Міністерство освіти і науки України

Запорізька державна інженерна академія

Факультет енергетики, електроніки та інформаційних технологій

Кафедра програмного забезпечення автоматизованих систем

**КУРСОВА РОБОТА**

з дисципліни **ШТУЧНИЙ ІНТЕЛЕКТ**

на тему: **Динамічне програмування: оптимальна тріангуляція полігону**

Спеціальність 6.050103 „Програмна інженерія”

Виконав студент групи СП-14-1

Кущ Богдан

Керівник роботи Безверхий А.І.

Оцінка

Члени комісії: доц. Безверхий А.І.

проф. Вербицький В.Г.

Запоріжжя

2018

**Реферат**

Об'єм роботи складає 40 сторінок. Кількість рисунків – 6. У даній роботі використовувалось 5 джерел.

Мета роботи – здобути та засвоїти навички динамічного програмування на базі реалізації застосунку редактор полігонів, з алгоритмом тріангуляції.

Результатами роботи є реалізація застосунку, який відповідає вимогам курсового завдання, закріплення знань про динамічне програмування, знань з алгоритмами тріангуляції зокрема Matrix Chain Multiplication, та вирішення проблем за допомогою динамічного програмування, здобуття практичного досвіду з програмування застосунку для редактору полігонів.

Ключовими словами в даній роботі були: полігон, тріангуляція, динамічне програмування, трикутник, дерево розбору, клас, метод, штучній інтелект.

Зміст

[1 ВСТУП 4](#_Toc515409378)

[1.1 Глосарій 5](#_Toc515409379)

[1.2 Опис предметної області 6](#_Toc515409380)

[1.3 Неформальна постановка задачі 7](#_Toc515409381)

[1.4 Огляд існуючих методів рішення 7](#_Toc515409382)

[1.5 План робіт 11](#_Toc515409383)

[2 ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ 12](#_Toc515409384)

[2.2 Вимоги до програмного забезпечення 13](#_Toc515409385)

[2.3 Вимоги до користувача 13](#_Toc515409386)

[3 ВАРІАНТИ КОРИСТУВАННЯ 14](#_Toc515409387)

[3.1. Архітектура системи 15](#_Toc515409388)

[3.2. Діаграма класів 17](#_Toc515409389)

[3.3. Основні атрибути і методи. Їх короткий опис 18](#_Toc515409390)

[4 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ 30](#_Toc515409391)

[4.1 Засоби реалізації 30](#_Toc515409392)

[4.2 Проект інтерфейсу 30](#_Toc515409393)

[4.3. Реалізація та тестування 35](#_Toc515409394)

[5 ВИСНОВКИ 36](#_Toc515409395)

[6 ЛІТЕРАТУРА 37](#_Toc515409396)

# 1 ВСТУП

Метою курсової роботи є реалізація редактору полігонів, та реалізація алгоритму оптимальної тріангуляції полігона з можливостю перегляду дерева перебору.

Багато проблем в області комп’ютерної графіки, обчислювальна геометрія, геоінформатики. Цій задачі приділяється велика увага адже цей алгоритм повинен бути швидким для будь-яких багатокутників, ми знаємо що GPU вміє малювати лише трикутники. Також алгоритм використовують для моделювання рельєфу, або швидкої замальовки багатокутників. Інші види тріангуляції використовують для побудови великих інтегральних схем, для оптимізації довжини ліній електропередач та телефонної мережі, на основі остового дерева може бути побудовано розв'язок задачі комівояжера, в геодезії, при вирішенні диференційних рівнянь. Взагалі усюди, де потрібно апроксимувати якісь поверхні.

## 1.1 Глосарій

**Многокутник (Polygon)** – геометрична фігура, замкнена ламана (сама, або разом із точками, що лежать усередині). Вершини цієї ламаної називають вершинами многокутника (Vertex), а відрізки ламаної — сторонами многокутника (Sides). Не самопересекающийся багатокутник (тільки такі ми і будемо, як правило, розглядати) називається простим (simple).

**Тріангуляція (у геометрії)** в найзагальнішому значенні – це розбиття геометричного об'єкта на симплекси. Наприклад, на площині це розбиття на трикутники, звідки й назва. Тріангуляція тривимірного об'єкта містить розбиття на тетраедри («піраміди» різноманітних форм та розмірів), що лежать один до одного.

**Динамічне програмування** – розділ математики, який присвячено теорії і методам розв'язання багатокрокових задач оптимального управління. У динамічному програмуванні для керованого процесу серед множини усіх допустимих управлінь шукають оптимальне у сенсі деякого критерію тобто таке яке призводить до екстремального (найбільшого або найменшого) значення цільової функції.

**Трикутник** – у евклідовій геометрії геометрична фігура, яка складається з трьох точок, що не лежать на одній прямій, і трьох відрізків, що їх сполучають. Трикутник з вершинами A, B, і C. Трикутник є многокутником і 2-симплексом. В евклідовій геометрії трикутник однозначно задає площину.

**Overlapping subproblems** – if the problem can be broken down into subproblems which are reused several times or a recursive algorithm for the problem solves the same subproblem over and over rather than always generating new subproblems.

## 1.2 Опис предметної області

Зазвичай якщо потрібно замалювати багатокутник, або створити тривимірний об’єкт, це обумовлено тим що через 3 точки можна провести площину а через 4 вже не факт, також для заливки складної фігури (4 і більше точок) якщо її не розбити на трикутники (а трикутники по своїй формі опуклі) обчислень буде набагато більше і т. д. багато з цих задач вирішує комп’ютерна графіка, тому і виникає необхідність в тріангуляції полігону, до того ж у швидкій спосіб та не ресурсномісткий, до того ж з якимись мінімальними критеріями (функція вартості).

У наш час ця проблема є актуальною, тому що на даний час усі алгоритми які існують для вирішення тріангуляції багатокутника з дірками найкраща важкість алгоритму є , нажаль це найкраща важкість на даний час якщо її можна б було поліпшити комп’ютерну графіку чекав би підйом та багато застосунків які використовують графіку запрацювали б швидше.

Якщо говорити про тривіальну необхідність в тріангуляції то для цого підійшов би застосунок в якому можна б було будувати, зберігати полігони та виконувати їх тріангуляцію, а також отримувати чітку інформацію про виконані дії.

Якщо більш розгорнуто описати це: застосунок дозволяє зберігати полігони та інформацію про тріангуляцію для подальшого збереження, передачі та нарешті у використанні. Також застосунок буде надавати опції налаштування відображення, в залежності від потреби користувача.

## 1.3 Неформальна постановка задачі

Програма призначена для редагування полігонів, застосунок повинен мати реалізовувати алгоритм оптимальної тріангуляції полігону з можливістю перегляду дерево розбору. Інтерфейс повинен мати меню для збереження, відкриття існуючих збережених полігонів, меню опцій для кастомізації відображення полігона за потреби користувача. И нарешті меню з опціями побудови, видаленні вже побудованого полігону з подальшою його оптимальною тріангуляцією. Також відобразити додаткову інформацію про тріангуляцію цього полігона а саме кількість вершин, діагоналей, трикутників, а також «вартість» тріангуляції.

## 1.4 Огляд існуючих методів рішення

На просторах інтернету дуже багато програм для побудови тріангуляції зображень, це окремий пласт програмного забезпечення розроблений для створення унікального зображення, для потреб дизайну але таке програмне забезпечення нас не влаштовує.

Здебільше застосунки які нам необхідні надаються у вигляді окремих бібліотек які підключаються до розроблювального застосунку до яких надається повна інструкція по API.

Аналоги бібліотек:

* **Clip2Tri Polygon Triangulation Library**
* **OpenGL**
* **earcut.hpp**
* **Earcut Polygon Triangulation Library**

Але також існують застосунки які дозволяють розраховувати математичні задачі та будувати графіки.

**MATLAB** – пакет прикладних програм для числового аналізу, а також мова програмування, що використовується в даному пакеті. Система створена компанією The MathWorks і є зручним засобом для роботи з математичними матрицями, малювання функцій, роботи з алгоритмами, створення робочих оболонок (user interfaces) з програмами в інших мовах програмування. Хоча цей продукт спеціалізується на чисельному обчисленні, спеціальні інструментальні засоби працюють з програмним забезпеченням Maple, що робить його повноцінною системою для роботи з алгеброю.

Отже за допомогою цього застосунку можна побудувати полігон (Рис. 1) та запустити його тріангуляцію (Рис. 2).

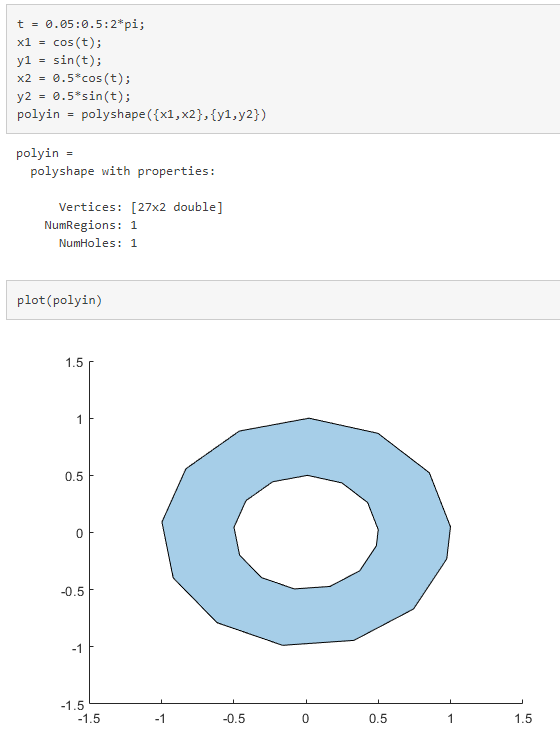


Рис. 1 побудова полігону у MATLAB

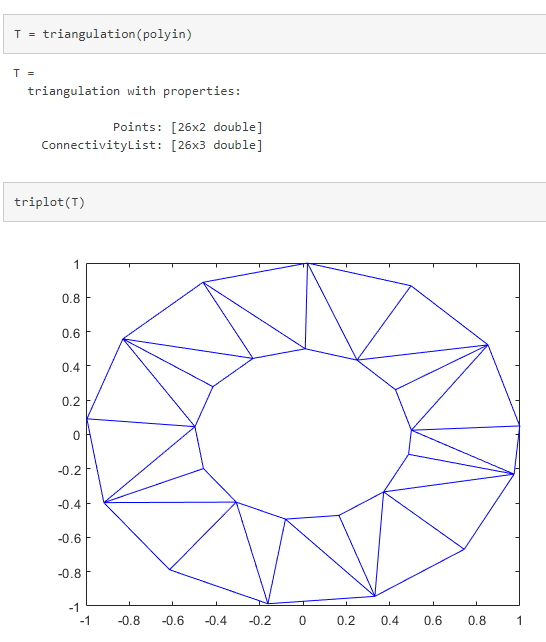


Рис. 2 тріангуляція полігону у MATLAB

Отже як ми бачимо для того щоб побудувати полігон, та виконати тріангуляцію потрібно чи мало знань, наприклад синтаксис та мову яку використовує програмне забезпечення.

**GNU Octave** – система для виконання математичних розрахунків, що надає інтерпретовану мову, багато в чому сумісну з Matlab. GNU Octave може використовуватися для вирішення лінійних завдань, нелінійних та диференціальних рівнянь, обчислень з використанням комплексних чисел і матриць, візуалізації даних, проведення математичних експериментів.

Отже за допомогою GNU Octave можна побудувати випадковий полігон (Рис. 3) та запустити його тріангуляцію (Рис. 4).

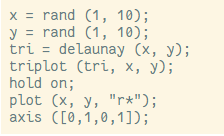


Рис. 3 код для побудови випадкового полігону та виклик його тріангуляції

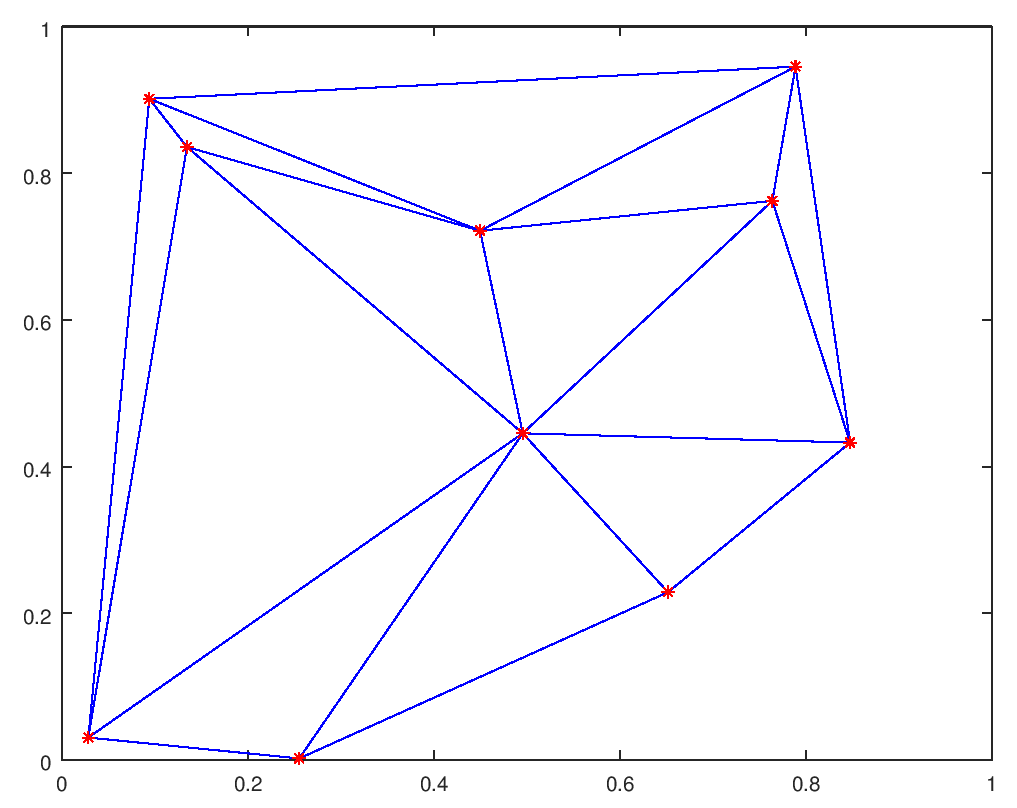


Рис. 4 результат тріангуляції у GNU Octave

Виходячи з того що у цьому застосунку також багато функцій, але потрібно знати мову та синтаксис, в прикладі (Рис. 3) ми використовували тріангуляцію по Делоне.

У висновку можна сказати що альтернативи які були розглянуті є бібліотеками або складають важкий комплекс застосунків власну мову яку потрібно вивчити та синтаксис. Аналоги дуже важкі у використанні.

## 1.5 План робіт

20.04.18 Було розроблено ТЗ

28.04.18 Складено план роботи, початок розробки

10.05.18 Перша alpha - версія програми

25.05.18 Тестування alpha - версії

27.05.18 Перехід до beta - версії

28.05.18 Тестування beta - версії програми

29.05.18 Остання версія програми

31.05.18 Презентація фінальної версії програмного продукту

# 2 ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

**Мінімальні вимоги до апаратного забезпечення:**

* **Система** Windows 10
* **Процесор** з тактовою частотою не менше 300 MHz.
* **Оперативна пам’ять** ємністю 128 Мб.
* **Жорсткий диск** об’ємом 100Мб
* **Монітор** з розподільною здатністю 800х600.

**Рекомендовані вимоги до апаратного забезпечення:**

* **Система** Windows 10
* **Процесор** з тактовою частотою не менше 1 ГГц.
* **Оперативна пам’ять** ємністю 1024Мб.
* **Жорсткий диск** об’ємом 100Мб
* **Графічний процесор** 128 Мб пам'яті (адаптер з підтримкою DirectX 9), драйвери WDDM версії 1.0 і старше.
* **Монітор** з розподільною здатністю 1280х720.

## 2.2 Вимоги до програмного забезпечення

Незалежно від операційної системи, присутність Java не меншої за 8 версію.

## 2.3 Вимоги до користувача

Користувач повинен пройти базовий курс користування операційною системою на якій він працює. Базові знання з геометрії.

В цьому випадку програмне забезпечення не потребує спеціальних навичок користування.

# 3 ВАРІАНТИ КОРИСТУВАННЯ

На діаграмі (Рис. 5) відображена use case діаграма яка розкриває усі дії які може робити користувач у нашому застосунку та у кій послідовності.

Як показано на діаграмі користувачу в першу чергу потрібно створити файл полігону, або відкрити вже створений на комп’ютері за допомогою меню яке знаходиться вверху. І вже після того колі з’явиться вкладка з полігоном можна буде його зберігати, закрити, побудувати.

Побудова полігону відбувається тільки у графічному вигляді, тобто користувач обирає де потрібна знаходитись вершина полігону и клацає туди.

Після побудови полігону відкриваються нові функції оптимальна тріангуляція цього полігону. Але тільки після тріангуляції полігону буде надано доступ до розверненої інформації про тріангуляцію. Пункт меню опції доступний на протязі всього часу як и вихід з програми.

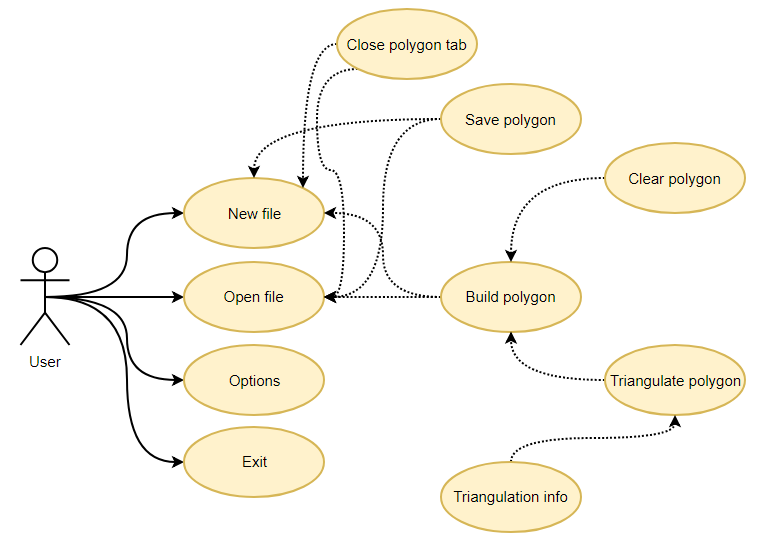


Рис. 5 діаграма варіантів використання застосунку

## 3.1. Архітектура системи

Проект застосунку побудовано на технології MVC (Model-View-Controller) View – частину представляє fxml файл який одразу за допомогою бінденга використовує методи та поля класу Controller. В якому за допомогою документованого коду усе було розбито по логічним часткам, а саме: частина яка рендерить окремий візуальній інтерфейс (верне та праве меню), обробники подій верхнього меню, обробники подій правого меню, та логіку контролера.

Кожна з логічних частин може звертатися до частини Model, але найголовніше виходячи с того що наш застосунок може працювати одразу з декількома полігонами, щоб реалізувати таку систему знадобилося у контролер додати дерево ключи якого будуть хеш-коди полігона а у значенні самі полігони. Таким чином ми зможемо контролювати відображення коректного інтерфейсу, та знаходити над яким полігоном працює користувач, це один з ключових моментів у застосунку, та розуміння як застосунок працює з декількома полігонами одразу.

Класи які включає наш основній пакет:

Класи клієнтської частини:

* Controller: компонент шаблону проектування MVC.
* CostCell: клас якій потрібен для побудови дерева розбору, використовується у матриці яка будується протягом алгоритму тріангуляції.
* DialogHelper: клас який спрощує роботу з діалоговими вікнами, використовується у контролері.
* DrawAssistant: клас який спрощує роботу з елементом Canvas, а саме якщо потрібно намалювати на ньому, то в даному класі існує такий метод щоб це зробити.
* OptionValues: клас за допомогою якого відбувається зміна налаштувань користувача, що до відображення полігону, цей клас зберігає у собі значення усіх налаштувань.
* Point2D: звичайна точка на площині, необхідність в цьому класу виникла через те що вбудовану точка не реалізує інтерфейс Serializable.
* Polygon: назва цього класу говорить сама за себе, цей клас описує полігон, а саме звичайний список точок на площині.
* Segment: цей клас описує відрізок на площині, також зберігає опис цього відрізку, та пару точок.
* TriangulatedPolygon: саме за допомогою цього класу відбувається тріангуляція полігону, зберігає в собі корінь дерева розбору, та обчислює необхідну інформацію.
* UIPolygon: за допомогою цього класу контролер може отримати візуальне представлення полігону, його відображення, видалення, тріанугялції.

Як ми бачимо (Рис. 6) проект складає з себе 10 класів, виключаючи класу Main який потрібен тільки для того щоб запустити JavaFx застосунок.

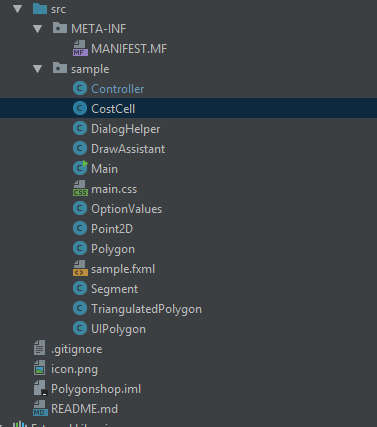


Рис. 6 вигляд проекту у IntelliJ IDEA

## 3.2. Діаграма класів

Діаграма класів нашого застосунку яка розкриває усю сутність розробки, та показує застосування патерну MVC. А також можна побачити інтерфейс Serializable який дозволяє зберігати необхідні дані у файли за допомогою рефлексії. Саме це нам знадобилось щоб зберігати полігони та зв’язну з ними інформацію про тріангуляцію.

Класи які відносяться до моделі даних використовуються тільки UIPolygon класом, це звичайна агрегація. Клас DrawAssistant зазвичай потрібен UIPolygon класу для відображення полігону а також дерева розбору.

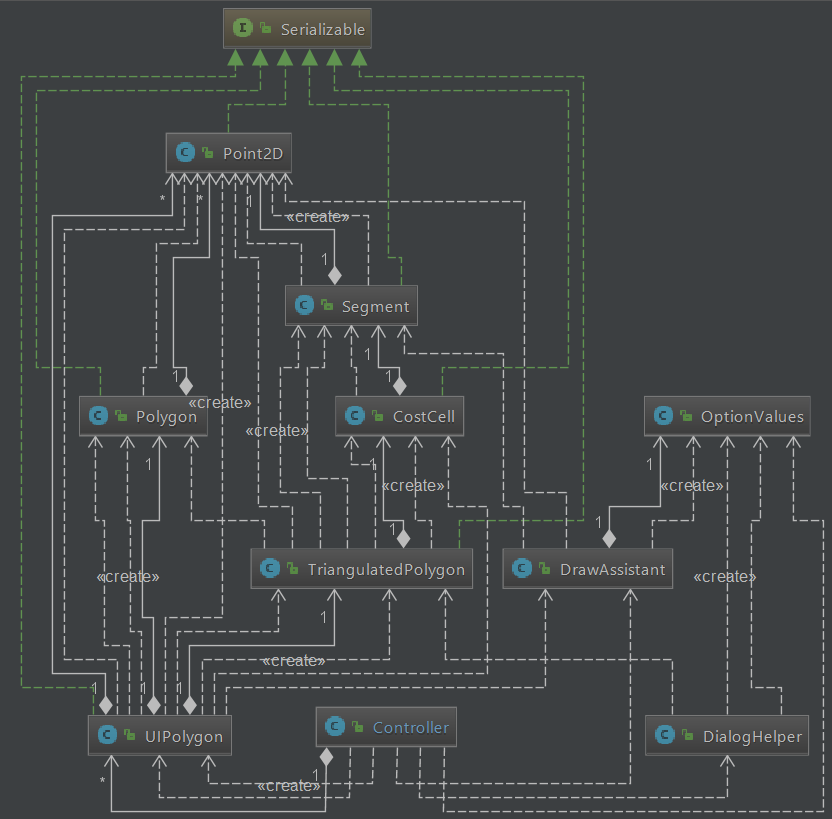


Рис. 7 діаграма класів проекту

## 3.3. Основні атрибути і методи. Їх короткий опис

**Клас Controller**

Атрибути:

* MenuItem MISave; – це біндінг даних від fxml форми, у даному випадку це елемент меню для зберігання файлів, потрібен для включення і відключення (коли е файли и їх немає).
* JFXTabPane TPPolygonFiles; – це біндінг даних від fxml форми, у даному випадку це елемент інтерфейсу TabPane в якому знаходяться вкладки з полігонами.
* JFXButton BBuildPolygon, BClearPolygon, BTriangulate, BTriangulateInfo; – це біндінг даних від fxml форми, у даному випадку це усі кнопки правого меню, це необхідно для правильного відображення.
* private String style; - це String який зберігає у собі стиль для передачі його у діалогові вікна коли це потрібно.
* private HashMap<Integer, UIPolygon> tabs; – дерево яке зберігає доступ до конкретного полігону, у ключі хеш-код табу а у значенні відповідний йому полігон (модель).
* private Integer selectedTab; – хеш-код обраного зараз табу, завдяки якому можна знайти у дереві відповідний йому полігон, якщо не обрано він буде мати null.

Методи:

* public void initialize() – початковий метод який викликається при відображенні View частини, в якому відбувається початкова ініціалізація атрибутів контролеру.
* private void renderAllUI() – метод який викликає в свою чергу рендер двох меню верхнього та правого.
* private void renderMenu() – метод рендерить верхнє меню.
* private void renderRightSideMenu() – метод рендерить меню яке знаходиться правіше.
* private void handleKeyInput(final InputEvent event) – обробник події на натискання клавіш швидкого доступу наприклад, Ctrl+N створити новий файл з полігоном.
* private void onCreateNewFile() – обробник події на натискання опції створення нового файлу.
* private void onOpenFile() – обробник події на натискання опції відкриття файлу.
* private void onSave() – обробник події на натискання опції збереження файлу.
* public void onOptions(ActionEvent actionEvent) – обробник події на натискання виклику опцій застосунку.
* private void onExit() – обробник події на натискання виходу з застосунку.
* private void onShowAbout() – обробник події на натискання виклику інформації о застосунку (версія розробник).
* private void onBuildPolygon() – обробник події за правого контекстного меню, який реагує на натискання кнопки побудови або закінчення побудови плігону.
* private void onClearPolygon() – обробник події за правого контекстного меню, який реагує на натискання кнопки очищення полігону.
* private void onTriangulatePolygon() – обробник події за правого контекстного меню, який реагує на натискання кнопки тріангуляції полігону.
* public void onTriangulateInfo() – обробник події за правого контекстного меню, який реагує на натискання кнопки відображення докладної інформації про проведену оптимальну тріангуляцію полігону.
* private UIPolygon getSelectedPolygon() – метод з логіки контролера, який повертає сутність полігону який зараз обрав користувач.
* private void addNewTab(Tab tab, UIPolygon polygonUI) – метод з логіки контролера, формально використовується коли потрібно додати новий таб (полігон) до застосунку, для подальшої роботи з ним.
* private void onCloseTab(int tabHashCode) – метод з логіки контролера, обробник який прив’язується до конкретного табу коли його закривають.
* private File createAndShowFileChooser(String type, String initialFileName) – метод з логіки контролера, який створює та викликає вікно відкриття або збереження файлу.
* private void showErrorMessage(String errMsg) – метод з логіки контролера, який виводить вікно з помилкою.

**Клас CostCell**

Атрибути:

* private Segment seg; – сегмент клітинки у матриці коштів, яка має геометричний сенс діагоналі або ребра полігону.
* private double cost; – вартість клітинки.
* private List<CostCell> subNodes; – під клітинки (під вузли дерева розбору).

Методи:

* public Segment getSeg() – метод отримання сегменту клітинки.
* public List<CostCell> getSubNodes() – метод отримання під вузлів дерева розбору.
* public double getCost() – метод отримання вартості клітинки.
* public CostCell(Segment seg) – простий конструктор.
* public CostCell(Segment seg, double cost) – конструктор класу.
* public CostCell(Segment seg, double cost, List<CostCell> subNodes) – конструктор класу.

**Клас DialogHelper**

Атрибути:

Відсутні (це «статичний» клас)

Методи:

* public static Alert getErrorDialog(String style, String msg) – метод отримання діалогу з помилкою, аргументи стиль та повідомлення помилки.
* public static Alert getConfirmSaveDialog(String style) – метод отримання діалогу підтвердження про збереження файлу.
* public static Alert getTriangulationInfoDialog(String style, TriangulatedPolygon tPol) – метод отримання діалогу про інформацію проведеної тріангуляції.
* public static Dialog<String[]> getCreateNewDialog(String style) – метод отримання діалогу про створення нового файлу полігону.
* public static Dialog<OptionValues> getOptionsDialog(String style, OptionValues values) – метод отримання діалогу з опціями застосунку.

**Клас DrawAssistant**

Атрибути:

* public static OptionValues options; – атрибут для збереження поточних налаштувань.

Методи:

* public static void drawDefaultPoint(Canvas canvas, Point2D point) – метод який дозволяє намалювати на Canvas який йому надсилають точку.
* public static void drawNodeOfTree(Canvas canvas, Point2D point) – метод який дозволяє намалювати на Canvas який йому надсилають вузол дерева розбору.
* public static void drawLeafOfTree(Canvas canvas, Point2D point) – метод який дозволяє намалювати на Canvas який йому надсилають лист дерева розбору.
* public static void drawDefaultLine(Canvas canvas, Point2D a, Point2D b) – метод який дозволяє намалювати на Canvas який йому надсилають простий відрізок.
* public static void drawDashedLine(Canvas canvas, Point2D p1, Point2D p2) – метод який дозволяє намалювати на Canvas який йому надсилають пунктирний відрізок.
* public static void drawDefaultDiagonal(Canvas canvas, Segment diagonal, boolean leaf) ) – метод який дозволяє намалювати на Canvas який йому надсилають діагональ.
* public static void drawTreeLine(Canvas canvas, Point2D a, Point2D b) ) – метод який дозволяє намалювати на Canvas який йому надсилають лінію дерева розбору.
* public static void drawText(Canvas canvas, String text, Point2D point, double xShift, double yShift) – метод який дозволяє намалювати на Canvas який йому надсилають текст.
* public static void fillCanvas(Canvas canvas) – метод який замальовує Canvas який йому надсилають одним кольором.

**Клас OptionValues**

Атрибути:

* public boolean showDescription; – опція яка відповідає за відображення опису полігону.
* public boolean showTree; – опція яка відповідає за відображення дерева розбору яке було побудовано в результаті тріангуляції.
* public double pointRadius; – опція яка відповідає за відображення точки а саме задає її радіус.
* public double lineWidth; – опція яка відповідає за відображення довжини ліній.
* public double treeLineWidth; – опція яка відповідає за відображення довжини ліній побудованого дерева у результаті тріангуляції.
* public Color polygonBackground; – опція яка відповідає за колір заднього фону полігона.
* public Color vertexColor; – опція яка відповідає за колір вершин побудованого полігону.
* public Color leafColor; – опція яка відповідає за колір вузла дерева розбору а саме листа дерева.
* public Color nodeColor; – опція яка відповідає за колір вузла дерева розбору а саме вузлів (діагоналей) дерева.
* public Color simpleLineColor; – опція яка відповідає за колір простої лінії полігону.
* public Color diagonalColor; – опція яка відповідає за колір діагоналі полігону, після його тріангуляції

Методи:

* public OptionValues(boolean showDescription, boolean showTree, double pointRadius, double lineWidth, double treeLineWidth, Color polygonBackground, Color vertexColor, Color leafColor, Color nodeColor, Color simpleLineColor, Color diagonalColor) – конструктор класу.

**Клас OptionValues**

Атрибути:

* private double x; – значення по осі х.
* private double y; – значення по осі у.
* private String desc; –опис точки (V0).

Методи:

* public double getX() – метод отримання координати х.
* public double getY() – метод отримання координати у.
* public String getDesc() – метод отримання опису точки.
* public void setDesc(String desc) – метод який призначає опис точки.
* public Point2D(double x, double y, String desc) – конструктор класу.
* public Point2D(double x, double y) – конструктор класу.
* public static double distance(Point2D a, Point2D b) – статичний метод який отримує дві точки та вираховує Евклідову відстань.
* public double distance(Point2D p) – метод який отримує іншу точку та повертає Евклідову відстань між ними.
* public String toString() – перевантажений метод який повертає опис точки.

**Клас Polygon**

Атрибути:

* private List<Point2D> points; – точки які описують полігон.

Методи:

* getPoints() – метод який повертає множину точок полігону.
* public Polygon(List<Point2D> points) – конструктор класу.

**Клас Segment**

Атрибути:

* private Point2D a, b; – точки які характеризують відрізок.
* private String desc; – опис відрізка який характеризує його.

Методи:

* public Point2D getA() – метод який повертає першу точку відрізка.
* public Point2D getB() – метод який повертає другу точку відрізка.
* public String getDesc() – метод який повертає опис відрізка.
* public void setDesc(String desc) – метод який встановлює опис відрізка.
* public Segment(Point2D a, Point2D b, String desc) – конструктор класу.
* public Segment(Point2D a, Point2D b) – конструктор класу.
* public double getCost() – метод який повертає коштовність відрізка, а саме Декартову відстань.
* public Point2D getMidpoint() – метод який повертає середину точку відрізка.
* public String toString() – перевантажений метод який повертає опис відрізка.

**Клас TriangulatedPolygon**

Атрибути:

* private CostCell rootNode; – корінь дерева розбору.
* private int n; – кількість вершин базового полігону.

Методи:

* public CostCell getRootNode() – метод який повертає корінь дерева розбору яке було побудовано після оптимального методу тріангуляції.
* public int getN() – метод повертає кількість вершин базового полігону.
* public int getTrianglesCount() – метод повертає кількість трикутників після проведення тріангуляції.
* public int getDiaglonalesCount() – метод повертає кількість діагоналей у результаті виконаної тріангуляції.
* public double getCostSum() – метод повертає вартість проведеної тріангуляції (сума діагоналей).
* public TriangulatedPolygon(Polygon polygon) – конструктор методу.

**Клас UIPolygon**

Атрибути:

* private transient Tab tab; – таб де зберігається полігон.
* private transient Canvas canvas; – об'єкт візуального інтерфейсу де малюється полігон.
* private transient List<Point2D> buildPoints; – список точок які зберігаються при побудові полігона.
* private double width; – ширина об’єкта візуального інтерфейсу де малюється полігон.
* private double height; – висота об’єкта візуального інтерфейсу де малюється полігон.
* private String fileName; – назва файлу з полігоном.
* private Polygon polygon; – геометричне представлення полігону (список точек).
* private TriangulatedPolygon tPol; – об’єкт полігону вже після тріангуляції.

Методи:

* public Tab getTab() – метод який повертає таб.
* public Canvas getCanvas() – метод який повертає об’єкт візуального інтерфейсу на якому малюється полігон.
* public double getWidth() – метод який повертає ширину об’єкту візуального інтерфейсу де малюється полігон.
* public double getHeight() – метод який повертає висоту об’єкту візуального інтерфейсу де малюється полігон.
* public String getFileName() – метод який повертає назву файлу полігона.
* public Polygon getPolygon() – метод який повертає геометричне представлення полігона.
* public TriangulatedPolygon getTriangulatedPol() – метод який повертає тріангуляцію полігона.
* public void setFileName(String fileName) – метод який назначає назву файлу полігона.
* public boolean isBuilding() – метод який повертає логічне значення, якщо полігон ще будується поверне true якщо ні то false.
* public boolean isBuilt() – метод який повертає логічне значення, якщо полігон вже побудован поверне true якщо ні то false.
* public boolean isTriangulate() – метод який повертає логічне значення, якщо полігон вже пройшов тріангуляцію поверне true якщо ні то false.
* public UIPolygon(UIPolygon p) – конструктор класу.
* public UIPolygon(String fileName, double width, double height) – конструктор класу.
* public void savePolygon(File file) – метод який зберігає полігон у файл.
* public void beginBuildPol() – метод який запускає побудову полігону.
* public void endBuildPol() – метод який завершує побудову полігону.
* public void triangulatePolygon() – метод який запускає тріангуляцію полігону.
* public void redrawPolygon() – метод який повністю перемальовує полігон.
* private void drawCostCell(CostCell cell) – рекурсивний метод для малювання дерева розбору.
* public void onClearPolygon() – метод який очищає полігон.

Повний код можна знайти у моєму репозіторії за посиланням: <https://github.com/hlus/Polygonshop>

**3.4. Відношення між класами**

Клас Controller керує усім сутностями полігонами, а також блокує або відкриває деякий візуальній функціонал. В свою чергу клас UIPolygon це так звана Model розділ патерну MVC саме він відповідає за геометричне представлення, тріангуляцію полігону. Також об’єкт цього класу звертається до DrawAssistant для малювання полігону, дерева розбору, та заливки робочої поверхні.

Щодо так званих помічників, а саме DrawAssistant надає допомогу у малюванні необхідних об’єктів на робочій поверхні, це там де ми будуємо полігон. Інший помічник це DialogHelper з його допомогою наш контролер значно зменшується, тому що цей клас помічник створює нам діалогові вікна, які нам потрібно, та повертає їх нам для подальшого використання в контролері.

Також буде цікавим об’єкт TriangulatedPolygon який використовується в об’єкті класу UIPolygon для тріангуляції полігону. Саме у TriangulatedPolygon відбувається тріангуляція полігона викликається конструктор туди надається звичайний полігон і у результаті отримуємо об’єкт класу TriangulatedPolygon який описує дерево розбору. Найцікавіше що алгоритм використовує спеціальні клітинки у оптимальному алгоритмі де будує матрицю, саме за допомогою цих клітинок будується дерево розбору, в оригінальному вигляді алгоритм не має такого. Саме це поліпшення дозволяє нам дізнатися усі вузли дерева з їх координатами та описом.

# 4 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ

## 4.1 Засоби реалізації

Модель програмного забезпечення була реалізована засобами мови Java. Середа розробки: Intellij IDEA 2018.

Інтерфейс розроблювався за допомогою javaFX. Середа розробки: SceneBuilder 2.0, за допомоги бібліотеки jfoenix, яка в свою чергу надає додаткові стилі та нові візуальні компоненти. Також для поліпшення роботи використовувалась система контролю версій github.

## 4.2 Проект інтерфейсу

Головне вікно відображає усі файли з полігонами які зараз знаходяться у роботі (Рис. 8).

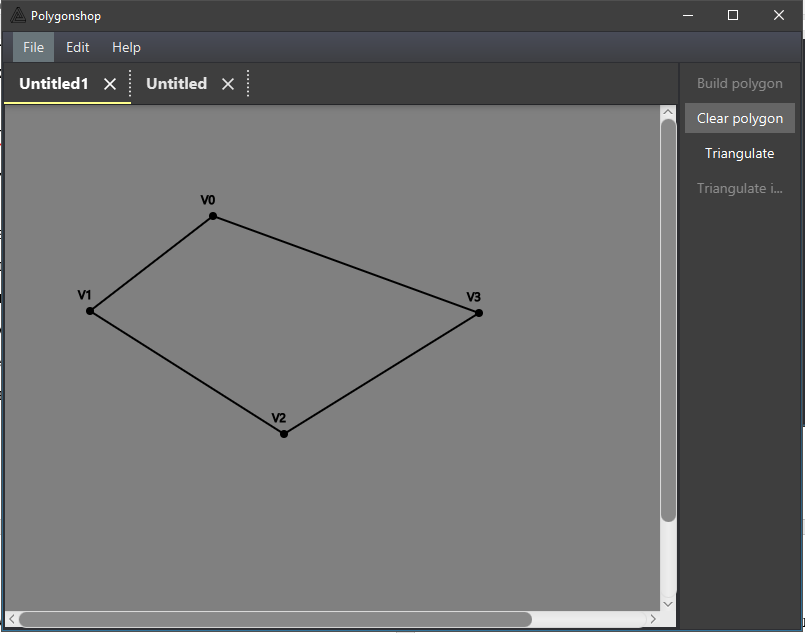


Рис. 8 головне вікно застосунку

Головне вікно надає доступ до меню в якому можна створювати файли (Рис. 9), відкривати файли (Рис. 10), змінювати опції відображення (Рис. 11).

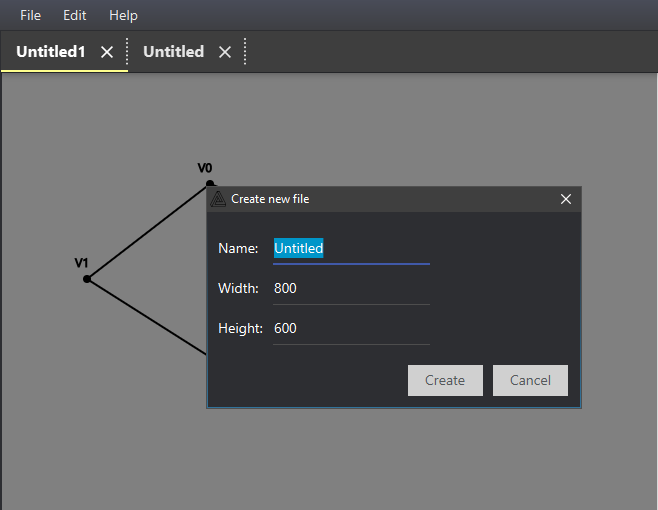


Рис.9 діалогове вікно для створення нового файлу-полігону

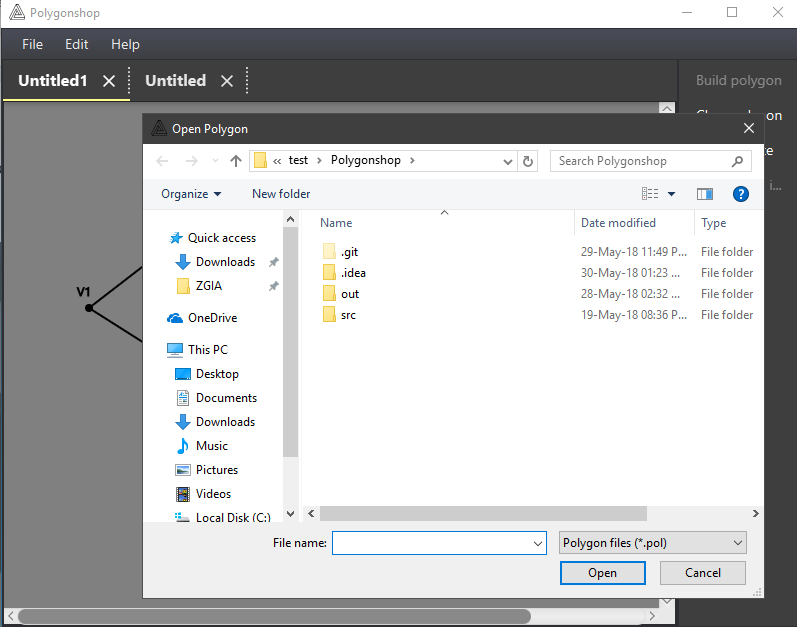


Рис.10 діалогове вікно відкриття файлів-полігонів

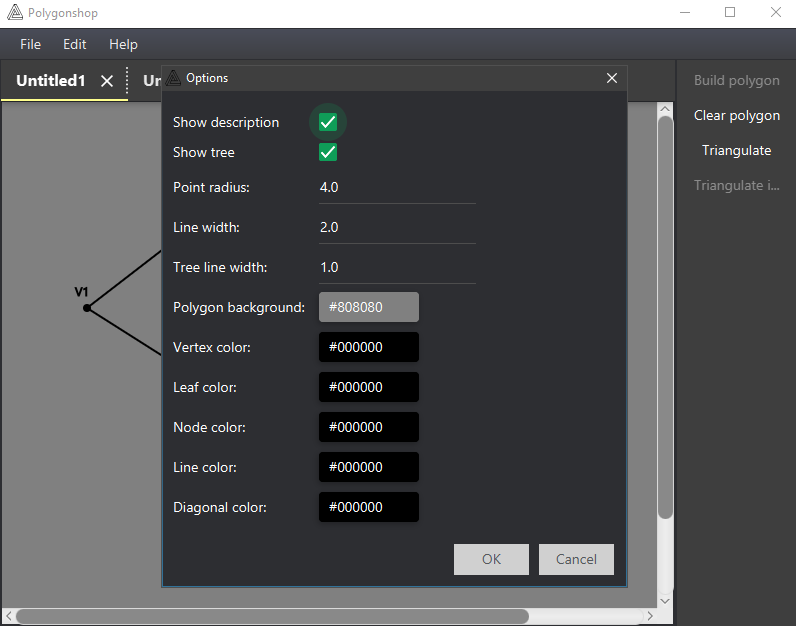


Рис. 11 діалогове вікно опцій відображення полігонів

Коли ви обрали файл для побудови полігону, є можливість почати побудову полігону, для цього у лівому меню потрібно натиснути Build polygon та нанести вершини (Рис. 12). Після чого потрібно натиснути кнопку Finish building для завершення нанесення вершин, та застосунок побудує ваш полігон (Рис. 13).

Надалі у разі необхідності можна викликати тріангуляцію полігона, натиснувши Triangulate, після чого застосунок виконає тріангуляцію, та побудує дерево розбору (Рис. 14). Також у разі необхідності, можна відобразити додаткову інформацію про виконану тріангуляцію натиснувши кнопку Triangulation, info після чого ви побачите додаткову інформацію про проведену тріангуляцію (Рис. 15).

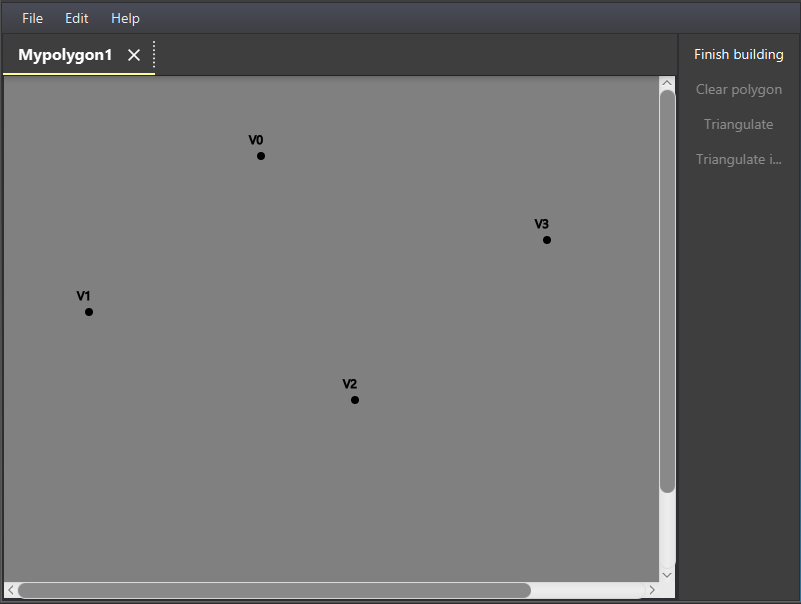


Рис. 12 нанесення вершин для побудови полігона

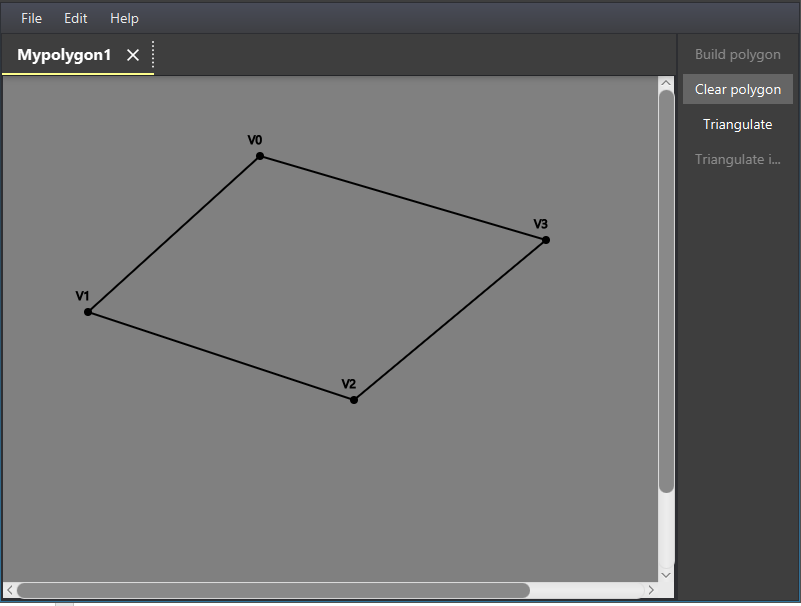


Рис. 13 побудова полігона

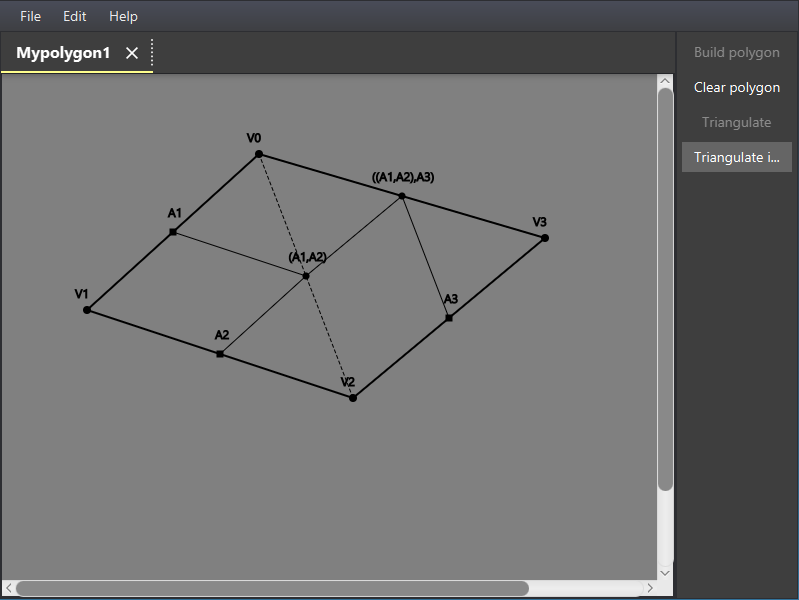


Рис. 14 вдала тріангуляція та побудова дерева

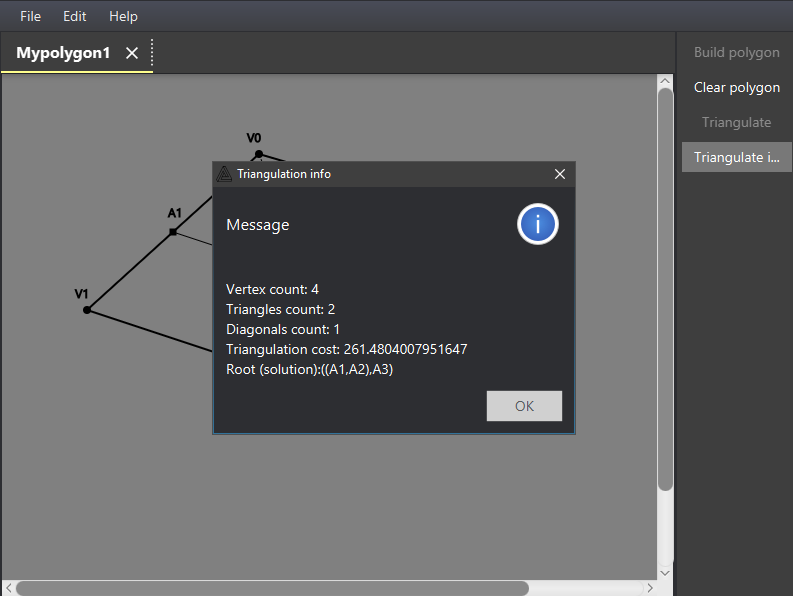


Рис. 15 додаткова інформація про триангуляцію

Маню налаштувань дозволяє змінювати вигляд полігона, якщо на щось потрібно звернути увагу (Рис. 16).

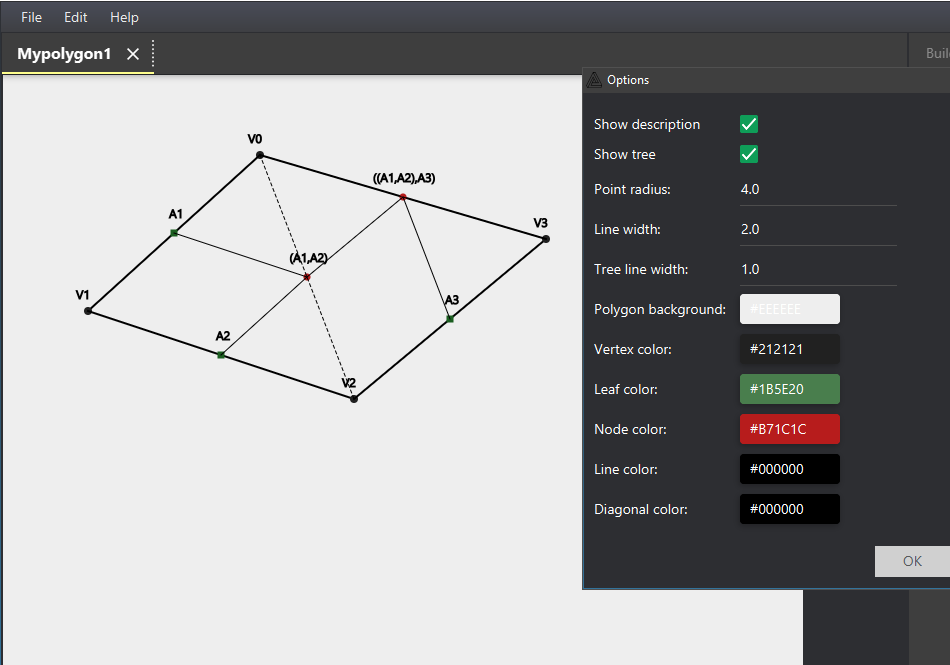


Рис. 16 зміна стандартних налаштувань відображення полігону

## 4.3. **Реалізація та тестування**

**Фізичні характеристики даної системи:**

1) Об’єм коду в строках– 1636.

2) Об’єм коду в КБ– 51,2.

3) Кількість форм – 1.

4) Витрати оперативної пам'яті МБ ~75.

5) Затрати часу на обробку основних операцій~ 0.2 сек.

# **5 ВИСНОВКИ**

* У процесі виконання курсової роботи було розроблено редактор полігонів з реалізацією оптимальної тріангуляції.
* Було спроектовано об'єктно–орієнтовану модель програмного забезпечення: виділено основні класи, встановлені залежності між ними і побудовано діаграму класів. Засвоєні принципи об'єктно – орієнтованого програмування (успадкування, інкапсуляція, поліморфізм), залежність між класами.
* Освоєні основи алгоритму оптимальної тріангуляції для полігонів.
* Освоєні принципи побудови дерева розбору.
* Були розібрані та ознайомлені інші алгоритми тріангуляції полігонів.
* Освоєні класи: Serializable, Point2D, Dialog і т.д.
* Були вивчення підключення та засоби використана бібліотеки для Material Design, яка називається jfoenix
* Також ознайомлення з системою контролю версій github.

# 6 ЛІТЕРАТУРА

1. Оформлення і захист курсових, дипломних та кваліфікаційних робіт Методичні вказівки для студ. ЗДІА спец. 080403 "ПЗАС" ден. та заоч. форм навчання / А. І. Безверхий, Н. П. Полякова, В. І. Попівщий, І. А. Скрипник ; ЗДІА. – Запоріжжя : ЗДІА, 2007.
2. Гайсарян С.С. Объектно– ориентированное программирование. М.: ЦИТ, 2002.
3. Иан Грэхем. Объектно– ориентированные методы. Принципы и практика— 3– е изд. — М.: «Вильямс», 2004.
4. Антони Синтес. Освой самостоятельно объектно– ориентированное программирование за 21 день— М.: «Вильямс», 2002.
5. Толок В.О., Киричесвкий В.В., Волкова Т.Д. Курс математики для економістів. Навчальний посібник в трьох частинах. Ч. 1. – К.: Наук. думка, 2002. – 336 с.
6. Толок В.О., Киричесвкий В.В., Волкова Т.Д. Курс математики для економістів. Навчальний посібник в трьох частинах. Ч. 2. – К.: Наук. думка, 2002. – 336 с.
7. Толок В.О., Киричесвкий В.В., Волкова Т.Д. Курс математики для економістів. Навчальний посібник в трьох частинах. Ч. 3. – К.: Наук. думка, 2002. – 336 с.
8. Курапов С.В., Савин В.В. Векторная алгебра и рисунок графа. Запорожье: ЗГУ, 2003. – 200с.