5. Полігональна сітка сфери

### 1.2.2 Циліндр

Циліндр - тіло, що знаходиться між двох площин, та обмежується циліндричною поверхнею.

Циліндрична система координат – система координат на якій точка задається за допомогою трьох координат:

1. Радіальна відстань – відстань від початку координат та перпендикулярної проекції точки на площину XZ.
2. Азимут — кут між віссю OX, та прямою яка містить точку початку координат і точку проекції заданої точки на площину XZ.
3. Висота ***y***(див. рис. 1.6.) дорівнює відстані від XZ до заданої точки.

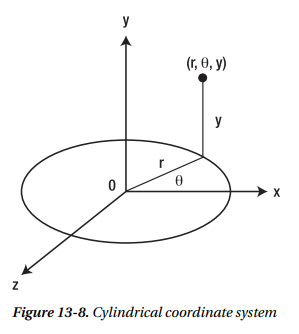


Рис. 1.6. Циліндрична система координат

Перехід до тривимірної декартової системи координат в нашому випадку здійснюється за допомогою наступних формул:

; ; (1.2.2.).

При побудові в графіці циліндр складається з елементів чотирикутних призм, що розташовані по колу і кількість яких дорівнює n. Утворюючи тим самим внутрішню циліндричну площину та зовнішню циліндричну площину. Для того щоб побудувати апроксимацію циліндру із можливістю отвору, аналогічно сфері потрібно знайти деякі ключові точки у просторі які будуть вершинами полігональної сітки. Для цього перейдемо до циліндричної системи координат із центром у точці що є центром симетрії циліндру.

Будемо змінювати φ від 0 до 2 радіан включно, на крок . Причому при одному значенні φ будуть визначатися чотири точки, дві точки з них задають внутрішню циліндричну площину, відповідно їх радіальна відстань дорівнює радіусу отвору, а висота дорівнює h/2 та -h/2. І дві точки що задають зовнішню циліндричну поверхню, радіальна відстань яких дорівнює радіусу циліндру, а висота дорівнює відповідно h/2 та -h/2. Координати цих точок можна для зручності розуміння занести до матриці A розміром n+1 ✕ 4 де, рядки будуть приймати значення пов’язані з кутом . Хоча при реалізації цього алгоритму на мові програмування двовимірний масив можна не створювати, а одразу рахувати і будувати, це дозволить пришвидшити алгоритм.

Наступним кроком буде з’єднання цих точок у правильній послідовності, так як наша побудова складається з чотирикутних призм які з трикутників будуються наступним чином(рис. 1.7.). Для її побудови потрібно знати координати восьми вершин призми. Однак необов’язково будувати всю призму а лише сторони що є видимими для користувача. Будуємо прямокутники на базі матриці A за наступним правилом:

ai,1; ai,2 ai+1,1; ai+1,2(зовнішня поверхня);

ai,2; ai,3; ai+1,2; ai+1,3(нижня поверхня);

ai,1; ai,4; ai+1,1; ai+1,4(верхня поверхня);

ai,3; ai,4; ai+1,3; ai+1,4,

що задовольняють i ∈ N та 1≤i≤n. Таким чином ми побудували фігуру що наближено є циліндром з отвором(рис. 1.7.).

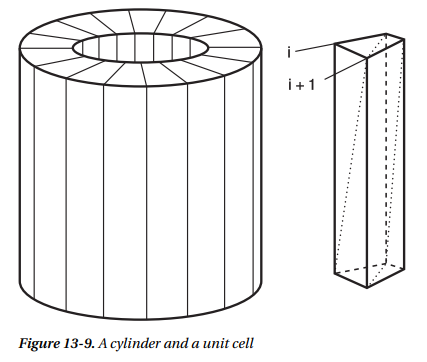


Рис. 1.7. Полігональна сітка циліндру та прямокутна призма

Також треба зазначити, що при наближені n до нескінченності побудована фігура буде наближатися до циліндру з отвором.

### 1.2.3 Конус

Зрізаний конус — тіло, що знаходиться між площиною що перетинає конус та паралельна основі цього конуса і основою конуса.

Для побудови зрізаного конуса будемо використовувати наступні дані: центр зрізаного конуса(точка що ділить висоту конуса на два рівні відрізки), висота, радіус основи, радіус вершини та кількість полігонів по горизонталі(n). Ми будемо створювати зрізаний конус, однак вказавши для радіуса вершини конуса значення 0, можна отримати стандартний конус. Зрізаний конус буде складатися з трикутних призм кількість яких дорівнює n і які розташовані по колу. Однак нам не обов’язково будувати всі сторони наших призми а тільки нижню та верхню грань і бічну грань що буде утворювати сам конус (сторони що є видимими для користувача), так можна оптимізувати алгоритм. Так як побудова складається з трикутних призм, то необхідно знати 6 точок, дві з них є сталими для одного зрізаного конуса. А саме, точка вершини конуса та точка центру основи конуса. Далі потрібно буде знайти координати точок що лежать на колі що утворює вершину зрізаного конуса, та на колі що утворює основу зрізаного конуса. Для цього перейдемо до циліндричної системи координат із центром в центрі зрізаного конуса. Та почнемо перебирати точки, буде змінюватись від 0 до 2 включно кроком 2/n, n – кількість полігонів що утворюють конус по горизонталі. Радіальна відстань однієї точки буде дорівнювати радіусу вершини зрізаного конуса, а радіальна відстань другої – радіусу основі зрізаного конуса. Координати цих точок можна для зручності розуміння занести до матриці A розміром n+1 ✕ 2 де, рядки будуть приймати значення пов’язані з кутом , перший стовбець – значення пов’язані з колом що утворюють вершину зрізаного кута, другий – основою зрізаного конуса. Хоча при реалізації цього алгоритму на мові програмування двовимірний масив можна не створювати, а одразу рахувати і будувати, це дозволить пришвидшити алгоритм.

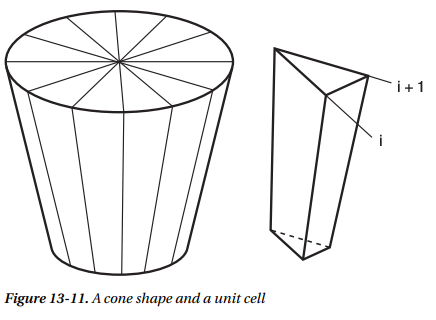


Рис. 1.8. Полігональна сітка конусу та трикутної призма

Потім всі точки потрібно перевести у глобальну тривимірну систему координат, використовуючи формули(див 1.2.2.), та додавши до координат точок відповідні координати центру зрізаного конуса. Наступним кроком буде утворення прямокутників на базі матриці A за наступним правилом: ai;1 ai;2 ai+1;1 ai+1;2. Та трикутників з вершинами: ai;1 ai+1;1 та точкою центру вершини зрізаного конуса. І трикутників з вершинами: ai;2 ai+1;2 та точкою центру основи зрізаного конуса. В результаті утвориться зрізаний конус.

### 1.2.4 Тор

Тор — геометричне тіло, що утворюється при обертанні кола навколо деякої вісі, такої що ось та утворююче коло знаходяться в одній площині, але ця вісь не перетинає коло.

Параметричні рівняння тора задаються за допомогою: відстані від центра тора до осі утворюючого кола(R), радіусу утворюючого кола(r), кута t та кута *p*.

Зображення, що містить текст, транспорт, літальний апарат, зелений

Description automatically generated

Рис. 1.9. Параметри тору в лівосторонній системі координат

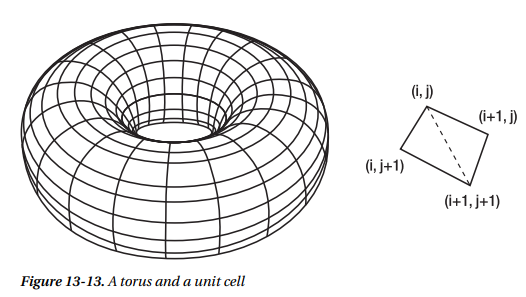


Рис. 1.10. Полігональна сітка тору та полігон

Та в нашому випадку описуються наступними рівняннями:

; ; .

За допомогою вище наведених формул визначимо точки у просторі які будуть вершинами полігональної сітки. Для цього у формули підставимо r та R що є сталими(введені користувачем), кут *t* будемо змінювати від 0 до 2 включно з кроком де n – кількість полігонів що утворюють тор по горизонталі, а кут *p* від 0 до 2 включно з кроком де m – кількість полігонів що утворюють тор по вертикалі. Координати цих точок ai,j можна для зручності розуміння занести до матриці A розміром n+1 ✕ m+1 де, стовпці будуть приймати значення пов’язані з кутом t, а рядки – з кутом p. Хоча при реалізації цього алгоритму на мові програмування двовимірний масив можна не створювати, а одразу рахувати і будувати, це дозволить пришвидшити алгоритм. Наступним кроком буде побудова множини чотирикутників на базі матриці A за наступним правилом ai,j; ai+1,j; ai+1,j+1; ai,j+1, де i,j ∈ N та 1≤i≤n,1≤j≤m (рис.). Таким чином була здійснена побудова площини що наближено є тором. Також треба зазначити що при наближені n та m до нескінченності побудована площина буде наближатися до тору.

# РОЗДІЛ 2

ТРАНСФОРМАЦІЯ ТРИВИМІРНИХ МОДЕЛЕЙ

Трансформації тривимірних об’єктів можна поділити на афінні та неафінні.

* При афінних трансформаціях зберігається паралельність, мимобіжність та пересічність.
* В неафінних трансформаціях взаємо розташування прямих не зберігається.

Наприклад до афінних належать наступні трансформації: повороту, масштабування та зсув.

## 2.1 Матриця трансформації

Радіус-вектор – вектор, початком якого є початок координат, а кінцем є точка положення якої він визначає.

Будь який радіус-вектор можна представити у вигляді суми кількох сталих векторів, ці вектори утворюють базис вектору. Наприклад для тривимірного простору ця сума має наступний вигляд: де, є числовими коефіцієнтами що називаються коефіцієнтами розкладу, а вектори є базисними. Зазвичай базисні вектори це одиничні вектори напрям яких співпадає з напрямками відповідних осей.

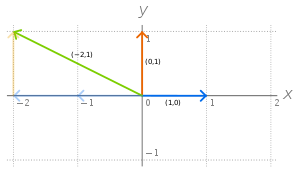


Рис. 2.1 Приклад базису двовимірного простору (синій та помаранчевий)

Для трансформацій зазвичай застосовують так звані матриці трансформації(переходу). Наприклад для того щоб перетворити вектор-стовбець потрібно матрицю трансформації помножити на цей вектор-стовбець, за правилами математики. Матриця трансформації змінює базис, що і дозволяє трансформувати будь-які об’єкти.

Проста матриця трансформації у тривимірному просторі має таку структуру:, де , , нові базисні вектори-стовбці. Наприклад матриця , є одиничною тобто при множені вектору-стовбця на неї він ніяк не зміниться. А ось наприклад , збільшить вектор-стовбець по осі Х у два рази. Матриці трансформацій є зручним математичним інструментом опису трансформацій, так як вони дуже компактні, наприклад нам потрібно вектор-стовбець перетворити спочатку за допомогою однієї матриці, а потім те що отримали за допомогою іншої. В такому випадку нам достатньо помножити матрицю що є добутком матриць трансформації у порядку трансформацій що помножується на вектор-стовбець. Це дозволяє оптимізувати розрахунки та програму, саме тому в DirectX, що є основою WPF, трансформації реалізовані за допомогою матриць.

### 2.2 Однорідні координати

Звичайні матриці трансформації можуть описувати масштабування та повороти однак зміщення в простих матрицях трансформації не реалізовується. Для вирішення цієї проблеми використовують однорідні координати.

Для цього розглянемо двовимірні однорідні координати. Вони складаються з координат u, v, w, де u, v, w числа що одночасно відмінні від нуля. Для того щоб перейти від однорідної до декартової системи координат потрібно u та v поділити на w: x = u/w, y = v/w. Відповідно щоб перейти від декартової системи координат до однорідної потрібно домножити координати x та y на будь-яке число w. Із вище сказаного слідує що одна точка в декартовій системі координат може задаватися безліччю варіантів в однорідних координатах: наприклад координати (1,2,1) та (2,4,2) задають на декартовій системі координат одну і ту ж точку – (1,2). Однак у однорідних координат є ще одна важлива властивість, якщо w = 0, то замість координат точки вже задаються координати вектору. В геометричному плані двовимірні однорідні координати мають три осі u, v, w, декартова площина паралельна площині UV та знаходиться на рівні w = 1(рис. 2.2). Візьмемо на цій декартовій площині точку та проведемо пряму до точки O3 що є центром однорідних координат. Проведена нескінченна пряма є множиною всіх однорідних координат що задають точку , окрім точки O3. На наведеному рисунку точка ***x*** також задає точку .

Зображення, що містить текст, небо, надворі

Description automatically generated

Рис. 2.2 Геометричне представлення однорідних координат

Аналогічно, однорідні тривимірні координати складаються з: координат u, v, w, t. І мають ті ж властивості що двовимірна однорідна система координат. Перехід тепер здійснюється за допомогою наступних формул: x = u/t,   
y =v/t, z = w/t.

Тепер задаючи базиси за допомогою тривимірної однорідної системи координат ми можемо записати зміщення. Для цього розширимо матрицю до розмірів 4 на 4. В загальному вигляді ця матриця буде мати наступний вигляд: , останній стовпець відповідає за зміщення. З теоретичної точки зору замість 1 в останньому рядку може стояти інше число, відмінне від 0, однак з практичної точки зору використання 1 більш оптимальне.

## 2.3 Різні види матриць трансформації

Розглянемо матриці трансформацій що реалізують типові перетворення.

Матриця трансформації що описує зміщення має вигляд: , де Dx,зміщення по осі X, Dy, по осі Y, Dz, по осі Z.

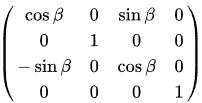
Наприклад := .

Матриця масштабування має наступний вигляд: , де Sx, Sy, Sz відповідно масштабування по осям X,Y та Z.

Матриця обертання навколо осей мають наступний записи:

Навколо осі X: Зображення, що містить текст, годинник

Description automatically generated, де кут α є кутом повороту.

Навколо осі Y: , де кут β кут повороту.

Навколо осі Z: Зображення, що містить текст, годинник

Description automatically generated, де кут γ кут повороту.

# РОЗДІЛ 3

ОГЛЯД КОДУ ПРОГРАМИ

## 3.1 Середовище розробки та мова програмування

Для написання додатку використовувалися мови XAML та C#, фреймворк – WPF. При написанні коду використовувалась IDE Microsoft Visual Studio Community 2019.

## 3.2 Бібліотеки

Для добавлення деяких функцій до проекту було додано необхідні бібліотеки.

Бібліотека *Extended WPF Toolkit* використовується для додавання елементу управління ColorPicker, який відсутній у WPF. ColorPicker використовується для задання кольору текстур для примітивів.

Також була використана бібліотека *3D Tools* для малювання координатних осей у Viewport3D, для орієнтації у просторі.

## 3.3 Інструменти роботи з 3D графікою у WPF

Відображення тривимірної сцени у WPF доступне завдяки елементу керування під назвою **Viewport3D**. **Viewport3D** є класом який має два поля. **CameraProperty** що визначає положення камери, напрямок погляду та її тип, у проекті він описується наступним чином:

<Viewport3D.Camera>

<PerspectiveCamera Position="10,4,30" LookDirection="-0.3,0,-1"/>

</Viewport3D.Camera>

Та поле **Children** яке приймає значення **ModelVisual3D**, який в свою чергу містить поле **Content** та **Transform**. Через поле **Content** ми задали світло та його характеристики. В нашому випадку світло описується наступним XAML кодом:

<ModelVisual3D>

<ModelVisual3D.Content>

<DirectionalLight Color="White" Direction="1,1,-2" />

</ModelVisual3D.Content>

</ModelVisual3D>

Також створюємо ще один екземпляр **ModelVisual3D** який буде містити створені нами примітиви та має назву **model**. Через цю назву будемо звертатися до нього з коду, де будемо задавати властивість **Content**. Ініціалізація model виглядає наступним чином:

<ModelVisual3D x:Name="model">

<ModelVisual3D.Transform>

<Transform3DGroup x:Name="vision">

<RotateTransform3D>

<RotateTransform3D.Rotation>

<AxisAngleRotation3D x:Name="visionX" Axis="0,1,0"/>

</RotateTransform3D.Rotation>

</RotateTransform3D>

<RotateTransform3D>

<RotateTransform3D.Rotation>

<AxisAngleRotation3D x:Name="visionY" Axis="1,0,0"/>

</RotateTransform3D.Rotation>

</RotateTransform3D>

<ScaleTransform3D x:Name="Zoom" CenterX="0" CenterY="0" CenterZ="0"/>

</Transform3DGroup>

</ModelVisual3D.Transform>

</ModelVisual3D>

Поле **Transform** відповідає за збільшення масштабу сцени та обертання її навколо осей, що дасть змогу роздивитися сцену з різних боків. Кут повороту буде задаватися користувачем через код.

## 3.4 Структура програми та код

Програма виконана за допомогою ООП. Побудова та трансформація виконуються через класи: **Shapes**,**Sphere**,**Torus**,**Cone**,**Cylinder**.(див. додаток А)

Shapes – батьківський клас який містить методи трансформації та будування а також поле **model** що представляє собою примітив.

Розглянемо методи цього класу:

//повертає об'єкт навколо осі X на задану кількість градусів

public void RotateX(double AngleX)

//повертає об'єкт навколо осі Y на задану кількість градусів

public void RotateY(double AngleY)

//повертає об'єкт навколо осі Z на задану кількість градусів

public void RotateZ(double AngleZ)

//масштабує об'єкт у 2 рази по осі X у задану кількість разів

public void ScaleX(double ScaleX)

//масштабує об'єкт у 2 рази по осі Y у задану кількість разів

public void ScaleY(double ScaleY)

//масштабує об'єкт у 2 рази по осі Z у задану кількість разів

public void ScaleZ(double ScaleZ)

//зміщує об'єкт на зазначену відстань по осі X

public void OffsetX(double OffsetX)

//зміщує об'єкт на зазначену відстань по осі Y

public void OffsetY(double OffsetY)

//зміщує об'єкт на зазначену відстань по осі Z

public void OffsetZ(double OffsetZ)

//цей метод створює трикутник за трьома точками(po,p1,p2) та додає його до полігональної сітки(mesh), а вершинам присвоюється номер починаючи з count.

protected void CreateTriangleFace(Point3D po, Point3D p1, Point3D p2, MeshGeometry3D mesh, ref int count)

**Sphere** – клас що відповідає за створення сфери.

Методи:

//Створює сферу з заданим кольором(color), центром(center) та радіусом(radius), а також із вказаною кількістю полігонів по вертикалі та горизонталі(u,v)

private void CreateSphere(Point3D center, double radius, int u, int v, Color color, ref Model3DGroup All)

//Переведення сферичних координат у декартові.

private Point3D GetPositionSP(double radius, double theta, double phi)

**Cylinder** – клас що відповідає за створення циліндру.

Методи:

//Створює циліндр з заданим кольором(color), центром(center), радіусами циліндру та порожнини(rout,rin), висотою(height), а також із вказаною кількістю полігонів горизонталі(n).

private void CreateCylinder(Point3D center, double rin, double rout, double height, int n, Color color, ref Model3DGroup All)

//Переведення циліндричних координат у декартові.

private Point3D GetPositionCY(double radius, double theta, double y)

**Cone** – клас що відповідає за створення конусу.

Методи:

//Створює зрізаний конус з заданим кольором(color), центром(center), радіусами вершини та основи (rtop,rbottom), висотою(height), а також із вказаною кількістю полігонів горизонталі(n).

private void CreateCone(Point3D center, double rtop, double rbottom, double height, int n, Color color, ref Model3DGroup All)

//Переведення циліндричних координат у декартові.

private Point3D GetPosition(double radius, double theta, double y)

**Torus** – клас що відповідає за створення тору.

Методи:

//Параметричні рівняння тору

private Point3D GetPosition(double R, double r, double t, double p)

//Створює тор з заданим кольором(color), центром(center), радіусом утворюючого кола та радіусом порожнини (r,R), а також із вказаною кількістю полігонів по вертикалі та горизонталі(n,N)

private void CreateTorus(Point3D center, double R, double r, int N, int n, Color color, ref Model3DGroup All)

Є також інші класи що відповідають за коректну роботу вікон.

## 3.5 Робота програми

При натиску на кнопку “Перейти до застосунку”, відкривається вікно що має наступний вигляд:

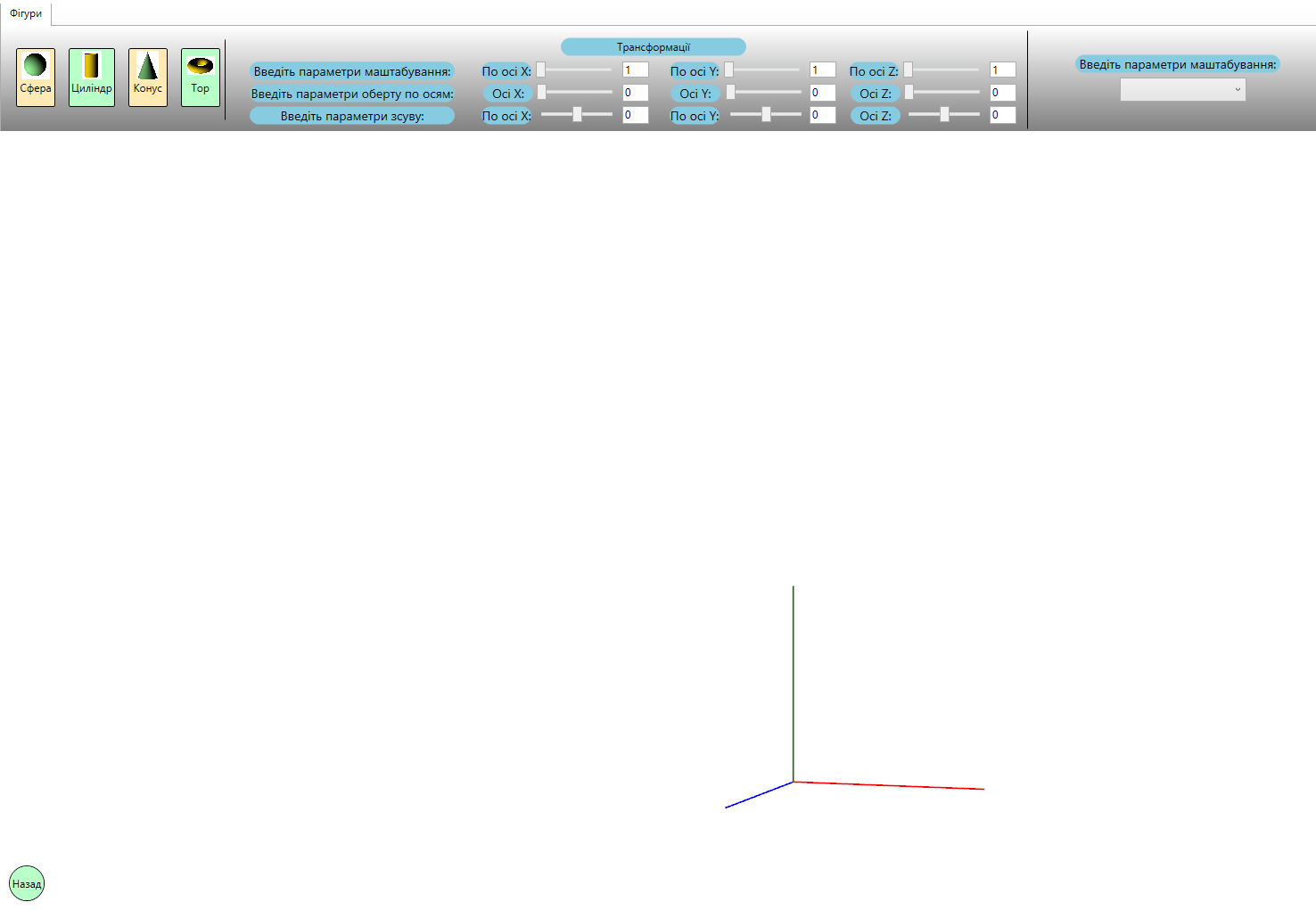


Рис. 3.1 Інтерфейс додатку

Кнопки «Сфера», «Циліндр», «Конус», «Тор» викликають вікна для вказання параметрів побудови примітивів. Серед яких є і «Ім’я примітиву» яке після створення з’являється у списку що розкривається(з права) та використовується для вказання фігури для трансформації. Після цього можна за допомогою повзунків або текстових полів трансформувати 3D примітив.

Повна версію коду знаходиться в додатку Б.

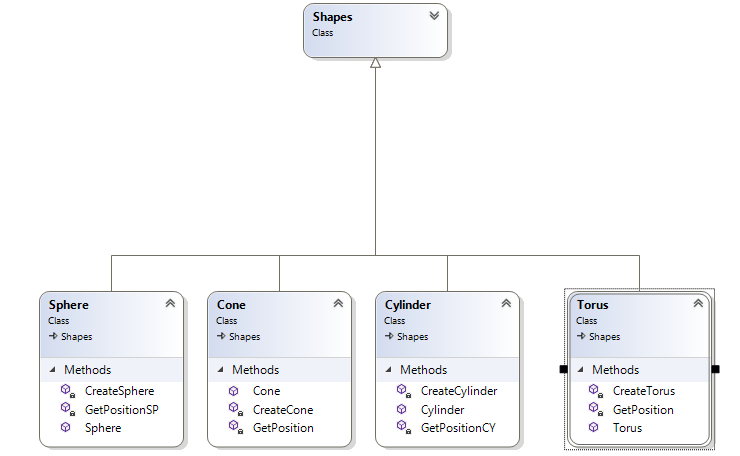
# ВИСНОВОК

В результаті опрацювання книг та сайтів по темі «Тривимірні зображення у WPF», та за допомогою деяких сторонніх бібліотек було створено додаток. Який дозволяє створювати стандартні тривимірні примітиви та робити деякі стандартні трансформації. Однак створення примітивів займає куди більше часу при великій кількості утворюючих полігонів. Звичайно у цій роботі було розглянуто відносно прості з точки зору розуміння алгоритми, що вже не використовуються в професіональних системах будування та конструювання, через свою набагато меншу швидкодію. Але використання DirectX та апаратного прискорення графіки дозволяє реалізовувати алгоритми набагато швидше.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* 1. Jack Xu.Practical C# Charts and Graphics.UniCAD Publishing, USA. 2007. р. 220- 277.
  2. Jack Xu.Practical WPF Charts and Graphics.APRESS, USA. 2009. р. 445- 529.
  3. Homogeneous Coordinates (Cyrill Stachniss, 2020) URL: <https://www.youtube.com/watch?v=MQdm0Z_gNcw> (дата звернення: 03.01.2023)
  4. Однородные координаты. URL: <https://ppt-online.org/16792> (дата звернення: 03.01.2023)
  5. 3D Graphics Overview URL: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/3-d-graphics-overview?view=netframeworkdesktop-4.8> (дата звернення: 15.07.2022)
  6. 3D Transformations Overview URL: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/3-d-transformations-overview?view=netframeworkdesktop-4.8> (дата звернення: 17.07.2022)
  7. Посібник з WPF URL: <https://metanit.com/sharp/wpf/> (дата звернення 10.06.2022)
  8. Extended.Wpf.Toolkit URL: <https://www.nuget.org/packages/Extended.Wpf.Toolkit/4.5.0?_src=template> (дата звернення 25.11.2022)

# ДОДАТОК А



# ДОДАТОК Б

MainWindow.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

mc:Ignorable="d"

Title="Титульна сторінка" Height="450" Width="800" BorderBrush="Black" BorderThickness="1" ResizeMode="NoResize">

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Window.Resources>

<ControlTemplate TargetType="Label" x:Key="btTemplate">

<ContentControl Margin="{TemplateBinding Padding}"

HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

Content="{TemplateBinding Content}" />

</ControlTemplate>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="390\*"/>

<RowDefinition Height="25\*"/>

<RowDefinition Height="19\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="272\*"/>

<ColumnDefinition Width="259\*"/>

<ColumnDefinition Width="269\*"/>

</Grid.ColumnDefinitions>

<Button Grid.Column="1" Content="Перейти до застосунку" Click="Button\_Click" Grid.Row="1" Style="{StaticResource YellowButton}"/>

<Border Grid.ColumnSpan="3" Margin="10" Background="White" CornerRadius="30">

<TextBlock Grid.ColumnSpan="3" TextAlignment="Center" Margin="10">

<Run>Дніпровський ліцей інформаційних технологій при Дніпровському національному університеті імені Олеся Гончара</Run>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<Run>Курсова робота за темою:</Run>

<LineBreak/>

<Run FontSize="15"> Дослідження 3D можливостей WPF</Run>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<LineBreak/>

<Run>Підготував учень 10-А класу Найденко Владислав Павлович</Run>

<LineBreak/>

<LineBreak/>

<Run>Керівник: Лавренюк Iрина Валерiївна</Run>

</TextBlock>

</Border>

</Grid>

</Window>

MainWindow.xaml.cs:

using System;

using System.Windows;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Project prog = new Project();

prog.Show();

Close();

}

}

}

Project.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.Project"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

mc:Ignorable="d"

xmlns:tool3d ="crl-namespace:\_3DTools;assembly=3DTools"

Title="Проект" Height="450" Width="1677" WindowState="Maximized" KeyDown="WindowKeyDown" MouseWheel="Grid\_MouseWheel">

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Window.Resources>

<ControlTemplate TargetType="Button" x:Key="btTemplate">

<Border x:Name="bord" CornerRadius="50" BorderBrush="#66cccc" BorderThickness="1">

<Border.Background>

<SolidColorBrush x:Name="BorderColor" Color="#66cccc" />

</Border.Background>

<ContentControl HorizontalAlignment="Center" VerticalAlignment="Center" Content="{TemplateBinding Content}"/>

<VisualStateManager.VisualStateGroups>

<VisualStateGroup Name="CommonStates">

<VisualState Name="MouseOver">

<Storyboard>

<ColorAnimation Storyboard.TargetName="BorderColor"

Storyboard.TargetProperty="Color" To="#99ffff" />

</Storyboard>

</VisualState>

<VisualState Name="Normal">

<Storyboard>

<ColorAnimation Storyboard.TargetName="BorderColor"

Storyboard.TargetProperty="Color" To="#66cccc" />

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

</Border>

</ControlTemplate>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="143"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Border Grid.Row="1" x:Name="Port" Background="White"/>

<TabControl Focusable="False" VerticalAlignment="Center" Height="150">

<TabControl.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</TabControl.Background>

<TabItem> <TabItem.Header>

<TextBlock Margin="3">Фігури</TextBlock>

</TabItem.Header>

<TabItem.Content>

<Grid>

<Border VerticalAlignment="Center" HorizontalAlignment="Left" BorderBrush="Black" BorderThickness="0,0,1,0" Height="90" Width="250">

<WrapPanel Height="90" Width="250" HorizontalAlignment="Left">

<RibbonButton LargeImageSource="Images\PhotoSphere.png" Label="Сфера" Margin ="15,10,0,0" HorizontalAlignment="Left" BorderBrush="Black" Click="RibbonButton\_SphereClick" Background="#ffeab4"/>

<RibbonButton LargeImageSource="Images\PhotoCylinder.png" Label="Циліндр" Margin ="15,10,0,0" HorizontalAlignment="Left" BorderBrush="Black" Click="RibbonButton\_CylinderClick" Background="#b9ffc8"/>

<RibbonButton LargeImageSource="Images\PhotoCone.png" Label="Конус" Margin ="15,10,0,0" HorizontalAlignment="Left" BorderBrush="Black" Click="RibbonButton\_ConeClick" Background="#ffeab4"/>

<RibbonButton LargeImageSource="Images\PhotoTorus.png" Label="Тор" Margin ="15,10,0,0" HorizontalAlignment="Left" BorderBrush="Black" Click="RibbonButton\_TorusClick" Background="#b9ffc8"/>

</WrapPanel>

</Border>

<Border VerticalAlignment="Center" HorizontalAlignment="Left" Margin="250,0,0,0" BorderBrush="Black" BorderThickness="0,0,1,0" Height="110" Width="900">

<Grid>

<Slider Focusable="False" Width="90" Value="{Binding ElementName=ScaleX, Path= Text}" Minimum="1" Maximum="10" HorizontalAlignment="Left" Margin="348,35,0,0" VerticalAlignment="Top"></Slider>

<TextBox x:Name="ScaleX" Text="1" HorizontalAlignment="Left" Margin="445,35,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="18" TextChanged="ScaleX\_TextChanged"/>

<Label FontSize="14" Content="Введіть параметри маштабування:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="230" Height="20" Margin="27,35,0,0"/>

<Label FontSize="14" Margin="287,35,0,0" Content="По осі X:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="20" Width="56"/>

<TextBox Text="1" x:Name="ScaleY" HorizontalAlignment="Left" Margin="655,35,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="18" TextChanged="ScaleY\_TextChanged"/>

<Slider Focusable="False" Width="90" Minimum="1" Value="{Binding ElementName=ScaleY, Path= Text}" Maximum="10" HorizontalAlignment="Left" Margin="559,35,0,0" VerticalAlignment="Top"/>

<Label FontSize="14" Content="По осі Y:" HorizontalAlignment="Left" Margin="498,35,0,0" VerticalAlignment="Top" Height="20" Width="56"/>

<Label FontSize="14" Height="20" Content="По осі Z:" HorizontalAlignment="Left" Margin="699,35,0,0" VerticalAlignment="Top" Width="56"/>

<Slider Focusable="False" Width="90" Minimum="1" Value="{Binding ElementName=ScaleZ, Path= Text}" Maximum="10" HorizontalAlignment="Left" Margin="760,35,0,0" VerticalAlignment="Top"/>

<TextBox Text="1" x:Name="ScaleZ" HorizontalAlignment="Left" Margin="857,35,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="18" TextChanged="ScaleZ\_TextChanged"/>

<Slider Focusable="False" Width="90" Value="{Binding ElementName=RotateX, Path= Text}" Minimum="0" Maximum="360" HorizontalAlignment="Left" Margin="349,60,0,0" Height="20" VerticalAlignment="Top"></Slider>

<TextBox x:Name="RotateX" Text="0" HorizontalAlignment="Left" Margin="445,60,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="20" TextChanged="RotateX\_TextChanged"/>

<Label Content="Осі X:" HorizontalAlignment="Left" Margin="289,60,0,0" VerticalAlignment="Top" FontSize="14" Height="20" Width="56"/>

<TextBox Text="0" x:Name="RotateY" HorizontalAlignment="Left" Margin="655,60,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="20" TextChanged="RotateY\_TextChanged"/>

<Slider Focusable="False" Width="90" Minimum="0" Value="{Binding ElementName=RotateY, Path= Text}" Maximum="360" HorizontalAlignment="Left" Margin="561,60,0,0" VerticalAlignment="Top" Height="20"/>

<Label FontSize="14" Height="20" Content="Осі Y:" HorizontalAlignment="Left" Margin="499,60,0,0" VerticalAlignment="Top" Width="56"/>

<Label FontSize="14" Height="20" Content="Осі Z:" HorizontalAlignment="Left" Margin="701,60,0,0" VerticalAlignment="Top" Width="56"/>

<Slider Focusable="False" Width="90" Minimum="0" Value="{Binding ElementName=RotateZ, Path= Text}" Maximum="360" HorizontalAlignment="Left" Margin="761,60,0,0" VerticalAlignment="Top" Height="20"/>

<TextBox Text="0" x:Name="RotateZ" HorizontalAlignment="Left" Margin="857,60,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="20" TextChanged="RotateZ\_TextChanged"/>

<Label FontSize="14" Content="Введіть параметри оберту по осям:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="230" Height="20" Margin="27,60,0,0"/>

<Label Content="Трансформації" HorizontalAlignment="Left" Height="20" Margin="376,8,0,0" VerticalAlignment="Top" Width="208"/>

<Label FontSize="14" Content="Введіть параметри зсуву:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="230" Height="20" Margin="27,85,0,0"/>

<Label FontSize="14" Margin="287,85,0,0" Content="По осі X:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="20" Width="56"/>

<Slider Focusable="False" Width="90" Value="{Binding Text, ElementName=OffsetX}" Minimum="-20" Maximum="20" HorizontalAlignment="Left" Margin="349,85,0,0" Height="20" VerticalAlignment="Top"/>

<TextBox x:Name="OffsetX" Text="0" HorizontalAlignment="Left" Margin="445,85,0,0" TextWrapping="Wrap" Width="30" TextChanged="OffsetX\_TextChanged" Height="20" VerticalAlignment="Top"/>

<Label FontSize="14" Content="По осі Y:" HorizontalAlignment="Left" Margin="498,85,0,0" VerticalAlignment="Top" Height="20" Width="56"/>

<Slider Focusable="False" Width="90" Minimum="-20" Value="{Binding Text, ElementName=OffsetY}" Maximum="20" HorizontalAlignment="Left" Margin="561,85,0,0" VerticalAlignment="Top" Height="20"/>

<TextBox Text="0" x:Name="OffsetY" HorizontalAlignment="Left" Margin="655,85,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="20" TextChanged="OffsetY\_TextChanged"/>

<Label FontSize="14" Height="20" Content="Осі Z:" HorizontalAlignment="Left" Margin="701,85,0,0" VerticalAlignment="Top" Width="56"/>

<Slider Focusable="False" Width="90" Minimum="-20" Value="{Binding Text, ElementName=OffsetZ}" Maximum="20" HorizontalAlignment="Left" Margin="761,85,0,0" VerticalAlignment="Top" Height="20"/>

<TextBox Text="0" x:Name="OffsetZ" HorizontalAlignment="Left" Margin="857,85,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="30" Height="20" TextChanged="OffsetZ\_TextChanged"/>

<Label FontSize="14" Content="Введіть параметри маштабування:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="230" Height="20" Margin="27,35,0,0"/>

</Grid>

</Border>

<Label FontSize="14" Content="Введіть параметри маштабування:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="230" Height="20" Margin="1203,30,0,0"/>

<ComboBox Focusable="False" Name="NamesOfShapes" HorizontalAlignment="Left" Margin="1253,56,0,0" VerticalAlignment="Top" Width="142" Height="27" SelectionChanged="NamesOfShapes\_SelectionChanged"/>

</Grid>

</TabItem.Content>

</TabItem>

</TabControl>

<Viewport3D Name="myViewport" Grid.Row="1">

<Viewport3D.Camera>

<PerspectiveCamera Position="10,4,30" LookDirection="-0.3,0,-1" />

</Viewport3D.Camera>

<Viewport3D.Children>

<ModelVisual3D>

<ModelVisual3D.Content>

<DirectionalLight Color="White" Direction="1,1,-2" />

</ModelVisual3D.Content>

</ModelVisual3D>

<ModelVisual3D x:Name="model">

<ModelVisual3D.Transform>

<Transform3DGroup x:Name="vision">

<RotateTransform3D>

<RotateTransform3D.Rotation>

<AxisAngleRotation3D x:Name="visionX" Axis="0,1,0"/>

</RotateTransform3D.Rotation>

</RotateTransform3D>

<RotateTransform3D>

<RotateTransform3D.Rotation>

<AxisAngleRotation3D x:Name="visionY" Axis="1,0,0"/>

</RotateTransform3D.Rotation>

</RotateTransform3D>

<ScaleTransform3D x:Name="Zoom" CenterX="0" CenterY="0" CenterZ="0"/>

</Transform3DGroup>

</ModelVisual3D.Transform>

</ModelVisual3D>

</Viewport3D.Children>

</Viewport3D>

<Button Content="Назад" Grid.Row="1" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="40" Height="40" Margin="10,0,0,10" Click="ClickBackProject" Style="{StaticResource GreenButton}"/>

</Grid>

</Window>

Project.xaml.cs:

using System;

using System.Collections.Generic;

using System.Text;

using Курсова\_робота\_Найденко\_Влад\_\_10\_А;

using System.Windows;

using System.Windows.Controls;

using System.Globalization;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Media3D;

using System.Collections.ObjectModel;

using \_3DTools;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for Project.xaml

/// </summary>

public partial class Project : Window

{

public static ObservableCollection<string> ShapesCollectionNames = new ObservableCollection<string>();

public static Dictionary<string, Shapes> Figures = new Dictionary<string, Shapes>();

public Project()

{

InitializeComponent();

NamesOfShapes.ItemsSource = ShapesCollectionNames;

model.Content = All;

ScreenSpaceLines3D Zax = new ScreenSpaceLines3D();

Zax.Points.Add(new Point3D(0, 0, 0));

Zax.Points.Add(new Point3D(0, 0, 3));

Zax.Color = Colors.Blue;

Zax.Thickness = 2;

ScreenSpaceLines3D Yax = new ScreenSpaceLines3D();

Yax.Points.Add(new Point3D(0, 0, 0));

Yax.Points.Add(new Point3D(0, 3, 0));

Yax.Color = Colors.Green;

Yax.Thickness = 2;

ScreenSpaceLines3D Xax = new ScreenSpaceLines3D();

Xax.Points.Add(new Point3D(0, 0, 0));

Xax.Points.Add(new Point3D(3, 0, 0));

Xax.Color = Colors.Red;

Xax.Thickness = 2;

myViewport.Children.Add(Zax);

myViewport.Children.Add(Yax);

myViewport.Children.Add(Xax);

}

Shapes SelectedObject;

DataTorusWindow TorusPaintWindow;

DataSphere SpherePaintWindow;

DataCylinder CylinderPaintWindow;

ConDataPaintWindow ConePaintWindow;

public static Model3DGroup All = new Model3DGroup();

#region Sphere

private void RibbonButton\_SphereClick(object sender, RoutedEventArgs e)

{

SpherePaintWindow = new DataSphere();

SpherePaintWindow.Show();

SpherePaintWindow.Start.Click += Button\_SphereCreate;

}

private void Button\_SphereCreate(object sender, RoutedEventArgs e)

{

if (!double.TryParse(SpherePaintWindow.RadiusSphere.Text, out double InRadiusSphere))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(SpherePaintWindow.SphereCenterX.Text, out double InSphereCenterX))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(SpherePaintWindow.SphereCenterY.Text, out double InSphereCenterY))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(SpherePaintWindow.SphereCenterZ.Text, out double InSphereCenterZ))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (SpherePaintWindow.ColorPicker1.SelectedColor == null)

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(SpherePaintWindow.uText.Text, out int u))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(SpherePaintWindow.vText.Text, out int v))

{

MessageBox.Show("Введені данні невалідні");

return;

}

string str = SpherePaintWindow.nameShape.Text;

if (ShapesCollectionNames.Contains(str)|| String.IsNullOrWhiteSpace(str))

{

MessageBox.Show("Введені данні невалідні");

return;

}

Figures.Add(str, new Sphere(new Point3D(InSphereCenterX, InSphereCenterY, InSphereCenterZ), InRadiusSphere,u,v, (Color)SpherePaintWindow.ColorPicker1.SelectedColor, ref All));

ShapesCollectionNames.Add(SpherePaintWindow.nameShape.Text);

SpherePaintWindow.Close();

}

#endregion

#region Cylinder

private void RibbonButton\_CylinderClick(object sender, RoutedEventArgs e)

{

CylinderPaintWindow = new DataCylinder();

CylinderPaintWindow.Show();

CylinderPaintWindow.ButtonCreateCylinder.Click += Button\_CreateCylinder;

}

private void Button\_CreateCylinder(object sender, RoutedEventArgs e)

{

if (!double.TryParse(CylinderPaintWindow.CylinderRadius.Text, out double InRadiusCylinder))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(CylinderPaintWindow.CylinderHeight.Text, out double InHeightCylinder))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(CylinderPaintWindow.CylinderCenterX.Text, out double InCylinderCenterX))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(CylinderPaintWindow.CylinderCenterY.Text, out double InCylinderCenterY))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(CylinderPaintWindow.CylinderCenterZ.Text, out double InCylinderCenterZ))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (CylinderPaintWindow.ColorPicker1.SelectedColor == null)

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(CylinderPaintWindow.nTextBox.Text, out int n))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(CylinderPaintWindow.CylinderRadiusIn.Text, out double rin))

{

MessageBox.Show("Введені данні невалідні");

return;

}

string str = CylinderPaintWindow.CylinderName.Text;

if (ShapesCollectionNames.Contains(str) || String.IsNullOrWhiteSpace(str))

{

MessageBox.Show("Введені данні невалідні");

return;

}

Figures.Add(str, new Cylinder(new Point3D(InCylinderCenterX, InCylinderCenterY, InCylinderCenterZ),rin, InHeightCylinder, InRadiusCylinder,n, (Color)CylinderPaintWindow.ColorPicker1.SelectedColor, ref All));

ShapesCollectionNames.Add(CylinderPaintWindow.CylinderName.Text);

CylinderPaintWindow.Close();

}

#endregion

private void RibbonButton\_ConeClick(object sender, RoutedEventArgs e)

{

ConePaintWindow = new ConDataPaintWindow();

ConePaintWindow.Show();

ConePaintWindow.ConeCreate.Click += Button\_CreateCone;

}

private void Button\_CreateCone(object sender, RoutedEventArgs e)

{

if (!double.TryParse(ConePaintWindow.ConeRadius.Text, out double InRadiusCone))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(ConePaintWindow.ConeHeight.Text, out double InHeightCone))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(ConePaintWindow.ConeCenterX.Text, out double InConeCenterX))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(ConePaintWindow.ConeCenterY.Text, out double InConeCenterY))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(ConePaintWindow.ConeCenterZ.Text, out double InConeCenterZ))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (ConePaintWindow.ColorPicker1.SelectedColor == null)

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(ConePaintWindow.ConeVertexRadius.Text, out int InConeVertexRadius))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(ConePaintWindow.ConeEdgeNumber.Text, out int InConeEdgeNumber))

{

MessageBox.Show("Введені данні невалідні");

return;

}

string str = ConePaintWindow.ConeName.Text;

if (ShapesCollectionNames.Contains(str) || String.IsNullOrWhiteSpace(str))

{

MessageBox.Show("Введені данні невалідні");

return;

}

Figures.Add(str, new Cone(InRadiusCone, InConeVertexRadius, InHeightCone, InConeEdgeNumber, new Point3D(InConeCenterX, InConeCenterY, InConeCenterZ), (Color)ConePaintWindow.ColorPicker1.SelectedColor, ref All));

ShapesCollectionNames.Add(ConePaintWindow.ConeName.Text);

ConePaintWindow.Close();

}

private void RibbonButton\_TorusClick(object sender, RoutedEventArgs e)

{

TorusPaintWindow = new DataTorusWindow();

TorusPaintWindow.Show();

TorusPaintWindow.TorusCreate.Click += Button\_CreateTorus;

}

private void Button\_CreateTorus(object sender, RoutedEventArgs e)

{

if (!double.TryParse(TorusPaintWindow.RTorus.Text, out double InRadiusTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(TorusPaintWindow.rTorus.Text, out double InradTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(TorusPaintWindow.NTorus.Text, out int InNTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!int.TryParse(TorusPaintWindow.nTorus.Text, out int InnTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(TorusPaintWindow.TorusCenterX.Text, out double InCenterXTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(TorusPaintWindow.TorusCenterY.Text, out double InCenterYTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (!double.TryParse(TorusPaintWindow.TorusCenterZ.Text, out double InCenterZTorus))

{

MessageBox.Show("Введені данні невалідні");

return;

}

if (TorusPaintWindow.ColorPicker1.SelectedColor == null)

{

MessageBox.Show("Введені данні невалідні");

return;

}

string str = TorusPaintWindow.TorusName.Text;

if (ShapesCollectionNames.Contains(str) || String.IsNullOrWhiteSpace(str))

{

MessageBox.Show("Введені данні невалідні");

return;

}

Figures.Add(str, new Torus(new Point3D(InCenterXTorus, InCenterYTorus, InCenterZTorus), InRadiusTorus, InradTorus, InNTorus, InnTorus, (Color)TorusPaintWindow.ColorPicker1.SelectedColor, ref All));

ShapesCollectionNames.Add(TorusPaintWindow.TorusName.Text);

TorusPaintWindow.Close();

}

private void WindowKeyDown(object sender, KeyEventArgs e)

{

if(Keyboard.IsKeyDown(Key.Right))

{

visionX.Angle++;

}

if (Keyboard.IsKeyDown(Key.Left))

{

visionX.Angle--;

}

if (Keyboard.IsKeyDown(Key.Up))

{

visionY.Angle--;

}

if (Keyboard.IsKeyDown(Key.Down))

{

visionY.Angle++;

}

}

private void Grid\_MouseWheel(object sender, MouseWheelEventArgs e)

{

if (e.Delta > 0)

{

Zoom.ScaleX += 0.1;

Zoom.ScaleY += 0.1;

Zoom.ScaleZ += 0.1;

}

if (e.Delta < 0 && !(Zoom.ScaleX <= 0.2))

{

Zoom.ScaleX -= 0.1;

Zoom.ScaleY -= 0.1;

Zoom.ScaleZ -= 0.1;

}

}

private void ClickBackProject(object sender, RoutedEventArgs e)

{

MainWindow mainWind = new MainWindow();

mainWind.Show();

base.Close();

}

private void ScaleX\_TextChanged(object sender, TextChangedEventArgs e)

{

if(SelectedObject!=null)

{

if (double.TryParse(ScaleX.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.ScaleX(p);

}

}

}

private void ScaleY\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(ScaleY.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.ScaleY(p);

}

}

}

private void ScaleZ\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(ScaleZ.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.ScaleZ(p);

}

}

}

private void RotateX\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(RotateX.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.RotateX(p);

}

}

}

private void RotateY\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(RotateY.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.RotateY(p);

}

}

}

private void RotateZ\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(RotateZ.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.RotateZ(p);

}

}

}

private void OffsetX\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(OffsetX.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.OffsetX(p);

}

}

}

private void OffsetY\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(OffsetY.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.OffsetY(p);

}

}

}

private void OffsetZ\_TextChanged(object sender, TextChangedEventArgs e)

{

if (SelectedObject != null)

{

if (double.TryParse(OffsetZ.Text, NumberStyles.Any, new NumberFormatInfo { NumberDecimalSeparator = "." }, out double p))

{

SelectedObject.OffsetZ(p);

}

}

}

private void NamesOfShapes\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

Figures.TryGetValue(comboBox.SelectedItem.ToString(), out SelectedObject);

}

}

}

DataSphere.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.DataSphere"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

mc:Ignorable="d"

Title="Введіть дані сфери" Height="304" Width="500" ResizeMode="NoResize">

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<Label Content="Введіть кількість полігонів по вертикалі" HorizontalAlignment="Left" Margin="15,35,0,0" VerticalAlignment="Top" Width="250" Height="15"/>

<Label Content="Введіть кількість полігонів по горизонталі" HorizontalAlignment="Left" Margin="15,10,0,0" VerticalAlignment="Top" Width="250" Height="15"/>

<TextBox x:Name="uText" TextWrapping="Wrap" Margin="280,34,0,0" Height="18" HorizontalAlignment="Left" VerticalAlignment="Top" Width="146"/>

<TextBox x:Name="vText" TextWrapping="Wrap" Margin="280,9,0,0" Height="18" HorizontalAlignment="Left" VerticalAlignment="Top" Width="146"/>

<Label Content="Введіть радіус сфери" HorizontalAlignment="Left" Margin="15,60,0,0" VerticalAlignment="Top" Width="120" Height="15"/>

<TextBox Text="3" Name="RadiusSphere" HorizontalAlignment="Left" Margin="280,59,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="146" Height="18"/>

<Button x:Name="Start" Content="Створити кулю" Style ="{StaticResource YellowButton}" HorizontalAlignment="Center" VerticalAlignment="Top" Width="130" Margin="0,220,0,0" Height="40"/>

<Label Content="Введіть координати центру кулі:" HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Margin="15,85,0,0" Height="15"/>

<xctk:ColorPicker Name="ColorPicker1" DisplayColorAndName="True" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="15,200,0,0" Width="100"/>

<Label Margin="15,110,0,0" Height="25" Width="20" HorizontalAlignment="Left" VerticalAlignment="Top">X:</Label>

<Label Margin="15,135,0,0" Height="25" Width="20" HorizontalAlignment="Left" VerticalAlignment="Top">Y:</Label>

<Label Margin="15,160,0,0" Height="25" Width="20" HorizontalAlignment="Left" VerticalAlignment="Top">Z:</Label>

<TextBox Text="3" x:Name="SphereCenterX" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="80,160,0,0" Width="120" Height="18"/>

<TextBox Text="2" x:Name="SphereCenterY" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="80,135,0,0" Width="120" Height="18"/>

<TextBox Text ="6" x:Name="SphereCenterZ" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="80,110,0,0" Width="120" Height="18"/>

<Label Content="Ім'я примітиву" HorizontalAlignment="Left" Height="20" Width="120" Margin="331,80,0,0"/>

<TextBox x:Name="nameShape" HorizontalAlignment="Left" Margin="331,200,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

</Grid>

</Window>

DataSphere.xaml.cs:

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for DataSphere.xaml

/// </summary>

public partial class DataSphere : Window

{

public DataSphere()

{

InitializeComponent();

}

}

}

DataCylinder.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.DataCylinder"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

mc:Ignorable="d"

Title="Введіть дані циліндру" Height="340" Width="400" ResizeMode="NoResize">

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<Label Content="Введіть радіус циліндру" HorizontalAlignment="Left" Margin="10,60,0,0" VerticalAlignment="Top" Height="15" Width="150"/>

<TextBox Text="5" x:Name="CylinderRadius" HorizontalAlignment="Left" Margin="240,59,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть висоту циліндру" HorizontalAlignment="Left" Margin="10,85,0,0" VerticalAlignment="Top" Height="15" Width="150"/>

<TextBox Text="5" x:Name="CylinderHeight" HorizontalAlignment="Left" Margin="240,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть центр циліндру:" HorizontalAlignment="Left" Margin="10,110,0,0" VerticalAlignment="Top" Height="15" Width="140"/>

<TextBox Text="2" x:Name="CylinderCenterX" HorizontalAlignment="Left" Margin="59,144,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="X:" HorizontalAlignment="Left" Margin="10,140,0,0" VerticalAlignment="Top" Height="25" Width="20"/>

<Label Content="Y:" HorizontalAlignment="Left" Margin="10,167,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.562,0.541" Height="25" Width="20"/>

<Label Content="Z:" HorizontalAlignment="Left" Margin="10,194,0,0" VerticalAlignment="Top" Height="25" Width="20"/>

<TextBox Text="2" x:Name="CylinderCenterY" HorizontalAlignment="Left" Margin="59,171,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<TextBox Text="2" x:Name="CylinderCenterZ" HorizontalAlignment="Left" Margin="59,198,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Button x:Name="ButtonCreateCylinder" Style="{StaticResource GreenButton}" Content="Створити циліндр" HorizontalAlignment="Center" Margin="0,259,0,0" VerticalAlignment="Top" Height="40" Width="130"/>

<xctk:ColorPicker Name="ColorPicker1" DisplayColorAndName="True" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="15,239,0,0" Width="100"/>

<Label Content="Введіть кількість фрагментів циліндру" HorizontalAlignment="Left" Margin="10,35,0,0" VerticalAlignment="Top" Height="15" Width="212"/>

<TextBox Text="20" x:Name="nTextBox" HorizontalAlignment="Left" Margin="240,34,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть радіус порожнини циліндру" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Height="15" Width="212"/>

<TextBox Text="2" x:Name="CylinderRadiusIn" HorizontalAlignment="Left" Margin="240,9,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<TextBox x:Name="CylinderName" HorizontalAlignment="Left" Margin="240,232,0,0" Text="hh" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Ім'я примітиву" HorizontalAlignment="Left" Height="20" Width="120" Margin="240,100,0,0"/>

</Grid>

</Window>

DataCylinder.xaml.cs:

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for DataCylinder.xaml

/// </summary>

public partial class DataCylinder : Window

{

public DataCylinder()

{

InitializeComponent();

}

}

}

ConDataPaintWindow.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.ConDataPaintWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

mc:Ignorable="d"

Title="Введіть дані конусу" Height="339" Width="408" ResizeMode="NoResize" >

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<Label Content="Введіть радіус основи конусу" HorizontalAlignment="Left" Margin="15,10,0,0" VerticalAlignment="Top" Height="15" Width="166"/>

<TextBox Text="2" x:Name="ConeRadius" HorizontalAlignment="Left" Margin="220,10,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть висоту конуса" HorizontalAlignment="Left" Margin="15,35,0,0" VerticalAlignment="Top" Height="15" Width="150"/>

<TextBox Name="ConeHeight" Text="5" HorizontalAlignment="Left" Margin="220,35,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть центр конуса" HorizontalAlignment="Left" Margin="15,110,0,0" VerticalAlignment="Top" Height="15" Width="150"/>

<Label Content="Введіть радіус вершини" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="15,60,0,0" Height="15" Width="150"/>

<TextBox x:Name="ConeVertexRadius" Text="5" TextWrapping="Wrap" Width="120" Height="20" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="220,60,0,0"/>

<Label Content="Введіть кількість граней" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="15,85,0,0" Height="15" Width="150"/>

<TextBox x:Name="ConeEdgeNumber" Text="20" TextWrapping="Wrap" Width="120" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="220,85,0,0"/>

<StackPanel>

<Label Content="X:" Margin="-80,140,253,0" Height="25" Width="20"/>

<Label Content="Y:" Margin="-80,0,253,0" Height="25" Width="20"/>

<Label Content="Z:" Margin="-80,0,253,0" Height="25" Width="20"/>

</StackPanel>

<StackPanel>

<TextBox x:Name="ConeCenterX" Text="2" Margin="-150,140,0,0" Width="120"/>

<TextBox x:Name="ConeCenterY" Text="2" Margin="-150,10,0,0" Width="120" />

<TextBox x:Name="ConeCenterZ" Text="2" Margin="-150,10,0,0" Width="120" />

</StackPanel>

<Label Content="Ім'я примітиву" Height="20" Width="120" Margin="242,100,0,0"/>

<TextBox x:Name="ConeName" TextWrapping="Wrap" Width="120" Height="22" Margin="242,224,0,0" VerticalAlignment="Top"/>

<Button x:Name="ConeCreate" Style ="{StaticResource YellowButton}" Content="Побудувати конус" HorizontalAlignment="Center" Margin="0,249,0,0" VerticalAlignment="Top" Width="130" Height="40"/>

<xctk:ColorPicker Name="ColorPicker1" DisplayColorAndName="True" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="15,224,0,0" Width="100"/>

</Grid>

</Window>

ConDataPaintWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for ConDataPaintWindow.xaml

/// </summary>

public partial class ConDataPaintWindow : Window

{

public ConDataPaintWindow()

{

InitializeComponent();

}

}

}

DataTorusWindow.xaml:

<Window x:Class="Курсова\_робота\_Найденко\_Влад\_\_10\_А.DataTorusWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

xmlns:local="clr-namespace:Курсова\_робота\_Найденко\_Влад\_\_10\_А"

mc:Ignorable="d"

Title="Введіть дані тору" Height="373" Width="459" ResizeMode="NoResize" >

<Window.Background>

<LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">

<GradientStop Color="#7e7e7e" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<Label Content="Введіть радіус полості тору" HorizontalAlignment="Left" Margin="15,20,0,0" VerticalAlignment="Top" Height="15" Width="162"/>

<TextBox x:Name="RTorus" HorizontalAlignment="Left" Margin="278,20,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Height="18"/>

<Label Content="Введіть радіус утворюючого кола" HorizontalAlignment="Left" Margin="15,45,0,0" VerticalAlignment="Top" Width="197"/>

<TextBox x:Name="rTorus" HorizontalAlignment="Left" Margin="278,45,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Height="18"/>

<Label Content="Вкажіть кількість полігонів по горизонталі" HorizontalAlignment="Left" Margin="15,70,0,0" VerticalAlignment="Top" Width="254"/>

<TextBox x:Name="NTorus" HorizontalAlignment="Left" Margin="278,70,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Вкажіть кількість полігонів по вертикалі" HorizontalAlignment="Left" Margin="15,95,0,0" VerticalAlignment="Top" Width="254"/>

<TextBox x:Name="nTorus" HorizontalAlignment="Left" Margin="278,95,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="Введіть центр тору:" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="15,130,253,0"/>

<StackPanel>

<Label Content="X:" Margin="-160,155,253,0" Height="25" Width="20"/>

<Label Content="Y:" Margin="-160,0,253,0" Height="25" Width="20"/>

<Label Content="Z:" Margin="-160,0,253,0" Height="25" Width="20"/>

</StackPanel>

<StackPanel>

<TextBox x:Name="TorusCenterX" Margin="-150,155,0,0" Width="120"/>

<TextBox x:Name="TorusCenterY" Margin="-150,10,0,0" Width="120" />

<TextBox x:Name="TorusCenterZ" Margin="-150,10,0,0" Width="120" />

</StackPanel>

<Label Content="Ім'я примітиву" Height="20" Width="120" Margin="300,100,0,0"/>

<TextBox Name="TorusName" TextWrapping="Wrap" Width="120" Height="22" Margin="300,150,0,0"/>

<Button x:Name="TorusCreate" Content="Побудувати тор" Style="{StaticResource GreenButton}" HorizontalAlignment="Center" Margin="0,260,0,0" VerticalAlignment="Top" Width="130" Height="40"/>

<xctk:ColorPicker Name="ColorPicker1" DisplayColorAndName="True" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="15,240,0,0" Width="100"/>

</Grid>

</Window>

DataTorusWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

/// <summary>

/// Interaction logic for DataTorusWindow.xaml

/// </summary>

public partial class DataTorusWindow : Window

{

public DataTorusWindow()

{

InitializeComponent();

}

}

}

Sphere.cs:

using System;

using System.Windows.Media;

using System.Windows.Media.Media3D;

using System.Collections.Generic;

using System.Text;

using System.Threading;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

class Sphere : Shapes

{

public Sphere(Point3D centre, double r,int u,int v, Color col, ref Model3DGroup All):base(centre)

{

CreateSphere(centre, r, u, v, col, ref All);

}

//Створює сферу з заданим кольором(color), центром(center) та радіусом(radius), а також із вказаною кількістю полігонів по вертикалі та горизонталі(u,v)

private void CreateSphere(Point3D center, double radius, int u, int v, Color color, ref Model3DGroup All)

{

int coun = 0;

if (u < 2 || v < 2) return;

Point3D[] p = new Point3D[4];

MeshGeometry3D mesh = new MeshGeometry3D();

for (int i = 0; i < u ; i++)

{

for (int j = 0; j < v; j++)

{

p[0] = GetPositionSP(radius, i \* 180 / u, j \* 360 / v);

p[0] += (Vector3D)center;

p[1] = GetPositionSP(radius, (i +1)\* 180 / u, j \* 360 / v);

p[1] += (Vector3D)center;

p[2] = GetPositionSP(radius, (i + 1) \* 180 / u, (j+1) \* 360 / v);

p[2] += (Vector3D)center;

p[3] = GetPositionSP(radius, i \* 180 / u, (j+1) \* 360 / v);

p[3] += (Vector3D)center;

CreateTriangleFace(p[0], p[1], p[2], mesh, ref coun);

CreateTriangleFace(p[2], p[3], p[0], mesh, ref coun);

}

}

SolidColorBrush brush = new SolidColorBrush();

brush.Color = color;

Material material = new DiffuseMaterial(brush);

GeometryModel3D geometry = new GeometryModel3D(mesh, material);

model = geometry;

model.Transform = transform;

All.Children.Add(model);

}

//Переведення сферичних координат у декартові.

private Point3D GetPositionSP(double radius, double theta, double phi)

{

double snt = Math.Sin(theta \* Math.PI / 180);

double cnt = Math.Cos(theta \* Math.PI / 180);

double snp = Math.Sin(phi \* Math.PI / 180);

double cnp = Math.Cos(phi \* Math.PI / 180);

return new Point3D(radius \* snt \* cnp, radius \* cnt, -radius \* snt \* snp);

}

}

}

Cylinder.cs:

using System;

using System.Windows.Media;

using System.Windows.Media.Media3D;

using System.Collections.Generic;

using System.Text;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

class Cylinder : Shapes

{

public Cylinder(Point3D centerFirst,double radiusIn, double height, double radius, int n, Color color, ref Model3DGroup All) : base(centerFirst)

{

CreateCylinder(centerFirst, radiusIn, radius, height, n, color, ref All);

}

//Переведення циліндричних координат у декартові.

private Point3D GetPositionCY(double radius, double theta, double y)

{

Point3D pt = new Point3D();

double sn = Math.Sin(theta \* Math.PI / 180);

double cn = Math.Cos(theta \* Math.PI / 180);

pt.X = radius \* cn;

pt.Y = y;

pt.Z = -radius \* sn;

return pt;

}

//Створює циліндр з заданим кольором(color), центром(center), радіусами циліндру та порожнини(rout,rin), висотою(height), а також із вказаною кількістю полігонів горизонталі(n).

private void CreateCylinder(Point3D center, double rin, double rout, double height, int n, Color color, ref Model3DGroup All)

{

if (n < 2 || rin == rout) return;

double radius = rin;

if (rin > rout)

{

rin = rout;

rout = radius;

}

double h = height / 2;

Point3D[] p = new Point3D[8];

int coun = 0;

MeshGeometry3D mesh = new MeshGeometry3D();

for (int i = 0; i < n; i++)

{

p[0] = GetPositionCY(rout, i \* 360 / n, h);

p[0] += (Vector3D)center;

p[1] = GetPositionCY(rout, i \* 360 / n, -h);

p[1] += (Vector3D)center;

p[2] = GetPositionCY(rin, i \* 360 / n, -h);

p[2] += (Vector3D)center;

p[3] = GetPositionCY(rin, i \* 360 / n, h);

p[3] += (Vector3D)center;

p[4] = GetPositionCY(rout, (i+1) \* 360 / n, h);

p[4] += (Vector3D)center;

p[5] = GetPositionCY(rout, (i+1) \* 360 / n, -h);

p[5] += (Vector3D)center;

p[6] = GetPositionCY(rin, (i+1) \* 360 / n, -h);

p[6] += (Vector3D)center;

p[7] = GetPositionCY(rin, (i+1) \* 360 / n, h);

p[7] += (Vector3D)center;

CreateTriangleFace(p[0], p[4], p[3], mesh, ref coun);

CreateTriangleFace(p[4], p[7], p[3], mesh, ref coun);

CreateTriangleFace(p[2], p[5], p[1], mesh, ref coun);

CreateTriangleFace(p[2], p[6], p[5], mesh, ref coun);

CreateTriangleFace(p[0], p[1], p[4], mesh, ref coun);

CreateTriangleFace(p[1], p[5], p[4], mesh, ref coun);

CreateTriangleFace(p[2], p[7], p[6], mesh, ref coun);

CreateTriangleFace(p[2], p[3], p[7], mesh, ref coun);

}

SolidColorBrush brush = new SolidColorBrush();

brush.Color = color;

Material material = new DiffuseMaterial(brush);

GeometryModel3D geometry = new GeometryModel3D(mesh, material);

model = geometry;

model.Transform = transform;

All.Children.Add(model);

}

}

}

Cone.cs:

using System;

using System.Windows.Media;

using System.Windows.Media.Media3D;

using System.Collections.Generic;

using System.Text;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

class Cone : Shapes

{

public Cone(double radiusBotom, double radiusTop, double height, int count, Point3D center, Color col, ref Model3DGroup All) : base(center)

{

CreateCone(center, radiusTop, radiusBotom, height, count, col, ref All);

}

//Переведення циліндричних координат у декартові.

private Point3D GetPosition(double radius, double theta, double y)

{

Point3D pt = new Point3D();

double sn = Math.Sin(theta \* Math.PI / 180);

double cn = Math.Cos(theta \* Math.PI / 180);

pt.X = radius \* cn;

pt.Y = y;

pt.Z = -radius \* sn;

return pt;

}

//Створює зрізаний конус з заданим кольором(color), центром(center), радіусами вершини та основи (rtop,rbottom), висотою(height), а також із вказаною кількістю полігонів горизонталі(n).

private void CreateCone(Point3D center, double rtop, double rbottom, double height, int n, Color color, ref Model3DGroup All)

{

if (n < 2)

return;

double h = height / 2;

Point3D[] p = new Point3D[6];

int coun = 0;

MeshGeometry3D mesh = new MeshGeometry3D();

for (int i = 0; i < n; i++)

{

p[0] = GetPosition(rtop, i \* 360 / n, h);

p[0] += (Vector3D)center;

p[1] = GetPosition(rbottom, i \* 360 / n, -h);

p[1] += (Vector3D)center;

p[2] = GetPosition(0, i \* 360 / n, -h);

p[2] += (Vector3D)center;

p[3] = GetPosition(0, i \* 360 / n, h);

p[3] += (Vector3D)center;

p[4] = GetPosition(rtop, (i+1) \* 360 / n, h);

p[4] += (Vector3D)center;

p[5] = GetPosition(rbottom, (i+1) \* 360 / n, -h);

p[5] += (Vector3D)center;

CreateTriangleFace(p[0], p[4], p[3],mesh, ref coun);

CreateTriangleFace(p[2], p[5], p[1],mesh, ref coun);

CreateTriangleFace(p[0], p[1], p[5],mesh, ref coun);

CreateTriangleFace(p[0], p[5], p[4],mesh, ref coun);

}

SolidColorBrush brush = new SolidColorBrush();

brush.Color = color;

Material material = new DiffuseMaterial(brush);

GeometryModel3D geometry = new GeometryModel3D(mesh, material);

model = geometry;

model.Transform = transform;

All.Children.Add(model);

}

}

}

Torus.cs:

using System;

using System.Windows.Media;

using System.Windows.Media.Media3D;

using System.Collections.Generic;

using System.Text;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

class Torus : Shapes

{

public Torus(Point3D Center, double R, double r, int N, int n, Color col, ref Model3DGroup All) : base(Center)

{

CreateTorus(Center, R, r, N,n,col, ref All);

}

//Параметричні рівняння тору

private Point3D GetPosition(double R, double r, double t, double p)

{

Point3D pt = new Point3D();

double snt = Math.Sin(t \* Math.PI / 180);

double cnt = Math.Cos(t \* Math.PI / 180);

double snp = Math.Sin(p \* Math.PI / 180);

double cnp = Math.Cos(p \* Math.PI / 180);

pt.X = (R + r \* cnp) \* cnt;

pt.Y = r \* snp;

pt.Z = -(R + r \* cnp) \* snt;

return pt;

}

//Створює тор з заданим кольором(color), центром(center), радіусом утворюючого кола та радіусом порожнини (r,R), а також із вказаною кількістю полігонів по вертикалі та горизонталі(n,N)

private void CreateTorus(Point3D center, double R, double r, int N, int n, Color color, ref Model3DGroup All)

{

if (N < 2 || n < 2)

return;

Point3D[] p = new Point3D[4];

int coun = 0;

MeshGeometry3D mesh = new MeshGeometry3D();

for (int i = 0; i < N; i++)

{

for (int j = 0; j < n; j++)

{

p[0] = GetPosition(R, r, i \* 360 / N, j \* 360 / n);

p[0] += (Vector3D)center;

p[1] = GetPosition(R, r, (i+1) \* 360 / N, j \* 360 / n);

p[1] += (Vector3D)center;

p[2] = GetPosition(R, r, (i+1) \* 360 / N, (j+1) \* 360 / n);

p[2] += (Vector3D)center;

p[3] = GetPosition(R, r, i \* 360 / N, (j+1) \* 360 / n);

p[3] += (Vector3D)center;

CreateTriangleFace(p[0], p[1], p[2], mesh, ref coun);

CreateTriangleFace(p[2], p[3], p[0], mesh, ref coun);

}

}

SolidColorBrush brush = new SolidColorBrush();

brush.Color = color;

Material material = new DiffuseMaterial(brush);

GeometryModel3D geometry = new GeometryModel3D(mesh, material);

model = geometry;

model.Transform = transform;

All.Children.Add(geometry);

}

}

}

Shapes.cs:

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Media;

using System.Windows.Media.Media3D;

namespace Курсова\_робота\_Найденко\_Влад\_\_10\_А

{

public class Shapes

{

protected GeometryModel3D model;

protected Transform3DGroup transform = new Transform3DGroup();

protected ScaleTransform3D ScaleValue = new ScaleTransform3D(1, 1, 1);

protected AxisAngleRotation3D AxX = new AxisAngleRotation3D(new Vector3D(1, 0, 0), 0);

protected AxisAngleRotation3D AxY = new AxisAngleRotation3D(new Vector3D(0, 1, 0), 0);

protected AxisAngleRotation3D AxZ = new AxisAngleRotation3D(new Vector3D(0, 0, 1), 0);

protected TranslateTransform3D offset = new TranslateTransform3D(0,0,0);

public Shapes(Point3D centre)

{

ScaleValue.CenterX = centre.X;

ScaleValue.CenterY = centre.Y;

ScaleValue.CenterZ = centre.Z;

//

transform.Children.Add(ScaleValue);

RotateTransform3D RotateTransformX = new RotateTransform3D(AxX);

RotateTransform3D RotateTransformY = new RotateTransform3D(AxY);

RotateTransform3D RotateTransformZ = new RotateTransform3D(AxZ);

//

RotateTransformX.CenterX = centre.X;

RotateTransformX.CenterY = centre.Y;

RotateTransformX.CenterZ = centre.Z;

RotateTransformY.CenterX = centre.X;

RotateTransformY.CenterY = centre.Y;

RotateTransformY.CenterZ = centre.Z;

RotateTransformZ.CenterX = centre.X;

RotateTransformZ.CenterY = centre.Y;

RotateTransformZ.CenterZ = centre.Z;

//

transform.Children.Add(RotateTransformX);

transform.Children.Add(RotateTransformY);

transform.Children.Add(RotateTransformZ);

transform.Children.Add(offset);

}

//повертає об'єкт навколо осі X на задану кількість градусів

public void RotateX(double AngleX)

{

AxX.Angle = AngleX;

}

//повертає об'єкт навколо осі Y на задану кількість градусів

public void RotateY(double AngleY)

{

AxY.Angle = AngleY;

}

//повертає об'єкт навколо осі Z на задану кількість градусів

public void RotateZ(double AngleZ)

{

AxZ.Angle = AngleZ;

}

//масштабує об'єкт у 2 рази по осі X у задану кількість разів

public void ScaleX(double ScaleX)

{

ScaleValue.ScaleX = ScaleX;

}

//масштабує об'єкт у 2 рази по осі Y у задану кількість разів

public void ScaleY(double ScaleY)

{

ScaleValue.ScaleY = ScaleY;

}

//масштабує об'єкт у 2 рази по осі Z у задану кількість разів

public void ScaleZ(double ScaleZ)

{

ScaleValue.ScaleZ = ScaleZ;

}

//зміщує об'єкт на зазначену відстань по осі X

public void OffsetX(double OffsetX)

{

offset.OffsetX = OffsetX;

}

//зміщує об'єкт на зазначену відстань по осі Y

public void OffsetY(double OffsetY)

{

offset.OffsetY = OffsetY;

}

//зміщує об'єкт на зазначену відстань по осі Z

public void OffsetZ(double OffsetZ)

{

offset.OffsetZ = OffsetZ;

}

//цей метод створює трикутник за трьома точками(po,p1,p2) та додає його до полігональної сітки(mesh), а вершинам присвоюється номер починаючи з count.

protected void CreateTriangleFace(Point3D po, Point3D p1, Point3D p2, MeshGeometry3D mesh, ref int count)

{

mesh.Positions.Add(po);

mesh.Positions.Add(p1);

mesh.Positions.Add(p2);

mesh.TriangleIndices.Add(count);

count++;

mesh.TriangleIndices.Add(count);

count++;

mesh.TriangleIndices.Add(count);

count++;

}

}

}