Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Міні-проект з предмету   
Основи об'єктно-орієнтованого програмування

Game TD

Виконав студент 2-го курсу

Групи ІПС-22

Дехтяренко Михайло Віталійович

Київ – 2023

**Розділ перший:  
Опис проекту**

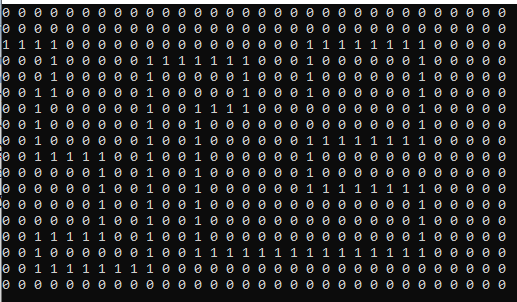
Проект Game TD - це гра в жанрі Tower Defense, де головна мета гравця - стратегічно розмістити вежі таким чином, щоб ворог не міг дійти до кінцевої точки. Цей жанр був обраний через його відносну простоту реалізації.

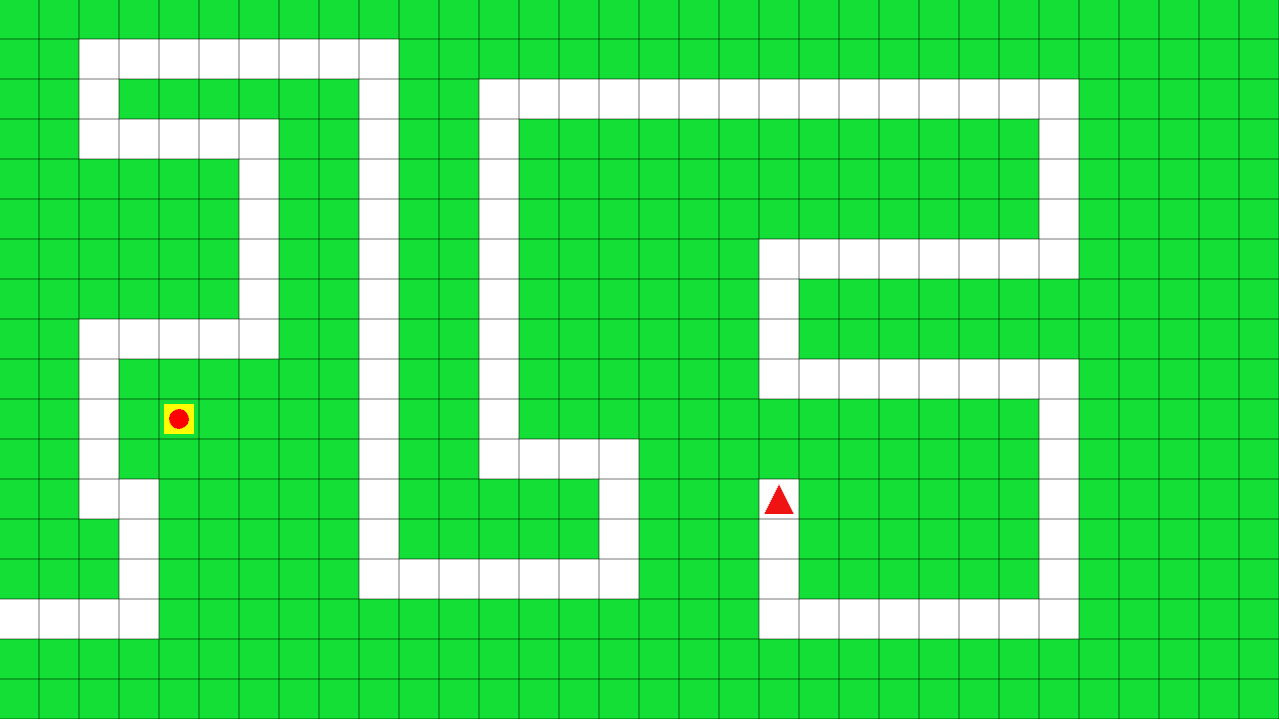
На даний момент у грі є один тип веж, що надає гравцям можливість зосередитися на вдосконаленні своїх стратегій розміщення веж, при урахуванні, що після кожної смерті ворог стає сильніше і сильніше. Гра має три зовнішні вигляди для веж та землі, що додає візуального різноманіття та дозволяє гравцям насолоджуватися приємною графікою під час гри.

У майбутньому планується додати більше типів веж та ворогів, що зробить гру. Крім того, будуть додані нові зовнішні вигляди для веж та землі, що забезпечить більше варіативності.

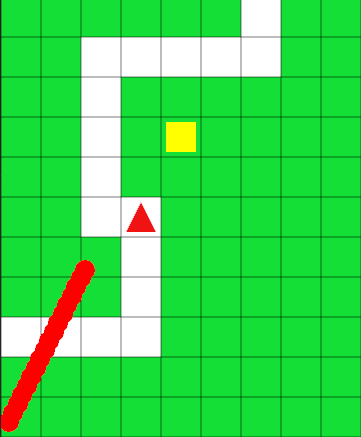
**Розділ другий:  
Розвиток проекта**

У багатьох людей перша гра робиться в спеціальних середовищах розробки, таких як Unreal Engine, Unity, Godot та інші. Те, що відокремлює проект Game TD від інших полягає в тому, що гра працює на графічній бібліотеці OpenGL FreeGlut, що в деяких моментах ускладнювало розробку, але давало чудовий досвід, який можна буде використати в майбутньому. Спочатку треба було зробити якесь ігрове поле, потім придумати, як зробити розташування веж та дороги. На заняттях з основ програмування на першому курсі викладач розповідав, що багато перших ігр працювали на графах та матрицях, тому для побудови ігрового поля за основу було взято матрицю з 0 та 1, де 0 – це поле для розташування веж, а 1 – дорога, по якій рухається ворог.   
Потім почалась розробка ворога, прописування його штучного інтелекту(наскільки це можливо назвати інтелектом) та тест вежі.  
На скріншотах нижче представлена перша працююча версія, де є ворог, який вміє рухатись та стоїть вежа, яка створює кулю, але в неї поки нема фізики руху.

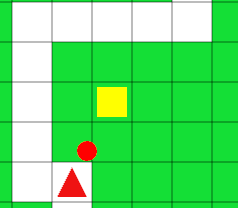
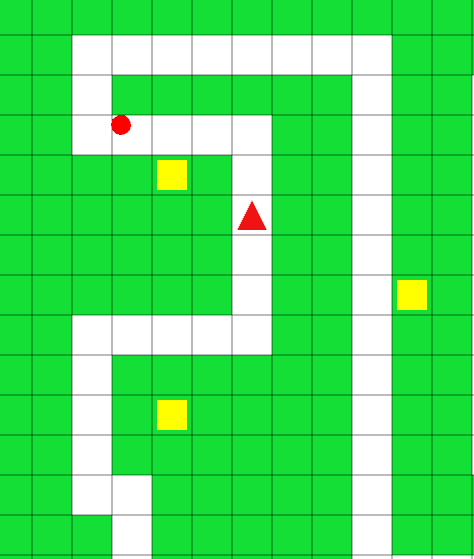
 Матриця що представляє ігрове поле (один елемент матриці дорівнює клітинці 40х40 пікселів)



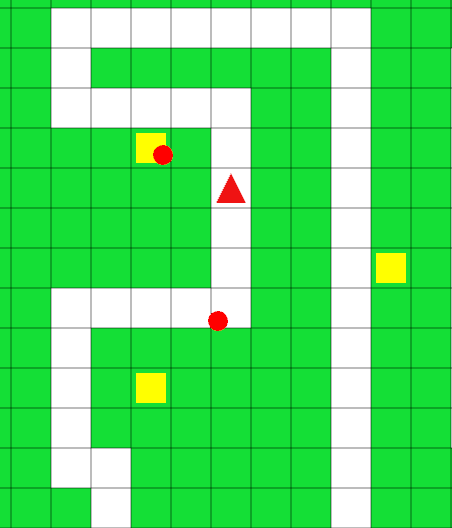
Само ігрове поле, на якому ворог дійшов до кінця.  
 Ігрове поле інвертовано відносно матриці, тому що рахунок пікселів (чи краще сказати координат OpenGL) було задано з нижнього лівого кута



Далі треба було розробити логіку руху кулі, зробити можливість вбивати ворога при дотику. Під час розробки цих механік, виникало декільках цікавих багів, такі як луч смерті, тобто замість однієї кулі, вилітала ціла купа куль.

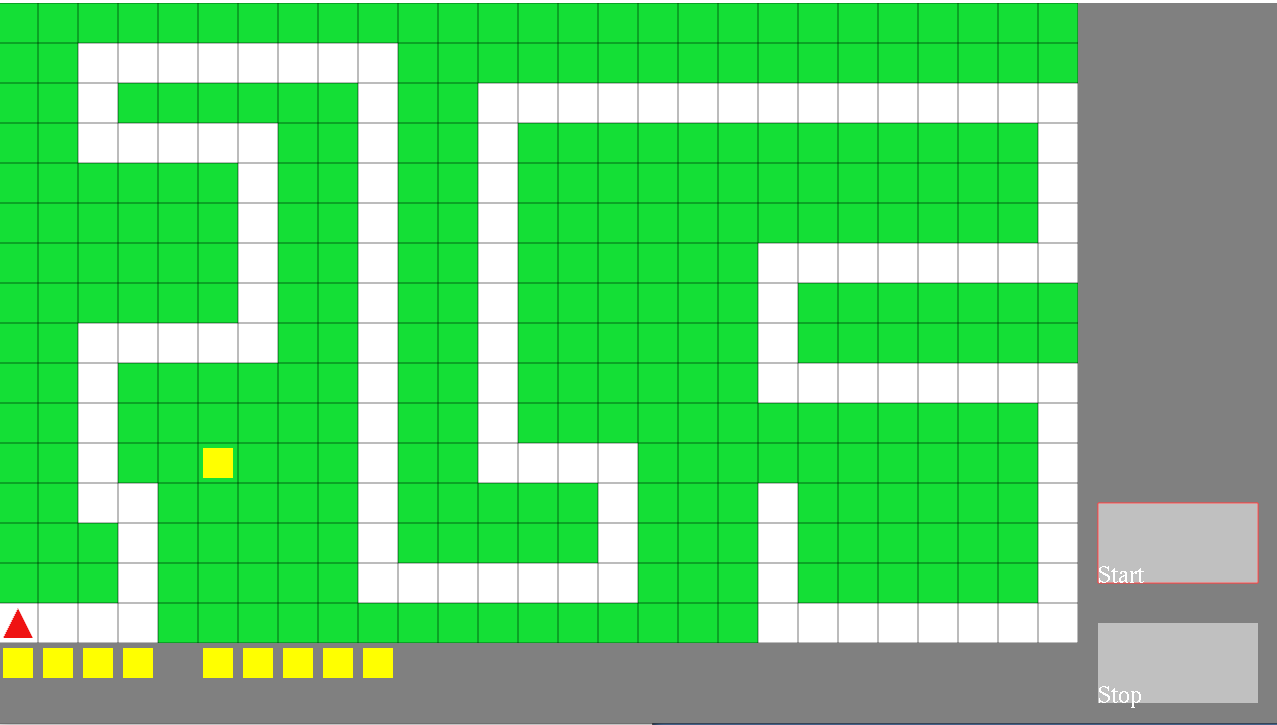
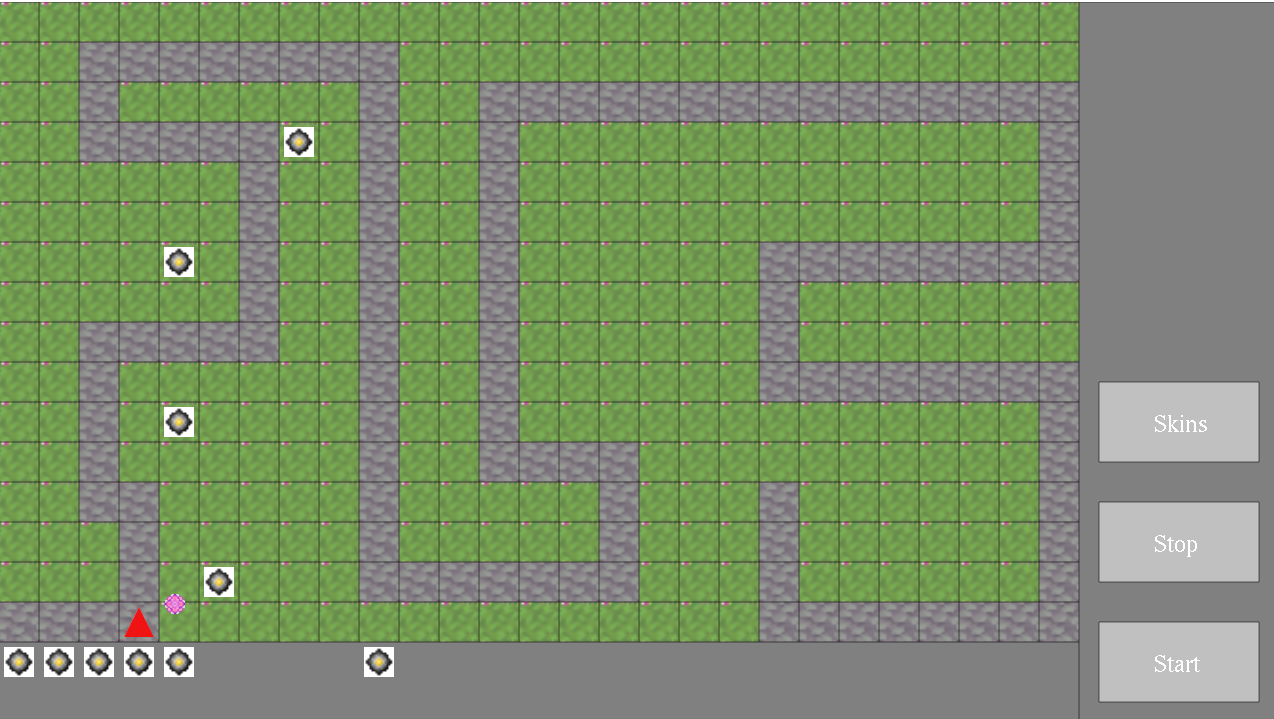


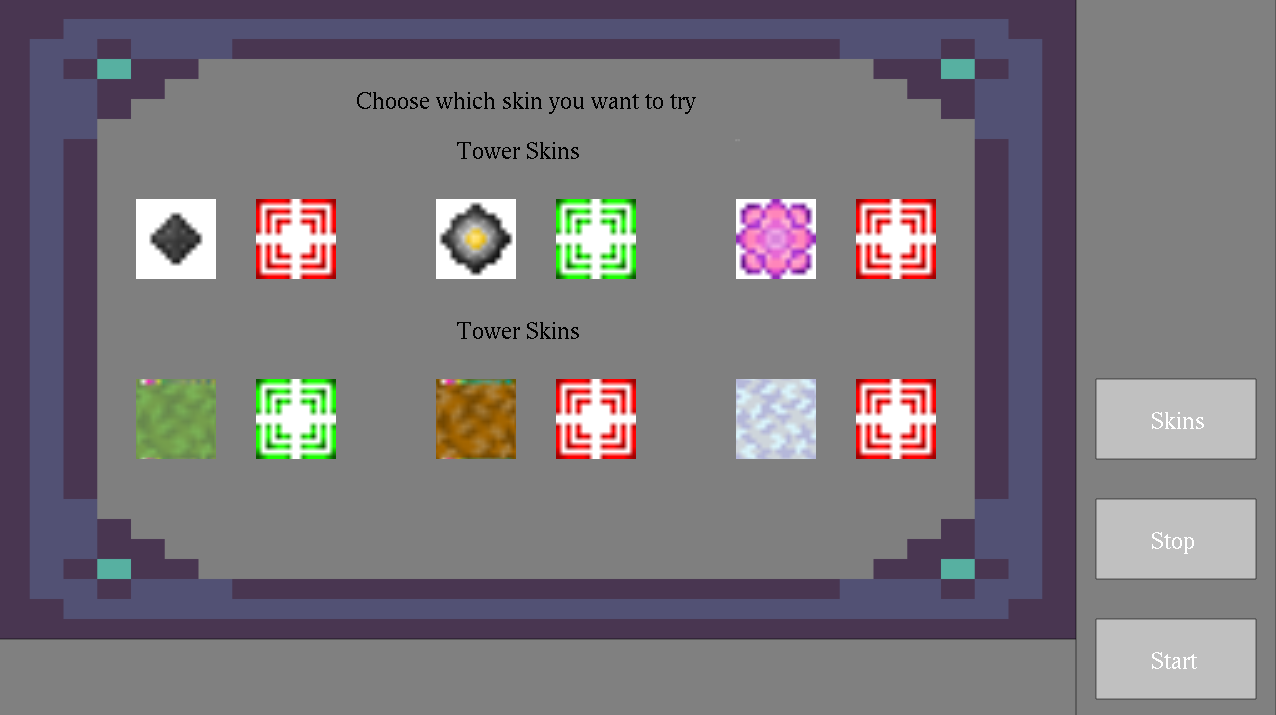
Працююча куля, що вбиває ворога  
  
  
  
  
  
  
Під час тестування   
декількох веж зявилась проблема, що ,коли ворог знаходиться в четвертій чверті, куля вилітає у протилежний бік.

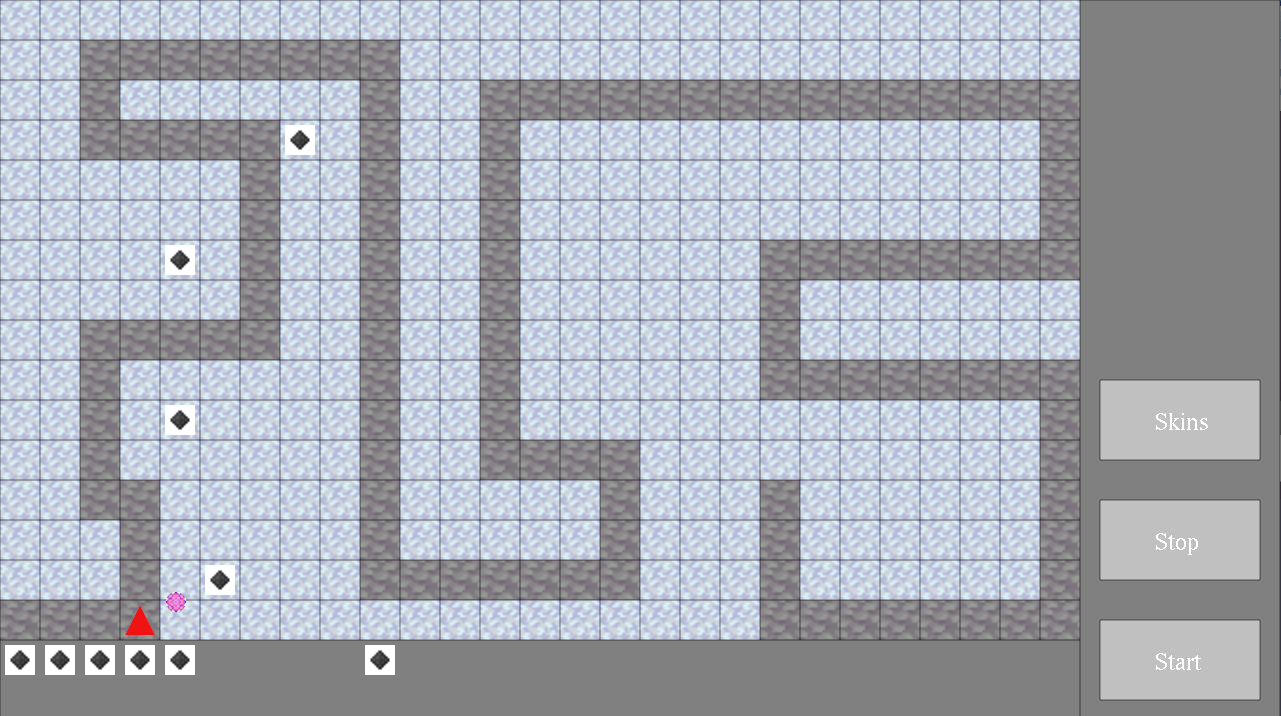
В спробах вияснити в чому проблема, виявилось що функція арктангенсу, яка була використана, бачить лише в першій та третій, при спробах штучно домножати на π чи робити якісь додаткові перевірки, завжди одна з чвертей залишалась сліпою зоною.

Ця проблема була вирішена завдяки функції арктангенсу від двох аргументів  
(Друге Джерело)

Оскільки логіку веж та вбивство ворога було вже реалізовано, треба було зробити розташування веж та мінімальний інтерфейс для користувача.

 На цьому етапі було реалізовано:  
 Перенесення веж з панелі, використовуючи мишку.  
 Створено кнопки Start та Stop. Гравець не має право ставити чи переносити вежі, пока гра в активному стані, її обов’язково треба зупинити, щоб маніпулювати з спорудами.  
  
 І остаточна версія проекту.

В ній було додано текстури для більшості об’єктів, кнопку Skins, яка відповідає за панель, де користувач може змінити зовнішній вигляд веж та полів, на яких можуть бути розташовані вежі.   
 Гра при інших текстурах  

**Розділ третій  
Код программи**

**Класи**

В останній версії программи 4 класи:  
  
Клас matrix, який було створенно під час лабораторної роботи з дисципліни Алгоритми та складність.

**Поля**:  
 **T\*\* a:** Двовимірний масив, який представляє матрицю.  
  
**int m, n**: Розміри матриці. m - кількість рядків, n - кількість стовпців.

**Методи**:  
 **matrix()**: Конструктор за замовчуванням, який створює порожню матрицю.  
  
**matrix(int n, int m):** Конструктор, який створює матрицю розміром n x m.  
  
**~matrix():** Деструктор, який звільняє пам’ять, виділену для матриці.

**Клас**target: Цей клас представляє ціль  
  
**Поля**:

**shape\* p**: Вказівник на форму цілі.

**int m, n**: Координати цілі.

**char shapeType**: Тип форми цілі (‘t’ - трикутники, ‘s’ - квадрати, ‘d’ - ромби).

**int currentDirection**: Поточний напрямок руху цілі (0 - праворуч, 1 - вгору, 2 - ліворуч, 3 - вниз).

**int health:** Здоров’я цілі.

**Клас**tower: Цей клас представляє вежу.  
  
**Поля:**

**int m, n**: Координати вежі.

**const int r**: Радіус дії вежі.

**bool active**: Стан вежі (якщо стан активний, вежа не працює, якщо не активний, вона може стріляти).

**Клас**bullet: Цей клас представляє кулю  
  
**Поля:**

**int m, n:** Координати кулі.

**int iTarget**: Індекс цілі, до якої направлена куля.

**double angle**: Кут, під яким випущена куля.

**double r:** Радіус кулі.

**Глобальні змінні**  
**long long counterT** Ця змінна відповідає за час, точніше частоту руху ворога в грі.

**(const int w = 1280 : та const int h = 720)** - ці константи визначають розміри   
вікна гри w - ширина, h - висота.

**int xx, yy** – ці змінні використовуються для відстеження координат миші.

**int ButtonStatus** – ця змінна відстежує стан кнопки. 1 – кнопка нажата, інше значення, кнопка не нажата

**int status** – ця змінна відстежує загальний стан гри. Значення 0 означає, що гра зупинена, 1 - що гра в активному стані, 2 - що гра закінчена, а 3 - що відображаються скіни.

**int Tstatus** – ця змінна відповадіє за те, яка вежі текстура на разі активна  
  
**int Fstatus** – ця змінна відповадіє за те, яка поля текстура на разі активна   
  
**int currentTower** – ця змінна може відстежувати поточну вежу, яка вибрана. Значення -1 означає, що жодна вежа не вибрана.

**target\* targets**: Вказівник на масив цілей.

**tower\* towers**: Вказівник на масив веж.

**int nTargets**: Кількість цілей.

**int nTowers**: Кількість веж.

**Функції программи**

**GameOver** – якщо координати ворога дорівнюють останньому тайлу дороги, закінчує гру

**displayText** – функція, що виводить повідомлення

**showTarget** – функція, що малює ворога, задаючи 3 координати, для відображення трикутника

**showTowers** – функція, що малює квадрат вежі та текстуру в залежності від Tstatus

**circle** – функція, яка малює коло та накладує на нього текстуру.

**skinPane** – функція, що відповідає за повне відображення панелі, де гравець може змінювати та виводить текст у меню текстур

**Skins** – функція, що відповідає за відображення текстур та кнопки, які показуют вибрану текстуру

**Detection** – функція, що відповідає за виявлення ворога та задання кута кулі

**Miss** – функція, що відповідає за видалення кулі та перезапуск вежі, якщо куля не попала в ворога

**Hit** – функці, що відповідає за потрапляння кулі у ворога та оновлення ворога при його смерті (більше поінтів здоровя та обнуління координат)

**glutIdleFunc(inc)** - ініціалізує функцію, що будет працювати фоном, в даному випадку функція inc, яка має в собі:  
Зупинку гри, якщо активен статус зупинки (2)  
Штучний інтелект ворога, що дозволяє йому рухатись по будь-якій карті

виклик функції detection

оновлення руху кулі на площині

виклик таких функцій: miss, hit, gameOver

**glutPostRedisplay()** – функція, що відповідає за оновлення зображення

**createTileRoad** – функція, що малює квадрати поля, в залежності від Fstatus, використовує текстуру, та накладує чорну окантовку, щоб поле було розбите на квадрати

**createTileField** - функція, що малює квадрати дороги, використовує текстуру, та накладує чорну окантовку

display - функція, що відповідає за відображення всього:   
  
   
for (int i = 0; i < N; i++)

{

for (int j = 0; j < M; j++)

{ //цикл, що відповідає, за відображення поля

if (pRoad->a[i][j] == 1)

createTileRoad(i, j);

else createTileField(i, j);

}

};

showBullets(bullets);

showTarget(targets);

if (status == 3) // умова, яка дозволяє відкривати панель скінів

{

skinPanel();

Skins();

}

showPanel();

showTowers(towers);

glutSwapBuffers();

}

**Mouse** – функція, яка відповідає за всі інтеракції з мишею. В ній прописані координати в пікселях,де потрібно нажати, щоб спрацювала та чи інша подія, також прописані умови розташування веж (не на дорогах, під час паузи, і не на інших вежах)

**MouseMotion** – функція, обробляє події руху миші в OpenGL, дозволяє переносити вежі.

Keyb – функція, за закриває гру, коли натискається кнопка esc.   
  
В **main** відбувається створення ворога та веж, перенесення матриці в матрицю рівня та ініціалізація OpenGL FreeGlut

**Висновок**

Проект Game TD – хоч і тривіальна гра в жанрі Tower Defense, розроблена з використанням OpenGL FreeGlut, але вона демонструє, як можна створити просту гру з використанням базових принципів комп’ютерної графіки та програмування ігор.

Основні елементи гри, такі як вежі, вороги та кулі, були успішно реалізовані, а також було введено декілька механік, таких як переміщення веж та вбивство ворогів. Проте, як і в будь-якому проекті, є місце для покращення та розвитку.

У майбутньому до гри можна додати більше типів веж та ворогів, що зробить гру більш різноманітною. Також можна вдосконалити код, створивши окремий клас для кнопок та оптимізувавши систему захоплення ворогів.

Загалом, проект Game TD - це відмінний і, мабудь, безцінний досвід, того, як можна створити гру з використанням OpenGL FreeGlut. Цей проект надає цінний досвід, який можна використати в майбутніх проектах

**Джерела**

1) Графическая библиотека OpenGL учебно-методическое пособие [https://studfile.net/preview/7588438/](https://studfile.net/preview/7588438/%20)

2) <https://en.wikipedia.org/wiki/Atan2>