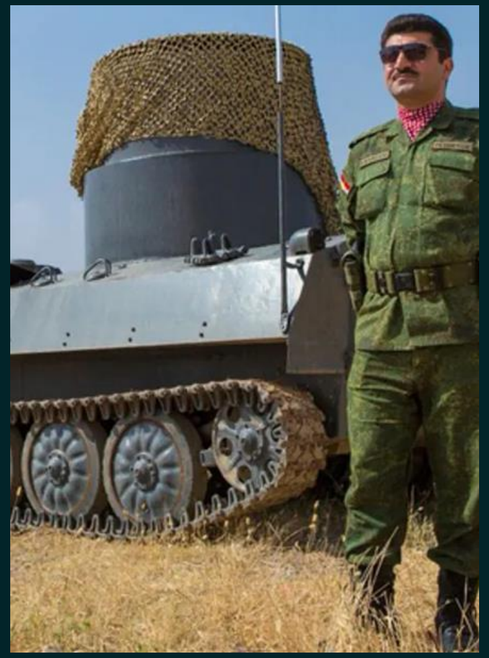
TANK TAEK

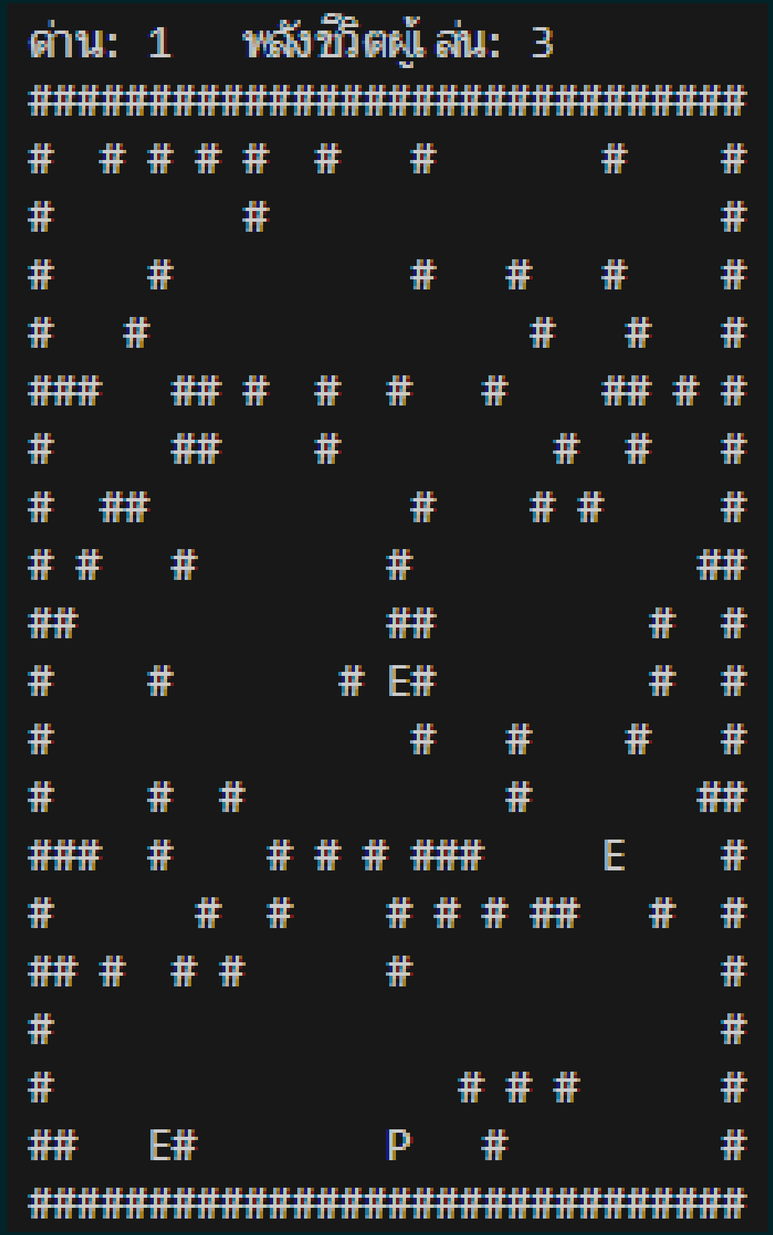


list OF CONTENTS

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

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

Source Code (Class) ---------- Slide 5-10



OVERVIEW

แนวคิดหลัก

เกม Tank Taek เป็นเกมยิงรถถังสองมิติแนวคลาสสิก ผู้เล่นควบคุมรถถัง (P) เพื่อทำลายรถถังศัตรูทั้งหมดในแต่ละด่าน เป้าหมายคือผ่านด่านต่างๆ ไปจนเจอบอสสุดท้ายที่แข็งแกร่งกว่าปกติ

แผนที่แต่ละด่านเป็นกริดอักขระ 30x20 มีป้อมและสิ่งกีดขวาง (#) วางแบบสุ่ม ผู้เล่นมี 3 ชีวิต แต่ละด่านชีวิตจะฟื้นกลับมาเต็มเมื่อผ่านด่านไป. เกมแสดงผลด้วยตัวอักษร ASCII บนหน้าจอคอนโซล 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 (*) เมื่อชนรถถังจะทำลายทันที และกระสุนจะหายไปเมื่อชนกำแพงหรือออกนอกเขต
- มีระบบด่านอัตโนมัติ 3 ด่าน ด่านสุดท้ายจะมี บอส (B) ซึ่งมีพลังชีวิตมากกว่าปกติ ต้องยิงหลายครั้งจึงจะทำลาย
- เมื่อผู้เล่นถูกยิงจนชีวิตหมด เกมจบ (แสดง “จบเกม! ผู้เล่นแพ้”) และเมื่อทำลายบอสได้สำเร็จ แสดงข้อความ “ยินดีด้วย คุณชนะเกมครบทุกด่าน!”

ข้อกำหนดของระบบ

Non-Functional Requirements

- ภาษาและเทคโนโลยี: พัฒนาโดยใช้ภาษา C++ ทั้งหมด และใช้งานบนคอนโซล (console) โดยไม่มีกราฟิกพิเศษ ใช้ไลบรารี <conio.h> สำหรับรับคีย์บอร์ดใน Windows
- ประสิทธิภาพ: วงลูปหลักรันได้ราบรื่นประมาณ 10 เฟรมต่อวินาที (หน่วง ~100ms) เพื่อให้การเคลื่อนที่ของรถถังและกระสุนเห็นได้ชัด และการคำนวณตำแหน่ง-การชนทำได้ในเวลาเชิงเส้น (O(n)) ซึ่งเพียงพอสำหรับจำนวนวัตถุตอนเล่น
- ความเข้ากันได้: เน้นรันบนระบบ Windows (ใช้คอนโซล Windows) เนื่องจากใช้ <conio.h> การแสดงผลภาษาไทยต้องรองรับ UTF-8 แต่โค้ดทั้งหมดจัดเก็บเป็น UTF-8 เพื่อภาษาไทยที่ถูกต้อง
- คุณภาพโค้ด: จัดโค้ดออกเป็นหลายไฟล์ตามหน้าที่ (เช่น Game, Tank, Player, Enemy, Map, Bullet, main) เพื่อให้อ่านง่ายและบำรุงรักษา โดยใช้ชื่อตัวแปร เมธอด และคอมเมนต์เป็นภาษาไทยชัดเจน
- ความเสถียร: โปรแกรมต้องทำงานได้นานโดยไม่ค้างหรือเด้งออกเอง มีการจัดการหน่วยความจำถูกต้อง (จัดสรร/ลบ (new/delete) สำหรับวัตถุที่สร้างไดนามิก เช่น รถถังศัตรู) เพื่อป้องกัน memory leak

SOURCE CODE (CLASS)

Class Tank

```cpp
#ifndef TANK_H
#define TANK_H

#include <vector>
#include <iostream>
#include <string>
class Map;       // forward declare Map (used in method parameters)
class Bullet;    // forward declare Bullet
class Tank {
protected:
    int ตำแหน่งX;         // ตำแหน่งแกน X ของรถถัง
    int ตำแหน่งY;         // ตำแหน่งแกน Y ของรถถัง
    int พลังชีวิต;        // พลังชีวิตของรถถัง (HP ของรถถัง)
    int พลังกระสุน;       // ความแรงของกระสุนที่ยิงออก
    char ตัวอักษร;        // ตัวอักษรที่ใช้แทนรถถังบนแผนที่
public:
    Tank(int x, int y, int hp, int bulletPower, char symbol)
        : ตำแหน่งX(x), ตำแหน่งY(y), พลังชีวิต(hp), พลังกระสุน(bulletPower), ตัวอักษร(symbol) {}
    virtual ~Tank() {}   // destructor ควรเป็น virtual (เพื่อให้ลบถูกชนิดเมื่อ delete ผ่าน pointer คลาสแม่)

    // Getter สำหรับค่าต่าง ๆ (เป็นการ encapsulation)
    int getX() const { return ตำแหน่งX; }
    int getY() const { return ตำแหน่งY; }
    int getHP() const { return พลังชีวิต; }
    char getSymbol() const { return ตัวอักษร; }
    int getBulletPower() const { return พลังกระสุน; }

    // ลดพลังชีวิตเมื่อโดนยิง (damage)
    void ลดพลังชีวิต(int damage) {
        พลังชีวิต -= damage;
        if (พลังชีวิต < 0) พลังชีวิต = 0;
    }
    bool isDestroyed() const {
        return พลังชีวิต <= 0;
    }

    // ฟังก์ชันเคลื่อนที่รถถัง 1 ก้าว (dx, dy) ถ้าไม่ชนกำแพงหรือตัวอื่น
    bool move(int dx, int dy, Map &map, const std::vector<Tank*> &tanks);

    // ฟังก์ชัน virtual สำหรับ AI ของศัตรู (ผู้เล่นจะไม่ override ฟังก์ชันนี้)
    // ใช้ polymorphism เพื่อให้ Enemy/Boss มีพฤติกรรมต่างกัน
    virtual void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) {
        // ค่า default ไม่ทำอะไร (สำหรับ Player ซึ่งควบคุมด้วยผู้เล่นเอง)
    }
};

#endif
```

- คลาส Tank เป็นคลาสฐาน (base class) สำหรับรถถังทั้งหมดในเกม ซึ่งนิยามคุณสมบัติพื้นฐานของรถถังทั่วไป เช่น พิกัดตำแหน่ง ทิศทางที่หันหน้า และพลังชีวิต (HP) คลาสนี้ใช้วาดตัวรถถังบนหน้าจอและจัดการการเคลื่อนไหวพื้นฐาน 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

- ทำหน้าที่พื้นฐานอะไรบ้าง: (คลาส Tank เองเป็นคลาสหลัก) คลาส Tank จัดการการเคลื่อนที่และสถานะพื้นฐานของรถถัง เช่น วิธีการเคลื่อนที่ไปยังตำแหน่งใหม่ การเช็คชน (collision) กับอุปสรรค หรือการลดพลังชีวิตเมื่อตกกระสุน ทำให้คลาสลูกสามารถนำวิธีเหล่านี้ไปใช้ได้ทันทีหรือปรับแต่งเพิ่มเติมภายหลัง

SOURCE CODE

Class Player

```cpp
#ifndef PLAYER_H
#define PLAYER_H

#include "Tank.h"

// คลาส Player สืบทอดจาก Tank (คลาสแม่)
// ใช้แทนผู้เล่นที่ควบคุมโดยคนเล่น
class Player : public Tank {
private:
    int ทิศทางX; // ทิศทางล่าสุดที่ผู้เล่นหัน (แกน X)
    int ทิศทางY; // ทิศทางล่าสุดที่ผู้เล่นหัน (แกน Y)
public:
    // Constructor: กำหนดตำแหน่งเริ่มต้น, พลังชีวิต=3, พลังกระสุน=1, สัญลักษณ์ 'P'
    Player(int startX, int startY);
    // Setter และ Getter สำหรับทิศทางที่หันของผู้เล่น
    void setDirection(int dx, int dy) {
        ทิศทางX = dx;
        ทิศทางY = dy;
    }
    int getDirX() const { return ทิศทางX; }
    int getDirY() const { return ทิศทางY; }
};

#endif
```

- คลาสนี้คืออะไร: คลาส Player เป็นคลาสลูกที่สร้างจากคลาส Tank เพื่อแทนรถถังของผู้เล่น คลาสนี้จะมีคุณสมบัติเพิ่มเติมเฉพาะสำหรับผู้เล่น เช่น การรับค่าปุ่มจากคีย์บอร์ด (W/A/S/D) เพื่อควบคุมการเคลื่อนที่ การจำกัดจำนวนชีวิต รวมถึงการตรวจจับการยิงกระสุนจากผู้เล่น (โดยปุ่ม Space)
- ประเภทของคลาส: เป็นคลาสลูก (derived class) ของคลาส Tank
- สืบทอดมาจากคลาสใด: สืบทอดมาจากคลาส Tank (ได้รับคุณสมบัติพื้นฐานของรถถังเช่น ตำแหน่ง และพลังชีวิตมาจาก Tank)
- บทบาทเฉพาะ: คลาส Player อาจเพิ่มฟังก์ชันพิเศษ เช่น ตรวจสอบสถานะชีวิตของผู้เล่น (จำนวนชีวิตที่กำหนดไว้ในเกม) และส่งการยิงกระสุนเมื่อผู้เล่นกดปุ่มยิง ภารกิจหลักของคลาสนี้คือรับการป้อนข้อมูลจากผู้ใช้และอัปเดตสถานะรถถังผู้เล่นตามคำสั่งเหล่านั้น

SOURCE CODE (CLASS)

Class Enemy

```cpp
#ifndef ENEMY_H
#define ENEMY_H

#include "Tank.h"

// คลาส Enemy สืบทอดจาก Tank
// ใช้แทนศัตรูทั่วไปที่ควบคุมด้วยคอมพิวเตอร์ (AI)
class Enemy : public Tank {
protected:
    int ทิศทางX; // ทิศทางปัจจุบันที่ศัตรูกำลังเคลื่อนที่ (แกน X)
    int ทิศทางY; // ทิศทางปัจจุบันที่ศัตรูกำลังเคลื่อนที่ (แกน Y)
public:
    // กำหนดค่าเริ่มต้นให้ Enemy: พลังชีวิต=1, พลังกระสุน=1, สัญลักษณ์ 'E'
    Enemy(int startX, int startY);
    // override ฟังก์ชัน updateAI จาก Tank
    // การเคลื่อนที่และยิงแบบสุ่มของศัตรู
    void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) override;
};

#endif
```

- คลาสนี้คืออะไร: คลาส Enemy เป็นคลาสลูกอีกหนึ่งคลาสของคลาส Tank ที่ใช้แทนรถถังศัตรูในเกม แต่ละวัตถุ Enemy จะเคลื่อนที่เองตามตรรกะ AI แบบง่าย (สุ่มหรือไล่ตามผู้เล่น) และทำหน้าที่ยิงกระสุนอัตโนมัติ โดยมีสัญลักษณ์แสดงบนแผนที่ (เช่น E)
- ประเภทของคลาส: เป็นคลาสลูก (derived class) ของคลาส Tank
- สืบทอดมาจากคลาสใด: สืบทอดมาจากคลาส Tank (ดังนั้นมีตำแหน่งและพลังชีวิตเช่นเดียวกับคลาส Tank)
- บทบาทเฉพาะ: คลาส Enemy จะเขียนตรรกะสำหรับการเคลื่อนไหวแบบ AI เช่น เลื่อนไปยังตำแหน่งอื่นสุ่มหรือเคลื่อนเข้าหาผู้เล่น รวมถึงยิงกระสุนเมื่อมีโอกาส ฟังก์ชันการทำงานพื้นฐานอื่นๆ มาจากคลาส Tank ทำให้สามารถปรับเปลี่ยนพฤติกรรมสำหรับศัตรูได้โดยไม่ต้องเขียนโค้ดเคลื่อนที่และตรวจชนใหม่ทั้งหมด

SOURCE CODE (CLASS)

Class Bullet

```cpp
#ifndef BULLET_H
#define BULLET_H

// คลาส Bullet แทนกระสุนที่ถูกยิง
class Bullet {
private:
    int ตำแหน่งX;        // ตำแหน่งแกน X ของกระสุน
    int ตำแหน่งY;        // ตำแหน่งแกน Y ของกระสุน
    int ทิศทางX;         // ทิศทางการเคลื่อนที่ของกระสุน (แกน X)
    int ทิศทางY;         // ทิศทางการเคลื่อนที่ของกระสุน (แกน Y)
    int พลังกระสุน;      // ความแรงของกระสุน (ใช้ลดพลังชีวิตเป้าหมาย)
    bool เป็นมิตร;       // ระบุว่ากระสุนนี้มาจากฝั่งผู้เล่นหรือไม่ (true = ผู้เล่น, false = ศัตรู)
public:
    static const char สัญลักษณ์; // สัญลักษณ์แทนกระสุนบนแผนที่
    // Constructor: กำหนดตำแหน่ง, ทิศทาง, ความแรง และฝ่ายของกระสุน
    Bullet(int startX, int startY, int dirX, int dirY, int power, bool friendly)
        : ตำแหน่งX(startX), ตำแหน่งY(startY), ทิศทางX(dirX), ทิศทางY(dirY),
          พลังกระสุน(power), เป็นมิตร(friendly) {}
    // ให้กระสุนเคลื่อนที่ไปข้างหน้า (ตามทิศทาง)
    void moveForward();
    // Getter ต่าง ๆ
    int getX() const { return ตำแหน่งX; }
    int getY() const { return ตำแหน่งY; }
    int getPower() const { return พลังกระสุน; }
    bool isFriendly() const { return เป็นมิตร; }
};

#endif
```

- คลาสนี้คืออะไร: คลาส Bullet ใช้แทนลูกกระสุนที่รถถังยิงออกไป (แสดงด้วยสัญลักษณ์ *) แต่ละวัตถุ Bullet จะเคลื่อนที่ไปตามทิศทางที่ยิงออก และตรวจสอบการชนกับสิ่งกีดขวางหรือรถถังคันอื่น
- ประเภทของคลาส: เป็นคลาสหลัก (base class) หรือคลาสเดี่ยวที่ไม่มีคลาสแม่ (ไม่ได้สืบทอดจากคลาสอื่นโดยเฉพาะ)
- ฟังก์ชันหลัก: จัดการการอัปเดตตำแหน่งของกระสุนในแต่ละเฟรม (ขยับกระสุนตามทิศทาง) ตรวจจับการชน (หากชนกับกำแพงหรือรถถังจะทำให้รถถังหายไปหรือลดพลังชีวิต และทำลายวัตถุกระสุนตัวเอง) และลบวัตถุกระสุนเมื่อออกนอกขอบเขตจอภาพ
- คลาสแม่ทำหน้าที่พื้นฐานอะไรบ้าง: (Bullet ไม่มีคลาสแม่) คลาสนี้ให้ฟังก์ชันพื้นฐานของกระสุน เช่น การเคลื่อนที่อย่างรวดเร็วและตรวจชนกับวัตถุในฉาก เพื่อให้รถถังต่างๆ ใช้สร้างกระสุนได้โดยไม่ต้องเขียนโค้ดเคลื่อนที่ใหม่

SOURCE CODE (CLASS)

Class Map

```cpp
#ifndef MAP_H
#define MAP_H

#include <cstdlib>  // สำหรับ rand()
#include <ctime>    // สำหรับ time()

// คลาส Map สำหรับแผนที่เกม
class Map {
private:
    static const int ความกว้าง = 30;  // ความกว้างของแผนที่ (จำนวนคอลัมน์)
    static const int ความสูง = 20;   // ความสูงของแผนที่ (จำนวนแถว)
    char grid[ความสูง][ความกว้าง];   // เมทริกซ์ของช่องแผนที่ (# กำแพง, ' ' พื้นที่ว่าง)
public:
    Map() {
        // Constructor อาจไม่ต้องทำอะไร เพราะจะเรียก generate ก่อนใช้แผนที่
    }
    int getWidth() const { return ความกว้าง; }
    int getHeight() const { return ความสูง; }
    bool isWall(int x, int y) const {
        return grid[y][x] == '#';
    }
    char getCell(int x, int y) const {
        return grid[y][x];
    }
    void setCell(int x, int y, char value) {
        grid[y][x] = value;
    }
    // สร้างแผนที่ใหม่สำหรับด่าน โดยสุ่มวางกำแพง
    void generate(int level);
};

#endif
```

- คลาสนี้คืออะไร: คลาส Map เป็นคลาสหลักสำหรับจัดการแผนที่ของเกม ซึ่งกำหนดตารางขนาด 30×20 พร้อมสิ่งกีดขวาง (กำแพงอิฐแสดงด้วย #) และพื้นที่ว่าง เมื่อเริ่มแต่ละด่าน คลาสนี้จะสุ่มสร้างตำแหน่งของกำแพงและเก็บข้อมูลสถานะของพื้นที่แต่ละช่องไว้
- ประเภทของคลาส: เป็นคลาสหลัก (base class) ใช้งานโดยคลาส Game ในการสร้างและวาดแผนที่
- ฟังก์ชันหลัก: มีหน้าที่สร้างแผนที่ (วางกำแพงแบบสุ่ม) และมีเมธอดในการแสดงผลแผนที่บนหน้าจอ (วาดกรอบสนามและกำแพง) รวมทั้งเก็บข้อมูลว่าช่องใดเป็นกำแพงหรือว่าง คลาสอื่นๆ เช่น Tank, Bullet จะตรวจการชนกับกำแพงเหล่านี้ผ่านคลาส Map
- คลาสแม่ทำหน้าที่พื้นฐานอะไรบ้าง: (Map ไม่มีคลาสแม่) ฟังก์ชันหลักคือการสร้างและอัปเดตตำแหน่งสิ่งกีดขวางบนแผนที่ เช่น การสุ่มวางกำแพงใหม่ในแต่ละด่าน และให้ข้อมูลพื้นฐานของแมปแก่คลาสเกมและวัตถุต่างๆ เพื่อใช้ตรวจชนและวาดฉาก

SOURCE CODE (CLASS)

Class Game

```cpp
#ifndef GAME_H
#define GAME_H

#include <vector>
#include "Map.h"
#include "Player.h"
#include "Enemy.h"
#include "Boss.h"
#include "Bullet.h"

// คลาส Game สำหรับควบคุมการเล่นเกมทั้งหมด
class Game {
private:
    Map แผนที่;                     // แผนที่ของเกม
    Player ผู้เล่น;                 // ผู้เล่น (รถถังคันของผู้เล่น)
    std::vector<Tank*> ศัตรูทั้งหมด; // รายชื่อศัตรูทั้งหมด (Enemy และ Boss) ใช้ Tank* เพื่อ polymorphism
    std::vector<Bullet> กระสุนทั้งหมด; // รายชื่อกระสุนทั้งหมดในเกม
    int ด่านปัจจุบัน;               // หมายเลขด่านปัจจุบัน (1-3)
    bool จบเกม;                     // สถานะว่าเกมจบแล้วหรือไม่
public:
    Game();
    ~Game();
    // เริ่มเกม (เล่นทุกด่านจนจบหรือจนผู้เล่นตาย)
    void run();
private:
    // ฟังก์ชันช่วยในการสร้างศัตรู
    void addEnemy(int x, int y);
    void addBoss(int x, int y);
    // ลบศัตรูทั้งหมด (เรียกตอนจบด่านหรือจบเกม เพื่อคืนหน่วยความจำ)
    void clearEnemies();
    // โหลดข้อมูลและตั้งค่าใหม่สำหรับด่านที่กำหนด
    void loadLevel(int level);
    // วาดสถานะปัจจุบันของเกม (แผนที่, รถถัง, กระสุน) ลงบนหน้าจอ
    void draw();
};

#endif
```

- คลาสนี้คืออะไร: คลาส Game เป็นคลาสหลักของเกม ซึ่งทำหน้าที่ควบคุมการทำงานหลักทั้งหมดของเกม รวมถึงการเริ่มเกม จัดการลูปของเกม การอ่านค่าจากผู้ใช้ และเรียกใช้งานการวาดหน้าจอและอัปเดตวัตถุต่างๆ ภายในเกม
- ประเภทของคลาส: เป็นคลาสหลัก (base class) ไม่ได้สืบทอดมาจากคลาสอื่น
- ฟังก์ชันหลัก: ทำหน้าที่เริ่มเกมและจบเกม ควบคุมการทำงานของเกมแต่ละด่าน เช่น สร้างแผนที่ใหม่ จัดการจำนวนศัตรู เรียกให้รถถังเคลื่อนที่ ยิงกระสุน และตรวจสอบการชนระหว่างวัตถุต่างๆ รวมถึงการบรรลุเงื่อนไขเมื่อผู้เล่นชนะหรือแพ้เกม
- ทำหน้าที่พื้นฐานอะไรบ้าง: (สำหรับคลาสหลัก ที่ไม่มีคลาสแม่) คลาส Game มีหน้าที่พื้นฐานของเกมโดยรวม เช่น การวนลูปอัปเดตสถานะเกม การจัดการด่าน และการรีเซ็ตสถานะเมื่อเริ่มด่านใหม่ ซึ่งคลาสลูกอื่นๆ (เช่น Player, Enemy) จะถูกใช้งานภายในคลาส Game

GET CONNECTED WITH US

นาย ศิรภพ สิทธิวงษ์
67340500074
นาย วนิช แซ่ยี่
67340500077

THANK YOU