МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
Об'єктно-орієнтоване конструювання програм»

Звіт до проекту
Гра на C++ з використанням штучного інтелекту «Tick_Tack_Toe»

Виконали:	Іванчук В., Косован І., Мартин А., Пантьо І., Прищепа В.	**Перевірив**:	Іванов Євген В'ячеславович
Група	ІПЗ-21	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність			

Вступ

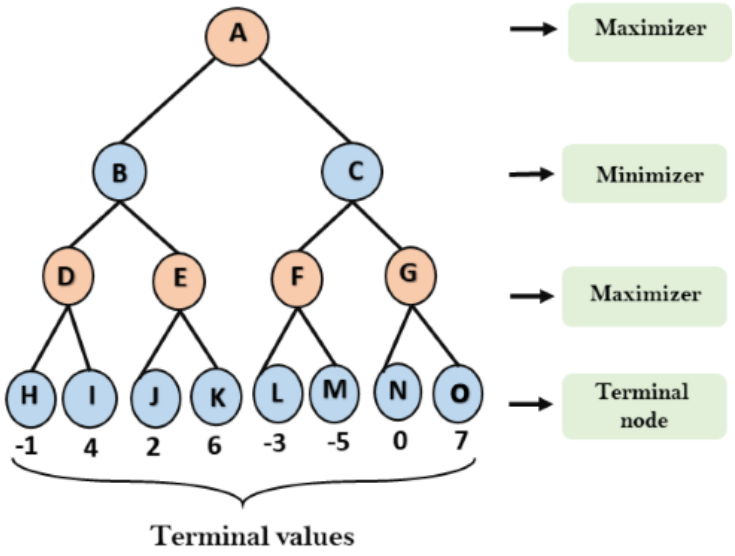
Мета роботи: розробка штучного інтелекту на C++ для гри в одну з варіацій хрестиків-нуликів.

Короткий опис проекту: гра, написана мовою C++ з бібліотекою SFML Має три ігрові моди: PvP, PvE, EvE. AI створений на основі Min-Max алгоритму

Особливості проекту:
На відміну від програм-аналогів, наша гра Tick_Tac_Toe, окрім від трохи зміненої назви гри, здатна підтримувати три ігрові моди, а саме PvP (гравець проти іншого гравця), PvE (гравець проти комп'ютера), EvE (комп'ютер проти комп'ютера). В той час як моди PvP та PvEможна користуватися для звичайного проведення дозвілля з іншою людиною або з комп'ютером за грою, мод EvE потрібен для демонстрації роботи AI та процесу самонавчання та розвитку штучного інтелекту.
Також наш варіант гри Tick_Tac_Toe є реалізацію не найпоширенішої версії гри (з полем 3x3), а ускладненої поширеної версії, в якій поле гри може бути в рази більшим, а отримує перемогу той гравець, який зможе розмістити 5 фігур в ряд.

Загальний опис алгоритму Min-Max:
Даний алгоритм спирається на здоровий глузд. Тобто, хід потрібно зробити таким чином, щоб максимізувати оцінку власної перемоги.
Перш за все, потрібно побудувати граф гри, де один гравець буде представлений як Maximizer, а інший – Minimizer. Далі гравець Maximize має набрати максимальний рахунок, а інший гравець – мінімальний.

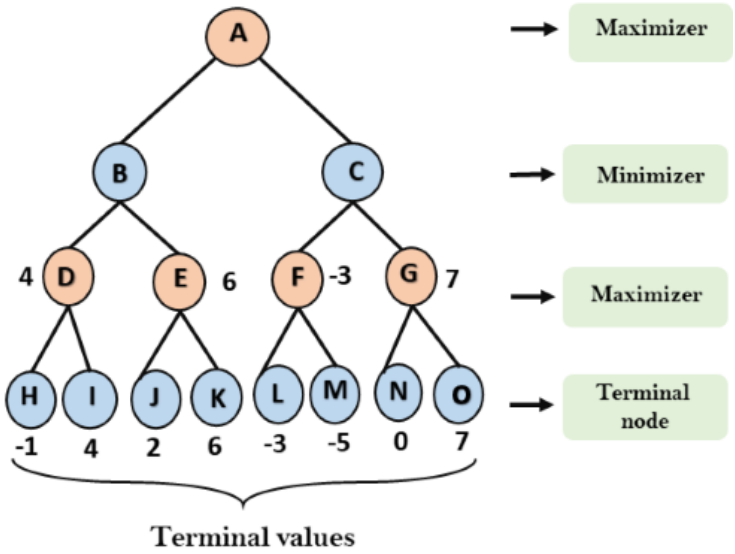
Крок 1



На цьому кроці алгоритм генерує все дерево гри і застосовує утилітарну функцію, щоб отримати значення залежно від кожного стану гри. В діаграмі вище A є початковим станом дерева. Якщо Maximizer ходить перший і, у найгіршому випадку, його початкове значення

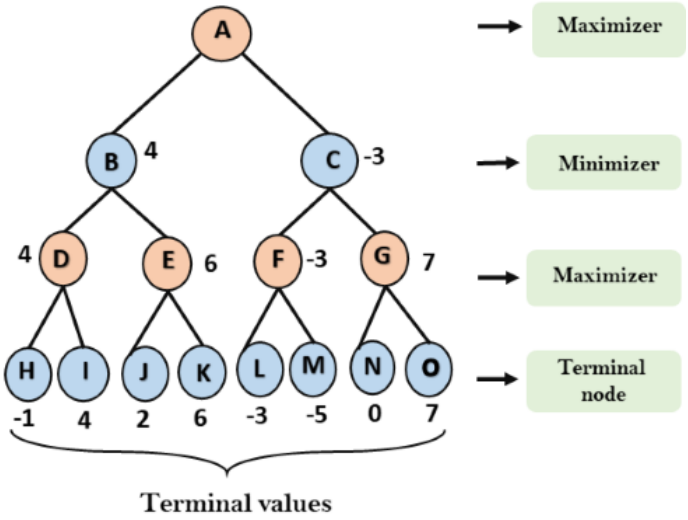
-∞, то у Minimizer у найгіршому випадку початкову значення +∞.

Крок 2



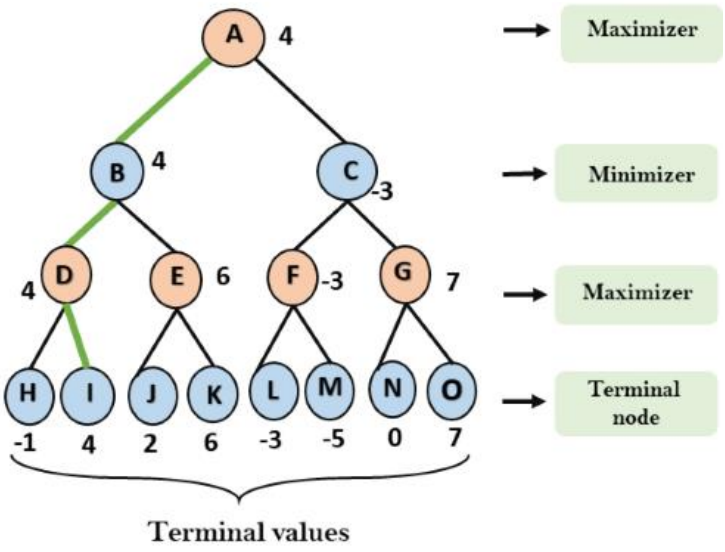
На цьому кроці мі знайдемо утилітарні значення для Maximizer. Його початкове значення -∞, тому ми будемо порівнювати кожне значення у кінцевому стані з початковим і визначати вузли з вищими значеннями. Таким чином знайдемо максимум серед них.

Крок 3



На цьому кроці черга Minimizer. Порівнюємо значення у вузлах з +∞ та обираємо найменше серед них.

Крок 4



Тепер знову черга Maximizer і знову обираємо максимум з усіх вузлів. Оскільки в даному прикладі лише 4 шари, крок 4 є кінцевим, але в реальних іграх алгоритм є рекурсивним та набагато довшим.

Опис алгоритму безпосередньо для гри:

Для визначення ходу, усім ходам надається певна вага. Максимальну вагу має хід, що приводить до перемоги. Після нього йде хід, що рятує від поразки.
Наступним є хід, що дозволяє поставити максимальну кількість «хрестиків» в один ряд. Далі хід, що ставить максимальну кількість «хрестиків» у ряд з пропуском одного «хрестика».
Алгоритм прораховує ваги всіх клітинок і робить хід в клітинку з найбільшою вагою.

Функціональні вимоги:

- Система повинна завершувати гру, коли один з гравців поставить 5 фігур в ряд.
- Система повинна надавати можливість користувачу вводити команди в консоль
- Система повинна надавати можливість користувачу зробити поле якої завгодно величини (на скільки вистачить місця)
- Система повинна надавати можливість користувачу взаємодіяти з грою
- Система повинна забезпечувати коректну роботу AI на основі Min-Max алгоритму
- Система повинна надавати можливість здійснити штучним інтелектом хід замість користувача, шляхом натискання гарячої клавіші, що викликає функцію
- Система повинна надавати можливість подивитися вагу клітинки, шляхом натискання гарячої клавіші, що викликає функцію
- Система повинна надавати можливість подивитися вагу усіх клітин, що були

виділені для аналізу, шляхом вводу в консоль відповідної команди

- Система повинна підтримувати три ігрові моди: PvP (гравець проти іншого гравця), PvE (гравець проти комп’ютера), EvE (комп’ютер проти комп’ютера)

Складові компоненти системи:

Гра реалізована мовою C++ з використанням бібліотеки SFML.

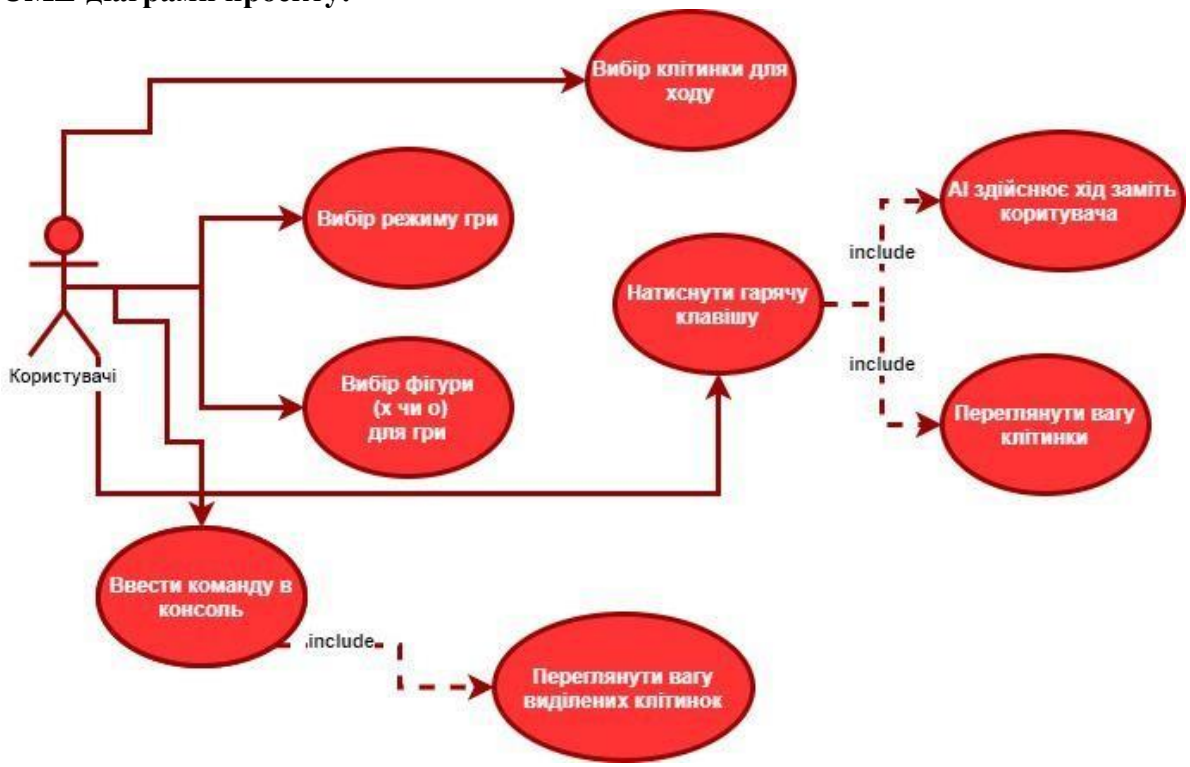
- Опис функцій компоненту:

SFML – це бібліотека для представлення мультимедійних даних. SFML забезпечує простий інтерфейс для розробки гри.

Програмна реалізація:

- SFML складається з п'яти модулів: system, window, graphics, audio і network.

UML-діаграми проекту:



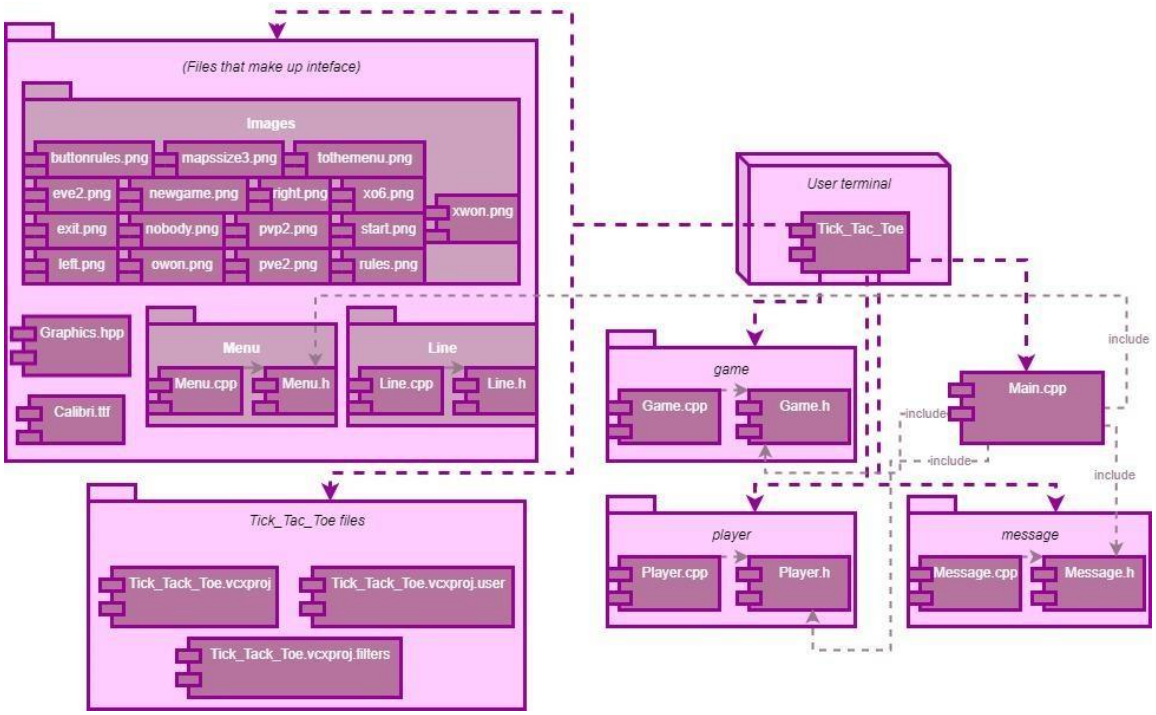
(Діаграма прецедентів)

Варіанти використання системи:

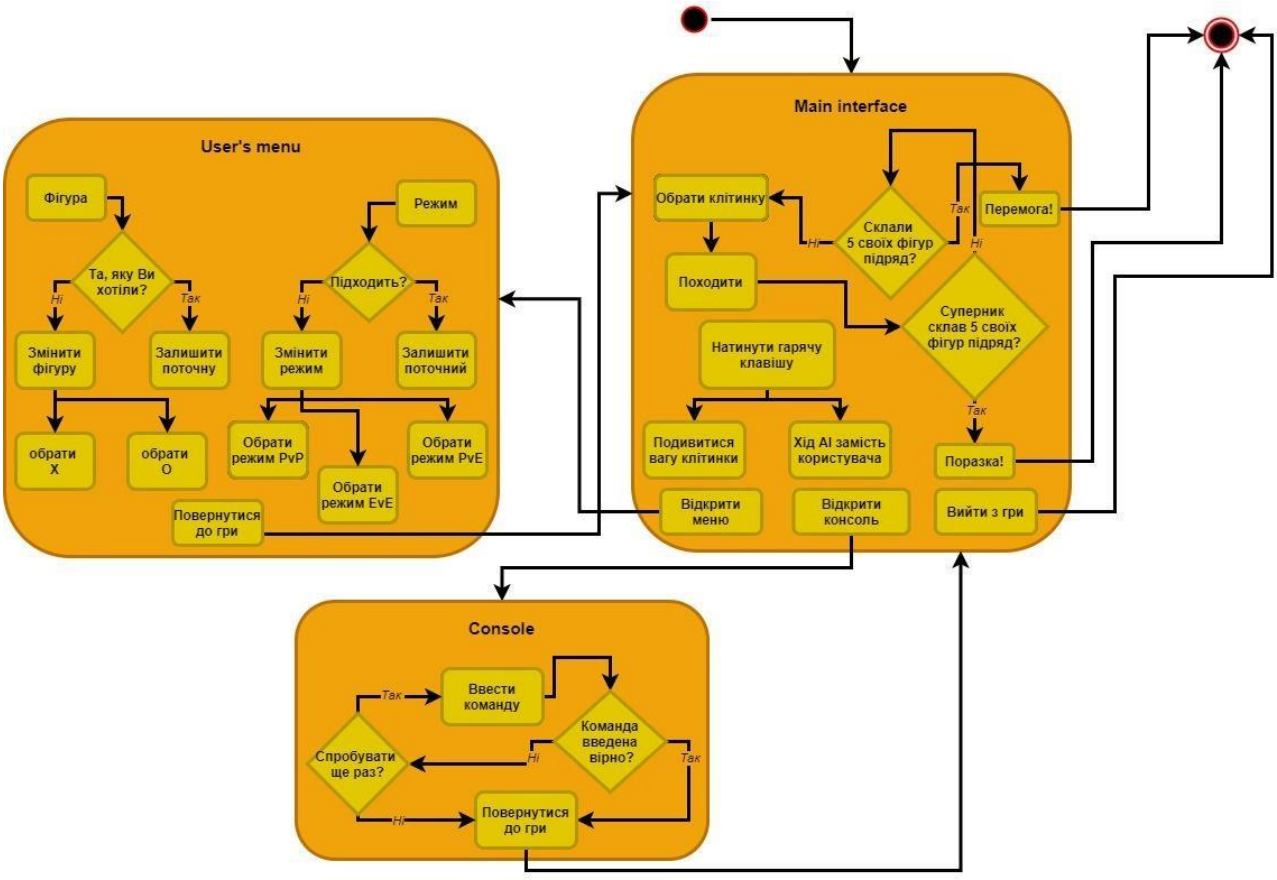
1. Вибір режиму гри:
   Короткий опис: користувач обирає один з трьох режимів гри
   1.2.Суб'єкт: користувач
   1.3. Передумова: користувач запускає гру
   1.4. Основний потік: користувач обирає один з трьох режимів гри (PvP, PvE,

2. Вибір фігури для гри:
   2.1. Короткий опис: користувач обирає фігуру, якою буде здійснювати подальші ходи
   2.2. Суб'єкт: користувач
   2.3. Передумова: користувач запускає гру
   2.4. Основний потік: користувач обирає фігуру серед двох запропонованих (х чи о), якою буде здійснювати подальші ходи
3. Введення команду в консоль:
   3.1.Короткий опис: користувач вводить в консоль команду
   3.2.Суб'єкт: користувач
   3.3. Передумова: користувач запускає гру
   3.4. Основний потік: користувач вводить в консоль команду, яка йому необхідна для виконання потрібної функції
   3.5. Альтернативний потік: користувач неправильно вводить в консоль команду, система повідомляє про помилку, виводячи в консоль повідомлення «Невідома команда, введіть, будь ласка, ще раз»

4. Переглянути вагу виділених клітинок:
    4.1. Короткий опис: користувач вводить в консоль команду, після чого може побачити вагу виділених клітинок
    4.2. Суб'єкт: користувач
    4.3. Передумова: користувач виділяє клітинки для аналізу
    4.4. Основний потік: користувач вводить в консоль команду, після чого може побачити вагу виділених клітинок за умови правильності введення команди
5. Вибір клітинки для ходу:
    5.1. Короткий опис: користувач обирає клітинку для ходу
    5.2. Суб'єкт: користувач
    5.3. Передумова: користувач обирає фігуру для ходу, або ж грає тією, яка була обрана автоматично
    5.4. Основний потік: користувач обирає клітинку для здійснення ходу і здійснює хід
    5.5. Альтернативний потік: користувач обирає клітинку, в якій вже стоїть фігура, але, оскільки клітинка занята, нічого не відбувається
6. Натиснути гарячу клавішу:
    6.1. Короткий опис: користувач натискає гарячу клавішу
    6.2. Суб'єкт: користувач
    6.3. Передумова: користувач запускає гру
    6.4. Основний потік: користувач натискає гарячу клавішу, для виконання необхідної функції
    6.5. Альтернативний потік 1: користувач натискає не ту клавішу, за умови, що клавіша не є гарячою – нічого не відбувається.
    6.6. Альтернативний потік 2: користувач натискає не ту клавішу, за умови, що клавіша є гарячою, але не виконує необхідну функцію – виконується інша функція, яка прив'язана якраз до цієї клавіші
7. AI
    7.1. Короткий опис: користувач натискає гарячу клавішу для виконання ходу, AI здійснює хід.
    7.2. Суб'єкт: користувач
    7.3. Передумова: користувач запускає гру
    7.4. Основний потік: користувач натискає гарячу клавішу для виконання функції ходу бота замість нього, а AI здійснює хід
8. Переглянути вагу клітинки:
    8.1. Короткий опис: користувач натискає гарячу клавішу для перегляду ваги клітинки
    8.2. Суб'єкт: користувач
    8.3. Передумова: користувач натискає на клітинку
    8.4. Основний потік: користувач натискає гарячу клавішу для перегляду ваги клітинки, яка була обрана



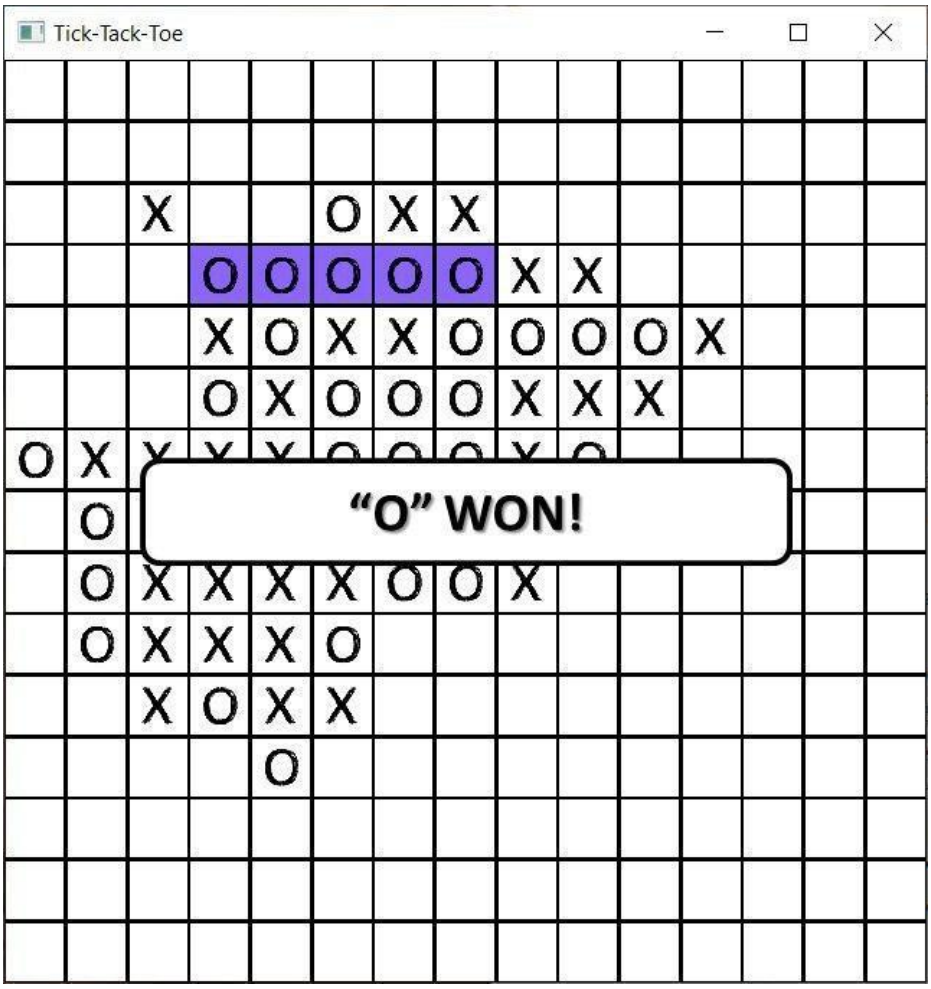
(Діаграма компонентів)



(Діаграма стану)

На діаграмі стану зображений спроектований інтерфейс користувача.

Скріншот роботи програми:



Висновок:
Була розроблена гра, написана мовою C++ з бібліотекою SFML, яка має три ігрові моди, і штучний інтелект якої створений на основі Min-Max алгоритму. До гри був повністю розроблений інтерфейс, додані додаткові функції, такі як гарячі клавіші та консоль для вводу команд. Для проекту було спроектовано три діаграми: діаграма прецедентів, діаграма компонентів та діаграма ст ану.

Посилання на код гри на github:
https://github.com/IVIVA69/OOKP_KNU/tree/master/Tick_Tack_Toe

В цьому ж репозиторії (https://github.com/IVIVA69/OOKP_KNU) знаходяться інші наші робочі проекти (операційна система та код, отриманий в ході вирішення альтернативної практичної роботи.