

Project Game OOP

Terrorist Hunt Game

บทที่ 1 บทนำ

ที่มาและความสำคัญของโปรเจ็ค

เกมนี้นี้พัฒนาขึ้นเพื่อตอบสนองความต้องการของผู้เล่นที่ชื่นชอบการผจญภัย โดยนำเสนอรูปแบบเกมแนวแอคชั่นที่ต้องใช้ทักษะการควบคุมเครื่องบินและการเล็งเป้าหมาย การออกแบบเกมถูกเน้นไปที่การเพิ่มระดับความยากและการตอบสนองต่อการโจมตีของผู้เล่น รวมถึงการจำลองสถานการณ์การต่อสู้ในอากาศที่สนุกสนาน

ประเภทของโครงการ

โครงการเกม 2D ที่พัฒนาโดยใช้ภาษา Java พร้อม GUI และการควบคุมเหตุการณ์

ประโยชน์

- ช่วยเสริมสร้างทักษะการควบคุมและการตอบสนอง
- ช่วยพัฒนาความคิดสร้างสรรค์ผ่านการออกแบบและการพัฒนาเกม
- ช่วยให้ผู้พัฒนาฝึกทักษะการเขียนโค้ด OOP

ขอบเขตของโครงการ

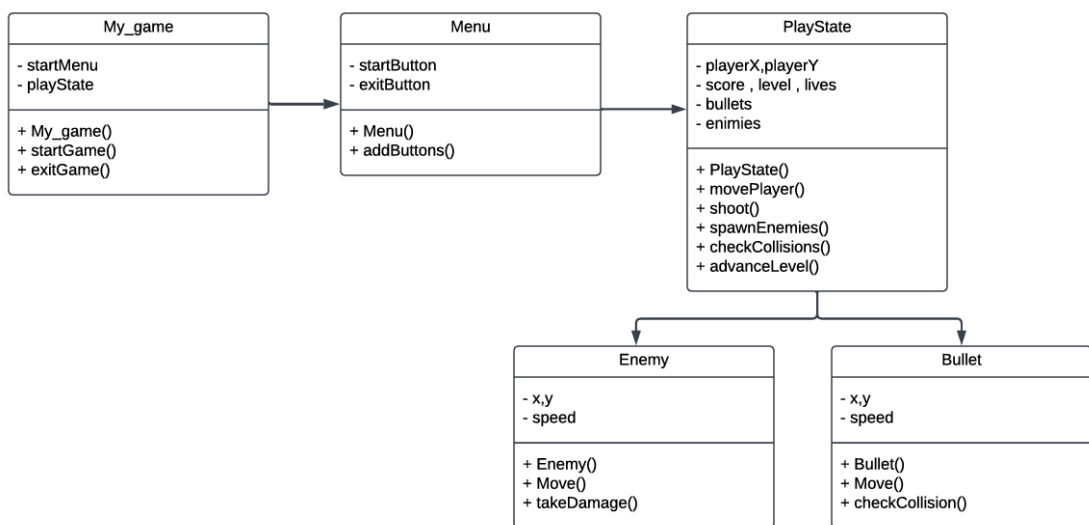
- มีฉากเริ่มต้นที่มีปุ่มให้เลือกเล่นเกม (Start) และออกจากเกม (Exit)
- เมื่อเริ่มเกม ผู้เล่นจะควบคุมเครื่องบินเพื่อยิงศัตรู
- มีระดับความยากที่เพิ่มขึ้นตามฉาก
- เกมจะจบเมื่อผู้เล่นเสียชีวิต

บทที่ 2 ส่วนการพัฒนา

เนื้อเรื่องย่อหรือวิธีการเล่น

ในเกมนี้ ผู้เล่นจะรับบทเป็นนักบินที่มีภารกิจคือกำจัดผู้ก่อการร้ายในแต่ละฉาก โดยในแต่ละฉากจะมีจำนวนศัตรูที่เพิ่มขึ้นและระดับความยากที่ท้าทายมากขึ้น การควบคุมจะเป็นการเคลื่อนเครื่องบิน ซ้าย-ขวา ขึ้น-ลง และกด Space เพื่อยิงกระสุน

Class Diagram



คำอธิบายคลาส

1. My_game (JFrame):

คลาสหลักที่สร้างหน้าต่างของเกมและควบคุมการเปลี่ยนแปลงระหว่างเมนูและเกม

Attributes:

Menu startMenu : เมนูเริ่มต้นของเกม

PlayState playState : สถานะการเล่นเกม

Methods:

My_game(): คอนสตรัคเตอร์ที่ใช้สร้างหน้าต่างเกม

startGame(): เริ่มต้นการเล่นเกม

exitGame(): ออกจากเกม

2. Menu (JPanel):

คลาสที่แสดงเมนูเริ่มต้นของเกม

Attributes:

JButton startButton: ปุ่มเริ่มเกม

JButton exitButton: ปุ่มออกจากเกม

Methods:

Menu(): คอนสตรัคเตอร์ที่ใช้สร้างเมนู

addButtons(): เพิ่มปุ่มเริ่มเกมและปุ่มออกจากเกม

3. PlayState (JPanel):

คลาสที่ควบคุมการเล่นเกมจริง ๆ ซึ่งรวมถึงการเคลื่อนที่ของ
เครื่องบิน, การยิงกระสุน, การสร้างศัตรู

Attributes:

int playerX, playerY: ตำแหน่งของเครื่องบิน

int score, level: คะแนนและระดับของเกม

List<Bullet> bullets: รายการของกระสุนที่ยิงออกมา

List<Enemy> enemies: รายการของศัตรู

Methods:

PlayState(): คอนสตรัคเตอร์ที่ใช้สร้างหน้าจอการเล่นเกม

movePlayer(): เคลื่อนที่เครื่องบิน

shoot(): ยิงกระสุน

spawnEnemies(): สร้างศัตรูแบบสุ่ม

checkCollisions(): ตรวจสอบการชนกันระหว่างผู้เล่นและกระสุนและศัตรู

advanceLevel(): เพิ่มระดับความยากของเกม

4. Bullet:

คลาสที่จัดการกระสุน

Attributes:

int x, y: ตำแหน่งของกระสุน

int speed: ความเร็วในการเคลื่อนที่ของกระสุน

Methods:

Bullet(): คอนสตรัคเตอร์ที่ใช้สร้างกระสุน

move(): เคลื่อนที่กระสุน

checkCollision(): ตรวจสอบการชนกับศัตรู

5. Enemy:

คลาสที่ดูแลการจัดการศัตรู

Attributes:

int x, y: ตำแหน่งของศัตรู

int health: ความแข็งแรงของศัตรู

int speed: ความเร็วในการเคลื่อนที่ของศัตรู

Methods:

Enemy(): คอนสตรัคเตอร์ที่ใช้สร้างศัตรู

move(): เคลื่อนที่ของศัตรู

takeDamage(): ลดพลังชีวิตของศัตรูเมื่อถูกโจมตี

รูปแบบการพัฒนา Application / Applet

เกมนี้ถูกพัฒนาในรูปแบบ Java Swing Application โดยใช้ JFrame และ JPanel เป็นส่วนประกอบหลักสำหรับ GUI เกมจะเปิดในหน้าต่าง JFrame และรองรับการเคลื่อนไหวและการทำงานต่าง ๆ ผ่านการฟังเหตุการณ์ (Event Handling) โดยใช้งาน KeyListener และ ActionListener

อธิบายส่วนของโปรแกรมที่มี

Constructor:

1. ในคลาส My_game มีการสร้าง Constructor เพื่อกำหนดค่าเริ่มต้นสำหรับเฟรม (frame) และเพิ่มเมนูเริ่มต้น startMenu ไปที่เฟรม รวมถึงตั้งค่า ActionListener ให้กับปุ่ม Start และ Exit เพื่อควบคุมการสลับระหว่างหน้าต่าง Menu และ PlayState หรือออกจากเกมเมื่อกดปุ่ม Exit

```

1  My_game() {
2
3      this.add(startMenu);
4
5      // Add an ActionListener to the "Start" button
6      startMenu.Start.addActionListener(new ActionListener() {
7          public void actionPerformed(ActionEvent e) {
8              // When the "Start" button is clicked, switch to PlayState
9              changeToPlayState();
10         }
11     });
12
13     // Add an ActionListener to the "Exit" button in PlayState
14     startMenu.Exit.addActionListener(new ActionListener() {
15         public void actionPerformed(ActionEvent e) {
16             // When the "Exit" button in PlayState is clicked, exit the game
17             System.exit(0);
18         }
19     });
20 }

```

2. Constructor ของคลาส Menu ถูกใช้งานเพื่อกำหนดค่าเริ่มต้นสำหรับ UI โดยในส่วนนี้มีการตั้งค่า Layout และตำแหน่งของปุ่ม Start และ Exit เมื่อเรียกสร้างวัตถุจากคลาส Menu (ผ่าน new Menu()), constructor นี้จะถูกเรียกใช้งานทันทีเพื่อเตรียมการตั้งค่าให้กับ UI

```

1  Menu() {
2
3      setLayout(null);
4
5      // Set Button Location
6      this.add(Start);
7      Start.setBounds(180, 200, 387, 67);
8      this.add(Exit);
9      Exit.setBounds(210, 300, 337, 52);
10
11 }

```

3. ในคลาส PlayState มีการใช้งาน Constructor เพื่อกำหนดค่าเริ่มต้นสำหรับ UI, เปิดใช้งาน focus, และเริ่มต้นเกม โดยการตั้งค่าปุ่ม restartButton, exitButton และ changeBackgroundButton เพื่อควบคุมการทำงานของเกม รวมถึงเริ่มการทำงานของ game loop ผ่าน startGameLoop

```

1  public PlayState() {
2      setupUI();
3      setFocusable(true);
4      requestFocus();
5      startGameLoop();
6  }

```

Encapsulation:

1. การใช้งาน private ในการประกาศตัวแปร starts และ exits ทำให้ตัวแปรเหล่านี้ไม่สามารถเข้าถึงโดยตรงจากภายนอกคลาส Menu เพื่อให้การใช้งานภายในคลาสนี้เท่านั้น ตัวแปร Start และ Exit ประกาศเป็น public เพื่อให้สามารถเข้าถึงได้จากภายนอกคลาสได้เช่นกัน

```

1 // Start&Exit button
2 private ImageIcon starts = new ImageIcon(this.getClass().getResource("start.png"));
3 public JButton Start = new JButton(starts);
4
5 private ImageIcon exits = new ImageIcon(this.getClass().getResource("exit.png"));
6 public JButton Exit = new JButton(exits);
7

```

2. การกำหนดตัวแปร private หลายตัว เช่น playerX, playerY, enemies, bullets ซึ่งจำกัดการเข้าถึงข้อมูลจากภายนอกคลาส PlayState ช่วยให้สามารถควบคุมการแก้ไขข้อมูลภายในคลาสได้ง่ายขึ้น

```

1 private int playerX = 300;
2 private int playerY = 450;
3
4 private int score = 0;
5 private int playerLives = 3;
6 private boolean isGameOver = false;

```

Composition:

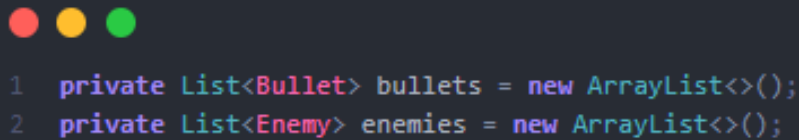
1. My_game มีการใช้งาน Menu และ PlayState ซึ่งแสดงให้เห็นว่า My_game ประกอบไปด้วย Menu และ PlayState ซึ่งทำให้คลาสนี้ใช้ความสามารถของคลาสอื่นๆ ได้ เช่น Menu สามารถเพิ่มปุ่ม Start และ Exit ได้ และ PlayState สามารถเปลี่ยนไปสู่สถานะการเล่นเกมได้

```

1 Menu startMenu = new Menu();
2 PlayState playState = new PlayState();

```

2. คลาส PlayState ประกอบไปด้วย Bullet และ Enemy เป็นองค์ประกอบที่อยู่ภายใน PlayState โดย PlayState ใช้ Enemy และ Bullet ในการจัดการตำแหน่งของศัตรูและกระสุน



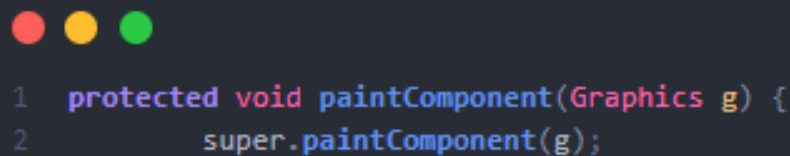
```

1 private List<Bullet> bullets = new ArrayList<>();
2 private List<Enemy> enemies = new ArrayList<>();

```

Polymorphism:

1. การ override เมธอด paintComponent เพื่อวาดองค์ประกอบต่าง ๆ ของเกมใน JPanel นอกจากนี้ยังมีการใช้ polymorphism ใน ActionListener ที่ปุ่ม changeBackgroundButton, restartButton, และ exitButton



```

1 protected void paintComponent(Graphics g) {
2     super.paintComponent(g);

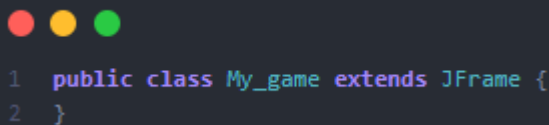
```

Abstract:

ไม่มีการใช้งาน abstract class หรือเมธอด abstract ในโปรแกรมนี้

Inheritance:

1. My_game สืบทอดคุณสมบัติจาก JFrame ซึ่งเป็นคลาสที่เป็นหน้าต่างหลักของโปรแกรม GUI การสืบทอดนี้ทำให้ My_game สามารถใช้งานเมธอดของ JFrame เช่น setSize, setVisible, setTitle



```

1 public class My_game extends JFrame {
2 }

```


2. PlayState สืบทอดจาก JPanel ทำให้สามารถใช้งานเมธอดและคุณสมบัติทั้งหมดของ JPanel ได้ การสืบทอดนี้ทำให้ PlayState สามารถใช้ paintComponent เพื่อวาดส่วนประกอบต่าง ๆ

```
1 public class PlayState extends JPanel {
2 }
```

โครงสร้าง GUI และ Component ที่ใช้

1. หน้าต่างหลัก (JFrame)

ในคลาส My_game หน้าต่างหลักของเกมถูกสร้างขึ้นโดยการสืบทอดจาก JFrame ซึ่งจะเป็นตัวหลักที่แสดงอินเทอร์เฟซของเกม โดยเพิ่ม Menu และ PlayState เป็นส่วนประกอบต่าง ๆ ของเกม

2. Menu Panel

เป็นหน้าจอเริ่มต้นที่ประกอบด้วยปุ่ม Start และ Exit ใช้ JButton และพื้นหลังเป็นรูปภาพเพื่อแสดงข้อความเริ่มต้นของเกม

เมื่อคลิกปุ่ม START เกมจะเปลี่ยนไปยัง PlayState ส่วนเมื่อคลิก EXIT จะออกจากโปรแกรม



3. PlayState Panel

เป็นหน้าจอหลักของเกมที่สืบทอดจาก JPanel และใช้สำหรับวาดองค์ประกอบในเกม เช่น ผู้เล่น ศัตรู กระสุน และพื้นหลัง

ประกอบด้วยปุ่ม Restart, Exit, และ changeBackgroundButton ซึ่งมีไว้สำหรับควบคุมสถานะของเกม เช่นการเริ่มเกมใหม่ เปลี่ยนพื้นหลัง และออกจากเกม

ตัวผู้เล่นและศัตรู: ใช้ภาพ ImageIcon เพื่อแสดงตำแหน่งผู้เล่นและศัตรู โดยตำแหน่งจะเปลี่ยนไปตามการเคลื่อนไหวและอัลกอริทึมของเกม

กระสุน: กระสุนของผู้เล่นถูกแสดงในรูปแบบของคลาส Bullet และมีการอัปเดตตำแหน่งผ่านลูปเกม

คะแนนและชีวิตผู้เล่น: แสดงผลผ่านการใช้ Graphics ในการวาดตัวเลขคะแนน ชีวิต และระดับของเกมที่มีอยู่บนหน้าจอ



Event Handling ที่ใช้ใน GUI

1. ActionListener ของปุ่มต่างๆ

ปุ่ม Start ใน Menu: ใช้ ActionListener เพื่อตรวจสอบเมื่อคลิกปุ่มนี้ และเรียกใช้เมธอด `changeToPlayState()` เพื่อสลับไปยังหน้าจอ PlayState

ปุ่ม Exit: ใช้ ActionListener เพื่อออกจากโปรแกรมโดยเรียกใช้ `System.exit(0)`

ปุ่ม Restart ใน PlayState: ใช้ ActionListener เพื่อเริ่มเกมใหม่โดยการเรียก `restartGame()` เมื่อคลิกปุ่มนี้หลังจากที่ผู้เล่นแพ้

ปุ่ม `changeBackgroundButton`: มี ActionListener สำหรับเปลี่ยนภาพพื้นหลังของเกมใน PlayState โดยเรียกใช้ `switchBackground()`

2. KeyListener สำหรับการควบคุมผู้เล่น

ใน PlayState ใช้ KeyListener ในเมธอด `handleKeyPress()` เพื่อควบคุมการเคลื่อนที่ของผู้เล่น และการยิงกระสุน การกดลูกศรซ้าย/ขวา/ขึ้น/ลง จะทำให้ตัวละครขยับทิศทางต่าง ๆ ในขณะที่การกด Space จะสร้างกระสุนขึ้นมาใหม่

อัลกอริทึมที่สำคัญในโปรแกรม

1. Game Loop (ลูปเกม)

ลูปเกมในเมธอด `startGameLoop()` ถูกตั้งค่าให้เรียก `gameUpdate()` ทุกๆ 1/60 วินาทีเพื่อให้เกิดความต่อเนื่องในเกม

`gameUpdate()` เรียกใช้เมธอดต่างๆ เช่น `spawnEnemies()` เพื่อสร้างศัตรู, `updatePositions()` เพื่ออัปเดตตำแหน่งของศัตรูและกระสุน, `checkCollisions()` เพื่อตรวจสอบการชนกันของศัตรูกับกระสุน และ `advanceLevel()` เพื่อเพิ่มระดับเกมเมื่อกำจัดศัตรูครบตามที่กำหนด

2. การอัปเดตตำแหน่งของศัตรูและกระสุน (`updatePositions`)

ในทุกๆ รอบของลูปเกม เมธอดนี้จะปรับตำแหน่งของศัตรูให้เคลื่อนลงมาและกระสุนให้เคลื่อนขึ้นไป นอกจากนี้ยังลบกระสุนที่ออกนอกขอบเขตหน้าจอ

3. การสร้างศัตรู (spawnEnemies)

มีการสร้างศัตรูใหม่โดยใช้ความน่าจะเป็น (โอกาส 2% ในทุกๆ รอบของลูป)

4. การตรวจสอบการชน (checkCollisions)

เมธอดนี้จะตรวจสอบว่ากระสุนชนกับศัตรูหรือไม่ หากมีการชนกัน จะลบศัตรูและกระสุนออกจากลิสต์ รวมถึงเพิ่มคะแนนและจำนวนศัตรูที่กำจัดได้ นอกจากนี้ยังตรวจสอบว่าศัตรูชนกับผู้เล่นหรือไม่ หากชนกันจะลดชีวิตของผู้เล่น และเมื่อผู้เล่นไม่มีชีวิตเหลือ จะกำหนดค่า isGameOver เป็น true เพื่อแสดงข้อความ Game Over และแสดงปุ่ม Restart และ Exit

5. การเพิ่มLevel (advanceLevel)

เมื่อกำจัดศัตรูครบตามจำนวนที่กำหนดในระดับปัจจุบัน (maxEnemies) เกมจะเพิ่มระดับ (level++) เพิ่มจำนวนศัตรูสูงสุด (maxEnemies) และรีเซ็ตการนับศัตรูที่กำจัด (enemiesDefeated) รวมถึงล้างลิสต์ของศัตรูและกระสุน และเรียก switchBackground() เพื่อเปลี่ยนภาพพื้นหลัง

6. การเปลี่ยนภาพพื้นหลัง (switchBackground)

ฟังก์ชันนี้ทำการเปลี่ยนภาพพื้นหลังใน PlayState โดยวนลูปผ่าน backgroundImages แล้วกำหนดภาพใหม่เป็นภาพพื้นหลัง ปรับค่าดัชนี (currentBackgroundIndex) ให้เป็นลำดับถัดไป

บทที่ 3 สรุป

ปัญหาที่พบระหว่างการพัฒนา

1. มีบัคตอนเริ่มเกมถ้าไม่รีบูตหน้าจอเกมจะค้าง
2. ศัตรูตอนเกิดยังมีการซ้อนกันอยู่

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

1. การเปลี่ยนฉากพื้นหลังเมื่อเลเวลเพิ่มขึ้น
2. การเพิ่มระดับความยากโดยจะเพิ่มจำนวนของผู้ก่อการร้ายและความเร็วของผู้ก่อการร้าย