ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О.ГОНЧАРА

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА МАТЕМАТИЧНОЇ

КІБЕРНЕТИКИ

Лабораторна робота №3

з теми «Наслідування. Поліморфізм»

з курсу «ООП»

Виконав:

студент групи ПА-18-3

Куц Микита

Дніпро, 2020

**Постановка задачі**

Скласти об’єктно-орієнтовану програму на мові C++, яка в діалоговому режимі керує графічними об’єктами, що відображаються на екрані дисплею.

По натисненню клавіши F1 програма виводить на екран коротку підказку по усіх наявних командах/клавішах; наприклад: як створити об’єкт, як зрушити з місця, як перейти до «наступного» об’єкту тощо.

Програма повинна підтримувати такі загальні елементи поведінки графічних об’єктів:

1. Активізація/візуалізація графічного об’єкту за вибором користувача.
2. Переміщення зі слідом/без.
3. Відновлення початкового стану об’єкту.
4. Зміна кольору.
5. Зміна стану видимий/невидимий.
6. Агрегація, тобто утворення нових об’єктів з вже створених. При агрегації об’єктів забезпечити можливість агрегації агрегатів.

Програма повинна надати користувачеві можливість зберігати поточну конфігурацію програми у вказаний користувачем текстовий файл на диску і завантажувати поточну конфігурацію програми із вказаного користувачем текстового файлу на диску.

Варіанти графічних образів пропонується згенерувати самостійно на основі таких графічних примітивів: точка, відрізок, трикутник, прямокутник, коло, зірка. Наприклад: кільце; зірка, вписана в коло; коло, вписане в трикутник, сузір’я і та ін. Під час здавання роботи викладачеві слід очікувати на пропозицію додати до програми графічний об’єкт нової форми.

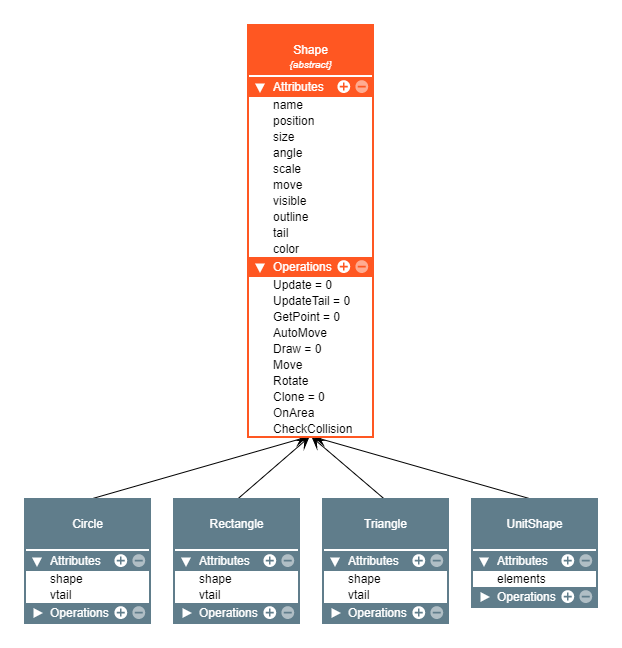
**Опис розв’язку**

Для реалізації графічної частини використовується бібліотека **SFML** (<https://www.sfml-dev.org/index.php>).

Через особливості бібліотеки кожен клас для відображення у вікні повинен мати посилання на необхідне вікно, тому кожне вікно та всі графічні елементи програми мають атрибут **std::shared\_ptr<sf::RenderWindow> window**.

Для всіх класів прописані сеттери та геттери, які не показані в діаграмах. Чисті віртуальні функції позначені « = 0».

**Реалізація фігур**



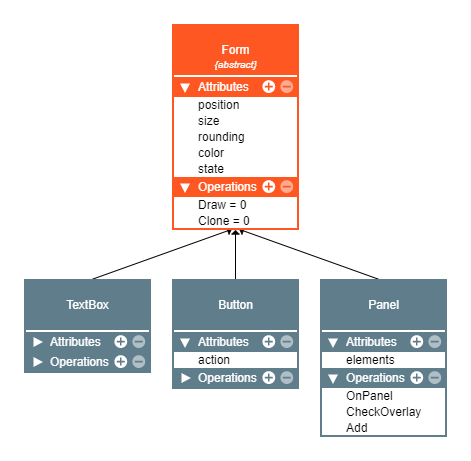
Через особливості **SFML** бібліотеки та наслідування в мові C++ кожен клас має свій атрибут **shape**, який відрізняється типом (**sf::CircleShape**, **sf::RectangleShape** й т. д.) для відображення кожної фігури.

Для відображення хвоста в кожної фігури є вектор, в який при зміщенні фігури записується вся інформація про її попередню позицію.

Агрегат фігур або **UnitShape** при об’явлені обчислює середнє значення кожного атрибуту для всіх фігур, які в нього входять. Надалі ж можна змінювати всі ці значення, які одночасно будуть змінюватись у всіх фігурах агрегату (в класі прописані сеттери, які змінюють значення для всіх фігур).

Якщо не чіпати програму 5 секунд, то вона переходить в режим очікування – всі фігури починають самостійно рухатись в довільному напрямку та с довільною швідкістю.

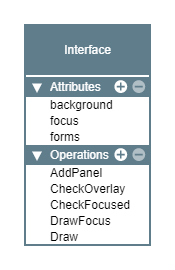
**Реалізація інтерфейсу**

****

Інтерфейс програми будується за подобою **Windows Forms**, але в дуже урізаному форматі (реалізовані лише **Button**, **TextBox**, **Panel**, які успадковані від класу **Form**).

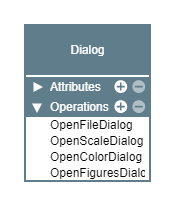
Клас **Buttom** має атрибут **action**, який має тип **enum Token**, що позволяє задавати дію на кожну кнопку.

Клас **Panel** має методи, що дозволяють перевірити, чи можна додати форму на панель.



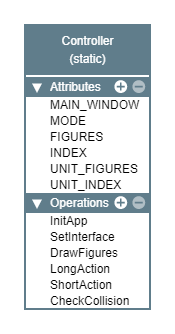
Саме вікно має екземпляр класу **Interface**, який виставляє фон, всі панелі в інтерфейсі та зберігає інформацію про фокус користувача.

Кожен тік програма перевіряє розташування курсора користувача, якщо він наводить його на якийсь елемент інтерфейсу, то цей елемент зберігається як фокус користувача. Користувач може натиснути лише на елемент з фокусом.



Також для вводу необхідних даних використовуються Dialog, які представлені у виді класу з методами OpenFileDialog, OpenColorDialog й т.д. (це окремі вікна, в які можна ввести всі необхідні дані)

**Реалізація головного вікна**

****

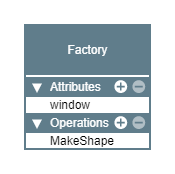
Клас **Controller** відповідає за роботу головного вікна.

Метод **InitApp** запускає головне вікно, **SetInterface** виставляє інтерфейс, **DrawFigures** малює фігури у вікні.

Методи **LongAction** та **ShortAction** відловлюють події, які можуть виконуватись продовж якогось часу або лише одноразово за один клик.

Метод **CheckCollision** перевіряє чи зіткнулась обрана фігура з іншими, якщо да, то вони мають стати середнього кольору.

**Патерн Factory**

****

В програмі також реалізован патерн **Factory** для побудови фігур для заданого вікна зі стандартними значеннями.

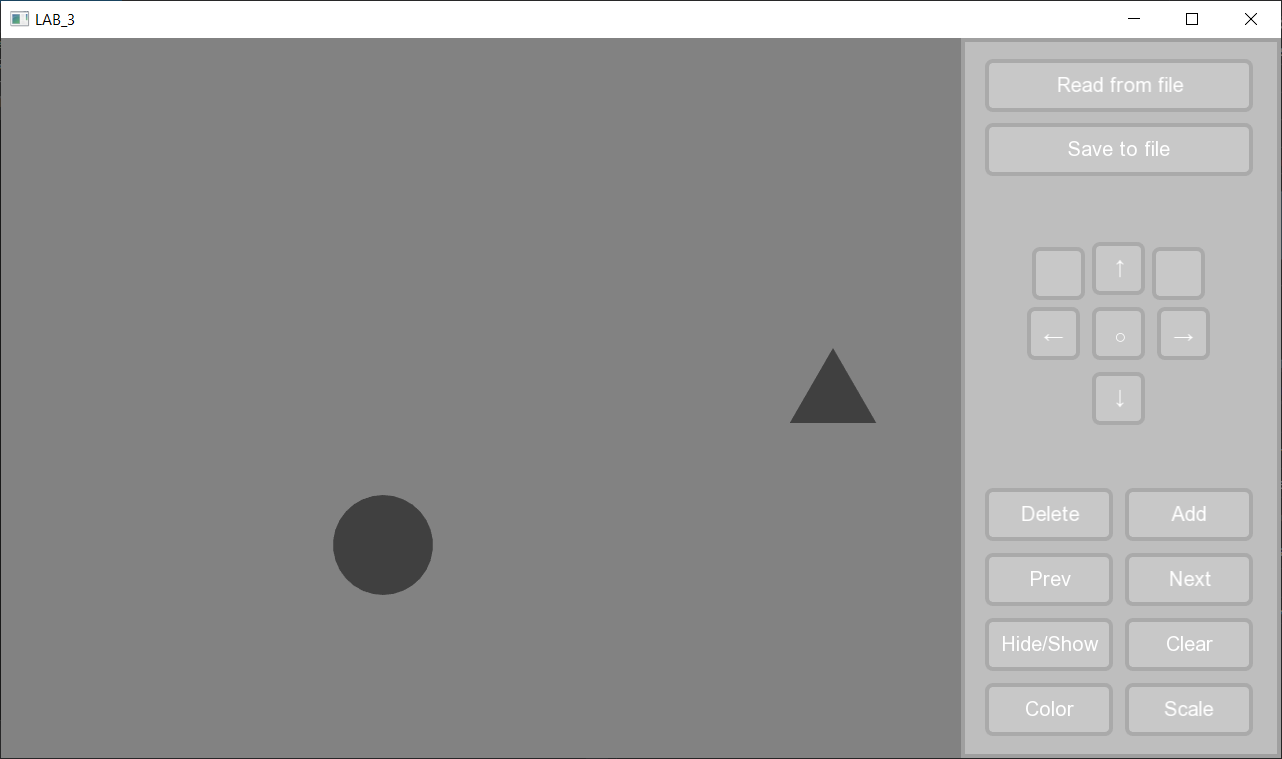
**Файл Settings**

Цей файл зберігає початкові структурні значення для програми (аналог файлу **Config**). Там можна виставити розмір вікна, режим (на весь екран, у вікні й т.д.), розміри, колір в інтерфейсі, значення при зміщенні фігур.

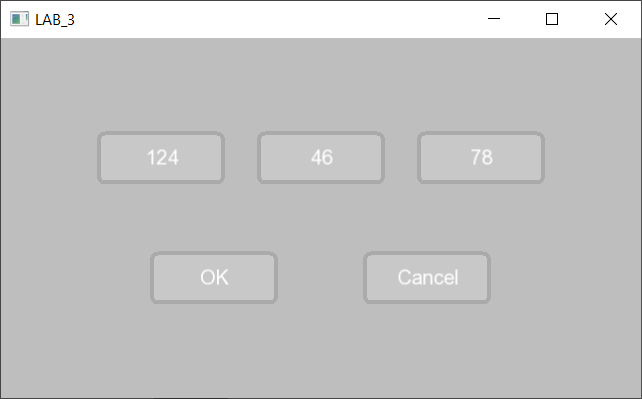
Також там приєднуються всі бібліотеки для роботи програми.

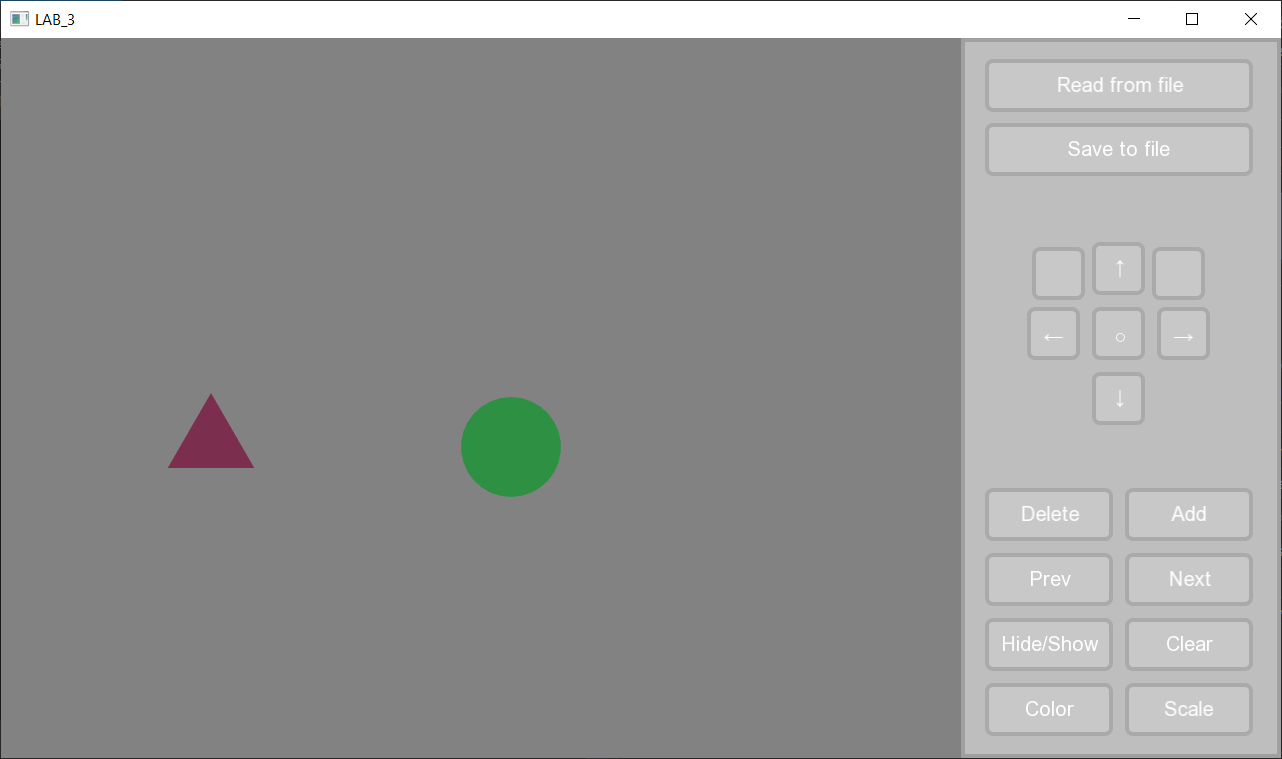
**Приклади роботи програми**

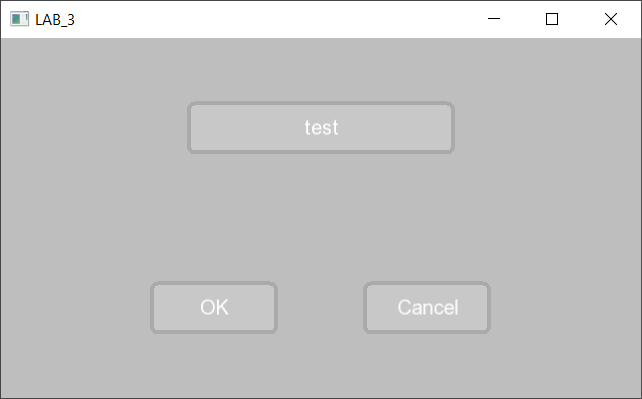




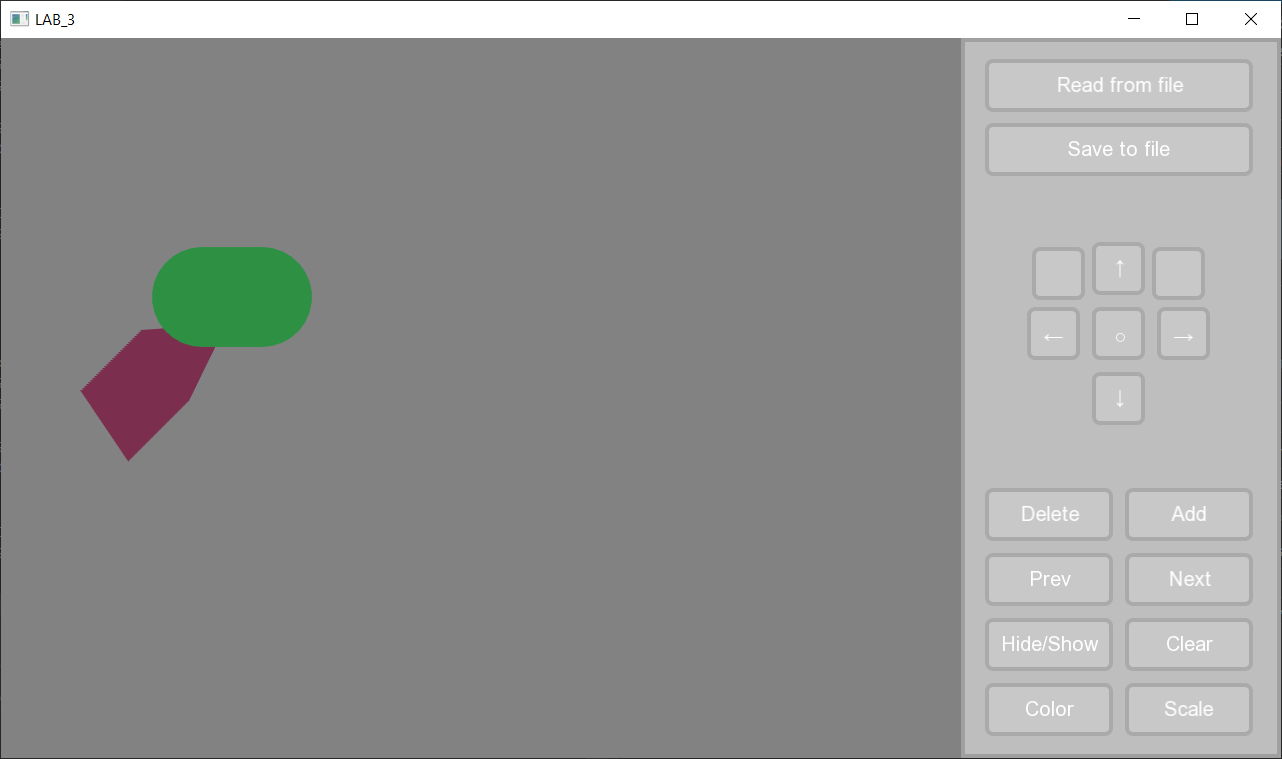
Мал.1-2 Додавання нових фігур



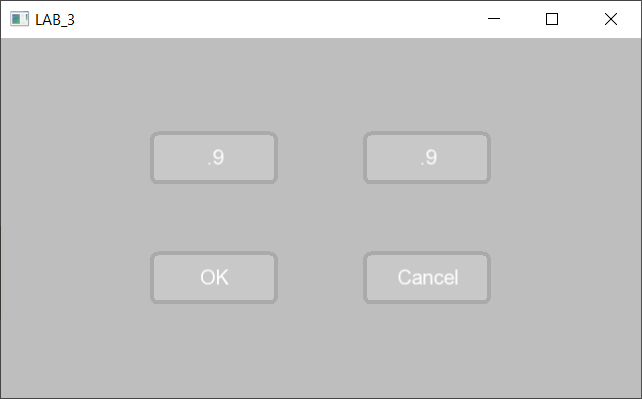
Мал.3-4 Вікно для вводу RGB кольору. Зміна кольору



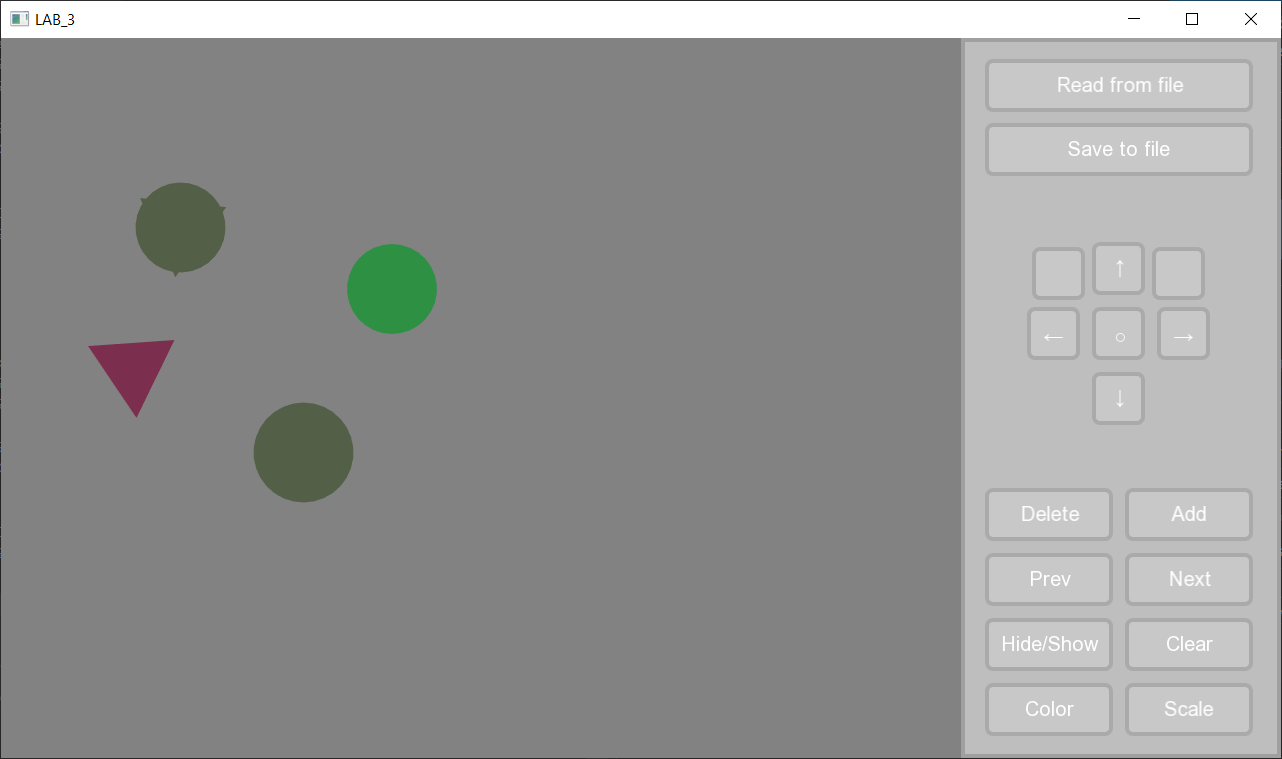
Мал.5 Вікно запису до файлу. Запис до файлу «test.txt»



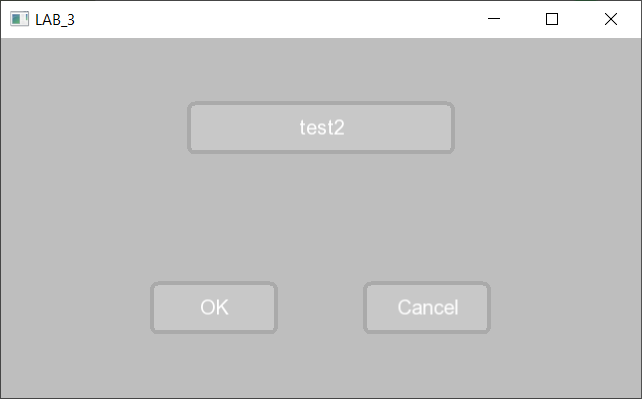
Мал.6 Приклад хвоста у фігури



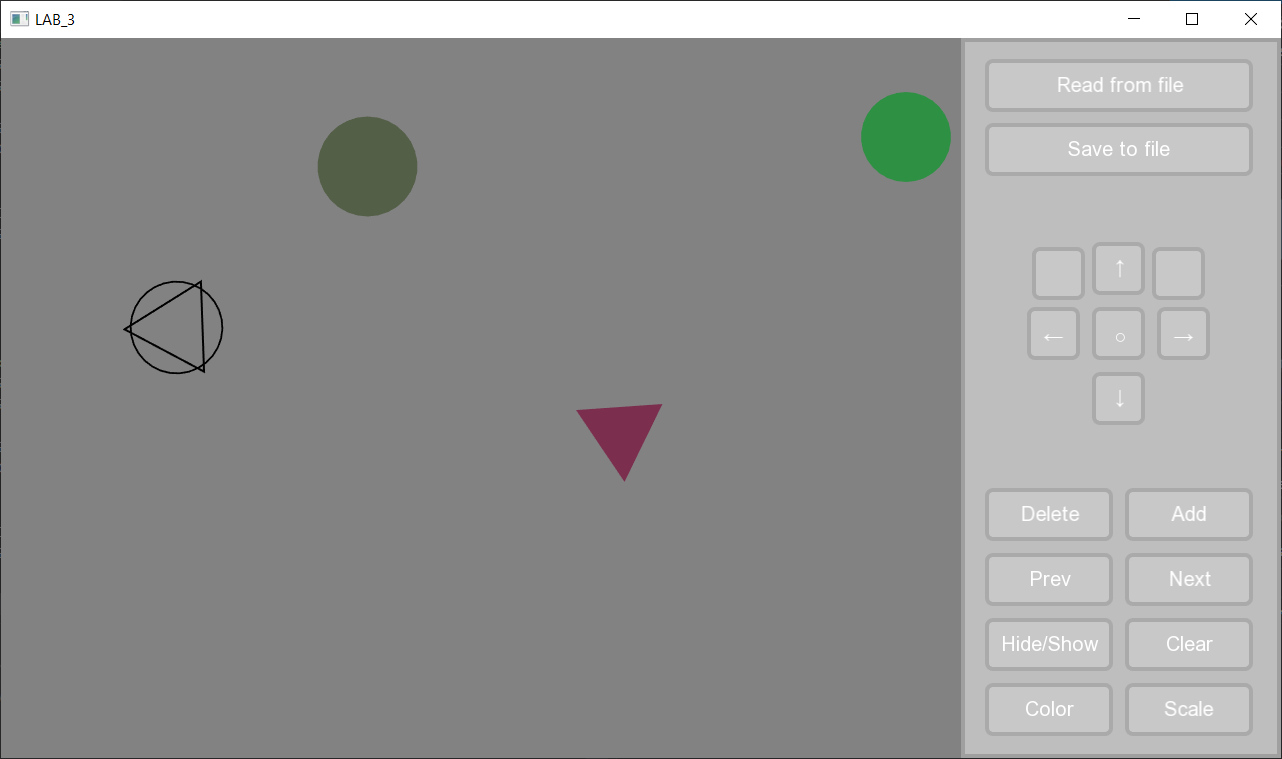
Мал.7 Вікно для ввода нового масштабу. Зміна масштабу



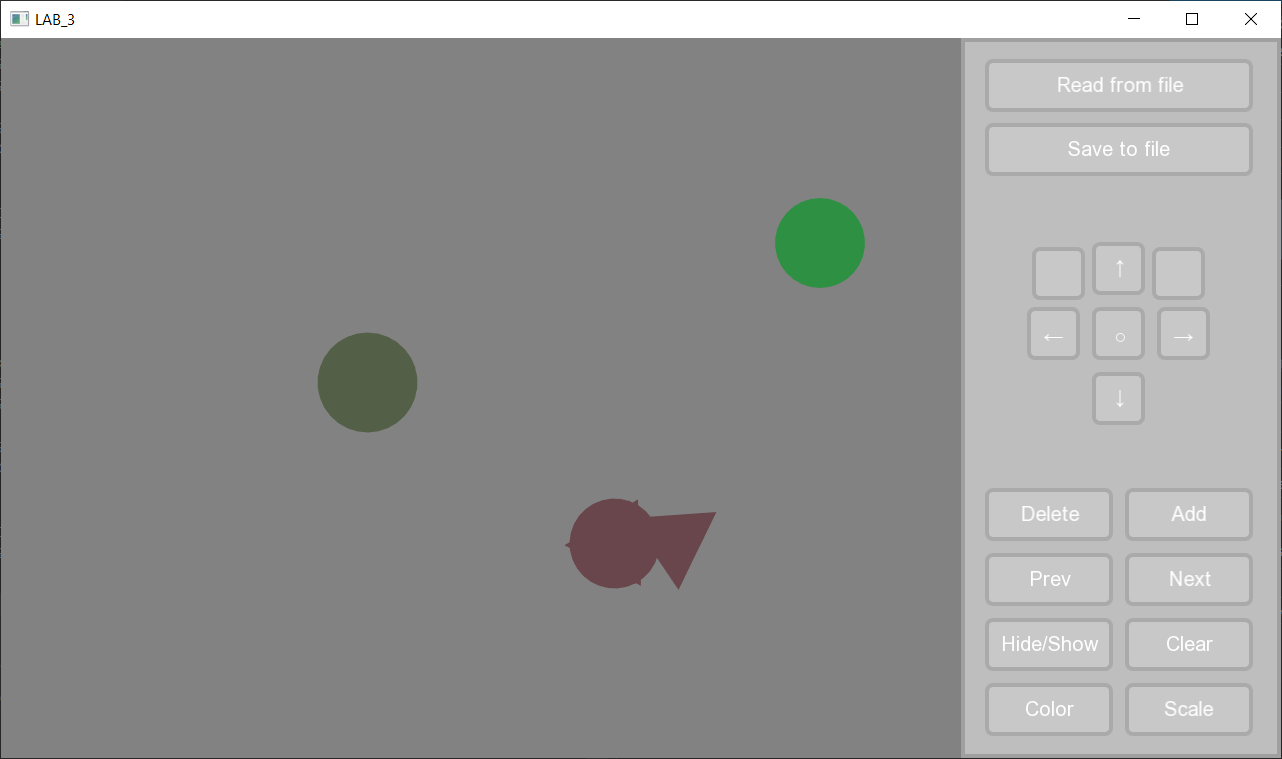
Мал.8 Додавання агрегатів



Мал.9 Запис до файлу «test2.txt»



Мал.10 Агрегат робимо невидимим



Мал.11 Зіткнення двох фігур (зміна кольору)

Зауваження: файли test.txt та test2.txt будуть причеплені разом зі звітом.