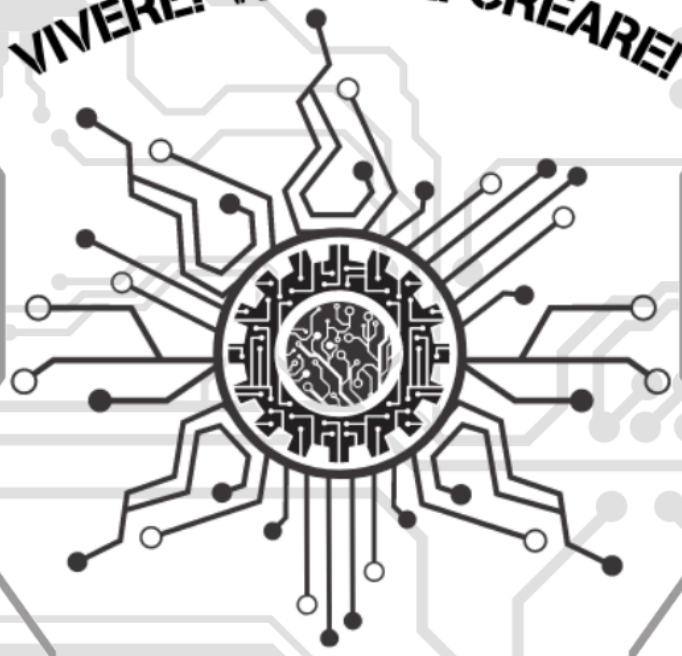
VIVERE! VINCERE! CREARE!



VladosBro

SKY DEFENDER

Зміст

Опис проекту

Небесні простори над Тихим океаном були потривоженні незліченими ескадрильями ворожих літаків. Небо потребує захисту, солдате! Сьогодні ви станете захисником повітряного простору і спробуєте себе в ролі пілота-винищувача. Сміливіше! сідайте за штурвал і вирушайте на боротьбу з небезпечними ворогами, які зазіхають на мир і спокій людства. Щасти, Солдате!

Sky Defender—це гра-аркада, в якій гравцеві надана можливість взяти під свій контроль літак. За допомогою літака, гравцеві необхідно знищувати ворожі літальні апарати. Протягом гри, необхідно звертати увагу на погодні умови, а саме велику хмарність, яка може зашкодити вчасно виявити ворога, який прямує на таран.
У правій стороні ігрового інтерфейсу наведені датчики, які допоможуть переглянути технічний стан літака і попередять про небезпеку.

ПРИМІТКА: це мій старий проект, коли я вивчав ООП C++, і вчився працювати з класами і наслідуванням

Демонстраційні матеріали

Примітки щодо реалізаії

Проблеми, що виникли у ході розробки

Програма писалася і розроблялася доволі давно, і я не пам’ятаю всіх проблем, з якими прийшлося зіткнутися у ході розробки.

Проте ось перелік деяких із проблем:

- Проблема руху літака відносно океану

  Опис: Через те, що геймплей гри передбачає симуляцію польоту літака над океаном, необхідно було зробити ілюзію «польоту» - тобто океан мав повільно зміщатися відносно літака. Тому створити звичайний map, який містив би коди спрайтів, з яких формувалася б карта, не вийш ло.

  Вирішення: Для цього було створене суцільне полотно океану (Game_powered_by_SFML_1 \images\ocean_map_2048x4096.png), текстура якого була «зациклина» (тобто нижня частина полотна—це продовжен ня верхньої). Програма, під час гри, сфокусовується на частині зображення полотна і повільно опускається, і при досягнені певної точки, фокус переміщається на початок полотна, що і забезпечує плавний і безкінечний рух океану. Інші елементи (рамка, інтерфейс кабіни, датчики), реалізо вані за допомогою map (map.h)

- Проблема очищення пам’яті при знищені об’єктів

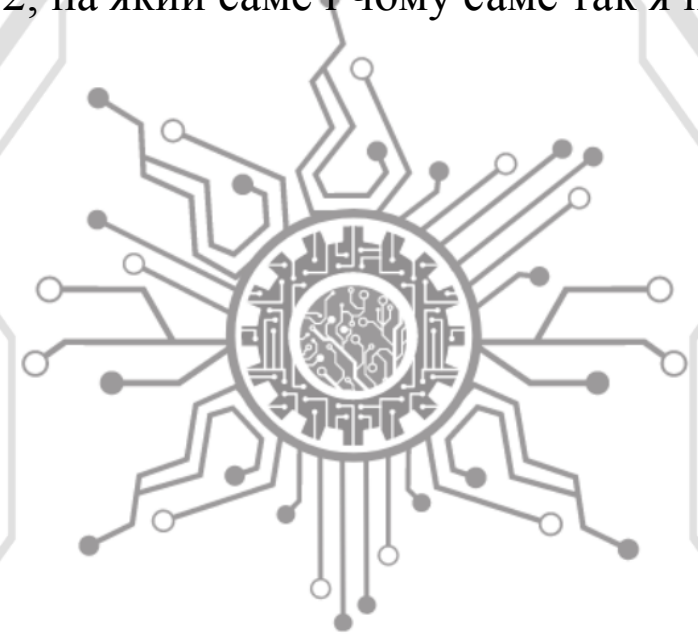
  Опис: Під час розробки, виникла проблема очищення пам’яті від об’єктів, які вже не використовуються (ворожі літаки, хмари, кулі). Під час очищення, програма видавала runtime помилку. Швидше за все, проблема полягала лише у тому, що я не мав до того моменту великої практики з дина мічним виділенням пам’яті, через це і виникали дані проблеми.

  Вирішення: проблему я вирішив частково. Достеменно я не пам’ятаю точного рішення, але здається я сформував статич ний масив вківників на об’єкти класів ворожих літаків і хмар, виділив на них пам’ять, і у момент, коли дані об’єкти виходять за межі ігрового екрану, вони переміщаються на верх поза екран, і потім зверху знову летять униз.

Ворожі літаки, у разі їх знищення, переміщаються на верх поза екран і знову летять униз, а на місці смерті відіграється анімація смерті (вибух).
Всім об’єктам (літаки та хмари), після переродження змінюється місце появи і швидкість за допомогою псевдови падкових чисел (rand()), що забезпечує різноманітну, динамічну гру.
Щодо куль, то тут при кожному пострілі, генерується новий об’єкт і виділяється динамічна пам’ять, яка ніяк не звільняється :) Проте виділення пам’яті на кожен об’єкт не значні, тому витік пам’яті майже непомітний.

-Проблема руху по діагоналі

Опис: під час написання логіки переміщення головного героя, була виявлена проблема переміщення по діагоналі, а саме рух був швидшим, аніж при русі по осям
Вирішення: це не стало вельми великою проблемою, через те що це все лише сума векторів. Враховуючм той факт, що швидкість по осям однакова, виходить, що вектор швидкості по діагоналі - це діагональ квадрата, і вона рівна добутку √2 на одну із сторін квадрата (вектор швидкості по осям). Тому вектор швидкості по діагоналі був розділений на коефіцієнт (но не на √2, на який саме і чому саме так я не пригадаю).



Особливості проекту

Сам проект передбачав за собою вивчення і закріплення здобутих знань і навичок при застосуванні концепції об’єктно-орієнтованого програмування мовою C++, тому як таких особливостей, не має.
Проте можна виділити знання, які я здобув чи повторив, під час створення цієї програми, а саме:
- робота з класами
- наслідування класів
- перевантаження операторів
- використання віртуальних функцій і поліморфізму
- інкапсуляція класів
- робота з динамічно виділеною пам’яттю
- робота з STL (використання vector, list)
- робота із сторонніми бібліотеками (SFML)

Технічний опис проекту

Компіляція репозиторія

Для того, щоб зкомпілювати проект, необхідно вказати компілятору шлях до бібліотек SFML.

Підключення бібліотек у Visual Studio 2017

УВАГА: за замовчуванням, проект вже містить необхідні бібліотеки у папці SFML, проте, у разі, якщо бібліотеки були якимось чином втрачені, виконуємо поану інсрукцію по встановленню
бібліотеки SFML за посиланням (Дивись пункт «Інше»)

УВАГА: більшість налаштувань, які описані нижче, вже задіяні у проекті, необхідно лише у певних місцях оновити шляхи до бібліотек.
Відкриваємо проект у Visual Studio.
На верхній панелі

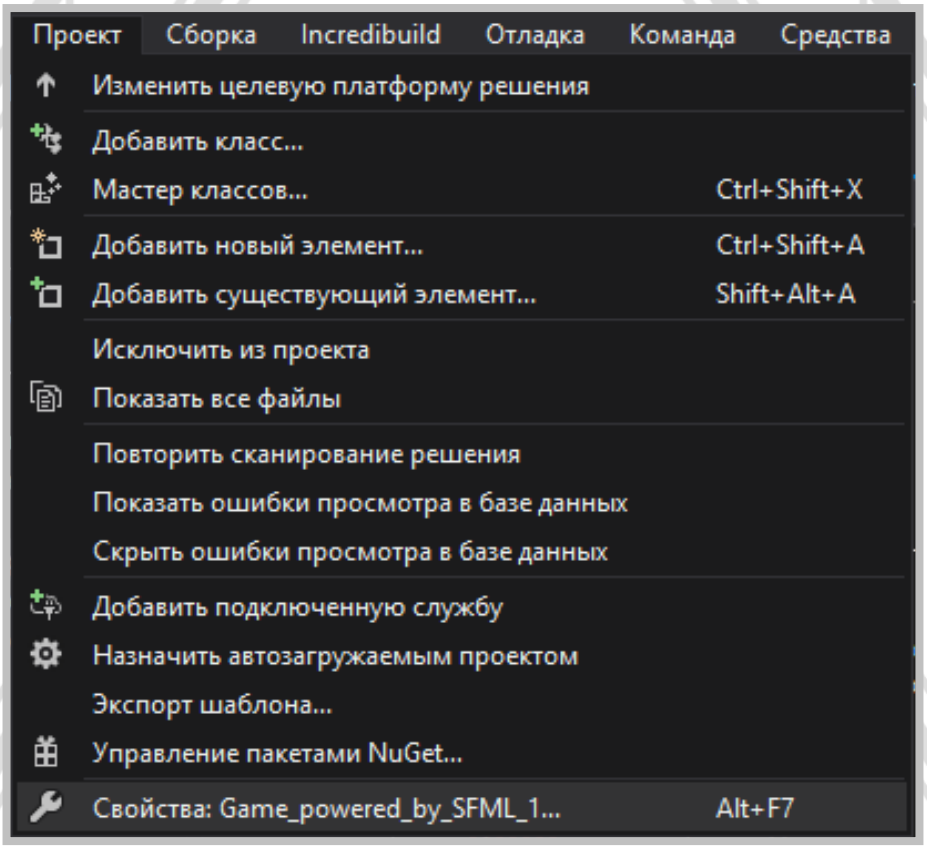


Рис. 1 Відкриття властивостей проекту

У вікні, що відкрилося, відкриваємо вкладку «Свойства конфигурации», у списку шукаємо «C\C++». Відкриваємо вкладку і натискаємо на «Общие». У правій панельці зверху знаходимо «Дополнительные каталоги включаемых

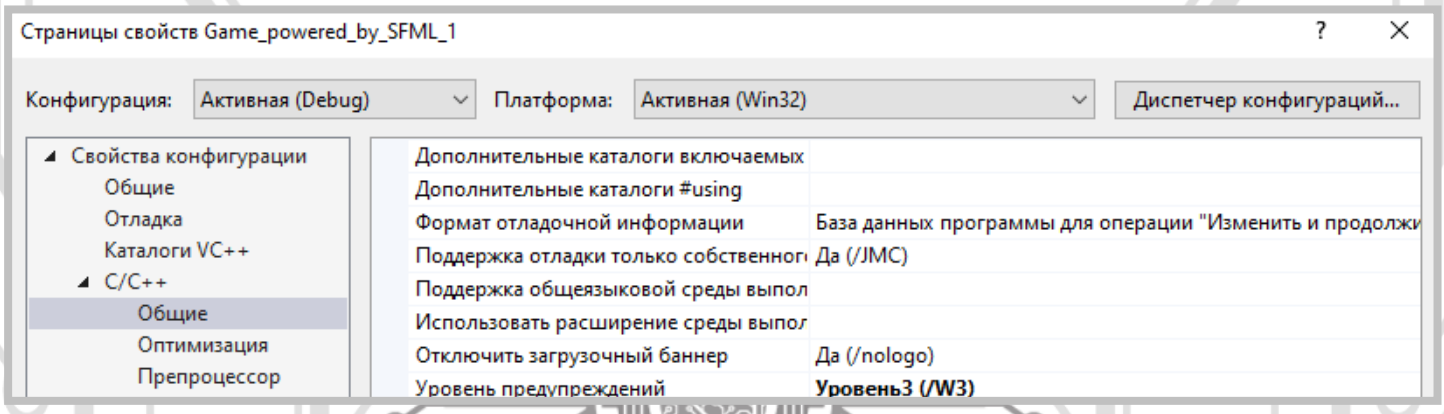


Рис. 2 Вказання шляху до папки з бібліотеками SFML

Тепер необхідно вказати/оновити шлях до папки include у папці з бібліотеками. Наприклад, мій шлях виглядає наступним чином: C:\Users\Glados\source\repos\Game_powered_by_SFML_1 \SFML\include
Після цього переходимо до вкладки «Препроцесор» і в «Определение препроцессора» добавляємо SFML_DYNAMIC; (Рис. 3)

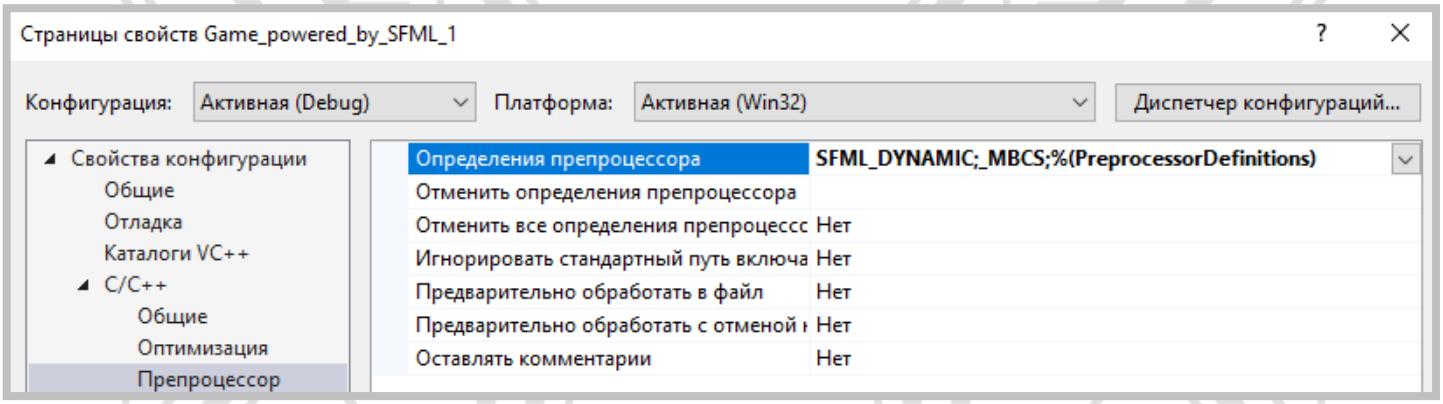


Рис. 3 Налаштування препроцесора

Переходимо у вкладку «Компоновщик», відкриваємо «Общие» і вводимо шлях до бібліотек у папці з даними SFML (x32\lib) Наприклад, мій шлях виглядає наступним чином: C:\Users\Glados\source\repos\Game_powered_by_SFML_1 \SFML\x32\lib

Проводимо оновлення конфігурацій компоговщика (Рис. 4):

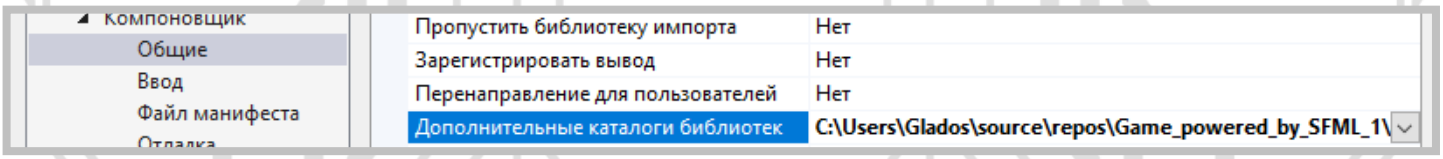


Рис. 4 Вказання шляху до папки з додатковими файлами SFML

У тепер натискаємо на вкладку «Ввод» і в полі «Дополнительные зависимости» прописуємо sfml-graphics-d.lib;sfml-window- (Рис. 5)

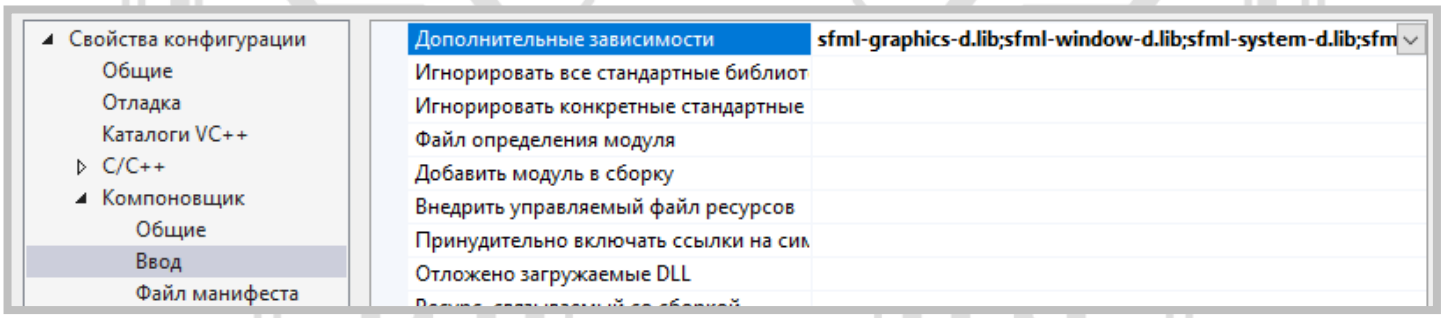


Рис. 5 Вказання додаткових залежностей

Все! Тепер можна зкомпілювати проект.

Встановлення

Проект не вимагає встановлення. Для запуску необхідно лише запустити exe-файл з Debug. Виконавчий файл має бути разом із іншими папками і файлами з Debug.
Для розповсюдження програми поза репозиторієм, необхідно, щоб файл-exe з файлами і папками з Debug були в окремій папці.

Налаштування

Проект не вимагає ніяких окремих налаштувань.

Керівництво по експлуатації

Головне меню

Після запуску програми, відкриється вікно з головним меню (Рис. 6)

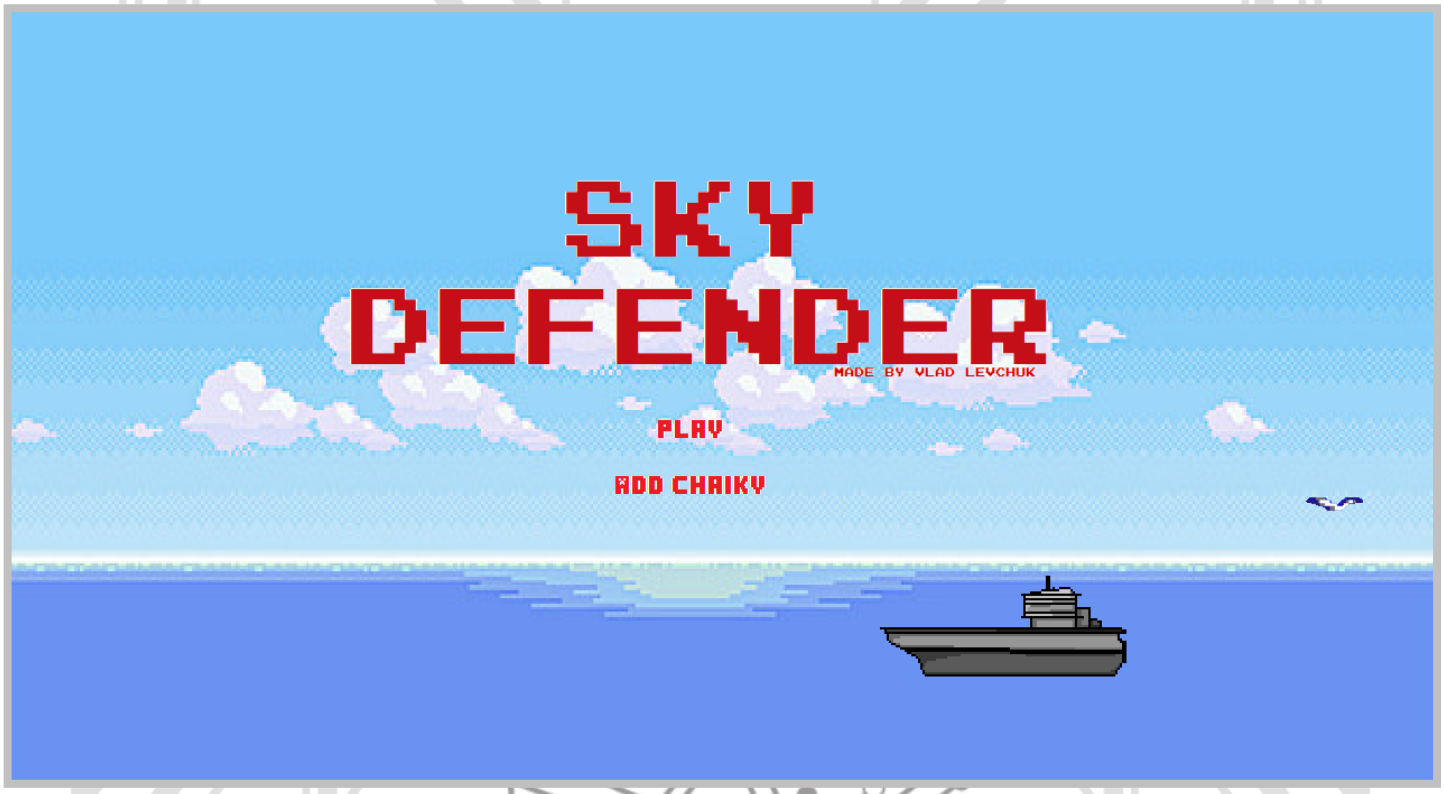


Рис. 6 Головний екран гри

На головному екрані присутні анімовані елементи у вигляді авіаносця, який пливає ліворуч-праворуч, і двох чайок, які рухаються
довільно, змінюючи напрямок при зустрічі з краями вікна
Головне меню складається з двох кнопок: «Play» та «Add chaiky» (Адд чайку).
При натиснені на кнопку «Play», негайно розпочнеться гра.
При натиснені кнопки «Add chaiky», на головному екрані появиться велика кількість чайок, які будуть довільно літати по екрану.
УВАГА: дане добавлення чайок виконується за допомогою динамічного виділення пам’яті, і пам’ять ніяк не звільняється під

Ігрове поле та гемплей

Ігрове поле представлено на рисунку 7

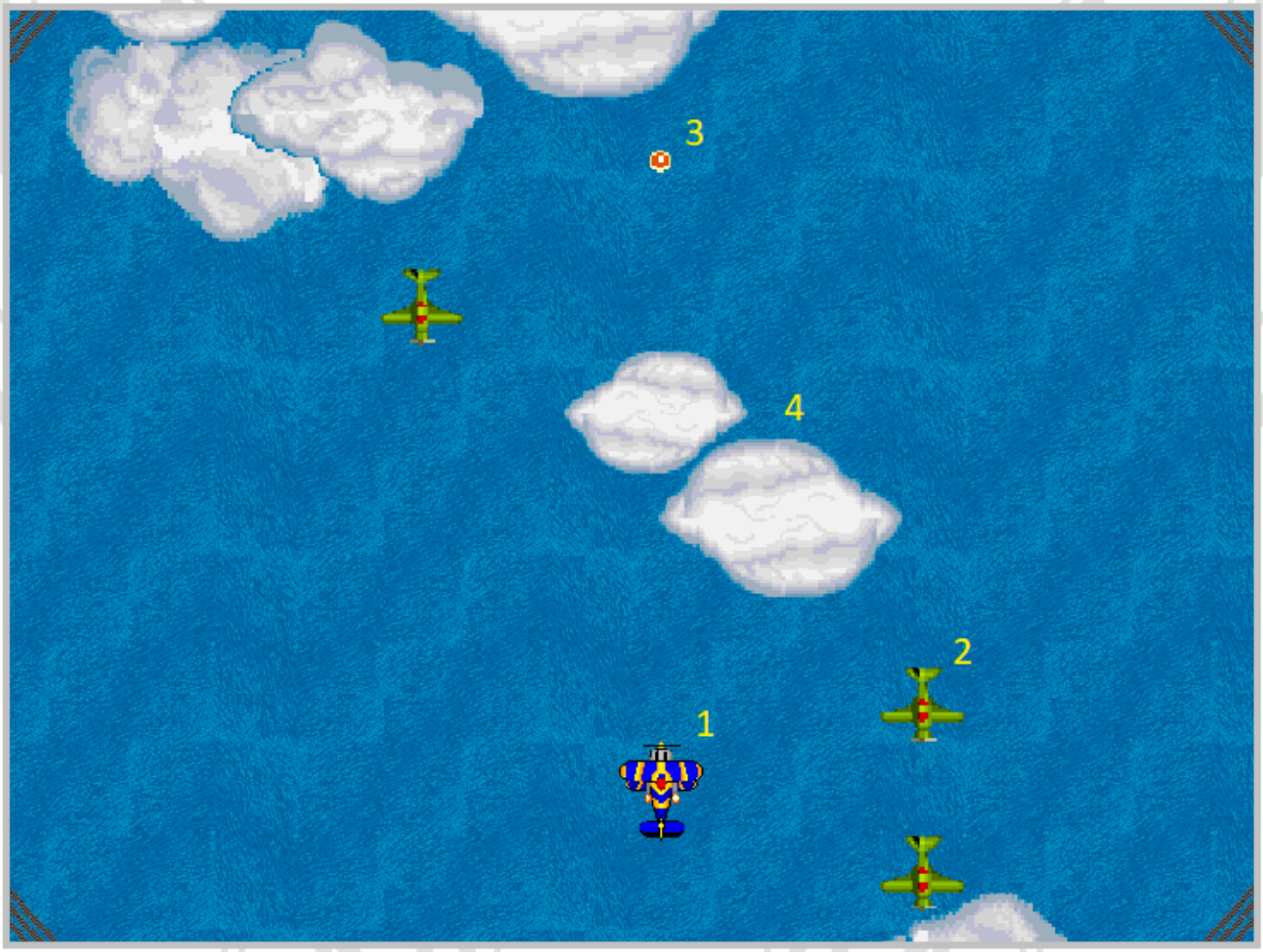


Рис. 7 Ігрове поле

Гравець керує літаком, який летить посеред океану. Під час гри на ігровому екрані можуть зустрітися певні об’єкти, а саме (рис. 7):
1 - Головний герой
2 - ворожі літаки
3 - кулі (постріли головного героя)
4 - хмари

Під час польоту, головний герой може пролітати під хмарами без шкоди для себе, але так само можуть пролітати і ворожі літаки, тому необхідно бути обачним.

Гравець має змогу знищувати ворожі літаки, здійснюючи постріли, але так само може бути знищеним, після 10 зіткнень з ворожими літаками (рис. 8)

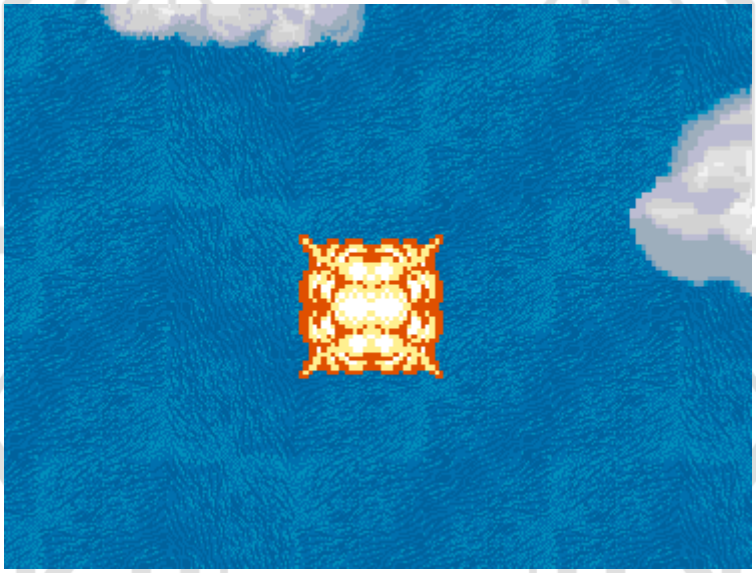


Рис. 8 Момент знищення ворога\гравця

У разі, якщо гравець буде знищеним, програється невелика музична композиція, яка сигналізуватиме про кінець гри, і відкриється вінальний екран, який виведе повідомлення про кінець гри і час ігрової сесії (рис. 9)

Рис. 9 Фінальне вікно

Після натиснення клавіші Enter, гравець перейде до головного екрану.

Інтерфейс

Загальний інтерфейс ігрового поля продемонстрований на рисунку 10

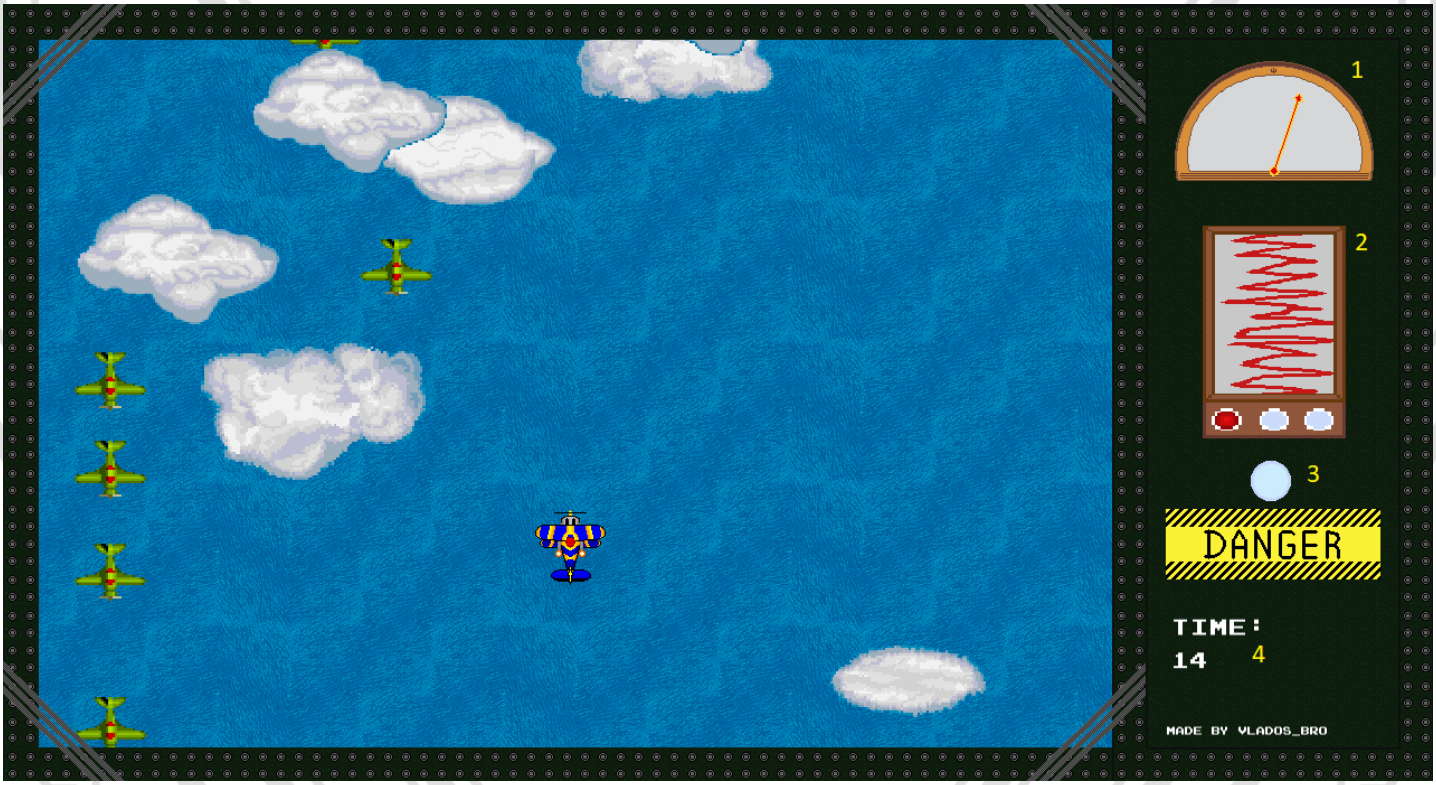


Рис. 11 Загальний інтерфейс гри

Ігровий інтерфейс складається з наступних елементів:

1 - «Тахометр»

Даний прибор показує «кількість» життів гравця. Відповідно, праве крайнє положення стрілки—максимальна кількість життів (10), а крайнє ліве положення - життя кінчилися (0).

2 - «Сейсмограф»

Просто елемент декору. Ніяких функцій не виконує.

3 - Лампа «Небезпека»

У разі, якщо кількість життів гравця менше 3, дана лампа починає світитися червоним кольором, що повідомляє про небезпеку.

4 - Таймер

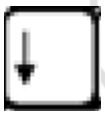
Відображає час від початку ігрової сесії

Керування

 - політ ліворуч

 - політ вверх

 - політ праворуч

 - політ униз

- вогонь

Інше

Посилання:
Інструкція по підключенню бібліотек SFML:
https://kychka-pc.ru/sfml/urok-1-podklyuchenie-biblioteki-k-srede-razrabotki-visual-studio-2013.html

Моя пошта для зв’язку: levchuk.ua.101@gmail.com