Міністерство освіти і науки України

Національний технічний університет "ХПІ"

кафедра "Інформатика та інтелектуальна власність"

**КУРСОВА РОБОТА**

з дисципліни «ООП»

**Тема**

"SFML гра"

Керівник курсового проєкту:

ст. викладач кафедри ІІВ Соболь М. О

Виконав:

студент групи КН-320Б Драбчак О. М

Харків 2021

Зміст

[Завдання 3](#_Toc89977207)

[Вступ 4](#_Toc89977208)

[1Аналіз предметної області 5](#_Toc89977209)

[1.1 Середовище програмування visual studio 5](#_Toc89977210)

[1.2 Графічна бібліотека SFML 6](#_Toc89977211)

[1.4 Інструкція як користуватися програмою 6](#_Toc89977212)

[2.Розробка програми 8](#_Toc89977213)

[2.1 Створення діалогового вікна 8](#_Toc89977214)

[2.2 Створення елементів діалогового вікна 10](#_Toc89977215)

[2.3 Опис використовуваних методів і подій 10](#_Toc89977216)

[2.4 Структура програми і основні функції 10](#_Toc89977217)

[3.Опис програмної реалізації 19](#_Toc89977218)

[3.1 Інтерфейс користувача 19](#_Toc89977219)

[3.2 Інструкція користувача 20](#_Toc89977220)

[Висновки 22](#_Toc89977221)

# Завдання

Опанувати набуті навички та знання з курсу ООП, опанувати фреймворк SFML та базові роботи з графікою і звуками, створивши просту в Visual Studio 2019.

**Мета:** мета, курсової роботи - систематизація, поглиблення і активне застосування знань з програмування, закріплення знань, отриманих в лекційному курсі, а також отримання навичок в роботі з SFML фреймворком.

# Вступ

Основне завдання в роботі це створити додаток в діалоговому вікні, при цьому в коді використовувати ООП принципи.

Об'єктно-орієнтоване програмування парадигма програмування, в якій основними концепціями є поняття об'єктів і класів.

У центрі ООП знаходиться поняття об'єкта. Об'єкт - це сутність, екземпляр класу, якій можна посилати повідомлення і яка може на них реагувати, використовуючи свої дані. Дані об'єкта приховані від решти програми. Приховування даних називається інкапсуляцією. Наявність інкапсуляції досить для об'єктно мови програмування, але ще не означає його об'єктної орієнтованості - для цього потрібна наявність успадкування. Але навіть наявність інкапсуляції і спадкування не робить мову програмування в повній мірі об'єктним з точки зору ООП. Основні переваги ООП проявляються тільки в тому випадку, коли в мові програмування реалізований поліморфізм, тобто можливість об'єктів з однаковою специфікацією мати різну реалізацію. Так само в ООП часто рекомендують виділяти ще одну важливу характеристику - **абстракцію**. Офіційно її не вносили в обов'язкові риси ООП, але списувати її з рахунків не варто.

**Абстрагування** - це спосіб виділити набір значущих характеристик об'єкта, виключаючи з розгляду не значимі Відповідно, **абстракція** - це набір всіх таких характеристик.

**Інкапсуляція** - це властивість системи, що дозволяє об'єднати дані і методи, що працюють з ними в класі, і приховати деталі реалізації від користувача.

**Спадкування** - це властивість системи, що дозволяє описати новий клас на основі вже існуючого з частково або повністю запозичує функціональністю. Клас, від якого виробляється спадкування, називається базовим, батьківським або суперкласом. Новий клас - нащадком, спадкоємцем або похідним класом

**Поліморфізм** - це властивість системи використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта.

# 1Аналіз предметної області

## 1.1 Середовище програмування visual studio

Microsoft Visual Studio — це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-програм, веб-сервісів та мобільних додатків.

Visual Studio дає змогу розробнику написати єдину програму для роботи на кількох платформах Windows, таких як мобільний, настільний і навіть експериментальний середовище Microsoft HoloLens. Він також забезпечує спосіб створення додатків, які взагалі не працюють на комп'ютерах Windows, але замість цього вони працюють на пристроях iOS або у веб-додатках у хмарі. 30 Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Вбудований відладчик працює як відладчик на рівні вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають програму для кодування, конструктор форм для побудови графічних інтерфейсів, веб-дизайнер, дизайнер класів та дизайнер схеми баз даних. Він приймає плагіни, які покращують функціональність практично на всіх рівнях, включаючи підтримку систем керування джерельними ресурсами (наприклад, Subversion та Gіt) та додавання нових наборів інструментів, таких як редактори та візуальні розробники для мов або наборів інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (наприклад, клієнт Team Foundation Server: Team Explorer). Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Вбудований відладчик працює як відладчик на рівні вихідного рівня, так і відладчик на рівні машини. Інші вбудовані інструменти включають програму для кодування, конструктор форм для побудови графічних інтерфейсів, веб-дизайнер, дизайнер класів та дизайнер схеми баз даних. Він приймає плагіни, які покращують функціональність практично на всіх рівнях, включаючи підтримку систем керування джерельними ресурсами (наприклад, Subversion та Gіt) та додавання нових наборів інструментів, таких як редактори та візуальні розробники для мов або наборів інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (наприклад, клієнт Team Foundation Server: Team Explorer). Visual Studio підтримує 36 різних мов програмування і дозволяє редакторові коду та відладчику підтримувати (в тій чи іншій мірі) майже будь-яку мову програмування, якщо існує певна мова-служба. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic. NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML та CSS.

## 1.2 Графічна бібліотека SFML

Бібліотека SFML надає простий інтерфейс для різних компонентів вашого комп'ютера, щоб полегшити розробку ігор і мультимедійних застосувань. Вона складається з п'яти модулів:

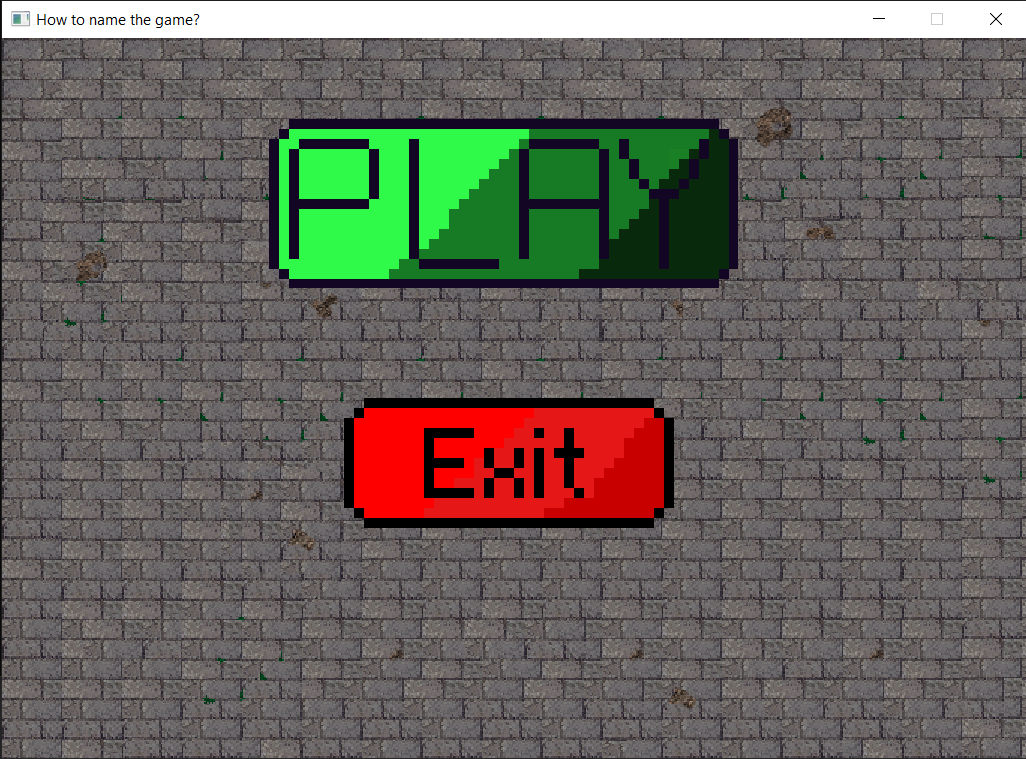
1. system – управління часом і потоками, він є обов'язковим, оскільки усі модулі залежать від нього;
2. window – управління вікнами і взаємодією з користувачем;
3. graphics – робить простим відображення графічних примітивів і зображень;
4. audio – надає інтерфейс для управління звуком;
5. network – для мережевих застосувань.

Використовуючи SFML, програма може бути скомпільоване і запущена на найбільш поширених платформах: Windows, Linux, Mac. SFML офіційно підтримує C і .NET. Також, завдяки своєму активному співтовариству, вона доступна на багатьох інших мовах, таких як Java, Ruby, Python, Go та ін.

## 1.4 Інструкція як користуватися програмою

Програма має стартове меню в якому в свою чергу присутні дві кнопки:

Play – запускає гру, Exit – виходить з гри.



UI гри має шкалу здоров’я та кількість очок.



Керування персонажем здійснюється в два способи:

1. Клавіші W,A,S,D (W - вперед, A - вліво, S – вправо, D - вниз).
2. Клавіші ↑, ↓, →, ← (↑ - вперед, ↓ - вниз, → - вправо, ← - вліво).

Головне завдання гри – зібрати якомога більше монеток. При зіткненні з ворогом здоров’я зменшується, при зіткненні з ресурсом здоров’я відновлюється на певну величину.

Гра закінчується коли в гравця закінчується здоров’я.

Після закінчення здоров’я з’являється відповідний напис про його закінчення та кнопка Restart, натиснувши яку гра перезапускається.



Щоби вийти з гри раніше ніж закінчиться здоров’я натисніть Esc клавішу або просто закрийте вікно.

# 2.Розробка програми

## 2.1 Створення діалогового вікна

Для малювання об'єктів, що надаються графічним модулем, необхідно використовувати спеціалізований клас вікна: RenderWindow. Цей клас є похідним від Window і успадковує всі його функції. Все, що ви дізналися про sf :: Window (створення, обробка подій, управління частотою кадрів, змішування з OpenGL і т.д.), можна застосувати і до RenderWindow.

Крім того, RenderWindow додає високо рівневі функції, щоб допомогти вам легко малювати. У цьому підручнику ми зосередимося на двох з цих функцій: clear і draw. Вони прості, як і випливає з їх назви: clear очищає все вікно вибраним кольором, а draw малює будь-який об'єкт, який ви йому передаєте.

На рисунку 2.1 показано, як виглядає типовий основний цикл з вікном рендеринга.

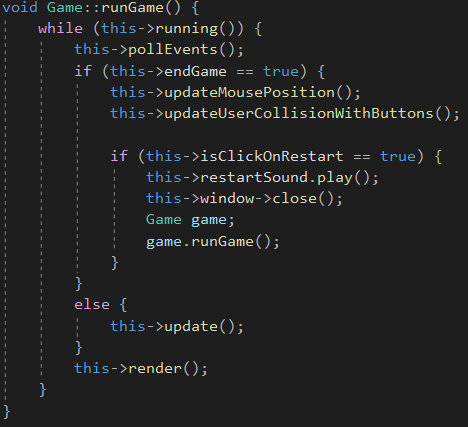


Рисунок 2.1 – Основний цикл з вікном рендеринга

Виклик методу render() є обов'язковим, інакше об’єкти не будуть відображатися в вікні. Render() в свою чергу має реалізацію

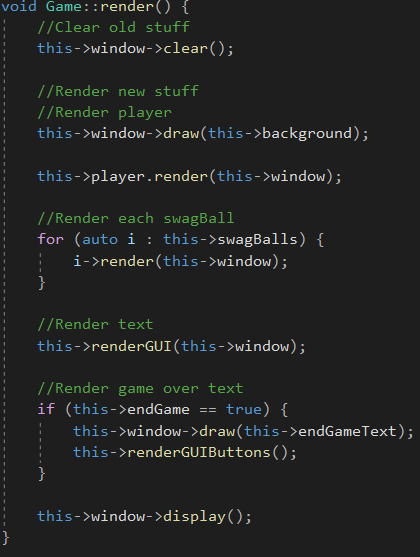


Рисунок 2.2 – Метод рендерингу

Виклик clear перед малюванням чого-небудь є обов'язковим, інакше вміст попередніх кадрів буде присутній за всім, що ви намалюєте. Єдиний виняток - коли ви закриваєте всі вікно тим, що малюєте, так що жоден піксель не буде відмалювали. В цьому випадку ви можете не викликати clear (хоча це не зробить помітного впливу на продуктивність).

Виклик display також обов'язковий, він бере те, що було намальовано з моменту останнього дзвінка display, і відображає це на вікні. Насправді, все малюється не прямо в вікно, а в прихований буфер. Цей буфер потім копіюється в вікно при виклику display - це називається подвійний буферизацією.

## 2.2 Створення елементів діалогового вікна

У SFML можливо зробити різні фігури і не тільки, для цього потребується створити об’єкт класу потрібної фігури. Щоб сформувати прямокутну кнопку у вікні, створюється об’єкт класу RectangleShape, а у конструктор з параметрами передається клас Vector2f(float X, float Y), де х ­­­­­­­­­– висота, у – ширина фігури. Далі використовуючи методи класу, задається колір фігурі setFillColor(), якому у параметри передається клас Color(), який приймає у параметри три цифри RGB та четвертий параметр Alfa так звану прозорість об’єкта. Після цього задається позиція фігури функцією setPosition(float x, float y), де х ­­­­­­­­­– ширина, у – висота. Після налаштування, фігура підмальовується на вікні, для цього викликається метод draw() об’єкту window, у параметри функції передається назва фігури.

## 2.3 Опис використовуваних методів і подій

У бібліотеці SFML є клас, який обробляє події, а саме натиснення клавішею мишки або клавіатури, кнопки джойстика и тд. Клас називається Event. За допомогою об'єкту цього класу, робиться взаємодія користувача з вікном і його елементами. Наприклад, при натисненні кнопки мишки, об'єкту класу Event вже зберігає про це інформацію. Таким чином можливо побудувати умову if () в якому буде порівняння об'єкту класу Event і потрібної дії, якщо вони рівні програма заходить в умову.

## 2.4 Структура програми і основні функції

Початок програми починається з функції main, де ми маємо об’єкт класу Menu який в своючергу виклакає цикл з головним меню, як показано на рис.

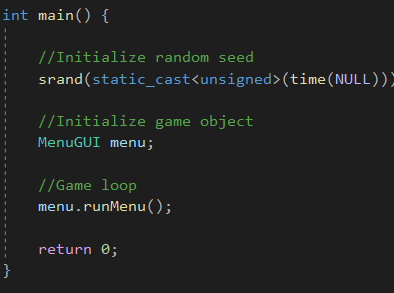


Рисунок 2.3 – main() функція

Далі ключовими є класи Game та MenuGUI. Клас MenuGUI надає користувачу дві можливості на вибір: Почати гру та вийти цілком з програми.

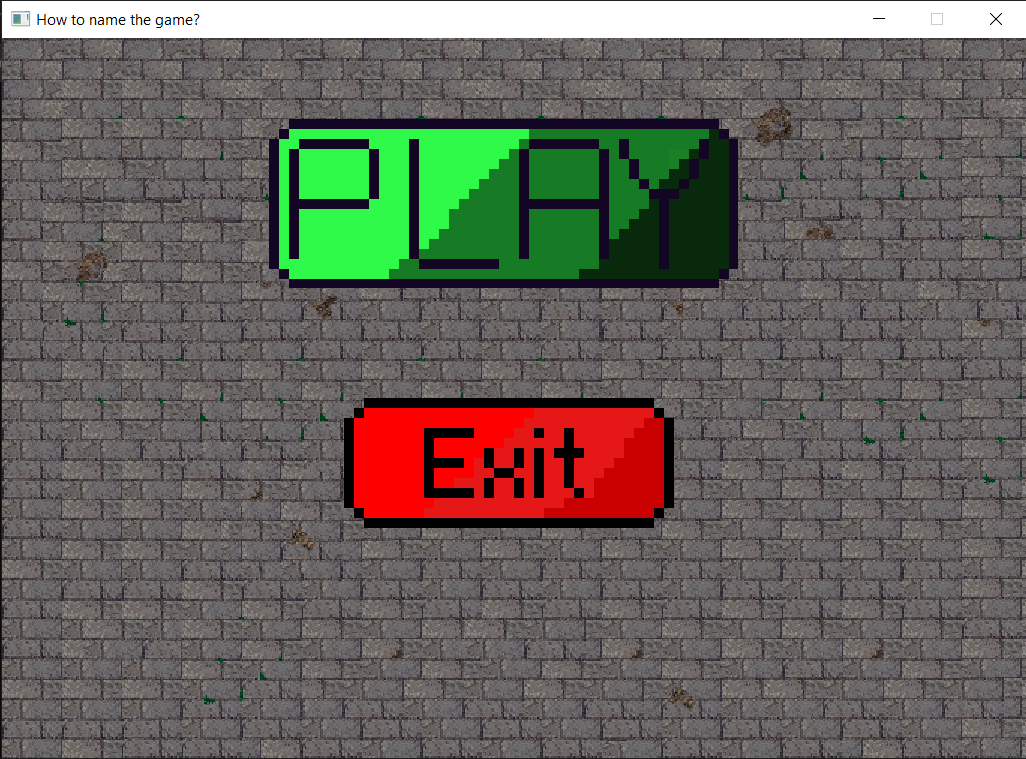


Рисунок 2.4 – ігрове меню

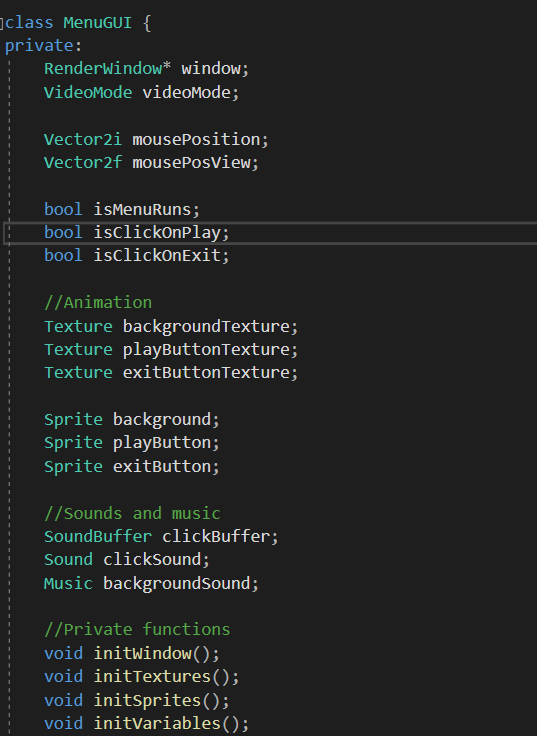


Рисунок 2.5 – оголошення класу меню

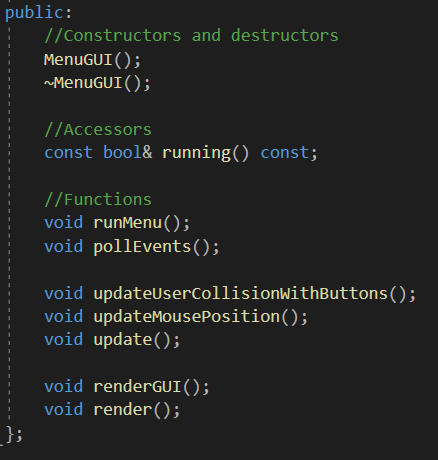


Рисунок 2.6 – клас меню (продовження)

Клас Game відповідає за всі події, основні механіки гри, їх обробку та відображення всіх подій у вікні. Даний клас був створений таким чином, щоб розділити процеси, як от: оновлення, зміна та перевірка даних при кожній ітерації, в коді цього класу цей метод має назву Update(), та відображення всіх об’єктів у методі Render(), які беруть візуальну і не тільки участь у програмі. В свою чергу дані методи об’єднують в собі методи update(), render() інших класів, об’єкти яких потребують зміни своїх властивостей, параметрів або відображення у вікні відповідно. Таким чином підвищується читабельність коду та в загалом логіки програми. Даний метод був взятий з книги «Game Programming Patterns» by Robert Nystrom.

Клас Game застосовує принцип композиції об’єктів та навіть агрегації: містить в собі об’єкти та вказівники на об’єкти та вектори вказівників на об’єкти інших класів.

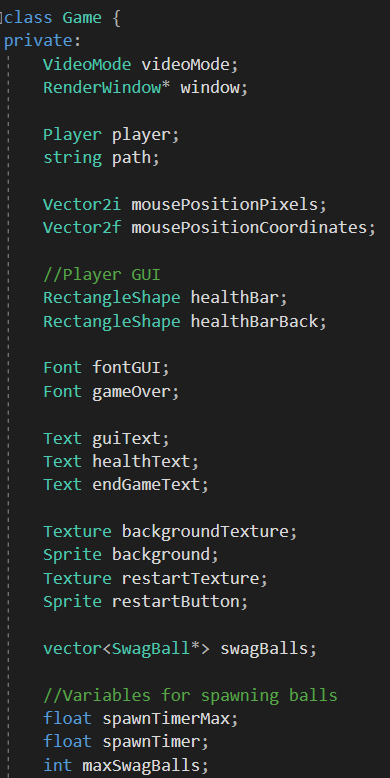


Рисунок 2.7 – оголошення класу гри

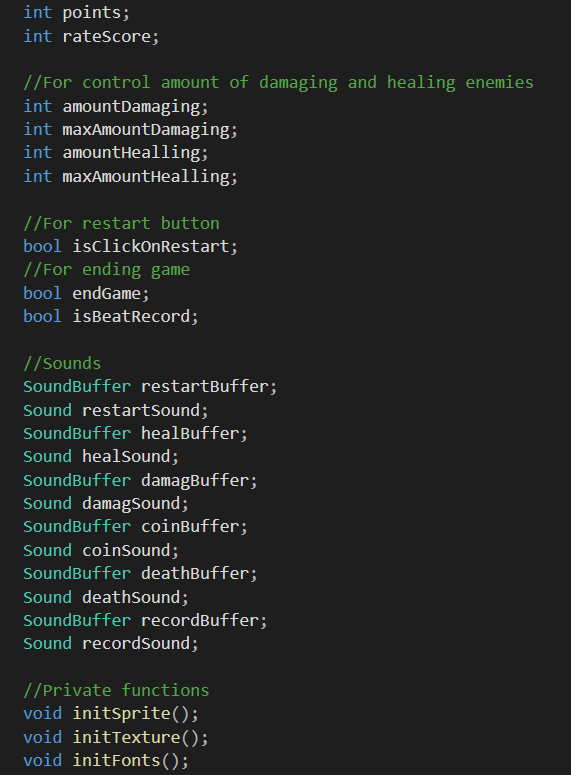


Рисунок 2.8 – оголошення класу гри (продовження)

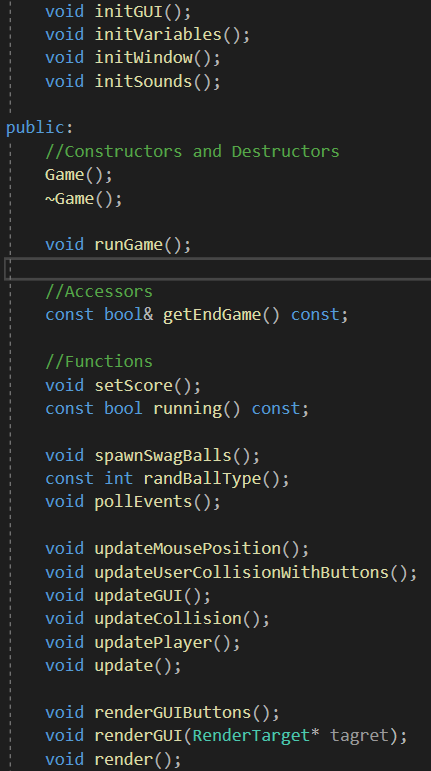


Рисунок 2.9 – оголошення класу гри (продовження)

Також, як видно з рис. даний містить приватні методи для ініціалізації певних об’єктів, змінних, текстур, елементів ігрового інтерфейсу, векторів з вказівниками на об’єкти і тощо. Такий підхід дає змогу написати наступний конструктор(див. Рисунок 2.10)

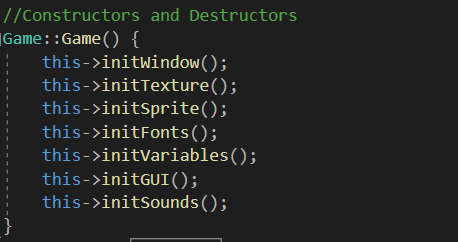


Рисунок 2.10 – визначення конструктору









Рисунки 2.11 – 2.14 – гра

# 3.Опис програмної реалізації

## 3.1 Інтерфейс користувача

Гравець має шкалу здоров’я яке при взаємодії з об’єктом ворог - зменшується а при взаємодії з об’єктом здоров’я – відновлюється. А також є шкала балів яка збільшуються при взаємодії з об’єктом бал.

Керування гравця здійснюється за допомогою клавіш (W, A, S, D) та

(↓, ←, →, ↑). Пересування гравця супроводжується анімацією руху.

Гру можна завершити двома способами:

1. Коли закінчиться здоров’я;
2. Натиснути клавішу Esc;
3. Закрити вікно з грою.

При закінченні здоров’я користувачу виводиться відповідне повідомлення та завершується гра, а також є змога заново почати гру натиснувши кнопку Restart.

## Інструкція користувача

Щоб усе працювало, потрібно підключити бібліотеку SFML. Для цього

завантажуються бібліотеки sfml-graphics 2.5.1 та sfml-audio 2.5.1.

Відкрийте Visual studio та створюється пустий проект, далі в пошуку пишемо

NuGet

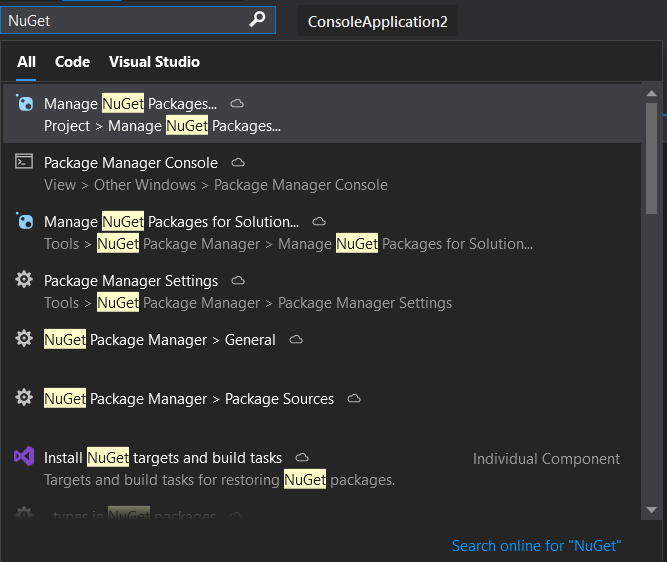


Рисунок 3.1 – встановлення бібліотек SFML

Далі пишемо в пошуку відкритого вікна спочатку sfml-graphics 2.5.1 потім sfml-audio 2.5.1. І встановлюємо по черзі.

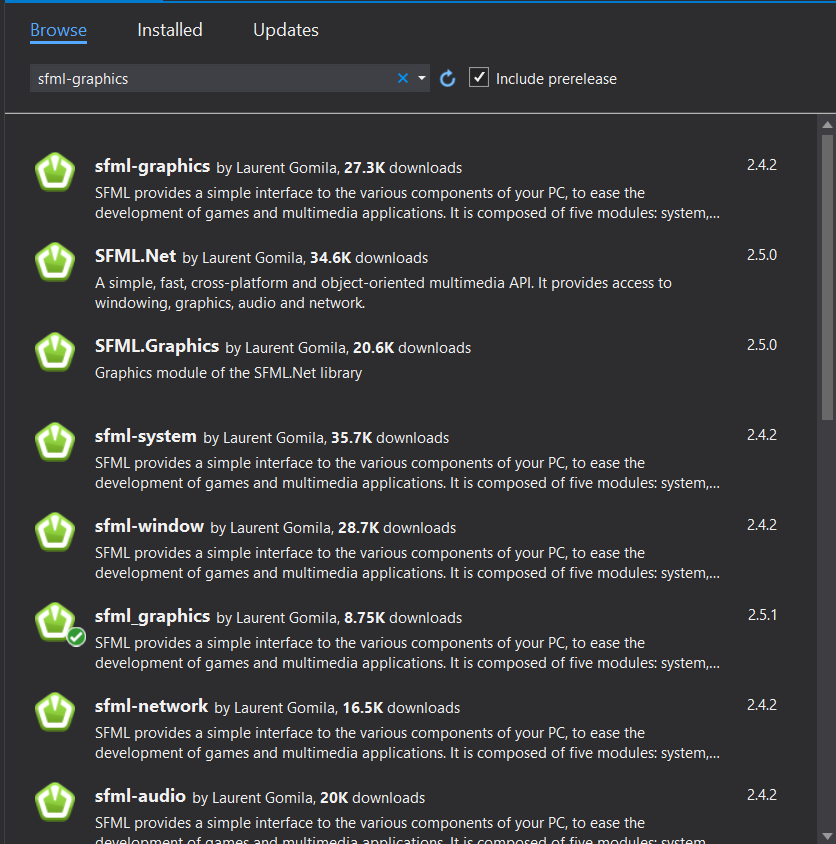


Рисунок 3.1 – встановлення бібліотек

Слідуйте інструкції:

Правою кнопкою миші по проекту → Properties→General→Platform Toolset→

→Visual Studio 2017(v141).

Далі Configuration Properties→ General→ Window SDK Version→ 10.0.19041.0→ Apply→Ok

І можна користуватися SFML Framework.

У самому проекті підключаються бібліотеки #include<SFML/Graphics.hpp>

#include<SFML/System.hpp>

#include<SFML/Window.hpp>

#include<SFML/Audio.hpp>

#include<SFML/Network.hpp>, та його простір using namespace sf;

# Висновки

У цій роботі були зроблена гра. В ході розробки гри була освоєна бібліотека SFML і закріплені базові знання щодо її використання. Так само закріплені навички програмування, які були отримані під час навчання і написання лабораторних робіт. У курсової були використані класи, а так само контейнери. Ще в проекті є можливість спостерігати ООП принципи на практиці. У грі є можливість додавання нових рівнів з найменшим зміною коду. Так само були помічені недоробки, в плані, тільки один тип монстрів, монстри не вміють ухилятися від атак гравця.

За допомогою практичної частини курсового проектування були закріплені теоретичні знання, що ми освоїли за курс ООП.