



TANK TAEK



list OF CONTENTS

Overview ----- Slide 2

System requirement ----- Slide 3-4

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

[illegible]

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



ข้อกำหนดของระบบ

Functional Requirements

- รองรับผู้เล่นเดี่ยว ควบคุมรถถังได้ด้วยคีย์บอร์ด (W,A,S,D เคลื่อนที่ และ Space ยิงกระสุน)
- มีระบบศัตรูอัตโนมัติหลายคัน เคลื่อนที่และยิงแบบสุ่ม ผู้เล่นต้องหลบเลี้ยว กระสุนของศัตรู
- 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: วงรูปหลักรันได้ราบรื่นประมาณ 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: โปรแกรมต้องทำงานได้นานโดยไม่ค้างหรือเด็งออกเอง 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 สำหรับค่าต่าง ๆ (เป็นการ 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     void ลดพลังชีวิต(int damage) {
30         พลังชีวิต -= damage;
31         if (พลังชีวิต < 0) พลังชีวิต = 0;
32     }
33     bool isDestroyed() const {
34         return พลังชีวิต <= 0;
35     }
36
37     // ฟังก์ชันเคลื่อนที่รถถัง 1 ก้าว (dx, dy) ถ้าไม่ชนกำแพงหรือตัวอื่น
38     bool move(int dx, int dy, Map &map, const std::vector<Tank*> &tanks);
39
40     // ฟังก์ชัน virtual สำหรับ AI ของศัตรู (ผู้เล่นจะไม่ override ฟังก์ชันนี้)
41     // ใช้ polymorphism เพื่อให้ Enemy/Boss มีพฤติกรรมต่างกัน
42     virtual void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) {
43         // ค่า default ไม่ทำอะไร (สำหรับ Player ซึ่งควบคุมด้วยผู้เล่นเอง)
44     }
45 };
46
47 #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 การเช็คนชน (collision) กับอุปสรรค หรือการลดพลังชีวิตเมื่อตกกระสุน ทำให้คลาสลูกสามารถนำวิธีเหล่านี้ไปใช้ได้ทันทีหรือปรับแต่งเพิ่มเติมภายหลัง



SOURCE CODE (CLASS)

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; // ทิศทางล่าสุดที่ผู้เล่นหัน (แกน X)
11     int ทิศทางY; // ทิศทางล่าสุดที่ผู้เล่นหัน (แกน Y)
12 public:
13     // Constructor: กำหนดตำแหน่งเริ่มต้น, พลังชีวิต=3, พลังกระสุน=1, สัญลักษณ์ 'P'
14     Player(int startX, int startY);
15     // Setter และ Getter สำหรับทิศทางที่หันของผู้เล่น
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 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 ภารกิจหลักของคลาสนี้คือการรับการป้อนข้อมูลจากผู้ใช้และอัปเดตสถานะรถถังผู้เล่นตามคำสั่งเหล่านั้น



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  // ใช้แทนศัตรูทั่วไปที่ควบคุมด้วยคอมพิวเตอร์ (AI)
8  class Enemy : public Tank {
9  protected:
10     int ทิศทางX; // ทิศทางปัจจุบันที่ศัตรูกำลังเคลื่อนที่ (แกน X)
11     int ทิศทางY; // ทิศทางปัจจุบันที่ศัตรูกำลังเคลื่อนที่ (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     // การเคลื่อนที่และยิงแบบสุ่มของศัตรู
17     void updateAI(Map &map, std::vector<Tank*> &tanks, std::vector<Bullet> &bullets) override;
18 };
19
20 #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

```
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
29
```

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



SOURCE CODE (CLASS)

Class Map

```
1  #ifndef MAP_H
2  #define MAP_H
3
4  #include <cstdlib> // สำหรับ rand()
5  #include <ctime>   // สำหรับ time()
6
7  // คลาส Map สำหรับแผนที่เกม
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]; // เมทริกซ์ของช่องแผนที่ (# กำแพง, ' ' พื้นที่ว่าง)
13 public:
14     Map() {
15         // Constructor อาจไม่ต้องทำอะไร เพราะจะเรียก 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     // สร้างแผนที่ใหม่สำหรับด่าน โดยสุ่มวางกำแพง
29     void generate(int level);
30 };
31
32 #endif
```

- คลาสนี้คืออะไร: คลาส Map เป็นคลาสหลักสำหรับการจัดการแผนที่ของเกม 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 จะตรวจสอบการชนกับกำแพงเหล่านี้ผ่านคลาส Map
- คลาสแม่ทำหน้าที่พื้นฐานอะไรบ้าง: (Map ไม่มีคลาสแม่) ฟังก์ชันหลักคือการสร้างและอัปเดตตำแหน่งสิ่งกีดขวางบนแผนที่ เช่น การสุ่มวางกำแพงใหม่ในแต่ละด่าน และให้ข้อมูลพื้นฐานของแมปแก่คลาสเกมและวัตถุต่างๆ เพื่อใช้ตรวจสอบและวาดฉาก



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 (รถถังคันของผู้เล่น)
16     std::vector<Tank*> ศัตรูทั้งหมด; // รายชื่อศัตรูทั้งหมด (Enemy และ Boss) ใช้ Tank* เพื่อ polymorphism
17     std::vector<Bullet> กระสุนทั้งหมด; // รายชื่อกระสุนทั้งหมดในเกม
18     int ด้านปัจจุบัน;     // หมายเลขด้านปัจจุบัน (1-3)
19     bool จบเกม;          // สถานะว่าเกมจบแล้วหรือไม่
20 public:
21     Game();
22     ~Game();
23     // เริ่มเกม (เล่นทุกด้านจนจบหรือจนผู้เล่นตาย)
24     void run();
25 private:
26     // ฟังก์ชันช่วยในการสร้างศัตรู
27     void addEnemy(int x, int y);
28     void addBoss(int x, int y);
29     // ลบศัตรูทั้งหมด (เรียกตอนจบด้านหรือจบเกม เพื่อคืนหน่วยความจำ)
30     void clearEnemies();
31     // โหลดข้อมูลและตั้งค่าใหม่สำหรับด้านที่กำหนด
32     void loadLevel(int level);
33     // วาดสถานะปัจจุบันของเกม (แผนที่, รถถัง, กระสุน) ลงบนหน้าจอ
34     void draw();
35 };
36
37 #endif
38
```

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



GET CONNECTED WITH US



นาย ศิรภพ สิทธิวงษ์

67340500074

นาย วณิช แซ่यी

67340500077



THANK YOU

