**ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ № 4**

**за курсом " Об’єктно-орієнтоване програмування"**

студента групи ПЗ-24у-1

Кондрачук Олександр Вадимович

кафедра математичного забезпечення ІПЗІТ, ДНУ ім. О. Гончара

2024 / 2025 н.р.

Тема: «Об’єктно-орієнтований дизайн. Рефакторинг коду»

1. Постановка задачі

Метою даної лабораторної роботи є проведення рефакторингу існуючої програми для роботи з геометричними фігурами з використанням патернів проектування:

1. Composite (Композит) – для групування графічних об’єктів у агреговані структури, що дозволяють виконувати операції над групою як над єдиним об’єктом.
2. Prototype (Прототип) – для створення нових об’єктів шляхом клонування вже існуючих прототипів, що дозволяє уникнути дублювання коду та забезпечує глибоке копіювання.
3. Singleton (Одинак) – для забезпечення існування лише одного контролера сцени, який відповідає за централізоване управління всіма графічними об’єктами.
4. Memento (Знімок) – для збереження стану наявних фігур (сцени) у файл та подальшого відновлення сцени з файлу.

Завдання:

1. Реалізувати патерн Composite через клас AggregateObject, який дозволяє об’єднувати окремі графічні об’єкти (типу GraphicObject) в єдину групу.
2. Використати патерн Prototype шляхом реалізації методу clone() та реєстрації прототипів у статичному репозиторії, що дозволяє створювати нові об’єкти через клонування.
3. Забезпечити єдиність контролера сцени за допомогою патерна Singleton (клас SceneController).
4. Додати можливість збереження та відновлення стану сцени за допомогою патерна Memento – реалізувати методи saveState()/loadState() в класах об’єктів і функції saveScene()/loadScene() у SceneController.
5. Опис реалізації

2.1. Патерн Composite

Composite дозволяє створити єдину структуру, яка складається з окремих об’єктів і груп об’єктів, забезпечуючи однаковий інтерфейс для роботи з ними.

У цьому проекті:

1. Кожний об’єкт має компонент Transform, що містить у собі Transform\* parent та std::vector<Transform\*> children, що дозволяє обходити усіх нащадків та викликати у них потрібні методи.
2. Це дозволяє, наприклад, перемістити або змінити колір групи фігур, викликаючи відповідний метод у об’єкті батьківського класу.

2.2. Патерн Prototype

Prototype – це порождаючий патерн, що дозволяє створювати нові об’єкти шляхом клонування вже існуючих.

У цьому проекті:

1. У класі Memento реалізовано метод save(), який створює дані, що характеризують об’єкт.
2. Також за допомогою класу PrototypeContainer, натиснувши клавішу F1, можна зберегти прототип та дати йому Id та натиснувши F2 можна створити копію збереженого об’єта.

2.3. Патерн Singleton

Singleton гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до цього екземпляра.

У цьому проекті:

1. Класи Input, Screen, Time, ObjectsContainer реалізовано як Singleton: вони мають публічний конструктор і статичну змінну instance, який повертає єдиний екземпляр.
2. Input відповідає за обробку натискання клавіш. Screen відповідаю за надання доступу до RenderWindow. Time відповідає за прорахування часу, який пройшов між кадрами. ObjectsContainer зберігає відомості про всі створені об’єкти.

2.4. Патерн Memento

Memento дозволяє зберігати та відновлювати внутрішній стан об’єкта, не порушуючи інкапсуляцію.

У цьому проекті:

1. У класі SceneSaverLoader, що має в собі методи save для зберігання всіх об’єктів сцени та запис їх у файл з усіма зв’язками між ними. Та метод load, що створює всі об’єкти, що були серіалізовані у файл.
2. Метод save викликається при натисканні клавіші L. Метод load викликається один раз при заході у програму.
3. Демонстрація роботи програми

Програма дозволяє користувачеві:

1. Зберігати прототипи за допомогою клавіші F1 та створювати їх за допомогою клавіші F2, дивись рисунок 3.1.

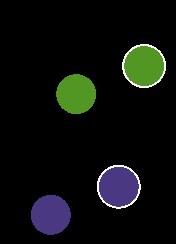


Рисунок 3.1 – Клонування об’єктів

1. Виділяти об’єкти за допомогою миші – при кліку на фігуру вона потрапляє до списку вибраних, дивись рисунок 3.2.

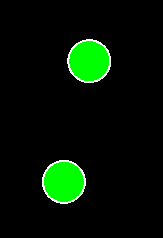


Рисунок 3.2 – Виділення об’єктів за допомогою миші

1. Змінювати властивості вибраних об’єктів (переміщення, зміна кольору, видимість) через відповідні клавіші, дивись рисунок 3.3.

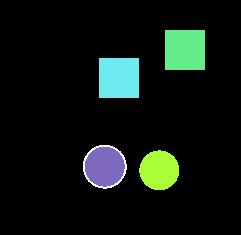


Рисунок 3.3 – Змінювання властивості вибраних об’єктів

1. Зберігати та відновлювати стан сцени:
   1. Клавіша S зберігає стан сцени у файл, дивись рисунок 3.4.

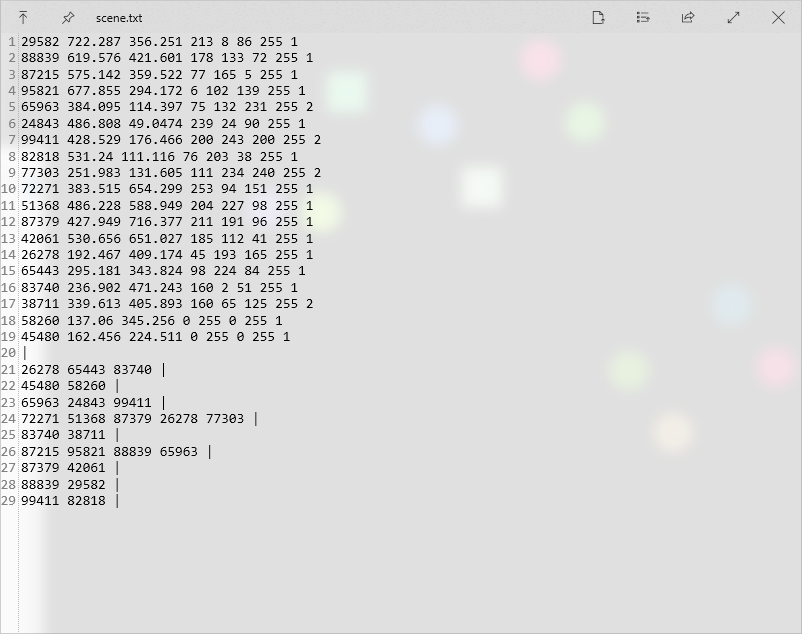


Рисунок 3.4 – Збережений файл фігур на сцені з їх розташуванням та іншими їхніми даними

1. Висновки

В результаті виконання лабораторної роботи було створено програму, яка демонструє високу гнучкість та розширюваність завдяки застосуванню сучасних об’єктно-орієнтованих рішень. Розроблена система дозволяє централізовано керувати графічними об’єктами, створювати їх копії, групувати у складні структури та зберігати/відновлювати стан сцени. Це значно полегшує підтримку та розширення функціональності застосунку, підтверджуючи переваги об’єктно-орієнтованого підходу у проєктуванні програмного забезпечення.