



# TANK TAEK

---



# LIST OF CONTENTS

**Overview** ----- **Slide 2**

**System requirement** ----- **Slide 3-4**

**Source Code (Class)** ----- **Slide 5-11**



ចំណាំ: 1 នាន់បានឱ្យទទួលឯកសារ: 3

# OVERVIEW

# ແນວຄົດຫລັກ

เกม Tank Taek เป็นเกมยิงรถถังสองมิติแนวคลาสสิก ผู้เล่นควบคุมรถถัง (P) เพื่อทำการยิงรถถังศัตรูทั้งหมดในแต่ละด่าน เป้าหมายคือผ่านด่านต่างๆ ไปจนเจอบอสสุดท้ายที่แข็งแกร่งกว่าปกติ

แผนที่แต่ละด่านเป็นกริดอักขระ  $30 \times 20$  มีป้อมและสิ่งกีดขวาง (#) ทางแบบสุ่ม ผู้เล่นมี 3 ชีวิต แต่ละด่านชีวิตจะฟื้นกลับมาเต็มเมื่อผ่านด่านไป. เกมแสดงผลด้วยตัวอักษร ASCII บนหน้าจอコンโซล ทำให้ได้บรรยากาศย้อนยุคของเกมคอนโซลรุ่นเก่า

# วิธีการเล่น

ผู้เล่นใช้ปุ่ม W,A,S,D เพื่อเคลื่อนที่และกด Spacebar เพื่อยิงกระสุน (\*) ตามทิศทางที่รถถังหันหน้าอยู่ รถถังของศัตรู (E) เคลื่อนที่และยิงได้เองแบบสุ่ม ผู้เล่นต้องหลบและยิงสวนเพื่อเอาชีวิตรอด.



# ข้อกำหนดของระบบ

## Functional Requirements

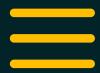
- รองรับผู้เล่นเดี่ยว ควบคุมรถถังได้ด้วยคีย์บอร์ด (W,A,S,D เคลื่อนที่ และ Space ยิงกระสุน)
- มีระบบศัตรูอัตโนมัติหลายคัน เคลื่อนที่และยิงแบบสุ่ม ผู้เล่นต้องหลบเลี่ยงกระสุนของศัตรู
- รองรับการสร้างและเคลื่อนที่ของกระสุน (\*) เมื่อชนรถถังจะกำลังกันที่ และกระสุนจะหายไปเมื่อชนกำแพงหรืออุปกรณ์
- มีระบบด่านอัตโนมัติ 3 ด่าน ด่านสุดท้ายจะมี บอส (B) ซึ่งมีพลังชีวิตมากกว่าปกติ ต้องยิงหลายครั้งจึงจะกำลัง
- เมื่อผู้เล่นถูกยิงจนชีวิตหมด เกมจบ (แสดง “จบเกม! ผู้เล่นแพ้”) และ เมื่อกำลังบอสได้สำเร็จ แสดงข้อความ “ยินดีด้วย คุณชนะเกมครบถ้วน!”



# ข้อกำหนดของระบบ

## Non-Functional Requirements

- ภาษาและเทคโนโลยี: พัฒนาโดยใช้ภาษา C++ กึ่งหนด และใช้งานบนコンโซล (console) โดยไม่มีกราฟิกพิเศษ ใช้ไลบรารี `<conio.h>` สำหรับรับคีย์บอร์ดใน Windows
- ประสิทธิภาพ: วงลูปหลักรันได้ราบรื่นประมาณ 10 เฟรมต่อวินาที (หน่วง ~100ms) เพื่อให้การเคลื่อนที่ของรถถังและกระสุนเห็นได้ชัด และการคำนวณตำแหน่ง-การชนทำได้ในเวลาเชิงเส้น ( $O(n)$ ) ซึ่งเพียงพอสำหรับจำนวนวัตถุต่อนเล่น
- ความเข้ากันได้: เน้นรันบนระบบ Windows (ใช้คอนโซล Windows) เนื่องจากใช้ `<conio.h>` การแสดงผลภาษาไทยต้องรองรับ UTF-8 แต่โค้ดกึ่งหนดจัดเก็บเป็น UTF-8 เพื่อภาษาไทยที่ถูกต้อง
- คุณภาพโค้ด: จัดโค้ดออกเป็นหลายไฟล์ตามหน้าที่ ( เช่น Game, Tank, Player, Enemy, Map, Bullet, main ) เพื่อให้อ่านง่ายและบำรุงรักษา โดยใช้ชื่อตัวแปร เมธอด และคอมเม้นต์เป็นภาษาไทย ชัดเจน
- ความเสถียร: โปรแกรมต้องทำงานได้บ้านโดยไม่ค้างหรือเด้งออกเอง มีการจัดการหน่วยความจำถูกต้อง (จัดสรร/ลบ (new/delete) สำหรับวัตถุที่สร้างไดนามิก เช่น รถถังศัตรู) เพื่อป้องกัน memory leak



# SOURCE CODE (CLASS)

## Class Tank

```
1 #ifndef TANK_H
2 #define TANK_H
3
4 #include <vector>
5 #include <iostream>
6 #include <string>
7 class Map; // forward declare Map (used in method parameters)
8 class Bullet; // forward declare Bullet
9 class Tank {
10 protected:
11     int ตำแหน่งX; // ตำแหน่งแกน X ของรถถัง
12     int ตำแหน่งY; // ตำแหน่งแกน Y ของรถถัง
13     int พลังชีวิต; // พลังชีวิตของรถถัง (HP ของรถถัง)
14     int พลังกระสุน; // ความแรงของกระสุนที่ยิงออก
15     char ตัวอักษร; // ตัวอักษรที่ใช้แทนรถถังบนแผนที่
16 public:
17     Tank(int x, int y, int hp, int bulletPower, char symbol)
18         : ตำแหน่งX(x), ตำแหน่งY(y), พลังชีวิต(hp), พลังกระสุน(bulletPower), ตัวอักษร(symbol) {}
19     virtual ~Tank() {} // destructor ควรเป็น virtual (เพื่อให้ลบถูกชนิดเมื่อ delete ผ่าน pointer คลาสแม่)
20
21     // Getter ส่าหริญค่าต่าง ๆ (มีการ encapsulation)
22     int getX() const { return ตำแหน่งX; }
23     int getY() const { return ตำแหน่งY; }
24     int getHP() const { return พลังชีวิต; }
25     char getSymbol() const { return ตัวอักษร; }
26     int getBulletPower() const { return พลังกระสุน; }
27
28     // ลดพลังชีวิตเมื่อโดนยิง (damage)
29     voidลดพลังชีวิต(int damage) {
30         พลังชีวิต -= damage;
31         if (พลังชีวิต < 0) พลังชีวิต = 0;
32     }
33     boolisDestroyed() const {
34         return พลังชีวิต <= 0;
35     }
36
37     // พังก์ชันเคลื่อนที่รถถัง 1 步 (dx, dy) ถ้าไม่เข้ากับเขตหรือตัวอื่น
38     boolmove(int dx, int dy, Map &map, const std::vector<Tank*> &tanks);
39
40     // พังก์ชัน virtual สำหรับ AI ของตัวถัง (ถูกเลนจะไม่ override พังก์ชันนี้)
41     // ใช้ polymorphism เพื่อให้ Enemy/Boss มีพฤติกรรมต่างกัน
42     virtualvoidupdateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) {
43         // คำ default ไม่ทำอะไร (สำหรับ Player ซึ่งควบคุมด้วยผู้เล่นเอง)
44     }
45 };
46
47 #endif
```

- คลาส Tank เป็นคลาสฐาน (base class) สำหรับรถถังทั้งหมดในเกม ซึ่งนิยามคุณสมบัติพื้นฐานของรถถังทั่วไป เช่น พิกัดตำแหน่ง ทิศทาง หน้า และพลังชีวิต (HP) คลาสนี้ใช้มาตราตัวรถถังบนหน้าจอและจัดการการเคลื่อนไหวพื้นฐาน เช่น การเดินหน้า การหมุนทิศทาง และการยิงกระสุน
- ประเภทของคลาส: เป็นคลาสหลัก (base class) ที่คลาสอื่นสืบทอดจากมัน
- ฟังก์ชันหลัก: ให้คุณสมบัติหลักของรถถัง เช่น เก็บสถานะตำแหน่งและพลังชีวิต กำหนดการเคลื่อนที่ (ขึ้น/ลง/ซ้าย/ขวา) และวิธีการยิงกระสุน (สร้างวัตถุกระสุนใหม่เมื่อรถถังยิง) การทำงานเหล่านี้ถูกใช้โดยคลาสลูก เช่น Player และ Enemy ที่สืบทอดคุณสมบัติเหล่านี้ไปใช้ต่อ
- กำหนดการเคลื่อนที่และสถานะพื้นฐานของรถถัง เช่น วิธีการเคลื่อนที่ไปยังตำแหน่งใหม่ การเช็คชน (collision) กับอุปสรรค หรือการลดพลังชีวิต เมื่อตกรถถัง ทำให้คลาஸลูกสามารถนำวิธีเหล่านี้ไปใช้ได้ทันทีหรือปรับแต่งเพิ่มเติมภายหลัง



# SOURCE CODE

## Class Player

```
1 ~ #ifndef PLAYER_H
2   #define PLAYER_H
3
4   #include "Tank.h"
5
6 ~ // คลาส Player สืบทอดจาก Tank (คลาสแม่)
7 // ใช้แทนผู้เล่นที่ควบคุมโดยคนเล่น
8 ~ class Player : public Tank {
9 private:
10    int ทิศทางX; // ทิศทางล่าสุดที่ผู้เล่นทิ้น (แกน X)
11    int ทิศทางY; // ทิศทางล่าสุดที่ผู้เล่นทิ้น (แกน Y)
12 public:
13    // Constructor: กำหนดตำแหน่งเริ่มต้น, พลังชีวิต=3, พลังกระสุน=1, สัญลักษณ์ 'P'
14    Player(int startX, int startY);
15    // Setter และ Getter สำหรับทิศทางที่ทิ้นของผู้เล่น
16    void setDirection(int dx, int dy) {
17        ทิศทางX = dx;
18        ทิศทางY = dy;
19    }
20    int getDirX() const { return ทิศทางX; }
21    int getDirY() const { return ทิศทางY; }
22 };
23
24 #endif
25
```

- คลาสนี้คืออะไร: คลาส Player เป็นคลาสสูกที่สร้างจากคลาส Tank เพื่อแทนรถถังของผู้เล่น คลาสนี้จะมีคุณสมบัติเพิ่มเติมเฉพาะสำหรับผู้เล่น เช่น การรับค่าปุ่มจากคีย์บอร์ด (W/A/S/D) เพื่อควบคุมการเคลื่อนที่ การจำกัดจำนวนชีวิต รวมถึงการตรวจจับการยิงกระสุนจากผู้เล่น (โดยปุ่ม Space)
- ประเภทของคลาส: เป็นคลาสสูก (derived class) ของคลาส Tank
- สืบทอดมาจากคลาสใด: สืบทอดมาจากคลาส Tank (ได้รับคุณสมบัติพื้นฐานของรถถัง เช่น ตำแหน่ง ตัวแหน่ง และพลังชีวิตมาจาก Tank)
- บทบาทเฉพาะ: คลาส Player อาจเพิ่มฟังก์ชันพิเศษ เช่น ตรวจสอบสถานะชีวิตของผู้เล่น (จำนวนชีวิตที่กำหนดไว้ในเกม) และส่งการยิงกระสุนเมื่อผู้เล่นกดปุ่มยิง การกิจหลักของคลาสนี้คือรับการป้อนข้อมูลจากผู้ใช้และอัปเดตสถานะรถถังผู้เล่นตามคำสั่งเหล่านี้



# SOURCE CODE (CLASS)

## Class Enemy

```
1 #ifndef ENEMY_H
2 #define ENEMY_H
3
4 #include "Tank.h"
5
6 // คลาส Enemy สืบทอดจาก Tank
7 // ใช้แทนตัวตุรุทั่วไปที่ควบคุมด้วยคอมพิวเตอร์ (AI)
8 class Enemy : public Tank {
9 protected:
10     int ทิศทางX; // ทิศทางปีจุบันที่ตัวรถถังเคลื่อนที่ (แกน X)
11     int ทิศทางY; // ทิศทางปีจุบันที่ตัวรถถังเคลื่อนที่ (แกน Y)
12 public:
13     // กำหนดค่าเริ่มต้นให้ Enemy: พลังชีวิต=1, พลังกระสุน=1, สัญลักษณ์ 'E'
14     Enemy(int startX, int startY);
15     // override ฟังก์ชัน updateAI จาก Tank
16     // การเคลื่อนที่และยิงแบบสุ่มของตัวตุรุ
17     void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) override;
18 }
19
20#endif
```

- คลาสนี้คืออะไร: คลาส Enemy เป็นคลาสสูกอีกหนึ่งคลาสของคลาส Tank ที่ใช้แทนรถถังศัตรูในเกม แต่ละวัตถุ Enemy จะเคลื่อนที่เองตามตระรักษ AI แบบง่าย (สุ่มหรือໄລ่ตามผู้เล่น) และทำหน้าที่ยิงกระสุนอัตโนมัติโดยมีสัญลักษณ์แสดงบนแผนที่ (เช่น E)
- ประเภทของคลาส: เป็นคลาสสูก (derived class) ของคลาส Tank
- สืบทอดมาจากคลาสใด: สืบทอดมาจากคลาส Tank (ดังนั้นมีตำแหน่งและพลังชีวิตเช่นเดียวกับคลาส Tank)
- บทบาทเฉพาะ: คลาส Enemy จะเขียนตระรักษสำหรับการเคลื่อนไหวแบบ AI เช่น เลื่อนไปยังตำแหน่งอื่นสุ่มหรือเคลื่อนเข้าหาผู้เล่น รวมถึงยิงกระสุนเมื่อมีโอกาส พังก์ซับการทำงานพื้นฐานอื่นๆ มาจากคลาส Tank ทำให้สามารถปรับเปลี่ยนพฤติกรรมสำหรับศัตรูได้โดยไม่ต้องเขียนโค้ดเคลื่อนที่และตรวจสอบใหม่ทั้งหมด



# SOURCE CODE (CLASS)

## Class Boss

```
#ifndef BOSS_H
#define BOSS_H

#include "Enemy.h"

// คลาส Boss สืบทอดจาก Enemy
// เป็นศัตรูในด่านสุดท้าย ยิงแรงกว่าและหนทางกว่า
class Boss : public Enemy {
public:
    // กำหนดค่าเริ่มต้นให้ Boss: พลังชีวิตมากกว่า (เช่น 3), พลังกระสุนแรงกว่า (2), สัญลักษณ์ 'B'
    Boss(int startX, int startY);
    // override ฟังก์ชัน AI สำหรับบอส
    void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) override;
};

#endif
```

- คลาสนี้คืออะไร: คลาส Boss เป็นคลาสพิเศษที่แทนรถถังbosในด่านสุดท้าย โดยรถถังคันนี้จะมีพลังชีวิตมากกว่าปกติ ต้องยิงหลายครั้งจึงถูกกำล่าย (แสดงโดยสัญลักษณ์ B)
- ประเภทของคลาส: เป็นคลาสลูก (derived class) ของคลาส Enemy หรือ Tank (เขียนกับคุณสมบัติของศัตรูก้าไปเพื่อเพิ่มพลังชีวิตและความยาก)
- สืบทอดมาจากคลาสใด: คาดว่าสืบทอดมาจากคลาส Enemy (ซึ่งเป็นคลาสลูกของ Tank) เพื่อใช้ฟังก์ชันกั้งหนดของศัตรูและเพิ่มพิเศษ เช่น พลังชีวิตจำนวนมากขึ้น
- บทบาทเฉพาะ: คลาส Boss อาจเพิ่มฟังก์ชันให้bosเคลื่อนที่หรือยิงแข็งแกร่งขึ้น เช่น เคลื่อนที่เร็วกว่า ยิงกระสุนถี่ขึ้น หรือมีรูปแบบการยิงพิเศษ คลาสนี้ทำหน้าที่เป็นศัตรูขั้นสุดท้ายที่มีคุณสมบัติพิเศษจาก Tank/Enemy พร้อมคุณสมบัติเพิ่มเติมเพื่อให้เป็นbosของเกม



# SOURCE CODE (CLASS)

## Class Bullet

```
1 #ifndef BULLET_H
2 #define BULLET_H
3
4 // คลาส Bullet แทนกระสุนที่ถูกยิง
5 class Bullet {
6 private:
7     int ตำแหน่งX;          // ตำแหน่งแกน X ของกระสุน
8     int ตำแหน่งY;          // ตำแหน่งแกน Y ของกระสุน
9     int ทิศทางX;          // ทิศทางการเคลื่อนที่ของกระสุน (แกน X)
10    int ทิศทางY;          // ทิศทางการเคลื่อนที่ของกระสุน (แกน Y)
11    int พลังกระสุน;        // ความแรงของกระสุน (ใช้ลดพลังชีวิตเป้าหมาย)
12    bool เป็นมิตร;         // ระบุว่ากระสุนนี้มาจากฝั่งผู้เล่นหรือไม่ (true = ผู้เล่น, false = ศัตรู)
13 public:
14     static const char สัญลักษณ์; // สัญลักษณ์แทนกระสุนบนแผนที่
15     // Constructor: กำหนดตำแหน่ง, ทิศทาง, ความแรง และฝ่ายของกระสุน
16     Bullet(int startX, int startY, int dirX, int dirY, int power, bool friendly)
17         : ตำแหน่งX(startX), ตำแหน่งY(startY), ทิศทางX(dirX), ทิศทางY(dirY),
18           พลังกระสุน(power), เป็นมิตร(friendly) {}
19     // ให้กระสุนเคลื่อนที่ไปข้างหน้า (ตามทิศทาง)
20     void moveForward();
21     // Getter ต่าง ๆ
22     int getX() const { return ตำแหน่งX; }
23     int getY() const { return ตำแหน่งY; }
24     int getPower() const { return พลังกระสุน; }
25     bool isFriendly() const { return เป็นมิตร; }
26 };
27
28 #endif
```

- คลาสนี้คืออะไร: คลาส Bullet ใช้แทนลูกกระสุนที่รถถังยิงออกไป (แสดงด้วยสัญลักษณ์ \*) แต่ละวัตถุ Bullet จะเคลื่อนที่ไปตามทิศทางที่ยิงออก และตรวจสอบการชนกับสิ่งกีดขวางหรือรถถังคันอื่น
- ประเภทของคลาส: เป็นคลาสหลัก (base class) หรือคลาสเดียวที่ไม่มีคลาสมี (ไม่ได้สืบทอดจากคลา索ื่นโดยเด็ดขาด)
- ฟังก์ชันหลัก: จัดการการอัปเดตตำแหน่งของกระสุนในแต่ละเฟรม (ขับกระสุนตามทิศทาง) ตรวจจับการชน (หากชนกับกำแพงหรือรถถังจะทำให้รถถังหายไปหรือลดพลังชีวิต และทำลายวัตถุกระสุนตัวเอง) และลบวัตถุกระสุนเมื่อออกนอกขอบเขตจอกภาพ
- คลาสมีทำหน้าที่พื้นฐานอะไรบ้าง: (Bullet ไม่มีคลาสมี) คลาสนี้ให้ฟังก์ชันพื้นฐานของกระสุน เช่น การเคลื่อนที่อย่างรวดเร็วและตรวจสอบกับวัตถุในจอก เพื่อให้รถถังต่างๆ ใช้สร้างกระสุนได้โดยไม่ต้องเขียนโค้ดเคลื่อนที่ใหม่



# SOURCE CODE (CLASS)

## Class Map

```
1 #ifndef MAP_H
2 #define MAP_H
3
4 #include <cstdlib> // ส่าหรับ rand()
5 #include <ctime> // ส่าหรับ time()
6
7 // คลาส Map ส่าหรับแผนที่เกม
8 class Map {
9 private:
10     static const int ความกว้าง = 30; // ความกว้างของแผนที่ (จำนวนคอลัมน์)
11     static const int ความสูง = 20; // ความสูงของแผนที่ (จำนวนแถว)
12     char grid[ความสูง][ความกว้าง]; // เมทริกซ์ของแผนที่ (# กำแพง, ' ' พื้นที่ว่าง)
13 public:
14     Map() {
15         // Constructor อาจไม่ต้องทำอะไร เพราจะเรียก generate ก่อนใช้แผนที่
16     }
17     int getWidth() const { return ความกว้าง; }
18     int getHeight() const { return ความสูง; }
19     bool isWall(int x, int y) const {
20         return grid[y][x] == '#';
21     }
22     char getCell(int x, int y) const {
23         return grid[y][x];
24     }
25     void setCell(int x, int y, char value) {
26         grid[y][x] = value;
27     }
28     // สร้างแผนที่ใหม่ส่าหรับต่าน โดยสุ่มวางกำแพง
29     void generate(int level);
30 };
31
32 #endif
```

- คลาสนี้คืออะไร: คลาส Map เป็นคลาสหลักสำหรับจัดการแผนที่ของเกม ซึ่งกำหนดตารางขนาด  $30 \times 20$  พร้อมสิ่งกีดขวาง (กำแพงอีชูและด้วย #) และพื้นที่ว่าง เมื่อเริ่มแต่ละต่าน คลาสนี้จะสุ่มสร้างตำแหน่งกำแพง และเก็บข้อมูลสถานะของพื้นที่แต่ละช่องไว้
- ประเภทของคลาส: เป็นคลาสหลัก (base class) ใช้งานโดยคลาส Game ในการสร้างและวัดแผนที่
- ฟังก์ชันหลัก: มีหน้าที่สร้างแผนที่ (วางแผนแบบสุ่ม) และมีเมธอดในการแสดงผลแผนที่บนหน้าจอ (วัดกรอบสนามและกำแพง) รวมทั้งเก็บข้อมูลว่าช่องใดเป็นกำแพงหรือว่าง คลาสอื่นๆ เช่น Tank, Bullet จะตรวจสอบกับกำแพงเหล่านี้ผ่านคลาส Map
- คลาสมีหน้าที่พื้นฐานอะไรบ้าง: (Map ไม่มีคลาสมี) ฟังก์ชันหลักคือ การสร้างและอัปเดตตำแหน่งสิ่งกีดขวางบนแผนที่ เช่น การสุ่มวางกำแพงใหม่ในแต่ละต่าน และให้ข้อมูลพื้นฐานของแมปแก่คลาสเกมและวัตถุต่างๆ เพื่อใช้ตรวจสอบและวัดจาก



# SOURCE CODE (CLASS)

## Class Game

```
1 #ifndef GAME_H
2 #define GAME_H
3
4 #include <vector>
5 #include "Map.h"
6 #include "Player.h"
7 #include "Enemy.h"
8 #include "Boss.h"
9 #include "Bullet.h"
10
11 // คลาส Game สำหรับควบคุมการเล่นเกมทั้งหมด
12 class Game {
13 private:
14     Map แผนที่;           // แผนที่ของเกม
15     Player ผู้เล่น;        // ผู้เล่น (รถถังศึ่งของผู้เล่น)
16     std::vector<Tank*> ศัตรูทั้งหมด; // รายชื่อศัตรูทั้งหมด (Enemy และ Boss) ใช้ Tank* เพื่อ polymorphism
17     std::vector<Bullet> กระสุนทั้งหมด; // รายชื่อกระสุนทั้งหมดในเกม
18     int ถ่านปั๊บัน;       // หมายเลขถ่านปั๊บัน (1-3)
19     bool จบเกม;          // สลากะเมื่อจบและหรือไม่
20 public:
21     Game();
22     ~Game();
23     // เริ่มเกม (เล่นทุกค่าถ่านจนจบหรือจบผ่านด้วย)
24     void run();
25 private:
26     // ฟังก์ชันช่วยในการสร้างศัตรู
27     void addEnemy(int x, int y);
28     void addBoss(int x, int y);
29     // ลบศัตรูทั้งหมด (เรียกตอนจบค่าถ่านหรือจบเกม เพื่อคืนหน่วยความจำ)
30     void clearEnemies();
31     // โหลดข้อมูลและตั้งค่าใหม่สำหรับค่าถ่านที่กำหนด
32     void loadLevel(int level);
33     // โหลดสถานะปั๊บันของเกม (แผนที่, รถถัง, กระสุน) ลงบนหน้าจอ
34     void draw();
35 };
36
37 #endif
```

- คลาสนี้คืออะไร: คลาส Game เป็นคลาสหลักของเกม ซึ่งทำหน้าที่ควบคุม การทำงานหลักทั้งหมดของเกม รวมถึงการเริ่มเกม จัดการลูปของเกม การอ่านค่าจากผู้ใช้ และเรียกใช้งานการวางแผนหน้าจอและอัปเดตวัตถุต่างๆ ภายในเกม
- ประเภทของคลาส: เป็นคลาสหลัก (base class) ไม่ได้สืบทอดมาจากคลาสอื่น
- ฟังก์ชันหลัก: ทำหน้าที่เริ่มเกมและจบเกม ควบคุมการทำงานของเกมแต่ละด้าน เช่น สร้างแผนที่ใหม่ จัดการจำนวนศัตรู เรียกให้รถถังเคลื่อนที่ ยิงกระสุน และตรวจสอบการชนระหว่างวัตถุต่างๆ รวมถึงการบรรลุเงื่อนไขเมื่อผู้เล่นชนะหรือแพ้เกม
- ทำหน้าที่พื้นฐานอะไรบ้าง: (สำหรับคลาสหลัก ก่อนมีคลาสมี) คลาส Game มีหน้าที่พื้นฐานของเกมโดยรวม เช่น การวนลูปอัปเดตสถานะเกม การจัดการด้าน และการเรียกใช้งานสถานะเมื่อเริ่มด้านใหม่ ซึ่งคลาสลูกอื่นๆ (เช่น Player, Enemy) จะถูกใช้งานภายในคลาส Game



# GET CONNECTED WITH US



นาย ศิรภพ สักธิวงศ์ 67340500074  
นาย วนิช แซ่ย় 67340500077



# THANK YOU

