Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Магістерська робота

на тему: **“ Рекомендаційна система для побудови планіметричних малюнків”**

Виконав: студент 2-го року навчання
спеціальності
122 Комп’ютерні науки
Димченко Олексій Віталійович
Керівник Жежерун. О.П.
доцент, к.н

Магістерська робота захищена
з оцінкою _____
Секретар ЕК _____
“____” _____ 2021 р.

Київ 2021

Національний університет “Києво-Могилянська академія”
Кафедра інформатики факультету інформатики
Освітній ступень магістра
Спеціальність 122 Комп’ютерні науки
Освітньо-наукова програма Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики

Гороховський С. С.

“____”_____ 2021 року

ЗАВДАННЯ

ДЛЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Димченку Олексію Віталійовичу

1. Тема роботи: Рекомендаційна система для побудови планіметричних малюнків
керівник роботи Жежерун Олександр Петрович, доцент, к.н

затверджені наказом вищого навчального закладу від

“____”_____ 2021 року №___

2. Строк подання студентом роботи 6 червня 2021 року

3. План роботи

Вступ

1. Аналіз існуючих систем для побудови планіметричних малюнків

2. Порівняння та удосконалення існуючих систем

3. Опис та особливості реалізації власного рішення із удосконаленням

Висновки

Список джерел

Додатки

ГРАФІК ПІДГОТОВКИ МАГІСТЕРСЬКОЇ РОБОТИ ДО ЗАХИСТУ

№	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника
1.	Вибір теми. Схвалення теми кафедрою. Закріплення наукового керівника.	жовтень		
2.	Складання календарного графіка роботи.	кінець жовтня		
3.	Вивчення літератури за темою роботи.	листопад-грудень		
4.	Складання плану магістерської роботи та узгодження з науковим керівником.	грудень		
5.	Робота над практичною частиною.	грудень-лютий		
6.	Проміжний контроль.	березень		
7.	Написання основної частини, ознайомлення наукового керівника з її першим варіантом.	березень-квітень		
8.	Завершення написання роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику.	початок травня		
9.	Підготовка до захисту роботи на засіданні кафедри.	початок травня		
10.	Попередній захист роботи на засіданні кафедри.	середина травня		
11.	Подання роботи на кафедру з усіма супроводжувальними документами.	початок червня		
12.	Публічний захист роботи перед екзаменаційною комісією.	середина червня		

Графік узгоджено “____”_____ 2021 року
Науковий керівник Жежерун О. П.
Виконавець магістерської роботи Димченко О. В.

Анотація

Тема магістерської роботи: Рекомендаційна система для побудови планіметричних малюнків

Студент Димченко Олексій Віталійович

Рік навчання, спеціальність, факультет: 2-й рік навчання (магістерська програма), комп'ютерні науки, факультет інформатики.

Науковий керівник Жежерун О. П., доцент, к.н.

Мета роботи полягає в аналізі існуючих рішень для побудови планіметричних малюнків, їх порівнянні та удосконаленні використання таких систем у навчальних цілях

Ключові слова: планіметрія, навчання, система для навчання, побудова малюнків, геометрія, задача, математика

Вступ

Системи для малювання геометричних задач в 2D призначені для побудови геометричних фігур таких як коло, трикутник, паралелепіпед та інших. У сучасному навчальному процесі на уроках геометрії виникає нагальна проблема серед школярів - у розумінні розв'язку задач, у рішенні як підійти до доведення теореми, тощо; зі сторони вчителів - як навчити школярів знайти правильний підхід.

Вирішенню цієї проблеми сприяє впровадження в навчальний процес інформаційних технологій з метою ефективного засобу управління пізнавальною діяльністю та формування просторових уявлень учнів. Використання системи для планіметричних малюнків та здійснення обрахунків над ними може сприяти розвитку учнів та полегшити викладацьку діяльність вчителів геометрії.

Актуальність теми.

Актуальність роботи полягає у широкому спектрі використання системи для малювання планіметричних малюнків у навчальному процесі шкіл та під час домашнього навчання для поглиблення знань дітей у предметі геометрії та для загального розвитку. Існуючі системи для побудови планіметричних малюнків виконують обмежену кількість операцій, а отже потребують удосконалення для ширшого використання.

Мета і завдання дослідження.

Метою магістерської роботи є аналіз існуючих рішень для побудови планіметричних малюнків, їх порівнянні та удосконаленні використання таких систем у навчальних цілях.

Для досягнення поставленої мети необхідно вирішити наступні *завдання*:

- Проаналізувати існуючі системи для побудови планіметричних малюнків і можливість їх використання для навчання
- Здійснити комплексний аналіз та порівняння систем, вказуючи на їх слабкі та сильні сторони
- Запропонувати удосконалення існуючих систем спираючись на попередній аналіз
- Розробити власну систему для побудови планіметричних малюнків

Об'єктом дослідження є існуючі системи для побудови планіметричних малюнків.

Предметом дослідження є порівняння існуючих систем для побудови 2д малюнків та пошук шляхів їх удосконалення.

Методи дослідження.

В рамках магістерської роботи використовуються методи опису та аналізу, а також методи синтезу, індукції та дедукції.

Наукова новизна отриманих результатів.

В результаті виконання магістерської роботи було *вперше розроблено:*

- Систему для побудови планіметричних малюнків із можливістю задання та контролю при побудові малюнків конкретних задач із обмеженими даними та впровадженням теоретичної бази

удосконалено:

- Існуючі системи для побудови планіметричних малюнків

набули подальшого розвитку:

- Дослідження методів навчання геометрії;

- Удосконалення системи для побудови планіметричних малюнків

Практичне значення отриманих результатів.

Можливість використання удосконалених систем для побудови планіметричних малюнків у навчальному процесі на уроках геометрії або під час самостійного домашнього навчання.

РОЗДІЛ 1: АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ ПОБУДОВИ ПЛАНІМЕТРЧНИХ МАЛЮНКІВ

1.1 Огляд системи систем для побудови планіметричних малюнків

У 1997 році Карл Фукс прочитав лекцію в Зальцбурзькому університеті про доцільність використання калькулятора TI-92 в математичній освіті ([1]). Цей калькулятор вже був запропонований як для системи динамічної геометрії (DGS), так і для комп'ютерних систем алгебри (CAS), але ці дві системи були цілком відокремленими програмами. Маркус Хогенвартер, один із студентів пана Фукса, запропонував більш тісний зв’язок між можливостями візуалізації CAS і динамічною мінливістю DGS.

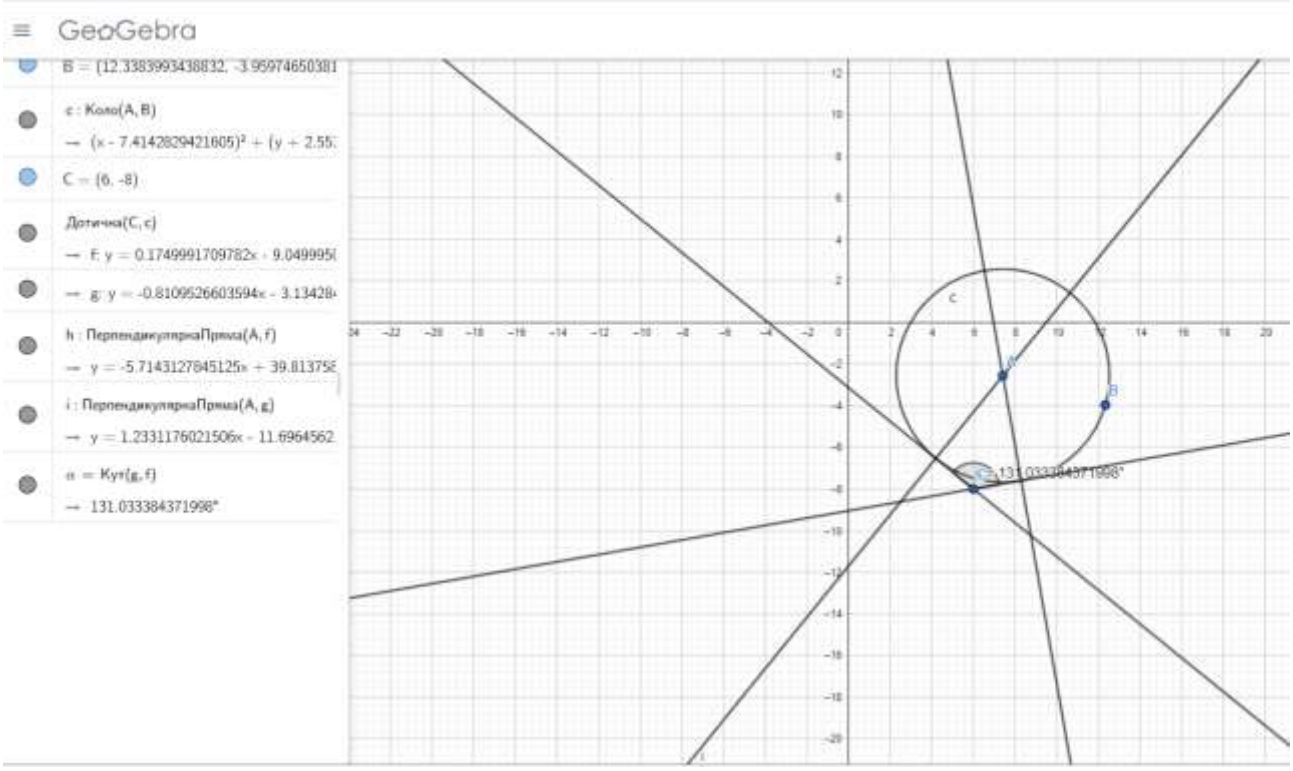
Таке бачення двостороннього поєднання динамічної геометрії та комп'ютерної алгебри вже виражали його попередники ([2], [3]): "Три протоколи рішення [графічний, числовий та алгебраїчний] не слід розглядати як окремі, а як такі, що становлять цілісний комплексний комп'ютерний підхід." [3, ст. 324] .

У 2001 році Маркус Гогенвартер розпочав роботу над магістерською роботою «GeoGebra». Метою цього проекту була розробка абсолютно нового виду інструменту для навчання математики в середніх школах.

«GeoGebra» (назва походить від скорочення двох слів Geometry та Algebra) - це інтерактивна програма для геометрії, алгебри, статистики та обчислень. «GeoGebra» доступна на багатьох платформах, із програмами для настільних ПК (Windows, MacOS та Linux), планшетів (Android, iPad та Windows) та Інтернету.

Програма орієнтована на учнів віком від 10 до 18 років та вчителів загальноосвітніх шкіл, вона заохочує підходити до математики експериментальним шляхом. Наприклад, можна дослідити параметри рівняння кола, перетягуючи коло за допомогою миші. З іншого боку, учні можуть

безпосередньо маніпулювати рівнянням і побачити змінене коло у вікні геометрії.



(Рис. 1)

Удосконалення інтерактивної побудови справді є значним. Бендер і Шрайбер вказували, що опис геометричної конструкції є до теоретичною основою ідеї алгоритму. Система побудови «GeoGebra» дозволяє змінювати конструкції в будь-який час вставляти нові елементи і навіть змінювати порядок з оглядом. Проте щоразу, коли учні вводять або видаляють математичні вирази, вони повинні знати про функціональні залежності [4].

Основними об’єктами в «GeoGebra» є точки, вектори, відрізки, багатокутники, прямі лінії та усі конічні перерізи і функції в x. За допомогою «GeoGebra» динамічні конструкції можна робити як у будь-якій іншій системі динамічної геометрії. Ці конструкції можуть бути динамічно змінені, можна вводити координати точок або векторів, рівняння прямих, конічних перерізів чи функцій

та чисел чи кутів безпосередньо.

«GeoGebra» це універсальна програма з точки зору викладання математики в середніх школах, вона може бути використана кількома шляхами:

1. Для демонстрації та візуалізації.

2. Як засіб конструювання.

3. Для розвитку математичних знань.

4. Для підготовки навчального матеріалу.

За допомогою «GeoGebra» можна створювати інтерактивні HTML-сторінки, так звані динамічні робочі аркуші, які можна використовувати з будь-яким Інтернет-браузером, що підтримує Java(Explorer, Mozilla, Netscape). Ці аркуші повністю незалежні від програми, тобто «GeoGebra» не потрібно встановлювати, щоб використовувати робочий аркуш.

«Desmos» - це програма для побудови графіків функцій та геометричних об'єктів. Система створена і працює як веб-додаток із існуючим API для розширення іншими користувачами написаний на JS. «Desmos» був створений математиком Елі Любероффом з Єльського університету і був запущений як startup на конференції TechCrunch Disrupt New York у 2011 році [5].

Модифікована версія калькулятора була використана в стандартизованих тестах, таких як тест штату Техас з оцінки академічної готовності та Віргінські стандарти навчання (SOL) [6].

У 2017 року «Desmos» випустив додаток 2D-інтерактивної геометрії - легкий, надзвичайно швидкий, браузерний інструмент, яким можна користуватись онлайн, підтримує різні інструменти для побудови ліній, відрізків, кутів, трикутників, полігонів, кіл та інших геометричних фігур [7].

«Desmos» складається із 5 основних частин

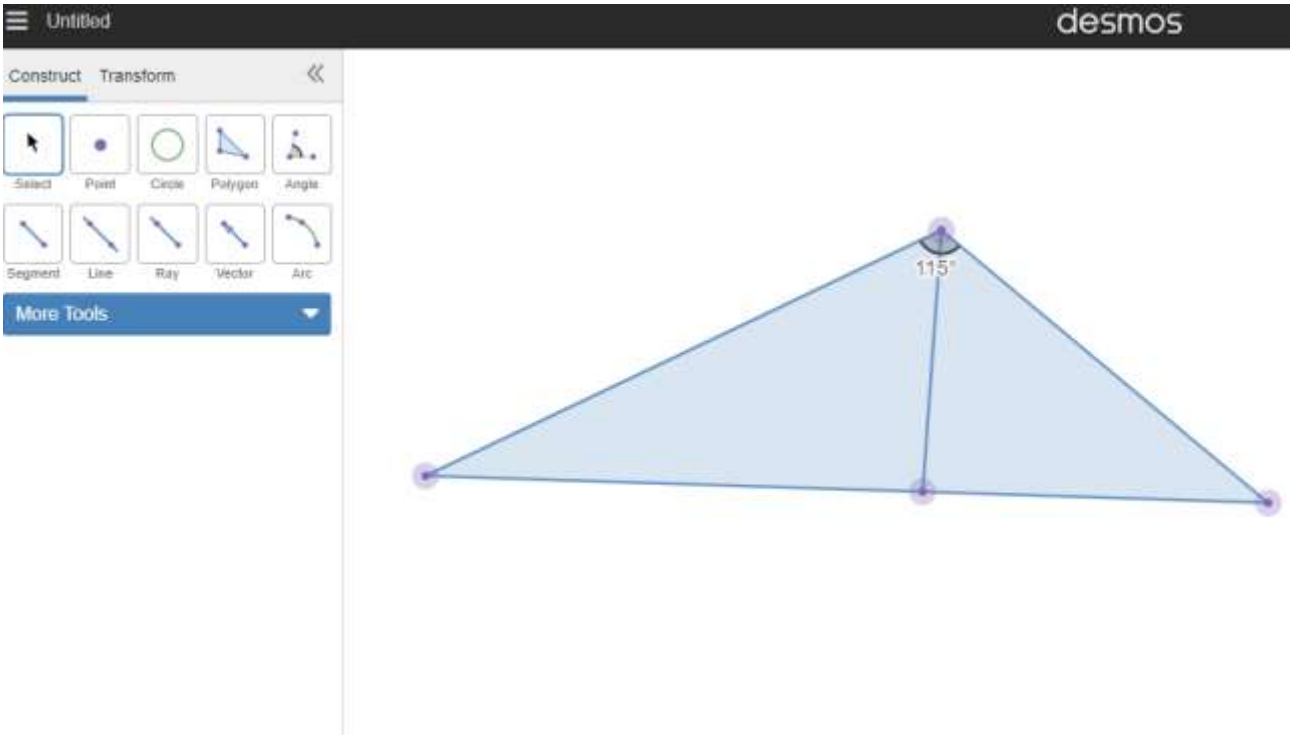
1. «Desmos» – ядро системи, через нього працюють усі інші додатки від «Desmos». Калькулятор в основному дозволяє будувати алгебраїчні графіки функцій за допомогою задання самих функцій, рівностей і нерівностей у текстовому форматі.

2. «Desmos fourfunction» –звичайний калькулятор для онлайн обчислень із багатьма можливостями як корені, степені, тощо.

3. «Desmos scientific» - інженерний калькулятор для онлайн обчислень із багатьма можливостями як корені, степені, синуси, косинуси тощо.

4. «Desmos geometry» - легкий, надзвичайно швидкий, браузерний інструмент, яким можна користуватись онлайн, підтримує різні

інструменти для побудови ліній, відрізків, кутів, трикутників, полігонів, кіл та інших геометричних фігур. Заняття в класі



(Рис. 2)

5. «Desmos teacher» дозволяють вчителю створити інтерактивну дошку, де учні можуть малювати, експериментувати, пояснювати своє мислення та взаємодіяти з процесом мислення один одного керовано та структуровано.

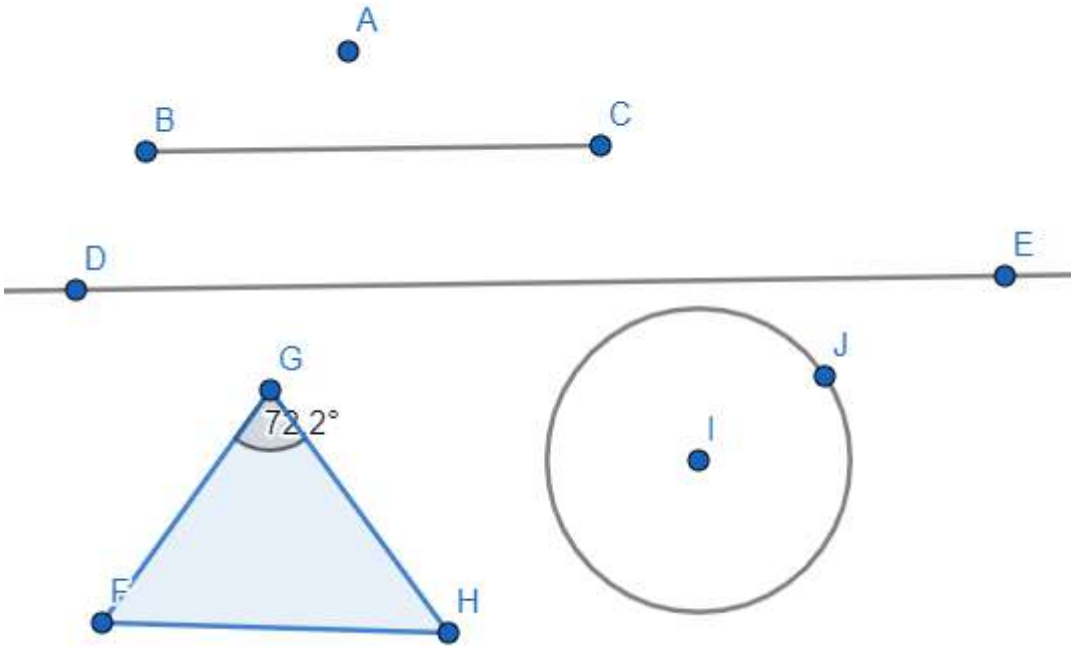
1.2 Огляд можливостей систем для побудови планіметричних малюнків

Опис можливостей «GeoGebra»

В рамках цієї роботи буде доцільніше розглянути можливості побудови саме геометрії і планіметричних малюнків, а не алгебри та графічних функцій.

Створення об’єктів, стандартні інструменти *(Рис. 3)*:

1. Малювати точки
2. Малювати відрізки
3. Малювати лінії
4. Малювати проміні
5. Малювати полігони
6. Малювати кола



(Рис. 3)

Також є можливість редагування та змінювати положення будь яких з елементів.

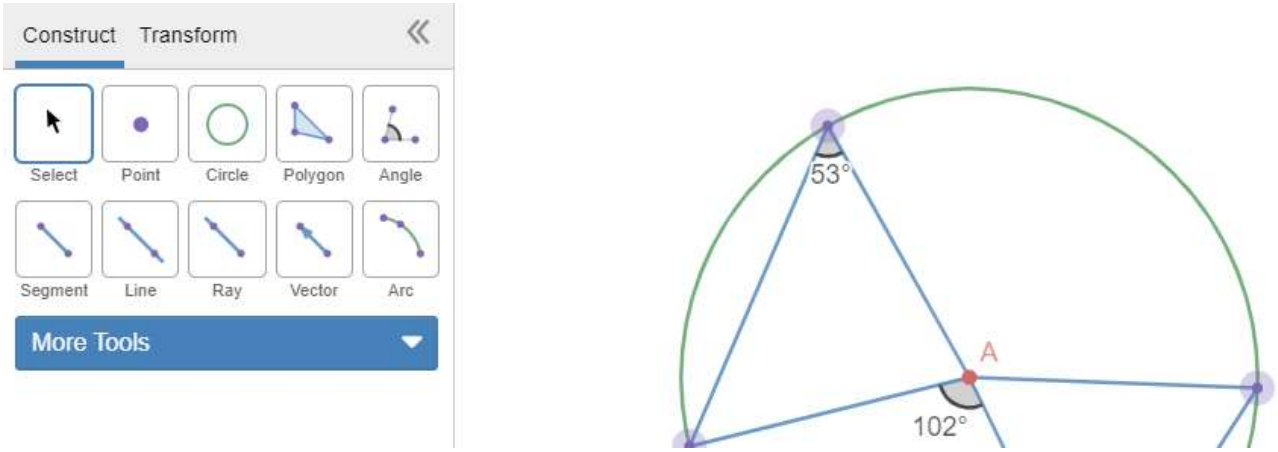
Якщо розглядати систему «GeoGebra» з точки зору навчання, то в неї є окремий модуль для вчителів. В цьому модулі люди, які працюють вчителями, можуть створювати спеціальні сторінки – книги із завданнями, а також додавати до цих завдань уже готові намальовані малюнки геометричних фігур або графіків функцій. Вчитель може додавати учнів до свого «класу» в якому учні можуть додаватись онлайн та переглядати завдання усім класом.

Після того як учень зайде до будь-якої книги і обере будь-яке завдання він зможе уже на готовому малюнку до задачі спробувати знайти відповідь. Далі учень може відправити результати вчителю, але уже іншими шляхами, не використовуючи систему «GeoGebra» або якщо учитель наглядає онлайн то він побачить правильну відповідь учня.

Опис можливостей «Desmos»

Можливості цієї системи дуже схожі на особливості системи «GeoGebra» якщо розглядати їх, як системи для малювання саме планіметричних малюнків.

У «Desmos» є можливість створювати лінії, відрізки, кути, кола, полігони, точки, промені, вектора, арки, тощо. Також гарною особливістю є те, що в «Desmos» на відміну від «GeoGebra» є можливість виконувати різного роду перетворення із об'єктами, такі як відображення, пересування, повертання.



(Рис. 4)

У «Desmos» є можливість створювати завдання, проте немає можливості створювати завдання з використанням геометричних фігур, а тільки з використанням графіків функцій. Тобто поки що немає підтримки «Desmos geometry».

У «Desmos» teacher» э власні завдання, які направленні на пояснення та навчання основних принципів математики і геометрії, а також існує можливість створювати свої завдання у яких є можливість колективом учнів та учителю дивитися за виконанням завдання іншим учнем. Учитель може корегувати дії учня, та керувати виконанням завдання. Так само можна робити і із власно створеними завданнями. Проте завдання не підтримують «Desmos geometry».

РОЗДІЛ 2: ПОРІВНЯННЯ ТА УДОСКОНАЛЛЕНЯ ІСНУЮЧИХ СИСТЕМ ДЛЯ ПОБУДОВИ ПЛАНІМЕТРЧНИХ МАЛЮНКІВ

2.1 Порівняння систем для побудови планіметричних малюнків

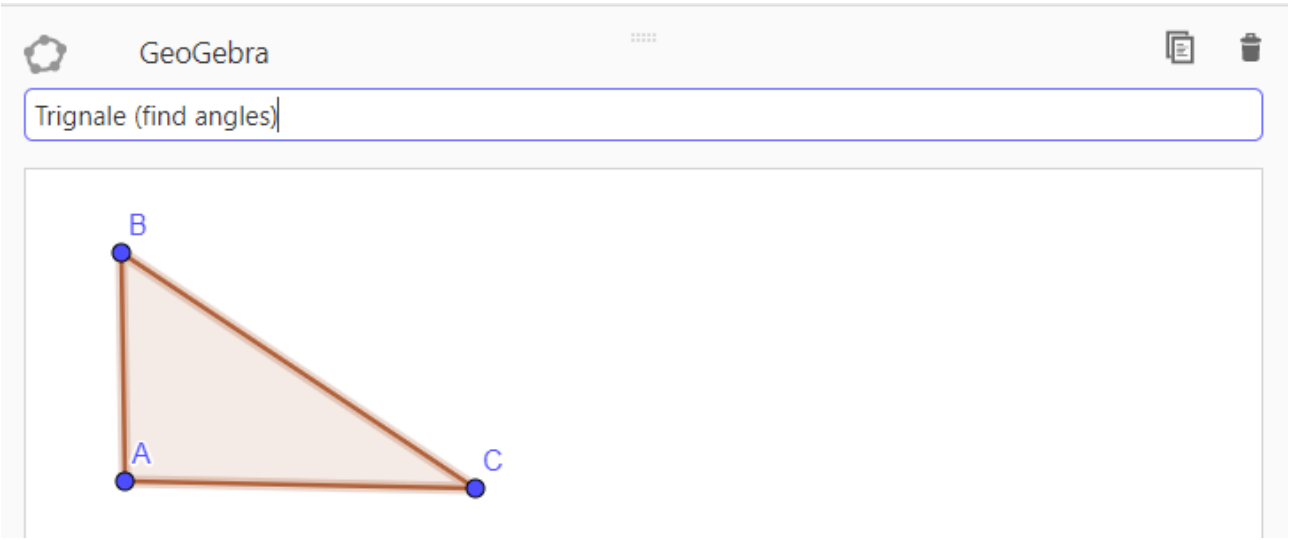
Проаналізувавши системи для планіметричних малюнків та розрахунків «GeoGebra» та «Desmos» ми прийшли до висновку, що системи досить схожі по своїй структурі та виконують і призначені для одних і тих самих задач. А саме в них можливо будувати базові геометричні примітиви такі як – трикутники, кола, відрізки, прямі, кути, точки, полігони, бісектриси, висоти, медіани тощо. Хоча їх інтерфейсі і відрізняються в цілому програми однакові, відрізняються тільки можливості для вчителів.

У системі «GeoGebra» є можливість створювати «книги» із завданнями. Кожна «книга» по суті є списком глав за якими учнем буде навчатись. В книгу можна додавати різні глави. Вчителі можуть наповнювати зміст глав за допомогою завдань. Завдання можна обирати серед публічно створених завдань або створити власне із списку варіантів.

1. Текст
2. Відео
3. Картинка
4. Нотаток
5. ПДФ файл
6. Запитання
7. Веб сторінка
8. Запропонований малюнок з «GeoGebra»

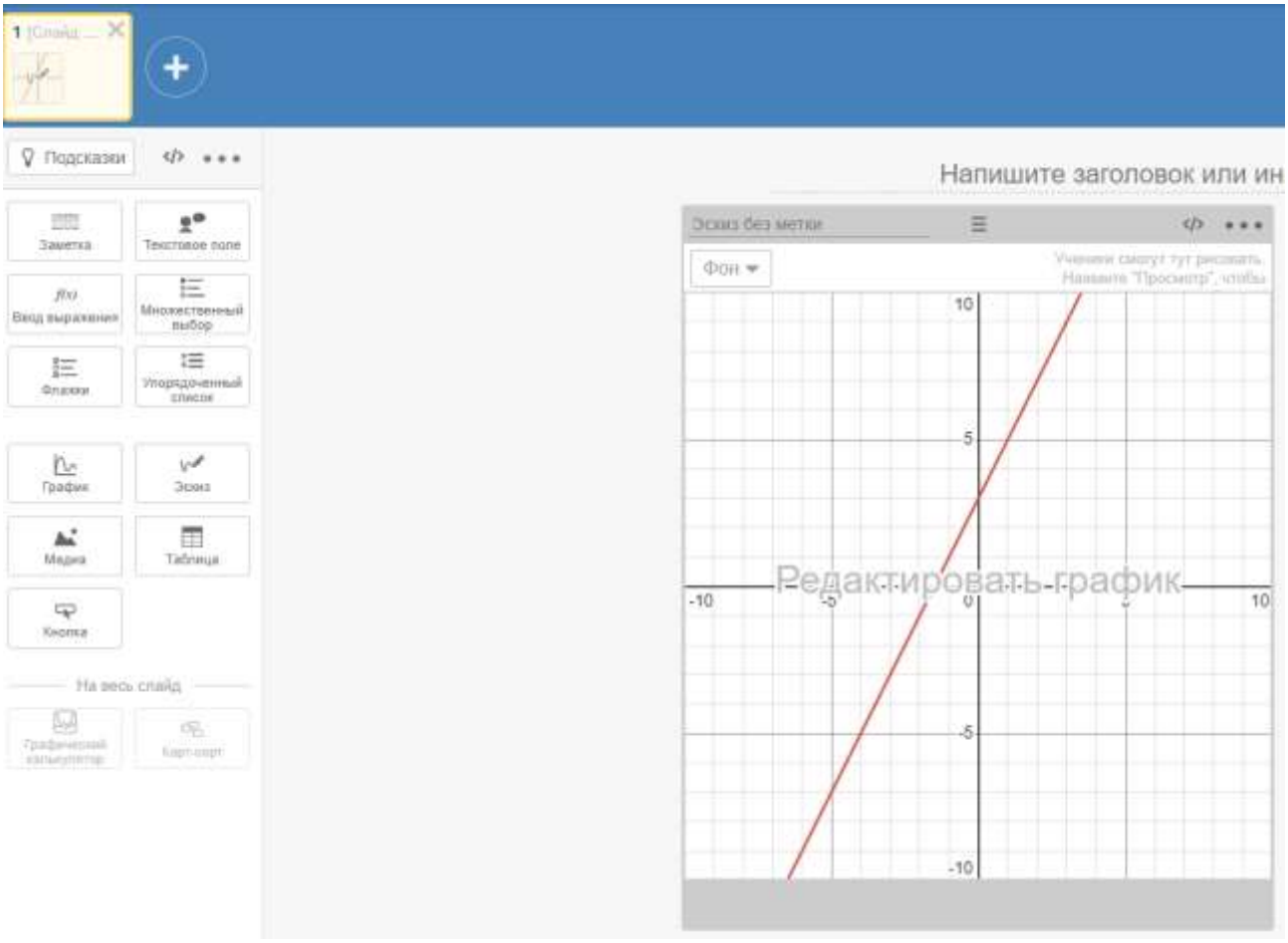
(Рис. 5)

Нас цікавить саме останній - восьмий пункт. Запропонований варіант малюнку з «GeoGebra» означає, що учитель завчасно створює малюнок до задачі та може прикріпити наприклад посилання на гугл форму, де учень дасть відповідь на поставлену задачу.



(Рис. 6)

Якщо розглядати «Desmos», то в цій системі для побудови планіметричних малюнків також є можливість створювати завдання, проте немає можливості створювати завдання з використанням геометричних фігур, а тільки з використанням графіків функцій.



(Рис. 7)

Хоча в системі для побудови планіметричних малюнків і є додаткові можливості, по типу підказок, яких немає в системі «GeoGebra», їх можна замінити тими самими прикріпленим файлами у форматі .pdf або нотаткам у «GeoGebra». Зате ««Desmos» teacher» також пропонує свої завдання у яких є можливість колективом учнів та учителю дивитися за виконанням завдання іншим учнем. Учитель може корегувати дії учня, та керувати виконанням завдання. Так само можна робити і із власно створеними завданнями.

2.2 Удосконалення існуючих систем для побудови планіметричних малюнків

Проаналізувавши декілька систем для побудови планіметричних малюнків, а саме «GeoGebra» та «Desmos», ми прийшли висновку, що обидві системи є дуже зручними у використанні для малювання різних задач та графіків, а також для використання цих систем задля навчання учнів.

В обох системах присутній функціонал який полегшує процес навчання учнів або студентів геометрії або інших математичних дисциплін. Проте усі ці засоби потребують нагляду вчителя, так наприклад у «Desmos» задачі надає і керує ними викладач онлайн, а в «GeoGebra» таку можливість можна отримати хіба що користуючись засобами по типу Teams або будучи присутнім на парі.

Нашою думкою є те, що в цих системах не вистачає контролю за якістю побудови планіметричних малюнків під час виконання учнями самостійних або домашніх завдань. Це є особливо актуальним у період карантину або виконання домашніх завдань під час канікул.

Системи можна вдосконалити додавши в них можливість додавати завдання, наприклад «Дано прямокутний трикутник зі стороною 8 см та один із кутів 45 градусів, знайдіть гіпотенузу.» у якомусь зрозумілому програмі вигляді, та допомагати і слідкувати за побудовою малюнків до цих задач за допомогою системи.

Система буде керувати учнем і повідомляти про наступні кроки побудови малюнку. Якщо учень зробить неправильне креслення малюнку система про це повідомить. А також як варіант допоможе підказкою, якщо помилка є досить розповсюдженою.

Тобто наше удосконалення має вирішувати 2 задачі. Перша – це можливість порахувати різну можливу інформацію про об’єкти задачі які були передані до системи, навіть якщо ця інформація про ці об’єкти не повна, а також порівняти малюнок який зробив учень із «приблизно» або «точно» правильним кресленням необхідним у задачі.

РОЗДІЛ 3: ОПИС ТА ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ВЛАСНОГО РІШЕННЯ ІЗ УДОСКОНАЛЕННЯМ

3.1 Використані технології

C++

Мова програмування яку використовують для розробки програмного забезпечення. Вона є однією із найпопулярніших мов програмування в світі. Її використовують для дуже широкого спектра задач, починаючи від простих додатків, закінчуючи операційними системами або «ААА» іграми. Великою переваго мови є те, що вона підтримує велику кількість парадигм, від чого розробка на цій мові є дуже гнучкою.

Головною переваго (або недоліком) мови є низькорівневий контроль за виконанням програми, а також її ресурсами, що надає програмісту можливість зробити код максимально швидким та не ресурсномістких. C++ добре підходить для додатків у яких використовується 2D або 3D графіка, під неї написано багато графічних бібліотек, бібліотек для малювання напряму спілкуючись з відеокартою, які потребують точності від програмістів, а також фреймворків, таких як QT.

C++ ми обрали тому, що ця мова є досить гнучкою і можна використовувати та запускати на різних операційних системах, також ми маємо досвід у цій мові і вважаємо, що вона ідеально підійде для подальшого розширення.

У своїй роботі ми використовуємо C++17-ого стандарту, який на даний момент є найновішою версією. C++11 додає в мову інтелектуальні указники [9]., а саме std::shared_ptr, std::unique_ptr та std::weak_ptr, які ми дуже часто використовуємо в нашій системі, оскільки об’єкти по типу трикутника є композитними об’єктами, які агрегують у собі точки, відрізки та інші елементи. При видаленні

трикутника, його точки не мають видалятись. Для цього ми досить вдало використовуємо std::shared_ptr. А також ми використовуємо std::function та lambda function, які дуже гарно використовувати для різних інструментів доступних у програмі.

OpenGL

OpenGL – відкрита графічна бібліотека [8]. По суті OpenGL – це специфікація, по суті документ який описує набір функцій та їх точну поведінку, що надає API для розробки під різні платформи та драйвери, який не залежить від мови або середовища для створення 2D та 3D графіки. Різні мови програмування та системи по різному імплементують цю специфікацію.

OpenGL був реалізований для різних операційних систем, таких як Windows, Linux, MacOS. OpenGL Включає більше 250-ти функцій для рендеру 2D та 3D графіки і використовується загалом для великих проектів, таких як відеоігри або додатків для малювання, таких як Photoshop, проте оскільки багато із реалізацій є Open source, люди створюють багато свої власних бібліотек-розширень з використанням OpenGL, які дуже спрощують проектування та розробку програмного забезпечення.

OpenGL стоїть на двох стовпах:

1. Приховати деталі реалізації рендеру за допомогою відеокарт надаючи розробнику єдиний API
2. Приховати відмінності для різних платформ

Можна сказати, що принципом роботи OpenGL є робота з точками – «вертекс» буфером, та списком того як їх поєднувати – «індекс» буфером. Після чого відбувається математична обробка. На цьому етапі відбувається створення зображення. Після чого відбувається векторна трансформація та растеризація зображення. І фіналом є передача всіх цих даних до відео карти.

Проте OpenGL нічого не знає про вікна та вхідні данні і про вхідні пристрої, оскільки OpenGL - це просто специфікація або інтерфейс для 2D та 3D рендеру.

Саме тому люди створили багато розширень, про які було згадано вище, таких як GLUT, GLSL, GLEW, GLFW, SFML тощо.

SFML

SFML – проста та швидка мультимедійна система. Являє собою Open Source бібліотеку, яка використовує OpenGl для відображення і складається з 5 модулів кожен із яких відповідає за різні задачі.

Системний модуль відповідає за зміну та керування системними налаштуваннями. Віконний модуль забезпечує користувачу можливість керувати вікнами, а також зчитувати вхідні та вихідні сигнали із девайсів вводу виводу, графічний модуль, який спрощує використання OpenGL та допомагає в відображенні графіки у вікнах. Саме цей модуль ми і використовуємо найбільше у системі. Аудіо модуль обробляє звуки та працює з апаратами відтворення та зчитування звуку. Мережевий модуль – за допомогою сокетів допомагає створити систему спілкування між мережею девайсів.

«Проста і Швидка Мультимедійна Бібліотека» - надає нам можливість яку не надає OpenGL, а саме керувати вікнами та дозволяє працювати із вхідними сигналами від мишки та клавіатури. Додатки які написані або частково використовують SFML можна легко перенести на будь-яку платформу, адже SFML є кросплатформленою бібліотекою

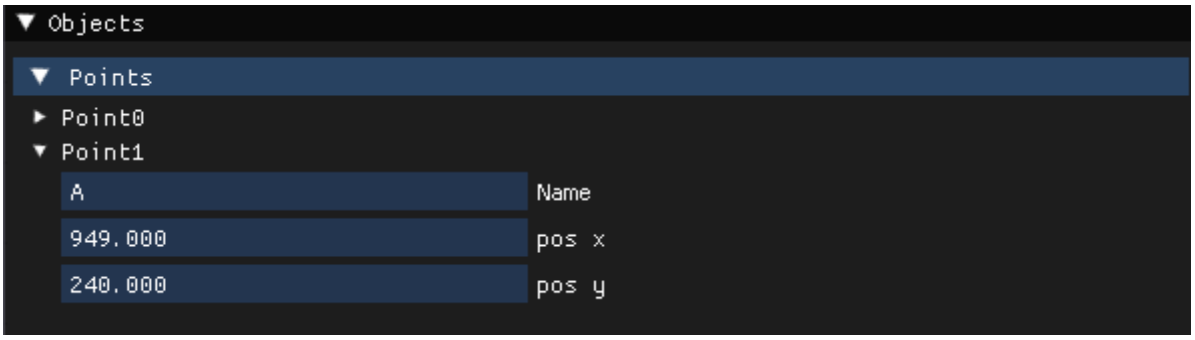
Бібліотека SFML зручна у використанні і дуже пришвидшує час програмування, а також має чудову документацію. Інтерфейсом бібліотеки SFML є так звана «state machine». Це означає, що SFML містить ряд змінних стану, які змінюються під час виконання програми.

SFML все таки бібліотека більш заточена для відображення графіки, тому нам знадобиться спосіб відображати інтерфейс користувача. Саме тут ми обрали Immediate mode GUI.

ImGUI

Графічний користувальницький інтерфейс безпосереднього режиму (GUI), також відомий як ImGUI, - це графічний шаблон дизайну інтерфейсу користувача, який використовує графічну бібліотеку безпосереднього режиму для створення графічного інтерфейсу [10].

За допомогою ImGUI можна будувати будь який користувацький інтерфейс і змінювати його прямо під час виконання програми, що є дуже зручним для нашої задачі. ImGUI використовує OpenGL для відображення інтерфейсу. Також перевагою бібліотеки є те, що вона є Open Source.



(Рис. 8)

3.2 Опис та особливості реалізації

Система для побудови планіметричних малюнків складається з декількох частин. Першою із таких частин є частина об’єктів. Об’єкт це базовий клас, який знає своє ім’я, унікальний ідентифікатор, а також свій тип, а також інформацію про те як його правильно відмалювати, та як дізнатись чи мишка наведена на нього чи ні. Усі інші об’єкти є похідним від цього.

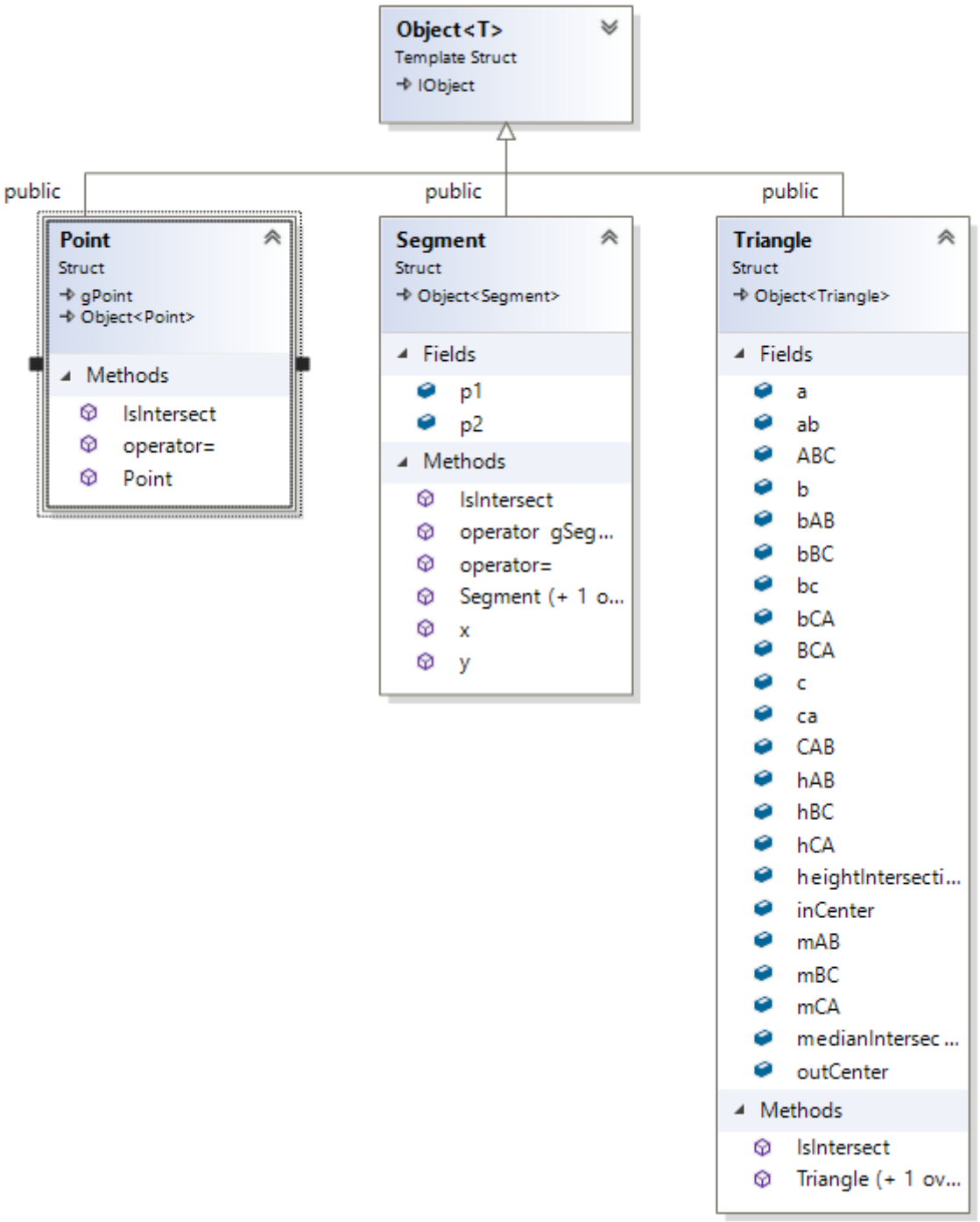
Об’єкти

Найпростішим об’єктом є точка. Вона зберігає в собі інформацію про своє положення. А також є базовим об’єктом для користувача без якого він\вона не зможе малювати інші об’єкти.

Об’єкт відрізок уже є більш складним оскільки зберігає в собі дані про дві точки. Типом цих точок в класі є std::shared_ptr<Point>, оскільки точки можуть існувати і без відрізка, а тому видалення відрізка не має призвести до видалення точок.

Об'єкт grid є дуже важливим, по суті це набір ліній які формують сітку по якій користувач зможе малювати потрібні йому об’єкти.

Об’єкт тексту – це і є просто текст але з інформацією про колір, розмір і т.д.



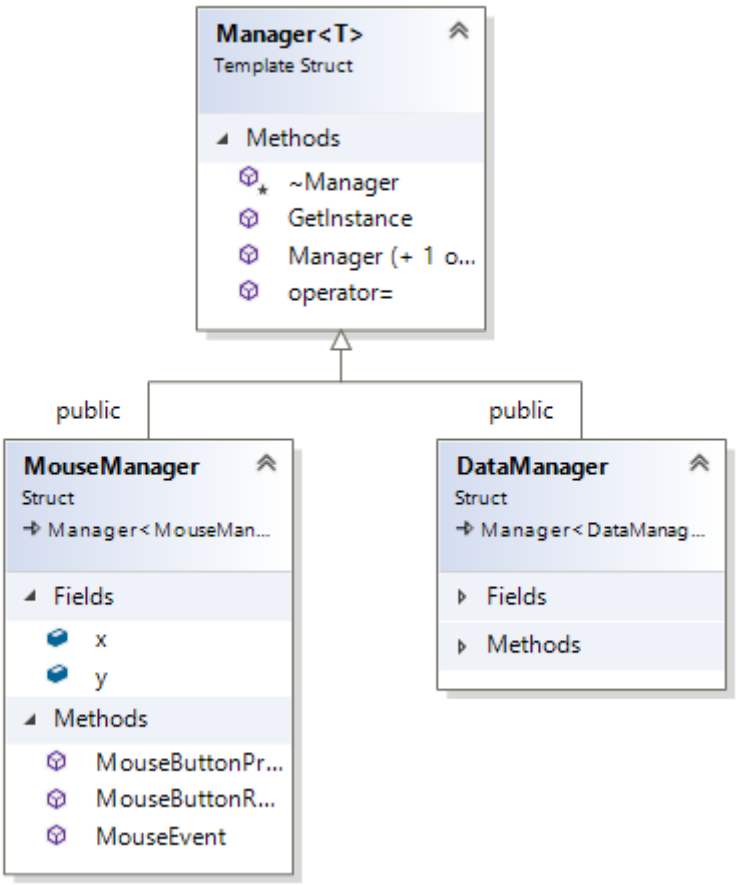
(Рис. 9)

Об’єкт трикутник уже є досить комлпексним. Він є вміщує в себе інтелектуальні указники спільного користування на усі точки які будують трикутник, а також на усі відрізки, на усі медіани, висоти, бісектриси якщо вони існують, а також на кола вписані та описані навколо нього. Під час того як будь який об’єкт додається на загальний малюнок, програма перевіряє чи не є цей

об’єкт частиною трикутника, бо якщо він є, то вона додасть його до цього трикутника.

Об’єкт коло – є об’єктом який зберігає інформацію про точку центра та радіус. Точка центра так само є унікальним указником, як і у випадку з трикутником.

Об’єкт кут – знає про три точки який його формує.



(Рис. 10)

Другою із цих частин є набір Менеджерів. Кожен із таких менеджерів є Singleton - шаблон проектування, який гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра. Всього в системі є 6 менеджерів кожен із яких відповідає за свою задачу.

Data Manager - головний із менеджерів який містить у собі усі дані які програма використовує. Тут зберігаються усі геометричні об’єкти які програма буде

відмальовувати, тимчасові об’єкти, шрифт і інша інформація про стан системи. Дуже зручно використовувати один об’єкт для цього. Якщо розширювати програму, в майбутньому буде дуже зробити додатковий клас який наповнював би Data Manager із бази даних.

Drawing Manager – менеджер який по своїй суті є обгорткою навколо SFML. Він використовує цю бібліотеку для малювання усіх об’єктів який він отримує із Data Manager. В головному циклі кожен тік програма просить у цього менеджера відмалювати всю сцену.

Keyboard Manager – менеджер який по своїй суті є обгорткою навколо SFML. Він використовує цю бібліотеку для обробки усіх вхідних і вихідних event`ів які надходять від клавіатури користувача. В головному циклі кожен тік програма просить у цього менеджера обробити усі події. Він знає про те на які клавіші було натиснуто і як правильно реагувати на ці події.

Mouse Manager – менеджер який по своїй суті є обгорткою навколо SFML. Він використовує цю бібліотеку для обробки усіх вхідних і вихідних event`ів які надходять від мишки користувача. В головному циклі кожен тік програма просить у цього менеджера обробити усі події.

Tool Manager – менеджер який відповідає за правильне переключання інструментів доступних користувачу для малювання або зміни об’єктів.

Останнім і не менш важливим є менеджер транзакцій. Створюючи або видаляючи об'єкт, програма користується цим менеджером, він запам'ятовує усі дії які відбувались в програмі. Кожна транзакція викликана в коді знає як саму себе відмінити. Таким чином в даних програми утворюються дві черги одна для redo, а інша для undo. Крім цього на етапі транзакцій може відбуватись додаткова обробка даних.

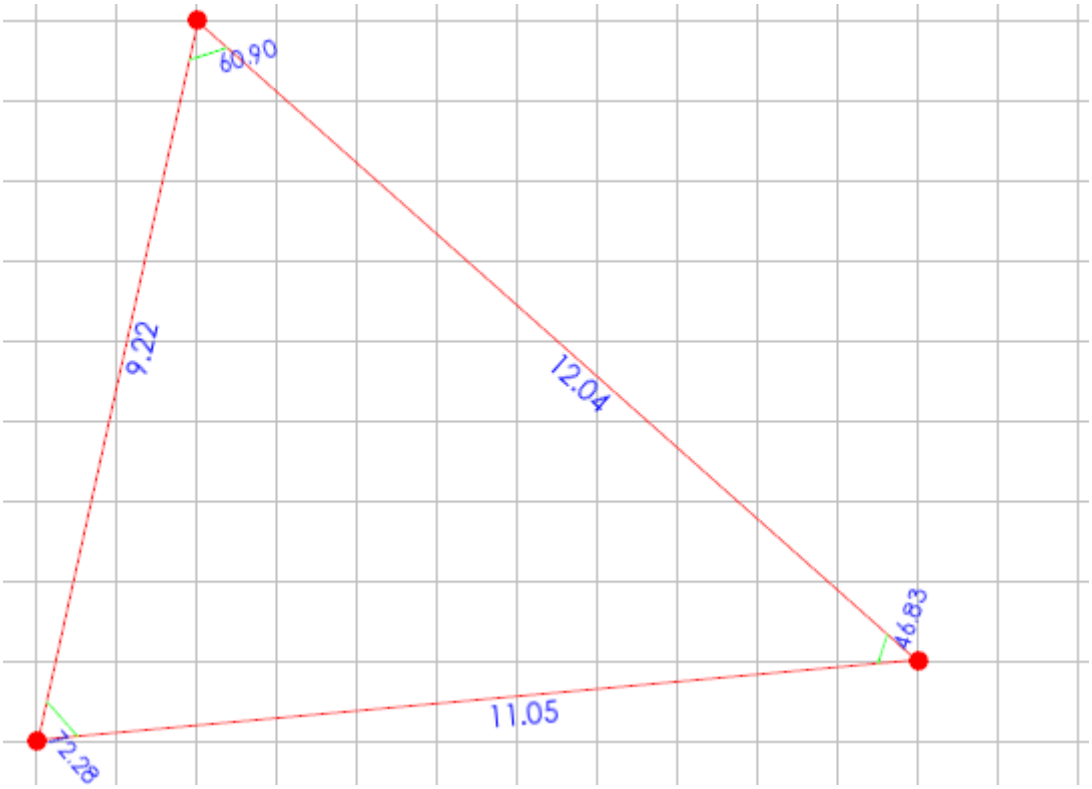
Третьою великою частиною є інструменти для малювання. За допомогою цих інструментів користувач може створити будь-який малюнок. Звісно є інструменти які створюють об’єкти, а є якість щось рахують.

Усі інструменти наслідуються від базового класу інструмента, який є інтерфейсом і вимагає від дочірніх класів імплементувати методи обробки подій мишки та\або клавіатури.

Першим і найпростішим інструментом є інструмент малювання точок. За його допомогою можна створити точку будь-де на екрані. Також точки явно прив’язані до так званих SnapPoint, це спеціальні місця на сітці для малювання в яких зручно і точно додавати точки.

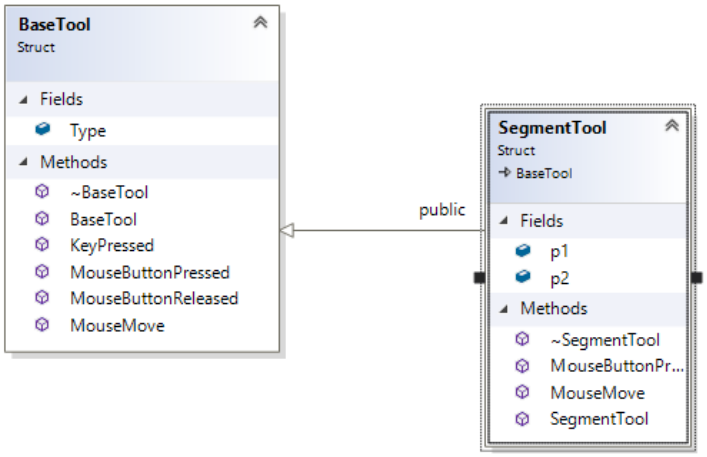
Інструмент малювання відрізків. Потребує обрати спочатку одну точку, а потім іншу. В цього, а також в деяких інших інструментах є спеціальні модифікатори клавіші, які трохи змінюють поведінку мишки. Якщо затиснути “CTRL”, то користувач зможе поставити точки де завгодно, якщо затиснути “ALT” ,то вертикаль зафіксується і користувач буде пересувати наступну точку тільки по осі x, а якщо затиснути “SHIFT”, то буде теж саме але по осі y.

Інструмент малювання трикутників має таку саму логіку, як і інструмент малювання відрізків, проте під час створення трикутника ми можемо побачити додаткову інформацію про довжину відрізків і градуси кутів.



(Рис. 11)

Інструмент для малювання кіл надає можливість користувачу намалювати коло обравши дві точки. Після чого одна з точок стане центром кола, а інша стане однією з точок на краю кола. Радіусом такого кола буде відстань між цими двома точками.



(Рис. 12)

Інструмент обирання інших об’єктів. За допомогою цього інструмента користувач зможе переміщати об’єкти, а також виділивши – зможе видалити їх.

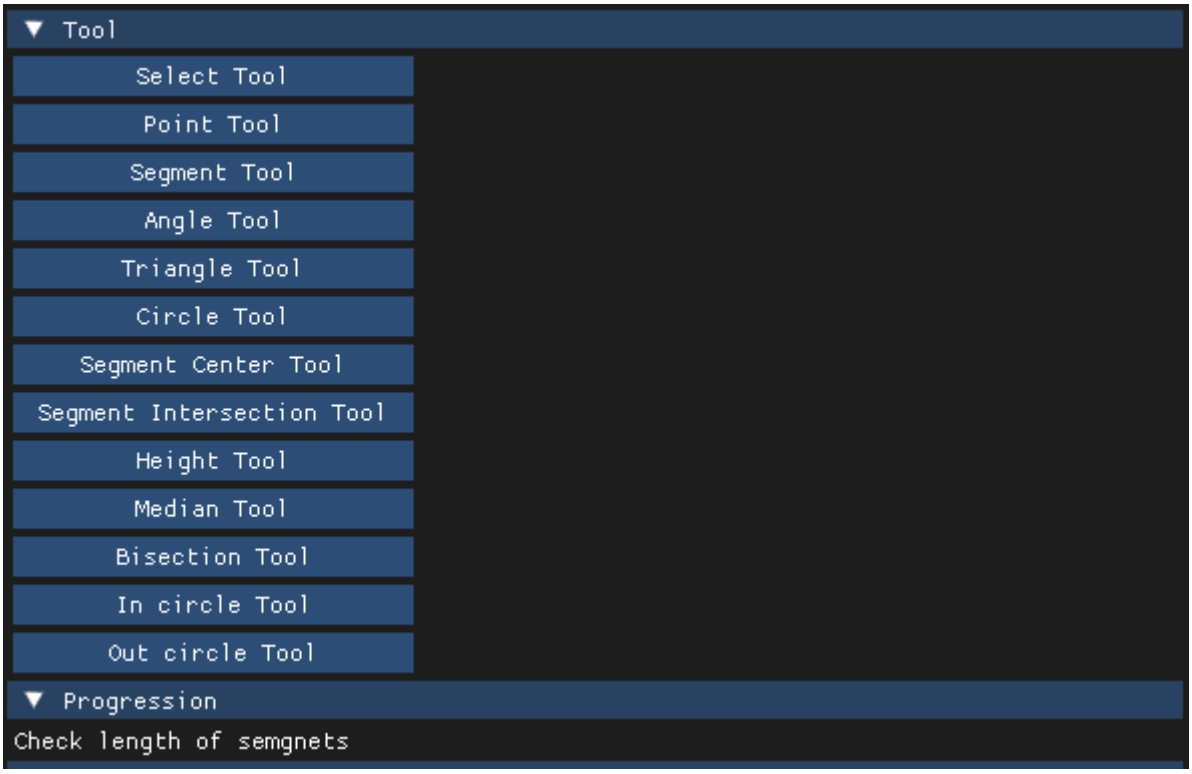
Інструмент для висот, бісектрис та медіан дозволяє відповідно малювати висоти, бісектриси, медіани і бісектрис і відрізняються хіба що вибором об’єктів.

Також є декілька додаткових інструментів, які дозволяють знаходити перетин двох відрізків та центр відрізка.

Додавати інструменти в програму дуже просто, усе що потрібно це створити клас нового інструмента та імплементувати відповідні методи його поведінки, після чого інструмент можна використовувати.

Наступною і немало важливою є частина користувацького інтерфейсу написаною за допомогою бібліотеки ImGUI. За допомогою користувацького інтерфейсу можна змінювати параметри об’єктів, а саме їх позиції та імена.

ImGUI можем змінювати будь які значення під час виконання програми за адресою, тому програма проходить циклом по усім об'єктам із менеджера даних та виписує інформацію про них (Додаток А).



(Рис. 13)

Крім того, користувацький інтерфейс дозволяє змінювати інструмент малювання за допомогою кнопок. А також показує прогрес побудови задачі, яка була обрана для виконання.
Тут ми переходимо до останньої частини програми, а саме це удосконалення системи для побудови планіметричних малюнків. Можливість додавати задачі для побудови планіметричного малюнку.

Для цього в програмі був створений спеціальний клас Reader. Який вміє зчитувати задачу написану на псевдомові. Варто зазначити, що інформація в

задачі може бути не повною. У роботі ми більше зосередились на задачах, які допомагають будувати планіметричні малюнки трикутників та кіл. Хоча нам і не вдалось зайти системний підхід до вирішення будь-якої задачі з не повною інформацією, ми прийшли висновку, що загалом найкращим способом вирішення цієї проблеми є перегляд великої кількості задач та модифікування існуючи системи враховуючи ці задачі. Оскільки на нашу думку не можливо створити загальний алгоритм для вирішення будь яких геометричних фігур при неповній інформації. Як приклад можна привести таку задачу. «Дано прямокутний трикутник один із кутів дорівнює 45 градусі. Прилеглий катет до цього кута дорівнює 8 см, знайдів гіпотенузу.». Таку задачу наша система може порахувати за відповідним алгоритмом.

1. Знайти кути, якщо це можливо (за допомогою відомих кутів і сторін за теоремою синусів чи косинусів).
2. Знайти сторони, якщо це можливо (за допомогою відомих кутів і сторін за теоремою синусів чи косинусів).
3. Порахувати периметр (якщо невідомий).
4. Порахувати площу (якщо невідома).
5. Порахувати висоти, медіани та бісектриси (якщо невідомі)
6. Порахувати центр і радіус вписаного та описаного кіл.
7. Порахувати точки перетину медіан, висот та бісектрис (якщо можливо)
8. Повторити усі кроки, якщо знайшли щось нове.

Був обраний саме такий підхід до вирішенню задач, оскільки не вдалось знайти алгоритм по вирішенню будь-якого трикутника із будь-якими значеннями. Під час виконання програма використовує такі формули, як

1. Теорема косинусів

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha$$
$$b^2 = a^2 + c^2 - 2ac \cdot \cos\beta$$
$$c^2 = a^2 + b^2 - 2ab \cdot \cos\gamma$$

(Рис. 14)

2. Теорема синусів

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

(Рис. 15)

3. Сума кутів трикутника,

$$\alpha + \beta + \gamma = 180^\circ$$

(Рис. 16)

4. Знаходження трикутника за трьома сторонами

$$\alpha = \arccos\frac{b^2 + c^2 - a^2}{2bc}, \quad \beta = \arccos\frac{a^2 + c^2 - b^2}{2ac}$$

(Рис. 17)

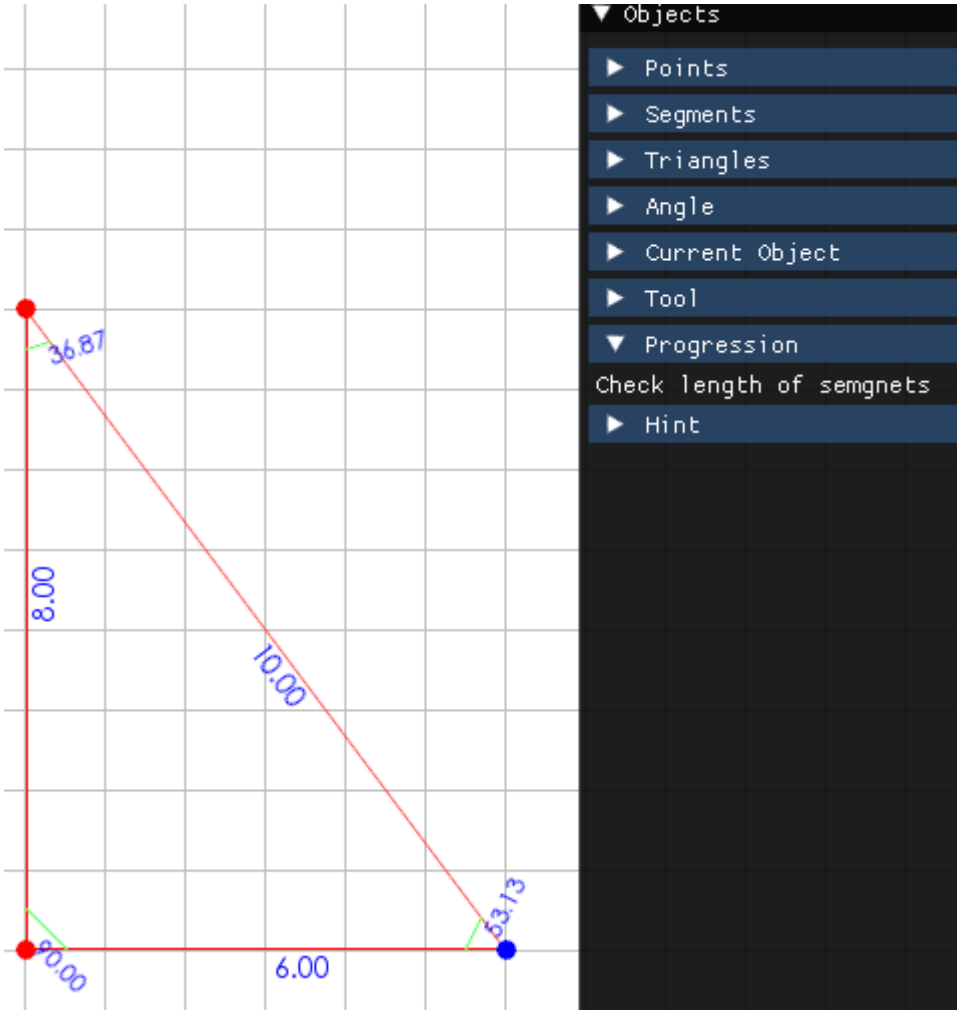
5. Дві сторони та кут між ними

$$c = \sqrt{a^2 + b^2 - 2ab\cos\gamma}.$$

(Рис. 17)

Якщо розглядати задачі із повною інформацією, для трикутників це означає знати усі сторони і кути. Такі завдання знають усю інформацію про трикутник, а тому може керувати да допомагат користувачу будувати планіметричні малюнки.

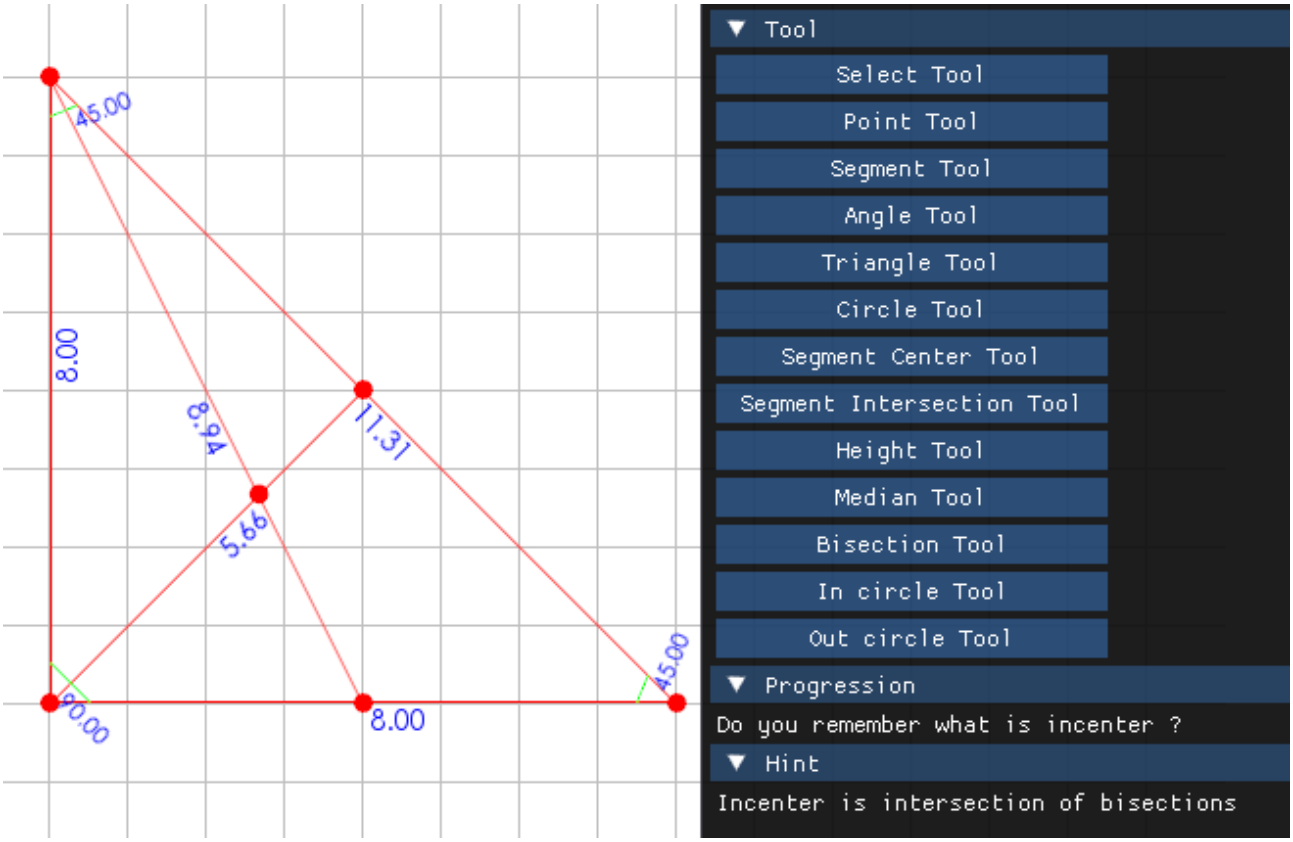
Приклад задачі - «Дано прямокутний трикутник, сторона 8 см, кут 45 градусів, знайти цент вписаного кола». Для виконання цього завдання необхідно правильно побудувати малюнок – трикутник. При не правильній побудові програма після кожної зміни стану буде перевіряти чи правильно ви побудували малюнок



(Рис. 18)

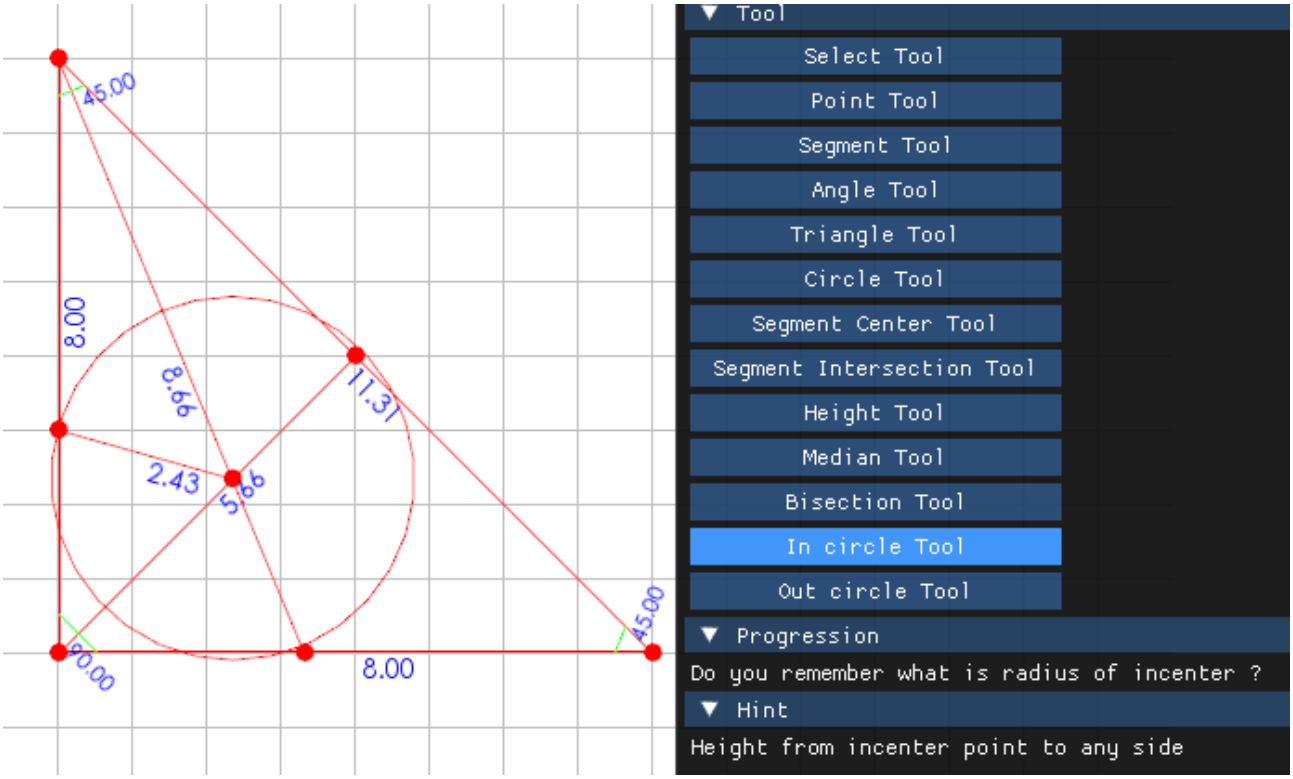
У графі прогресу написано, що потрібно перевірити довжину катетів. Якщо наприклад заборонити використовувати інструменту для вписаного кола, то для вирішення цієї задачі потрібно взяти інструмент для малювання бісектриси, провести ці бісектриси (хоча б дві) із кожного із кутів і знайти точку перетину цих бісектрис. Якщо користувач побудує перетин медіан, то програма скаже про те, що центр знайдено не правильно, а також з’явиться підказка для виконання

завдання. Підказки з’являються тоді коли помилки є поширеними. Знову ж таки нам не вдалось знайти якогось методу для виводу підказок для будь-яких побудов задач.



(Рис. 19)

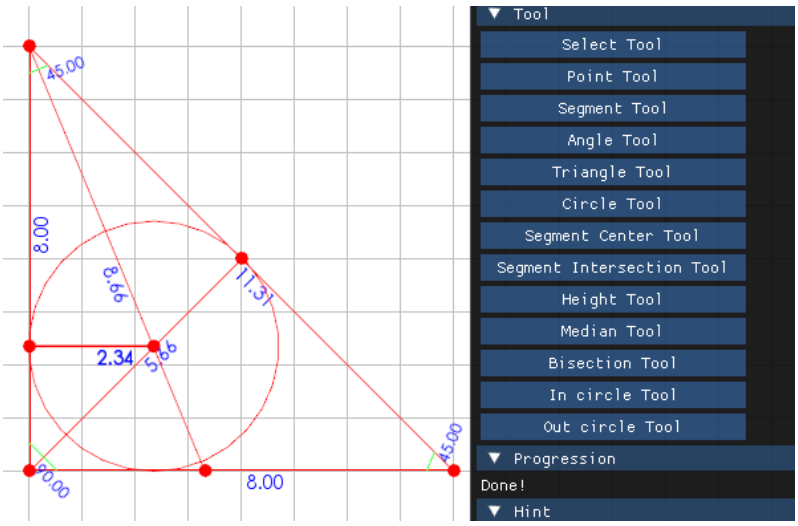
Підказка до задачі напише про те, що насправді центром вписаного кола у трикутник є точка перетину бісектрис. Для того щоб додавати такі підказки потрібно розглядати більше і більше задач та модифікувати на наповнювати код програми ними. Наступним кроком буде побудова самого вписаного кола, а його радіус – це довжина висоти проведеного із центра кола до будь-якої із сторін трикутника.



(Рис. 20)

Наступний крок виконано не правильно. Прогрес повідомить про це, а підказка нагадає, що радіус це висота проведена із центра до сторони.

Тобто потрібно провести висоту із центра кола до сторони. Якщо користувач це зробить і створить коло за допомогою інструмента для побудови кіл – задача буде виконана. В графі «прогрес» буде написано, що задача про побудову повністю виконана.



(Рис. 21)

Яким чином система зчитує та порівню малюнки до задачі. Зараз задачі описуються псевдомовою із синтаксисом, а на стороні програми створен класс який може зчитувати .txt файл та переводити його до програмних об'єктів.

Кожен тип об'єкта має свій трошки різний синтаксис. Для прикладу розглянемо трикутник

=
Triangle A B C
ab a 8
cab _ 90
abc _ 45
io _ true
=

Знак дорівнює повідомляє про початок об’єкта, після чого має бути ім’я типу об’єкта, в цьому випадку це трикутник після чого іде назви кожного із точок трикутника. Наступні рядки мають інформацію відповідно про відрізки його назву та довжину, кут (якщо назва _) то це неважливо так його значення, а також останній код io означає, що має бути побудоване вписане в цей трикутник коло. Після закриття елемента можна додавати будь-які інші – кути, точки, відрізки, кола, трикутники і т.д. і усі вони будуть валідуватись після цього (Додаток Б).

Зчитування відбувається із файла після чого із вхідної інформації будується трикутник в координатах 0 та 0 із вказаними або порахованими значеннями якщо задача не є повною. Характеристики порівнюються із характеристиками об’єктів, які були намальовані користувачем.

Така сама логіка використовується і для кіл, відрізків, кутів та точок. Проте для кожного із цих об’єктів алгоритм порівняння відрізняється. Очевидно, що об’єкти мають різну кількість характеристик, тощо.

Система є досить гнучкою, саме тому є багато різних шляхів до її розширення. Є можливість додати нові об'єкти для малювання або інструменти для малювання. Головною задачею для розширення є опанування великої кількості задач та модифікування вище описаних алгоритмів для вирішення наприклад задач в яких про трикутних відомо тільки інформація про висоту та кути між лініями проведеними у трикутнику.

Також можна спробувати створити систему, яка буде перетворювати людський текст задач в програмний опис задач, описаний вище. Після чого стане можливим занесення цілих підручників із геометрії до системи. Звісно без повного тестування усіх цих задач не обійтись, оскільки вони можуть бути дуже різними за своїм наповненням.

Можна також піти в сторону стереометрії, та добудувати 3D функціонал. Все що для цього потрібно, це налаштувати OpenGl та додати 3D об’єкти та інструкції як їх малювати.

Крім того можна додати систему акаунтів для учнів та вчителів, зробити для кожного учня щоденник та список завдань для самостійного опрацювання. Вирішення трикутника з неповною інформацією також наштовхнуло мене на те, що система може не тільки допомагати будувати малюнки, але також і вирішувати їх. Оскільки система і так знає як вирішувати велику кількість задач, проте це досить складно, оскільки учні можуть вивчати конкретні теми на які системі доведеться опиратись.

Висновок

В ході виконання роботи були проаналізовані системи для малювання планіметричних малюнків такі як “GeoGebra” та “Desmos”. Системи були порівняні та в процесі було помічено, що вони мають дуже обмежений функціонал для навчання. Тому був запропонований варіант удосконалення цих систем, а саме додання можливості задання власних задач, процес побудови малюнків до яких система буде контролювати та допомагати із побудовою правильного планіметричного малюнка.

Як результат була створена власна система для побудови таких малюнків із вище описаним удосконаленням. Яка може зчитувати задачу написану на псевдомові та перетворювати її на об’єкти з якими потім порівнюється малюнок користувача. Також система може обраховувати параметри деяких задач, якщо вони мають не повну інформацію про себе, для повноти картини та правильності побудови користувачем. Крім того система може надавати повідомлення про помилку побудови, а також в деяких випадках надавати підказки про побудову.

Було перевірено на великій кількості задач про трикутники із підручника для сьомих класів і на більшій частині програма працює коректно, а інші потребують специфічних змін, на які ми готові піти. Гарним варіантом є постійне оновлення та розширення системи для охоплення все більшої кількості задач.

В майбутньому можна запровадити систему у шкільний процес і використовувати на уроках геометрії для школярів та студентів.

Список використаних джерел

[1] Markus Hohenwarter. Komplexe Zahlen zum Anfassen. TI-Nachrichten, (2):16–17, 1998.

[2] Heinz Schumann. Schulgeometrisches Konstruieren mit dem Computer. Teubner und Metzler, Stuttgart, 1991.

[3] Heinz Schumann and David Green. New protocols for solving geometric calculation problems incorporating dynamic geometry and computer algebra software. International Journal of Mathematical Education in Science and Technology, 31(3):319–339, 2000.

[4] Hans Joachim Vollrath. Funktionales Denken. Journal für Mathematikdidaktik, 10:3–37, 1989.

[5] Rip Empson (May 25, 2011). "Build And Share Rich Educational Content With «Desmos»". TechCrunch.

[6] "Texas District Pilots «Desmos» as Alternative to Graphing Calculators (EdSurge News)". EdSurge. 2015-04-30. Retrieved 2016-03-16.

[7] "The «Desmos» Geometry Tool - Des-blog". blog.»Desmos».com. Retrieved 2019-06-21.

[8] "The OpenGL Graphics System: A Specification" (PDF). 4.0 (Core Profile). March 11, 2010.

[9] Stroustrup, Bjarne. "C++11 FAQ". stroustrup.com.

[10] Radich, Quinn (May 30, 2018). "Retained Mode Versus Immediate Mode". Win32 apps. Microsoft. Retrieved 21 December 2019.

Додаток А

(Приклад виводу інформації до ImGUI)

```
void IMGUITriangle(Triangle * p, int i = 0)
{
	std::string s = p->Name.empty() ? "" : p->Name.c_str();
	auto name = "Triangles" + std::to_string(i) + " " + s;

	if (ImGui::TreeNode(name.c_str()))
	{
		ImGui::BeginChild(name.c_str(), ImVec2(500, 200));
		{
			ImGui::InputText("Name segment ab", (char*)(p->ab->Name.c_str()), 64);
			ImGui::InputText("Name segment bc", (char*)(p->bc->Name.c_str()), 64);
			ImGui::InputText("Name segment ca", (char*)(p->ca->Name.c_str()), 64);

			ImGui::InputText("Name point 1", (char*)(p->a->Name.c_str()), 64);

			ImGui::InputFloat("pos A x", &((p->a)->x));
			ImGui::InputFloat("pos A y", &((p->a)->y));

			ImGui::InputText("Name point 2", (char*)(p->b->Name.c_str()), 64);

			ImGui::InputFloat("pos B x", &((p->b)->x));
			ImGui::InputFloat("pos B y", &((p->b)->y));

			ImGui::InputText("Name point 3", (char*)(p->c->Name.c_str()), 64);

			ImGui::InputFloat("pos C x", &((p->c)->x));
			ImGui::InputFloat("pos C y", &((p->a)->y));
		}
		ImGui::EndChild();

		ImGui::TreePop();
	}
}
```

Додаток Б

(Приклад зчитування для об’єкта «трикутник»)

```
if (res[0] == "ab")
{
	if (res[1] != "_")
		tri->ab.name = res[1];
	if (res[2] != "_")
		tri->ab.length = std::stof(res[2]);
	if (res[2] != "_")
		tri->ab.valid = true;
}
else if (res[0] == "bc")
{
	if (res[1] != "_")
		tri->bc.name = res[1];
	if (res[2] != "_")
		tri->bc.length = std::stof(res[2]);
	if (res[2] != "_")
		tri->bc.valid = true;
}
else if (res[0] == "ca")
{
	if (res[1] != "_")
		tri->ca.name = res[1];
	if (res[2] != "_")
		tri->ca.length = std::stof(res[2]);
	if (res[2] != "_")
		tri->ca.valid = true;
}
else if (res[0] == "abc")
{
	if (res[1] != "_")
		tri->angleABC.name = res[1];
	if (res[2] != "_")
		tri->angleABC.v = std::stof(res[2]);
	if (res[2] != "_")
		tri->angleABC.valid = true;
}
else if (res[0] == "bca")
{
	if (res[1] != "_")
		tri->angleBCA.name = res[1];
	if (res[2] != "_")
		tri->angleBCA.v = std::stof(res[2]);
	if (res[2] != "_")
		tri->angleBCA.valid = true;
}
else if (res[0] == "cab")
{
	if (res[1] != "_")
		tri->angleCAB.name = res[1];
	if (res[2] != "_")
		tri->angleCAB.v = std::stof(res[2]);
	if (res[2] != "_")
		tri->angleCAB.valid = true;
}
else if (res[0] == "io")
{
	if (res[1] != "_")
		tri->incenter.name = res[1];
	if (res[2] != "_")
		tri->incenter.shouldDraw = res[2] == "true" ? true : false;
                                                }
```