



Mini Project

Take Maho

จัดทำโดย

6704062612251 นายกันตภณ โคจร

เสนอ

ผู้ช่วยศาสตราจารย์สติติ ประสมพันธ์

วิชา 040613204 Object-Oriented Programming ภาคเรียนที่ 1 ปีการศึกษา 2568

ภาควิชาวิทยาศาสตร์คอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทนำ

เกี่ยวกับโปรเจค

ชื่อ : Take Maho

นำเสนอโดย: นายกันตภณ โคงร

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์สกิตติ์ ประสมพันธ์

Source code: [Github](#)

ที่มาและความสำคัญ

โครงการนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดย การ

นำเรื่องที่เรียนมาสร้างเป็นขั้นงานในรูปแบบเกมโดยใช้แนวคิดการเขียนโปรแกรมแบบเชิงวัสดุ และยัง ช่วยให้ผู้จัดทำเรียนรู้อุปกรณ์และเครื่องมือ ผู้ จัดทำได้สร้างเกมนี้ขึ้นมา

ประเภทของโครงการ

เกมคอมพิวเตอร์(Desktop Java application)

ประโยชน์

- เพื่อความสนุกสนาน
- ฝึกความร่วงไวในการคิด
- ฝึกการตัดสินใจ

ขอบเขตของโครงการ

รายการ	Week 1 1-7 ก.ย.	Week 2 8-14 ก.ย.	Week 3 15-21 ก.ย.	Week 4 22-30 ก.ย.	Week 5 1-7 ต.ค.
1. Desing					
2. Setup					
3. Core Gameplay					
4. Graphics & Refinement					
5. Testing & Submission					

บทที่ 2

ส่วนการพัฒนา

รายละเอียดเกม

เกมแนวต่อสู้ระหว่าง “พ่อมดสองคน” ที่ใช้การ ร่ายเวทด้วยลำดับคีย์คอมโบ แผนการโจมตีแบบกดปุ่มเดียว ผู้เล่นต้องใช้การเคลื่อนไหว หลบหลีก และจังหวะการร่ายคาถาเพื่อเอาชนะอีกฝ่ายในสนามประลองพิเศษ เป้าหมายคือ “กำจัดคู่ต่อสู้ครบ 10 ครั้ง” เพื่อคว้าชัยชนะในยกนั้น เกมนี้ สมมติฐานความเร็ว การตัดสินใจเฉียบพลัน และกลยุทธ์การจัดลำดับสกิลเข้าด้วยกัน หนึ่งในการดูแลระหว่างจอมเวทสองฝ่าย

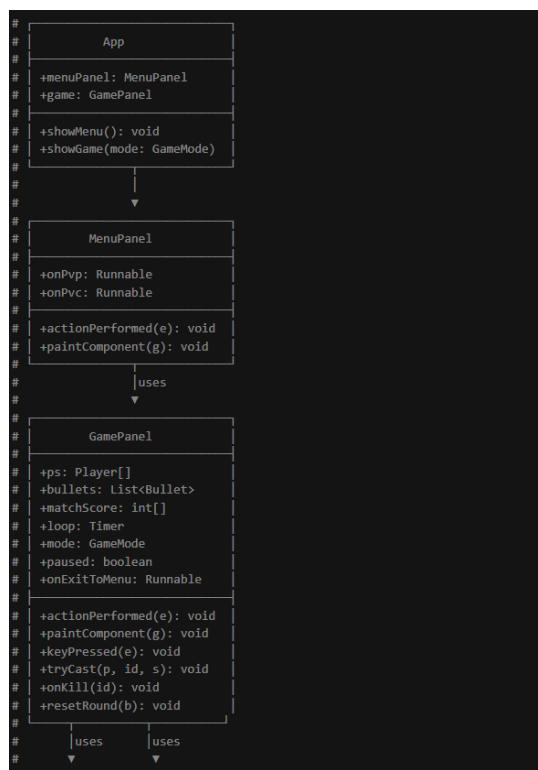
เนื้อเรื่องย่อ

ในโลกแห่งเวทมนตร์ที่สมดุลของธรรมชาติถูกทำลาย พลังแห่ง “Maho” คือพลังเวทโบราณที่แบ่งออกเป็นสองสาย — แสง และ เงา เมื่อกฎแห่งเวทมนตร์ถูกบิดเบือนโดยผู้ทรายศ พ่อมดผู้ยิ่งใหญ่สองคนจึงต้องต่อสู้กัน เพื่อแย่งชิง “คทาแห่งสมดุล” อันเป็นสิ่งเดียวที่สามารถกอบกู้หรือทำลายโลกนี้ได้ การต่อสู้ครั้งนี้ไม่ใช่เพียงการประลองเวท แต่คือการตัดสินชะตาแห่งโลกเวททั้งใบ

วิธีการเล่น

- เกมนี้เป็นการต่อสู้ระหว่างพ่อมดสองคนในสนามประลอง ผู้เล่นสามารถเลือกเล่นแบบ ผู้เล่นสองคน (PVP) หรือ ผู้เล่นคนเดียวต่อสู้กับบท (PVC) ได้
- เมื่อเริ่มเกม ผู้เล่นแต่ละฝ่ายจะยืนอยู่คนละฝั่งของสนาม และสามารถ เคลื่อนไหวไปมา ด้วยปุ่มลูกศร (สำหรับผู้เล่นขวา) หรือปุ่ม W A S D (สำหรับผู้เล่นซ้าย)
- การโจมตีไม่ได้ใช้ปุ่มเดียว แต่ต้อง ร่ายคาถาด้วย “คอมโบปุ่ม” (Spell Combo) ซึ่งเป็นลำดับปุ่มเฉพาะของแต่ละเวท เช่น
 - **JKL** หรือ **123** : ยิงเวทพลังงานเส้นตรง (Straight Shot)
 - **LKJ** หรือ **321** : ยิงเวทติดตาม (Homing Shot)
 - **KJL** หรือ **213** : บลิ๊คหลบกระสุน (Blink)
 - **KLJ** หรือ **231** : สะท้อนเวทกลับ (Reflect)
 - **LJK** หรือ **312** : แปลงร่างเป็นหินป้องกันเวท (Stone Form)
- เวทแต่ละชนิดมี ระยะเวลาฟื้นฟู (Cooldown) เมื่อใช้ไปแล้วต้องรอระยะเวลาที่กำหนด ก่อนจึงจะใช้ซ้ำได้ ผู้เล่นต้องบริหารจังหวะให้ดี
- หากผู้เล่นถูกเวทของอีกฝ่ายโดนจน HP หมด จะ “แพ้ในรอบนั้น” และอีกฝ่ายจะได้ 1 คะแนนคิล (Kill Point)
- เมื่อสะสมครบ 10 คะแนนคิล จะถือว่าชนะใน “หนึ่งยก (Round)”

Class diagram



1) Core Layer (แกนหลักของเกม)

App (JFrame)

- หน้าต่างหลักของเกม ใช้ **CardLayout** สลับระหว่าง **MenuPanel** ↔ **GamePanel**
- เมธอดสำคัญ
 - showMenu()** แสดงหน้าเมนู
 - showGame(mode: GameMode)** เปิดฉากร่องสู่ตามโหมด (PVP / PVC)
และส่ง callback **onExitToMenu** ให้ **GamePanel**

MenuPanel (JPanel)

- เมนูเริ่มเกม ปุ่ม **Play vs Player**, **Play vs COM**
- สร้างด้วย **GridBagLayout** แนวไข้งานง่ายบนจอ 1280×720 **GamePanel** (JPanel + ActionListener + KeyListener)
- ลูปเกมด้วย Timer ~60 FPS (**FPS_MS = 16**)
- คุมทุกระบบของฉากร่องสู่: ผู้เล่น กระสุน การชน การชน-แพ้ ยั่ดคูลดาวน์ บอท
- โหมดเกม: **PVP** / **PVC** (COM) ผ่าน **GameMode**
- Pause / Resume & กลับเมนู:
 - [ESC] toggle pause, ในหน้าหลังชนะกด ESC กลับเมนู
 - ระหว่าง pause: [R] รีเเมตซ์, [M] เมนู

GameMode (enum)

- PVP**, **PVC** (กำหนดชื่อฝ่ายแดงเป็น COM และเปิด AI)

2) Systems Layer (ระบบย่อยที่ **GamePanel** ใช้)

Input System (ภายใน GamePanel)

- รับ **KeyPressed/Released** สำหรับการเคลื่อนที่ P1 (**WASD**) / P2 (**Arrows**)
- ระบบคอมโบร่ายสกิลผ่าน **onComboKey()**
 - P1: J K L
 - P2/COM: 1 2 3
 - mapping ผ่าน **Keybinds** และสตวิ่งสูตรใน **Player** (เช่น "jkl", "lkj")

Cooldown / Skill System

- คูลดาวน์ต่อสกิล (ms): **CD_STRAIGHT**, **CD_HOMING**, **CD_BLINK**, **CD_REFLECT**, **CD_STONE**
- ระยะเวลาพล (ms): **DUR_REFLECT**, **DUR_STONE**, **DUR_BLINK_IFRAME**

- `Cooldowns` เก็บ/คำนวณเวลา: `set(now, skill, cd)`, `ready(now, skill)`, `remaining(now, skill)`

AI / Bot System (เฉพาะ PVC)

- ทำงานใน `botTick(now)`
 - คิดถี่ๆทุก `BOT_THINK_MS`
 - จำกัดความถี่ร่ายด้วย `BOT_CAST_GAP`
 - ตัดสินใจ: หลบกระสุน (`side-step`), รักษาระยะ `sweet-spot`, ใช้สกิลตามสถานการณ์ (`Reflect/Blink/Stone/ยิง`)

HUD / UI System (ใน GamePanel)

- สกอร์กลางจอ, Kills ของแต่ละฝ่าย, Pill คูลดาวน์ 5 แ夸/ฝ่าย
- หน้าชัชนะยก: แบนเนอร์ + `hint [R]` Rematch • ESC Menu
- Pause overlay: เมนูตัวหนังสือกลางจอ

Rendering System

- ใช้ `Animator` + `SpriteSheet` แสดง `idle/run/cast` ของสองฝ่าย (สเกล 2x)
- วางแผนรุ่น 2 แบบ: วงกลม (ตรง) / เพชร (ไฮเมิ่ง)
- วางแผนรุ่น 2 แบบ: วงกลม (ตรง) / เพชร (ไฮเมิ่ง)

3) Gameplay Objects Layer (วัตถุที่เกิดในฉาก)

Player

- ค่าหลัก: `x, y, vx, vy, hp, kills`, ทิศ `facingRight`, สถานะผลของสกิล (`reflectOn/stoneOn/blinkIFrameOn`)
- เวลาหมดอายุผล: `reflectUntil, stoneUntil, blinkIFrameUntil, lastBlinkAt`
- การควบคุม: `flags left/right/up/down` + `buffer` คอมโบ + `Cooldowns`
- อินพุตคอมโบ: สตริงสูตร (`S_STRAIGHT, S_HOMING, ...`)
- เมธอดเด่น: `pushCombo, comboString, clearCombo`

Bullet

- ตำแหน่ง/ความเร็ว `x, y, vx, vy`, เจ้าของ `ownerId` (0/1), คง `homing`, สี
- อัปเดตมุมเลี้ยวสำหรับไฮเมิ่งด้วย `HOMING_TURN_RATE`

- ชีนขอบจอ → `dead=true`
- ชีนโลผู้เล่น → สะท้อน + เปลี่ยนเจ้าของ
- ชีนลำตัว → ฝ่าเป้าหมายเว้นแต่มี `i-frame/blink/stone`

Skill (enum)

- `STRAIGHT, HOMING, BLINK, REFLECT, STONE`

4) Support / Utility Layer

Animator

- รับ `SpriteSheet` + ความเร็วเฟรม
- `update(), frame(), sheet().drawFrame(...)`

SpriteSheet

- โหลดสีไปร์ตจาก `Assets.img(path)`, กำหนดขนาดเฟรม (64×64)
- ช่วยเลือกเฟรมสำหรับนักเวท blue/red (idle/run/cast)

Assets

- ตัวช่วยโหลด `BufferedImage` จาก resource path (ภาพ UI/ตัวละคร/โล่/ทิ่น/พื้นหลัง)

Keybinds

- กำหนด mapping คีย์ของผู้เล่นสองฝ่ายทั้งทิศทางและคอมโบ

Geometry helpers (ใน GamePanel)

- `bodyHit()` ตรวจด้วยวงรีแทนลำตัว
- `nearShield()` ตรวจชนรัศมีเลื่อนตามสเกลรูป
- `clamp(), normalizeAngle()` พังค์ชันคณิตพื้นฐาน

5) Game State / Match Flow Layer

Match/Kill Flow

- ชนะ “ยก” เมื่อ kill ถึง `KILL_TO_WIN = 10` → บวก `matchScore` ฝ่ายนั้น
- หลังแต่ละ kill (ยังไม่ครบ 10) → รีเซ็ตตำแหน่ง (“คงคิลล์”)
- กด `R` ระหว่างเกม → รีรับ “ล้างคิลล์” (คะแนนรวมคงไว้)
- จบยก (ขึ้นแบบเนอร์) → `R` รีเมิกใหม่ หรือ `ESC` กลับเมนู

Pause/Resume

- **ESC** toggle pause (ขณะไม่จบยก)
- ตอน pause: **[R]** รีแมตช์, **[M]** เมนู, **[ESC]** Resume

PVC (COM)

- ตั้งชื่อผู้เล่นแดงเป็น **COM**, เปิด **isBot = true**
- ใช้ **botTick()** ตลอดเมื่ออยู่ในโหมดนี้

รูปแบบการพัฒนา

- ภาษา: Java
- IDE: IntelliJ idea
- การพัฒนา GUI: javax.swing (JFrame, JPanel, Graphics2D)
- การจัดการรูปภาพ: java.awt.image.BufferedImage
- ลูปการทำงาน: javax.swing.Timer (60 FPS)
- ระบบอินพุต: java.awt.event.KeyListener

แนวคิดการเขียนโปรแกรมเชิงวัตถุ

Constructor

```
public class App extends JFrame {
    private final JPanel root = new JPanel(new CardLayout()); 9 usages
    private MenuPanel menuPanel; 2 usages
    private GamePanel game; 5 usages

    public App() { 1 usage
        super(title: "Take Maho");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);

        menuPanel = new MenuPanel(app: this);
        root.add(menuPanel, constraints: "menu");

        setContentPane(root);
        pack();
        setLocationRelativeTo(null);

        showMenu();
    }
}
```

เป็นคอนสตรัคเตอร์หลัก สร้างหน้าต่าง, ติดตั้ง **CardLayout**, ใส่หน้าเมนูเริ่มต้น และ **pack()** ให้พอดีหน้าจอ

```
public MenuPanel(Runnable onPvp, Runnable onPvc) { 1 usage
    setPreferredSize(new Dimension(GamePanel.W, GamePanel.H));
    setLayout(new GridBagLayout());

    // โหลดภาพพื้นหลังเมนู
    try {
        bgMenu = Assets.img(path: "/duel/resources/Menu1.png");
    } catch (Throwable t) {
        bgMenu = new ImageIcon(filename: "src/duel/resources/Menu1.png").getImage();
    }

    GridBagConstraints c = new GridBagConstraints();
    c.gridx=0; c.fill=GridBagConstraints.HORIZONTAL;
    c.insets = new Insets(top: 10, left: 0, bottom: 10, right: 0);

    JLabel title = new JLabel(text: "Take Maho", SwingConstants.CENTER);
    title.setForeground(Color.WHITE);
    title.setFont(title.getFont().deriveFont(Font.BOLD, size: 40f));

    JButton pvp = new JButton(text: "Play vs Player");
    JButton pvc = new JButton(text: "Play vs COM");

    pvp.addActionListener(ActionEvent e -> onPvp.run());
    pvc.addActionListener(ActionEvent e -> onPvc.run());

    c.gridy=0; add(title, c);
    c.gridy=1; add(pvp, c);
    c.gridy=2; add(pvc, c);
}
```

สร้าง UI เมนูและ bind ปุ่มกับ callback (ส่งมาเป็น **Runnable**) → ใช้วิธี Encapsulation + Composition + Event Handling ไปด้วย

```

public MenuPanel(App app) { 1 usage
    this(
        () -> app.showGame(GameMode.PVP),
        () -> app.showGame(GameMode.PVC)
    );
}

```

Polymorphism (overloading) – ค่อนสตรัคเตอร์ที่สองท่อ callback ให้เอง สะดวกเวลาเรียกจาก App

```

public GamePanel(GameMode mode, Runnable onExitToMenu) { 1 usage
    this.mode = mode;
    this.onExitToMenu = onExitToMenu;

    setPreferredSize(new Dimension(W, H));
    setBackground(new Color(r: 20, g: 22, b: 26));
    setDoubleBuffered(true);
    setFocusable(true);
    addKeyListener(this);

    // โหลดภาพหลังหน้า
    try {
        bgBattle = Assets.img(path: "/duel/resources/Bg.png");
    } catch (Throwable t) {
        // ถ้าไม่ได้ไฟล์ Assets จะอ่านไฟล์ตรง
        bgBattle = new ImageIcon(filename: "src/duel/resources/Bg.png").getImage();
    }

    // Blue (JKL)
    ps[0] = new Player(
        name: "P1", x: W*0.25, y: H*0.5,
        Keybinds.P1_MOVE, Keybinds.P1_COMBO,
        mapJ: "J", mapK: "K", mapL: "L", mapI: "I",
        straight: "jkl", homing: "lkj", blink: "kjl", reflect: "klj", stone: "ljk",
        new Color(r: 120, g: 170, b: 255), new Color(r: 120, g: 200, b: 255, a: 120)
    );

    // Red (右手 + 123)
    String p2Name = (mode==GameMode.PVC) ? "COM" : "P2";
    ps[1] = new Player(
        name: "P2", x: W*0.75, y: H*0.5,
        Keybinds.P2_MOVE, Keybinds.P2_COMBO,
        mapJ: "1", mapK: "2", mapL: "3", mapI: "5",
        straight: "123", homing: "321", blink: "213", reflect: "231", stone: "312",
        new Color(r: 255, g: 160, b: 120), new Color(r: 255, g: 180, b: 120, a: 120)
    );
    ps[1].isBot = (mode==GameMode.PVC);

    // Sprites
    p1Idle = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_blue_idle.png"), frameW: 64, frameH: 64), ticksPerFrame: 10);
    p1Run = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_blue_run.png"), frameW: 64, frameH: 64), ticksPerFrame: 12);
    p1Cast = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_blue_cast.png"), frameW: 64, frameH: 64), ticksPerFrame: 12);
    p2Idle = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_red_idle.png"), frameW: 64, frameH: 64), ticksPerFrame: 10);
    p2Run = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_red_run.png"), frameW: 64, frameH: 64), ticksPerFrame: 12);
    p2Cast = new Animator(new SpriteSheet(Assets.img(path: "/duel/resources/wizard_red_cast.png"), frameW: 64, frameH: 64), ticksPerFrame: 12);

    shieldBlue = Assets.img(path: "/duel/resources/shield_blue.png");
    shieldRed = Assets.img(path: "/duel/resources/shield_red.png");
    stoneBlue = Assets.img(path: "/duel/resources/stone_blue.png");
    stoneRed = Assets.img(path: "/duel/resources/stone_red.png");

    loop.start();
}

```

ตั้งค่าพื้นฐานการเรนเดอร์, listener, สร้างผู้เล่น, โหลดสีปรterr, เริ่มเกมคุป → Constructor + Composition

Encapsulation

```

// ===== Rules =====
public static final double PLAYER_SPEED = 4.0; 2 usages
public static final int MAX_BUFFER = 8; 1 usage
public static final int KILL_TO_WIN = 10; 3 usages

// Cooldowns (ms)
public static final int CD_STRAIGHT = 700; 3 usages
public static final int CD_HOMING = 1200; 2 usages
public static final int CD_REFLECT = 700; 2 usages
public static final int CD_STONE = 1000; 2 usages
public static final int CD_BLINK = 900; 2 usages

// Durations (ms)
public static final int DUR_REFLECT = 1000; 1 usage
public static final int DUR_STONE = 1000; 1 usage
public static final int DUR_BLINK_IFRAME = 200; 1 usage
public static final int REFLECT_LENIENCE = 100; 1 usage

// Bullets
public static final double BULLET_SPEED = 7.0; 13 usages
public static final double HOMING_TURN_RATE = 0.12; 2 usages

// Blink tune
public static final double BLINK_DISTANCE = 200.0; 2 usages
public static final double BLINK_ESCAPE_RADIUS = 72.0; 1 usage
public static final int BLINK_ESCAPE_WINDOW_MS = 220; 1 usage

// Rendering
private static final int SPR_SCALE = 2; 14 usages

private final Player[] ps = new Player[2]; 70 usages
private final List<Bullet> bullets = new ArrayList<>(); 12 usages
private final javax.swing.Timer loop = new javax.swing.Timer(FPS_MS, [listener: this])

// บัน/แมตช์
private final int[] matchScore = new int[]{0,0}; 3 usages
private boolean pendingReset = false; 3 usages
private boolean resetKeepKills = true; 2 usages
private boolean matchOver = false; 6 usages

private final GameMode mode; 1 usage
private final Random rng = new Random(); 1 usage

// Pause / back to menu
private boolean paused = false; 6 usages
private final Runnable onExitToMenu; 5 usages

```

ซ่อนการจัดการสถานะ/ตระรกะไว้หลังเมธอด **private** ภายนอกเรียกไม่ได้ → ลดผลกระทบ/การใช้พิเศษ

Composition

```
public class App extends JFrame {  
    private final JPanel root = new JPanel(new CardLayout());  
    private MenuPanel menuPanel; 2 usages  
    private GamePanel game; 5 usages
```

App เป็นองค์ประกอบระดับสูง ประกอบด้วย **MenuPanel**, **GamePanel** บน **CardLayout** → สลับจอง่าย

```
private final Player[] ps = new Player[2]; 70 usages  
private final List<Bullet> bullets = new ArrayList<>(); 12 usages  
private final javax.swing.Timer loop = new javax.swing.Timer(FPS_MS, [listener: this]); 1 usage  
  
// ยก/แมมด้วย  
private final int[] matchScore = new int[]{0,0}; 3 usages  
private boolean pendingReset = false; 3 usages  
private boolean resetKeepKills = true; 2 usages  
private boolean matchOver = false; 6 usages  
  
private final GameMode mode; 1 usage  
private final Random rng = new Random(); 1 usage  
  
// Pause / back to menu  
private boolean paused = false; 6 usages  
private final Runnable onExitToMenu; 5 usages  
  
// พื้นหลังจากต่อสู้  
private Image bgBattle; 4 usages  
  
// Character animators / overlays  
private Animator p1Idle, p1Run, p1Cast, p2Idle, p2Run, p2Cast; 3 usages  
private BufferedImage shieldBlue, shieldRed, stoneBlue, stoneRed; 3 usages
```

รวมผู้เล่น, กระสุน, เกมคลุป, แอนิเมชัน → ขัดเจนว่าเป็น has-a relationship

Inheritance

```
public class App extends JFrame {
```

```
    public class GamePanel extends JPanel implements ActionListener, KeyListener {
```

```
        public class MenuPanel extends JPanel {
```

Inheritance จาก **JFrame/JPanel**; และ implements interfaces เพื่อรับอีเวนต์

Event Handling

```
@Override public void keyReleased(KeyEvent e) {  
    int k = e.getKeyCode();  
  
    if (k==KeyEvent.VK_A) ps[0].left=false;  
    if (k==KeyEvent.VK_D) ps[0].right=false;  
    if (k==KeyEvent.VK_W) ps[0].up=false;  
    if (k==KeyEvent.VK_S) ps[0].down=false;  
  
    if (!ps[1].isBot) {  
        if (k==KeyEvent.VK_LEFT) ps[1].left=false;  
        if (k==KeyEvent.VK_RIGHT) ps[1].right=false;  
        if (k==KeyEvent.VK_UP) ps[1].up=false;  
        if (k==KeyEvent.VK_DOWN) ps[1].down=false;  
    }  
}  
  
@Override public void keyPressed(KeyEvent e) {  
    int k = e.getKeyCode();  
  
    // PAUSE toggle  
    if (k==KeyEvent.VK_ESCAPE) {  
        if (matchOver) {  
            if (onExitToMenu != null) onExitToMenu.run();  
        } else {  
            paused = !paused;  
            repaint();  
        }  
        return;  
    }  
  
    if (paused) {  
        if (k==KeyEvent.VK_R) { resetRound(keepKills: false); paused=false; }  
        else if (k==KeyEvent.VK_M) { if (onExitToMenu != null) onExitToMenu.run(); }  
        return;  
    }  
  
    if (matchOver) {  
        if (k==KeyEvent.VK_R) {  
            matchOver = false;  
            resetRound(keepKills: false);  
        }  
        return;  
    }  
  
    if (k==KeyEvent.VK_A) ps[0].left=true;  
    if (k==KeyEvent.VK_D) ps[0].right=true;  
    if (k==KeyEvent.VK_W) ps[0].up=true;  
    if (k==KeyEvent.VK_S) ps[0].down=true;  
  
    if (!ps[1].isBot) {  
        if (k==KeyEvent.VK_LEFT) ps[1].left=true;  
        if (k==KeyEvent.VK_RIGHT) ps[1].right=true;  
        if (k==KeyEvent.VK_UP) ps[1].up=true;  
        if (k==KeyEvent.VK_DOWN) ps[1].down=true;  
    }  
  
    onComboKey(k);  
  
    if (k==KeyEvent.VK_R) resetRound(keepKills: false);  
}
```

ระบบที่ทำให้เกมตอบสนองต่อ “เหตุการณ์” จากภายนอก (เช่น การกดปุ่ม) และ “เหตุการณ์เวลา” (timer loop) ได้แบบเรียลไทม์

Algorithm ที่สำคัญ

```
private boolean bodyHit(Player p, double bx, double by) {  
    double cx = p.x;  
    double cy = p.y + 8 * SPR_SCALE;  
    double rx = 18 * SPR_SCALE;  
    double ry = 26 * SPR_SCALE;  
  
    double dx = (bx - cx) / rx;  
    double dy = (by - cy) / ry;  
    return dx*dx + dy*dy <= 1.0;  
}
```

ตรวจสอบสกิลโดนตัว

```

private void tryCast(Player caster, int casterId, Skill s) { @usages
    long now = System.currentTimeMillis();
    if (!caster.cds.ready(now, s)) return;
    if (caster.stoneOn && s != Skill.STONE) return;

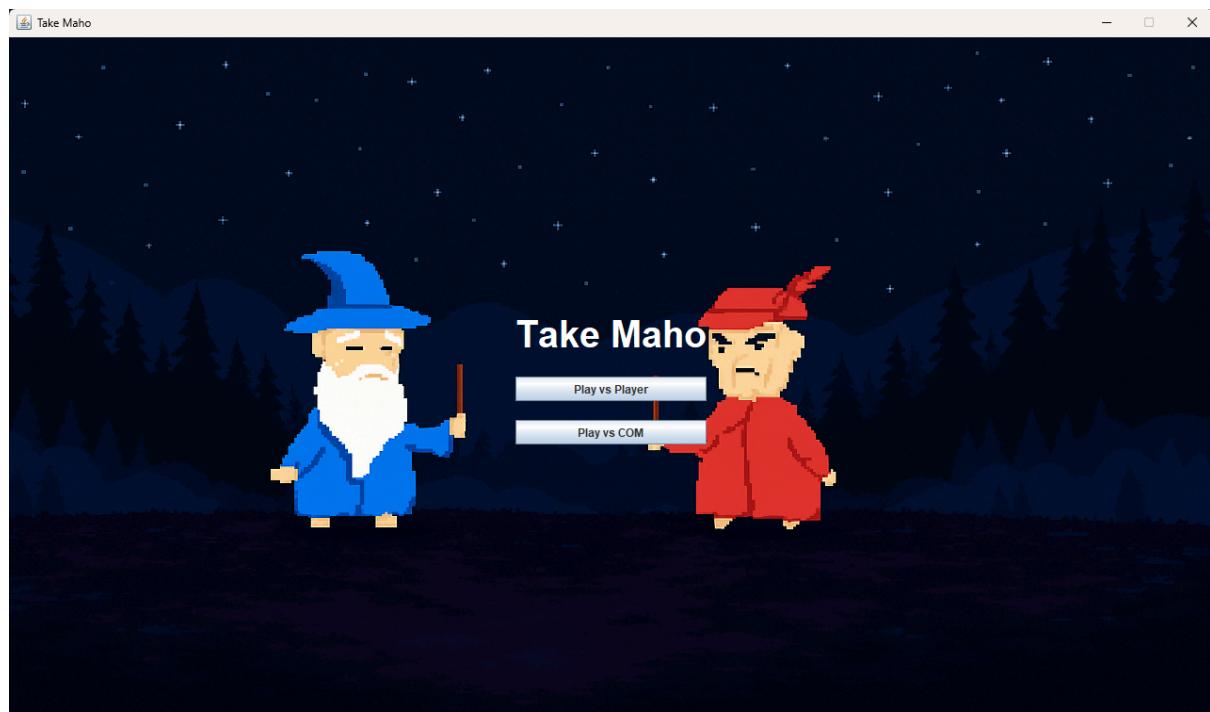
    switch (s) {
        case STRAIGHT -> {
            Player tgt = ps[casterId==0?1:0];
            double ang = Math.atan2(tgt.y - caster.y, tgt.x - caster.x);
            Color col = (casterId==0)? new Color( 120, 200, 255 ) : new Color( 255, 170, 120 );
            bullets.add(new Bullet(
                caster.x, caster.y,
                vx: Math.cos(ang)*BULLET_SPEED, vy: Math.sin(ang)*BULLET_SPEED,
                casterId, homing: false, col
            ));
            caster.cds.set(now, Skill.STRAIGHT, CD_STRAIGHT);
            caster.castUntil = now + 120;
            Sfx.play( name: "shoot" );
        }
        case HOMING -> {
            Player tgt = ps[casterId==0?1:0];
            double ang = Math.atan2(tgt.y - caster.y, tgt.x - caster.x);
            Color col = (casterId==0)? new Color( 120, 200, 255 ) : new Color( 255, 170, 120 );
            bullets.add(new Bullet(
                caster.x, caster.y,
                vx: Math.cos(ang)*BULLET_SPEED, vy: Math.sin(ang)*BULLET_SPEED,
                casterId, homing: true, col
            ));
            caster.cds.set(now, Skill.HOMING, CD_HOMING);
            caster.castUntil = now + 140;
            Sfx.play( name: "homimg" );
        }
        case REFLECT -> {
            caster.reflectUntil = now + DUR_REFLECT;
            caster.cds.set(now, Skill.REFLECT, CD_REFLECT);
            caster.castUntil = now + 100;
            Sfx.play( name: "reflect" );
        }
        case STONE -> {
            caster.stoneUntil = now + DUR_STONE;
            caster.cds.set(now, Skill.STONE, CD_STONE);
            caster.castUntil = now + 100;
            Sfx.play( name: "stone" );
        }
    }

    case BLINK -> {
        Player tgt = ps[casterId==0?1:0];
        double dx = caster.vx, dy = caster.vy;
        double len = Math.hypot(dx, dy);
        if (len == 0) { dx = tgt.x - caster.x; dy = tgt.y - caster.y; len = Math.hypot(dx, dy); }
        if (len > 0) { dx/=len; dy/=len; }
        caster.x = clamp( v: caster.x + dx*BLINK_DISTANCE, lo: 40, hi: W-40 );
        caster.y = clamp( v: caster.y + dy*BLINK_DISTANCE, lo: 60, hi: H-60 );
        caster.blinkIFrameUntil = now + DUR_BLINK_IFRAME;
        caster.lastBlinkAt = now;
        caster.cds.set(now, Skill.BLINK, CD_BLINK);
        caster.castUntil = now + 90;
        Sfx.play( name: "blink" );
    }
}

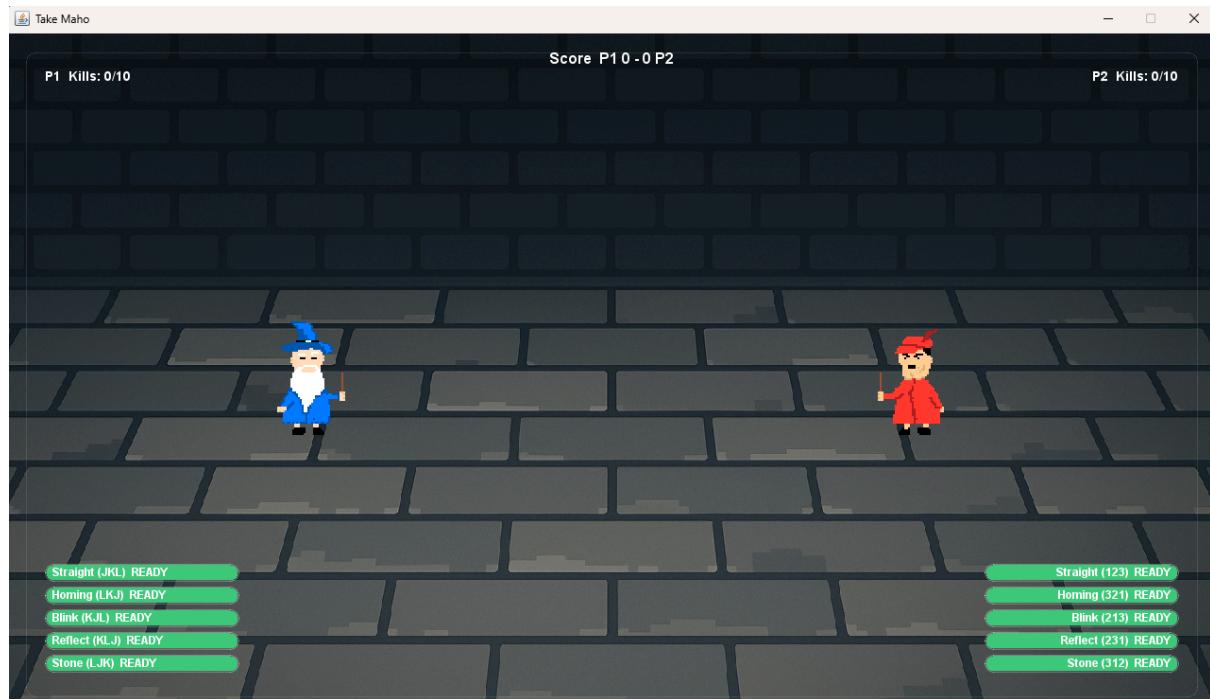
```

การร่ายสกิล

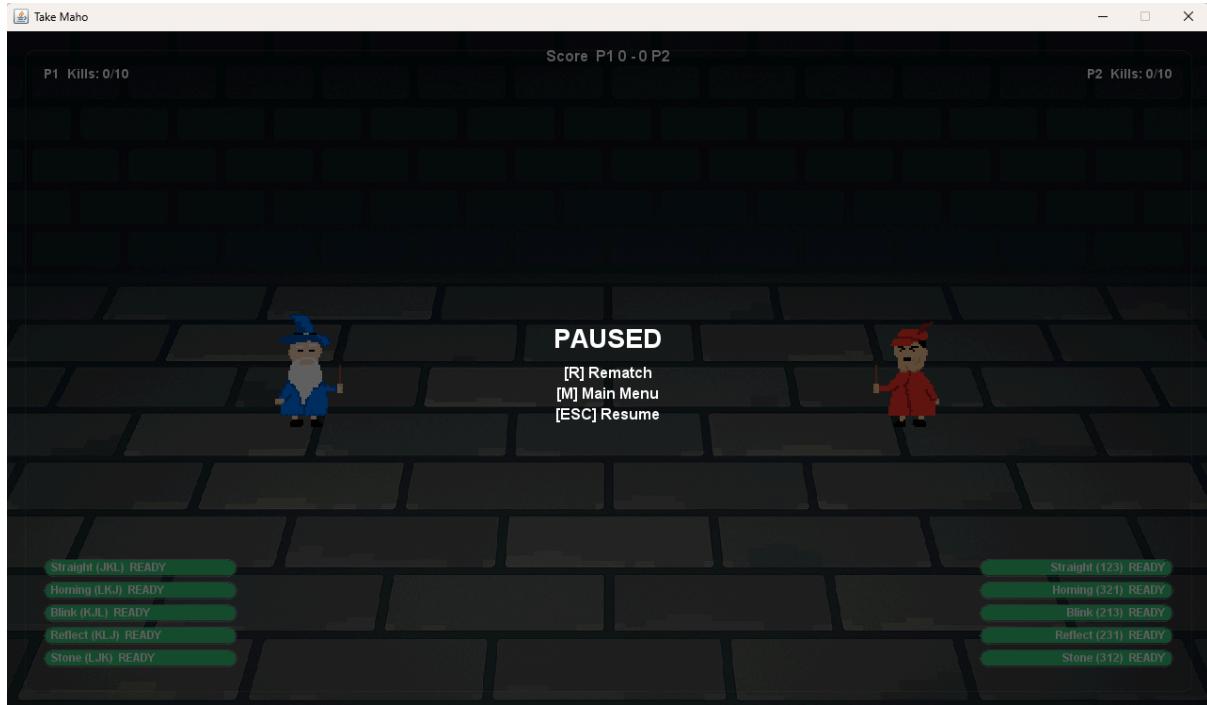
GUI



หน้าหลัก กดเล่นกับผู้เล่นอีกคนหรือกดเล่นกับบอท



หน้าเกมจากต่อสู้ แสดง จำนวน win แสดงสถิติคุณดาวน์



กด ESC เพื่อหยุดแล้วกดต่อ ออกรีปหน้าหลัก หรือออกจากรายการ

บทที่ 3

สรุปผลและข้อเสนอแนะ

ปัญหาที่พบ

- การกระตุก/หน่วงระหว่างเล่นบางจังหวะ
- ไฟกั้นคีย์บอร์ดหลุดตอนสลับหน้าจอ (Menu ↔ Game)
- บอท “รัวสกิล” เกินมนุษย์ในบางสถานการณ์
- สมดุลสกิลยังไม่นิ่ง
- ชนวัตถุกับกระสุนใช้ hitbox วางเดียว (เร็ว แต่ไม่ล่าเอี้ยด)
- เกมไม่มีเสียง

จุดเด่นของโปรแกรม

- คอนโทรลง่าย แต่มีขั้นเชิงตัวย “คอมโบสกิล” (Straight/Homing/Blink/Reflect/Stone)
- ระบบประทีลเล่นได้สนุก
- มีโหมด PvP และ PvC (บอท)
- รอบ/ยก และคะแนนรวม (match score) ชัดเจน

ข้อเสนอแนะ

- เพิ่มด่าน และฉากต่อสู้หลายแบบ เช่น ปราสาท ป่า ถ้ำ เพื่อให้เกมหลากหลายมากขึ้น
- เพิ่มระบบเสียงและเพลงประกอบเพื่อเพิ่มความตื่นเต้นในการเล่น
- เพิ่มตัวละครพอมต์ใหม่ ๆ ที่มีสกิลแตกต่าง เช่น เวทนำแข็ง เวทสายฟ้า
- ทำให้ UI หน้าเมนูและ HUD สวยงามขึ้น เช่น ไอคอนสกิล
- ปรับสมดุลของสกิล เช่น ลดความแรงของเวทติดตาม และเพิ่มคุณภาพให้เหมาะสม