โครงงาน

Mini Project

จัดทำโดย

6504062620051 ธนพ แสงจันทร์

เสนอ

ผู้ช่วยศาสตราจารย์ สถิต ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2566

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เกี่ยวกับโครงงาน

ชื่อโปรเจค: Jaunt of Urikaka

นำเสนอโดย: นาย ธนพ แสงจันทร์

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

บทที่ 1 ที่มาและความสำคัญของโปรเจ็ค

โครงงานนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดย การนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบโปรเจ็คเกม ผู้จัดทำได้สร้างเกมนี้ขึ้นมาเพื่อความสนุก และ เพื่อการศึกษา

ประเภทโครงการ

โครงการทางวิทยาศาสตร์และเทคโนโลยี

โปรเจ็คเกม

ประโยชน์

1.เพื่อความสนุกสนาน

2.ได้ความรู้เกี่ยวกับ วิชา Object Oriented Programming

3.ช่วยทำให้คิดเป็นขั้นเป็นตอน

ตารางงานแผนการทำงาน

ลำดับ	รายการ	5 ก.ย. - 15 ก.ย.	1 ต.ค. – 15 ต.ค.	> 15 ต.ค
1	หารูปจัดทำตัวละครในเกม			
2	ศึกษาข้อมูลต่างๆ			
3	เขียนโปรแกรม			
4	จัดทำเอกสาร			
5	ตรวจสอบข้อผิดพลาด			

6504062620051 ธนพ แสงจันทร์ sec.1

Proposal (Update)

การออกเดินทาง ของ อูริกาก้า

(Jaunt of Urikaka)

- **รายละเอียดเกม**

  เกม Jaunt of Urikaka เป็นเกมที่จะให้ผู้เล่นได้ออกผจญภัยและ เผชิญหน้ากับ Dimon ราชาปีศาจโกเลม ต้องคอยละวังอันตรายต่างๆระหว่างทางไปปราบ ราชาปีศาจโกเลม Dimon เกมจะจบลงเมื่อทำภารกิจสำเร็จ หรือเมื่อหัวใจหมด

- **วิธีการเล่น**

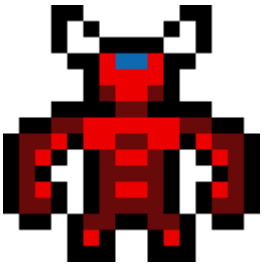
  กดปุ่ม w, a, s, d เพื่อบังคับทิศทางการเคลื่อนไหวโดยปุ่ม w จะเป็นการกระโดด a, s เป็น การเคลื่อนที่ไปซ้ายหรือขวา และปุ่ม ENTER เพื่อทำการตี

- Storyboard

ตัวละคร

เจ้าหนู อูริกาก้า

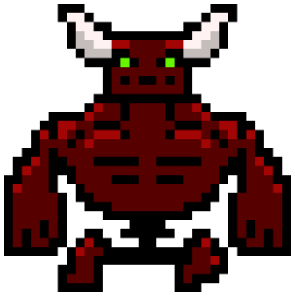
NPC เจ้าสไลม์แสน



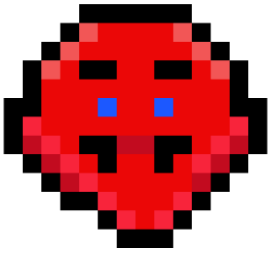
ปีศาจโกเลม

NPC มังกรตัวเล็ก

NPC หมาป่า



ราชาปีศาจ



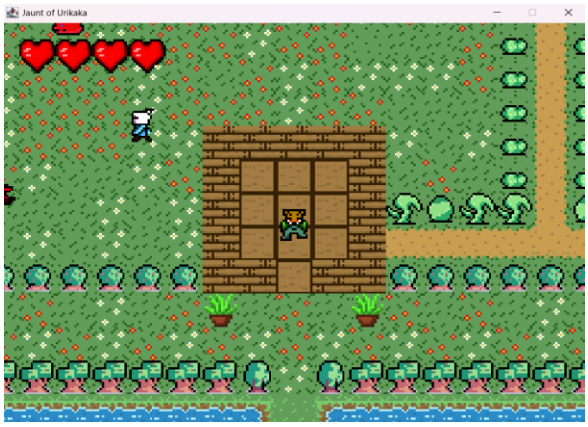
สไลม์แสนน่ากลัว

ฉาก

- ฉากหน้าเข้าเกม

ถ้ากด ENTER จะเข้าเกม

- ฉากเริ่มเกม



- ฉาก Game Over

- ฉาก ชนะเกม

บทที่ 2 ส่วนการพัฒนา

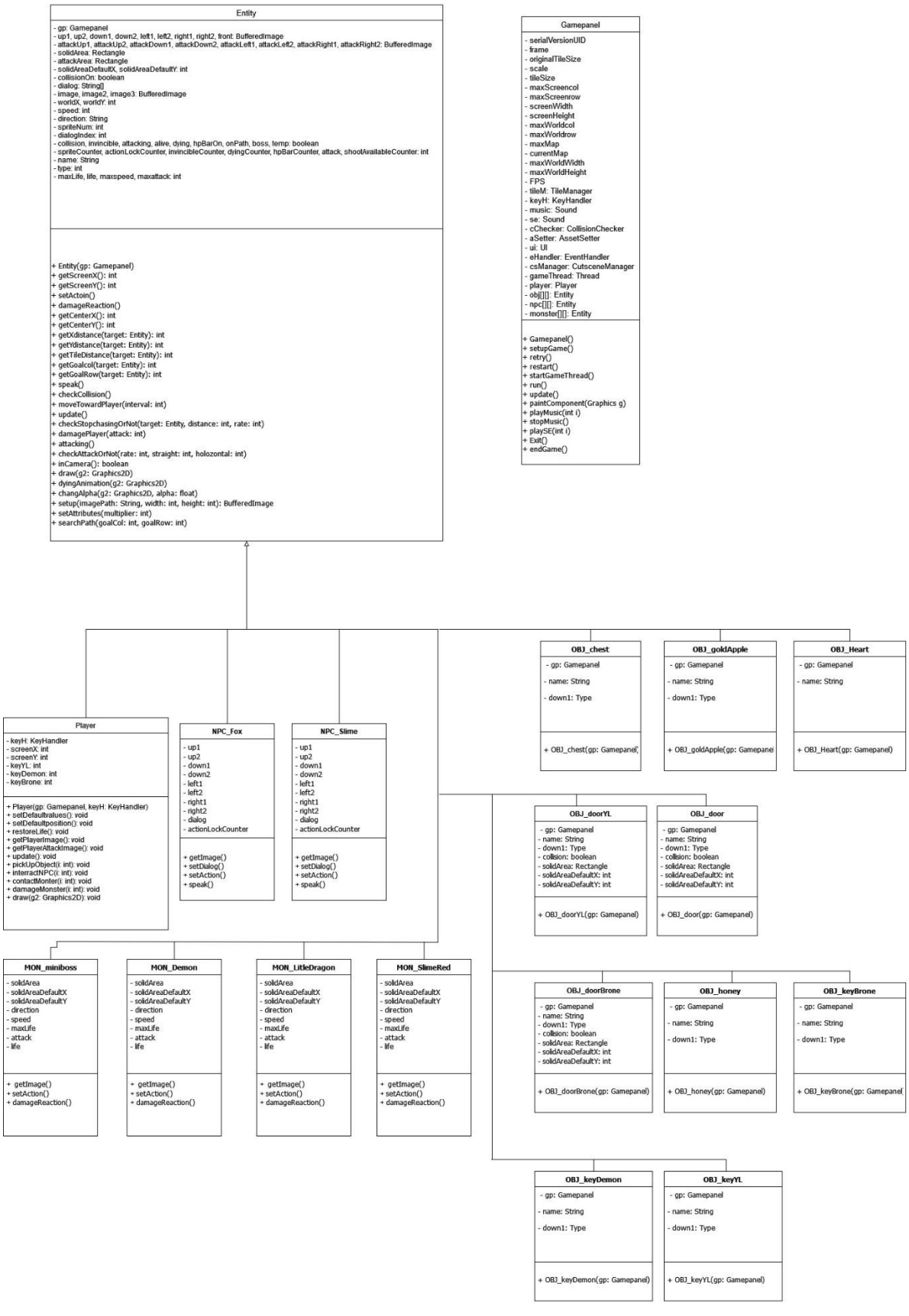
เนื้อเรื่องย่อ

- เจ้าหนูอูริกาก้าได้ตื่นขึ้นมาในกระท่อมแห่งหนึ่ง ด้วยความมึนงงจึงทำให้เขาต้องออกไปสำรวจ และเขาก็พบว่าจะต้องช่วยโลกใบนี้ไว้จากบางสิ่ง ทำให้เขาจะต้องเริ่มออกตามหาสิ่งๆนั้น และจะต้องกำจัดมัน !

วิธีการเล่น

กด W , A , S , D เพื่อ ทำการเดิน และ กด ENTER เพื่อทำการตี หรือ ทำการคุยกับ NPC

Class Diagram



คำอธิบาย Class Diagram

Class Entity เป็นคลาสหลักในการจัดเก็บ และควบคุมพฤติกรรมของ ตัวละคร NPC Monster และ Object ต่างๆ ภายในเกม

Class Player เป็นคลาสที่จัดเก็บข้อมูลและพฤติกรรมของตัวละครหลักที่ผู้เล่นบังคับ

Class NPC_Fox , NPC_Slime เป็นคลาสที่จัดเก็บข้อมูลของ NPC ภายในเกม

Class MON_miniboos , MON_Demon , MON_LitleDragon , MON_slimeRed จะเป็นคลาสที่จัดเก็บข้อมูลและพฤติกรรม ของ Monster ในเกม

Class OBJ_chest , OBJ_goldApple , OBJ_Heart , OBJ_doorYL , OBJ_door , , OBJ_doorBrone , OBJ_keyBrone , OBJ_keyDemon , OBJ_keyYL จะเป็นคลาสที่จัดเก็บข้อมูลของ Object ภายในเกม

รูปแบบการพัฒนา > Application

อธิบายส่วนของโปรแกรมที่มี

- Constructor

```
public Gamepanel() {

        this.setPreferredSize(new DimensionUIResource(screenWidth, screenHeight));
        this.setBackground(Color.black);
        this.setDoubleBuffered(true);
        this.addKeyListener(keyH);
        this.setFocusable(true);
    }
```

- **Constructor** ของ Class Gamepanel มีการทำงานดังนี้
  this.setPreferredSize(new DimensionUIResource(screenWidth, screenHeight));
  - ตั้งค่าขนาดที่ต้องการสำหรับ panel นี้ โดยใช้ความกว้างและความสูงของหน้าจอ
  this.setBackground(Color.black); - ตั้งค่าสีพื้นหลังของ panel เป็นสีดำ
  this.setDoubleBuffered(true); - เปิดใช้งาน double buffering ซึ่งช่วยลดปรากฏการณ์ flickering ในการแสดงผลกราฟิก this.addKeyListener(keyH); - เพิ่ม KeyListener ที่ชื่อ keyH ให้กับ panel นี้ ซึ่งจะรับการทำงานเมื่อมีการกดปุ่มคีย์บอร์ด this.setFocusable(true); - ทำให้ panel นี้สามารถรับ focus ได้ ซึ่งจำเป็นสำหรับการรับอินพุตจากคีย์บอร์ด

```java
public Player(Gamepanel gp, KeyHandler keyH) {

        super(gp);
        this.keyH = keyH;

        screenX = gp.screenWidth / 2 - (gp.tileSize / 2);
        screenY = gp.screenHeight / 2 - (gp.tileSize / 2);

        solidArea = new Rectangle();
        solidArea.x = 8;
        solidArea.y = 16;
        solidAreaDefaultX = solidArea.x;
        solidAreaDefaultY = solidArea.y;
        solidArea.width = 32;
        solidArea.height = 32;

        attackArea.width = 36;
        attackArea.height = 36;

        setDefaultvalues();
        getPlayerImage();
        getPlayerAttackImage();
    }
```

- **Constructor** ของ Class Player มีการทำงานดังนี้

  super(gp); - เรียก constructor ของคลาส parent ด้วย Gamepanel ที่ส่งมา

  this.keyH = keyH; - กำหนด KeyHandler สำหรับ Player

  screenX = gp.screenWidth / 2 - (gp.tileSize / 2); และ screenY = gp.screenHeight / 2 - (gp.tileSize / 2); - กำหนดตำแหน่งเริ่มต้นของ Player ให้อยู่ตรงกลางหน้าจอ

  solidArea = new Rectangle(); - สร้าง Rectangle ใหม่สำหรับ solidArea ซึ่งอาจจะใช้สำหรับการตรวจสอบการชนกัน

solidArea.x = 8; และ solidArea.y = 16; - กำหนดตำแหน่งเริ่มต้นของ solidArea

solidAreaDefaultX = solidArea.x; และ solidAreaDefaultY = solidArea.y; - จำตำแหน่งเริ่มต้นของ solidArea

solidArea.width = 32; และ solidArea.height = 32; - กำหนดขนาดของ solidArea

attackArea.width = 36; และ attackArea.height = 36; - กำหนดขนาดของ attackArea ซึ่งอาจจะใช้สำหรับการตรวจสอบการโจมตี

setDefaultvalues(); - เรียก method setDefaultvalues ซึ่งอาจจะกำหนดค่าเริ่มต้นสำหรับ Player

getPlayerImage(); และ getPlayerAttackImage(); - เรียก methods ที่อาจจะโหลดรูปภาพของ Player และรูปภาพการโจมตีของ Player

```java
public UI(Gamepanel gp) {
        this.gp = gp;

        try {
            InputStream is = getClass().getResourceAsStream("/res/font/x12y16pxMaruMonica.ttf");
            maruMoica = Font.createFont(Font.TRUETYPE_FONT, is);
            is = getClass().getResourceAsStream("/res/font/Purisa Bold.ttf");
            purisaB = Font.createFont(Font.TRUETYPE_FONT, is);
        } catch (FontFormatException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        // heart
        Entity heart = new OBJ_Heart(gp);
        heart_full = heart.image;
        heart_half = heart.image2;
        heart_blank = heart.image3;
    } // constructor
```

- **Constructor** ของ Class UI มีการทำงานดังนี้

  this.gp = gp; - กำหนด Gamepanel สำหรับ UI

  โค้ดในบล็อก try โหลดฟอนต์จากไฟล์ .ttf ที่กำหนด และสร้าง Font จากฟอนต์ที่โหลดมา

  ถ้ามีข้อผิดพลาดในการโหลดฟอนต์หรือสร้าง Font จะจับข้อผิดพลาดและพิมพ์ stack trace

  Entity heart = new OBJ_Heart(gp); - สร้าง Entity ใหม่ที่เป็น OBJ_Heart ด้วย Gamepanel ที่กำหนด

  heart_full = heart.image;, heart_half = heart.image2;, heart_blank = heart.image3; - กำหนดรูปภาพสำหรับสถานะต่าง ๆ ของหัวใจ (เต็ม, ครึ่งหนึ่ง, ว่าง) จาก Entity ที่สร้างขึ้น

```java
public TileManager(Gamepanel gp) {
        this.gp = gp;

        tile = new Tile[100];

        mapTileNum = new int[gp.maxMap][gp.maxWorldcol][gp.maxWorldrow];

        getTileImage();

        loadMap("/res/maps/worldmap01.txt",0);
        loadMap("/res/maps/worldmap02.txt",1);
        loadMap("/res/maps/worldmap03.txt",2);

    }
```

- **Constructor** ของ Class TileManager มีการทำงานดังนี้

  this.gp = gp; - กำหนด Gamepanel สำหรับ TileManager

  tile = new Tile[100]; - สร้าง array ของ Tile ที่มีขนาด 100

  mapTileNum = new int[gp.maxMap][gp.maxWorldcol][gp.maxWorldrow]; - สร้าง array 3 มิติของ int ที่มีขนาดตาม maxMap, maxWorldcol, maxWorldrow ของ Gamepanel

  getTileImage(); - เรียก method getTileImage ซึ่งอาจจะโหลดรูปภาพของ Tile

  loadMap("/res/maps/worldmap01.txt",0);, loadMap("/res/maps/worldmap02.txt",1);, loadMap("/res/maps/worldmap03.txt",2); - เรียก method loadMap สำหรับโหลดแผนที่จากไฟล์ .txt ที่กำหนด และกำหนด index ของแผนที่

➢ Composition

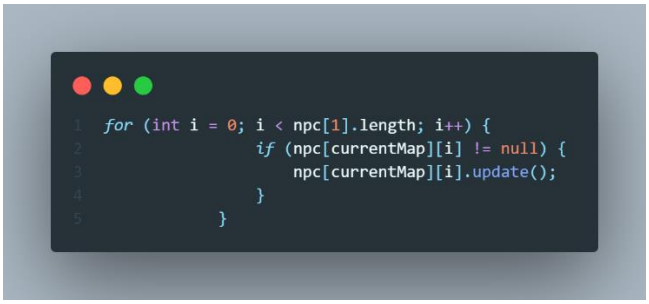
Composition ใน OOP Java เป็นเทคนิคการออกแบบเพื่อใช้สำหรับการสร้างความสัมพันธ์แบบ "has-a" ซึ่งหมายความว่า เมื่อคลาสหนึ่งมี instance

```
TileManager tileM = new TileManager(this);
    public KeyHandler keyH = new KeyHandler(this);
    Sound music = new Sound();
    Sound se = new Sound(); // sound effect
    public CollisionChecker cChecker = new CollisionChecker(this);
    public AssetSetter aSetter = new AssetSetter(this);
    public UI ui = new UI(this);
    public EventHandler eHandler = new EventHandler(this);
    public CutsceneManager csManager = new CutsceneManager(this);
    public Player player = new Player(this, keyH);
```

- Composition ใน class Gamepanel

➢ Polymorphism

```
public Entity obj[][] = new Entity[maxMap][20];
    public Entity npc[][] = new Entity[maxMap][10];
    public Entity monster[][] = new Entity[maxMap][20];
    ArrayList<Entity> entityList = new ArrayList<Entity>();
```



```
for (int i = 0; i < npc[1].length; i++) {
                if (npc[currentMap][i] != null) {
                    npc[currentMap][i].update();
                }
            }
```

```
for (int i = 0; i < entityList.size(); i++) {
            entityList.get(i).draw(g2);
        }
```

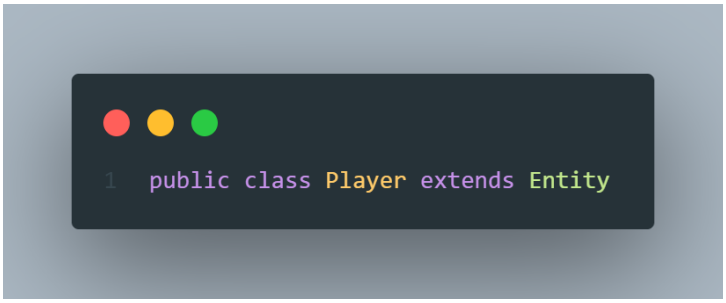
- Entity เป็นคลาสแม่และ obj, npc, และ monster เป็นคลาสลูก อาร์เรย์ Entity สามารถเก็บวัตถุใดๆที่เป็นอินสแตนซ์ของคลาส Entity หรือคลาสลูกของ Entity
- เมธอด update() ถูกเรียกบนวัตถุ npc นี่เป็นตัวอย่างของโพลีมอร์ฟิซึมเพราะการทำงานจริงๆของเมธอด update() จะขึ้นอยู่กับประเภทของวัตถุ npc ในระหว่างรันไทม์
- เมธอด draw() ถูกเรียกบนวัตถุ Entity นี่เป็นตัวอย่างของโพลีมอร์ฟิซึมเพราะการทำงานจริงๆของเมธอด draw() จะขึ้นอยู่กับประเภทของวัตถุ Entity ในระหว่างรันไทม์

➢ Inheritance

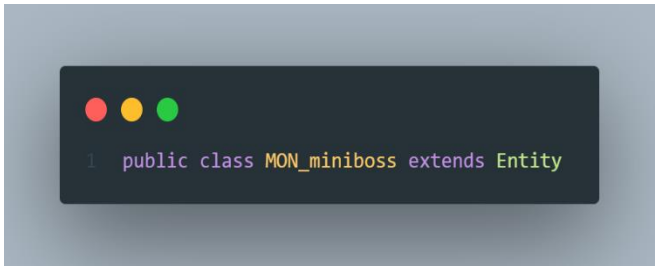
```
public class Gamepanel extends JPanel implements Runnable
```

คลาส Gamepanel นี้เป็นการสืบทอดจากคลาส JPanel และมีการสร้าง interface จาก Runnable ซึ่งเป็นส่วนหนึ่งของ Java Swing และ Java Threading ตามลำดับ

คลาส Player นี้เป็นคลาสที่สืบทอดมาจาก Entity และสามารถมีคุณสมบัติและเมธอดเพิ่มเติมที่เฉพาะเจาะจงสำหรับ Player ได้



```
public class Player extends Entity
```



```
public class MON_miniboss extends Entity
```

คลาส MON_miniboss นี้เป็นคลาสที่สืบทอดมาจาก Entity และสามารถมีคุณสมบัติและเมธอดเพิ่มเติมที่เฉพาะเจาะจงสำหรับ MON_miniboss ได้

6504062620051 ธนพ แสงจันทร์ sec.1

GUI ประกอบด้วย Component อะไรบ้าง

- หน้า TitleState

GUI หน้านี้เกิดจาก

g2.setColor(new Color(70, 120, 80)); > background

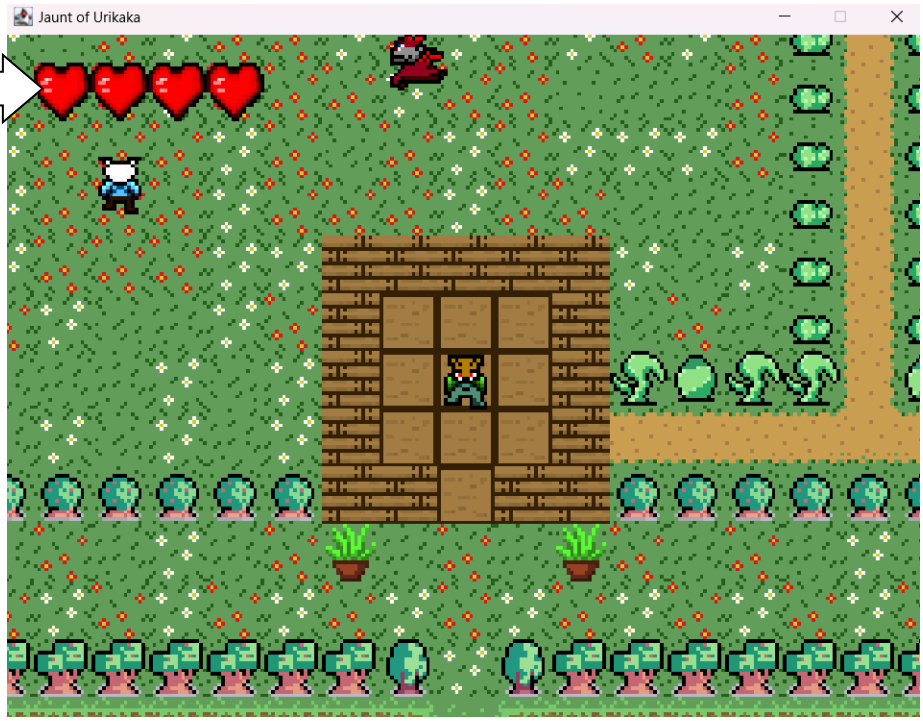
g2.drawImage(gp.player.front, x, y, gp.tileSize * 2, gp.tileSize * 2, null); > รูป ตัวละคร

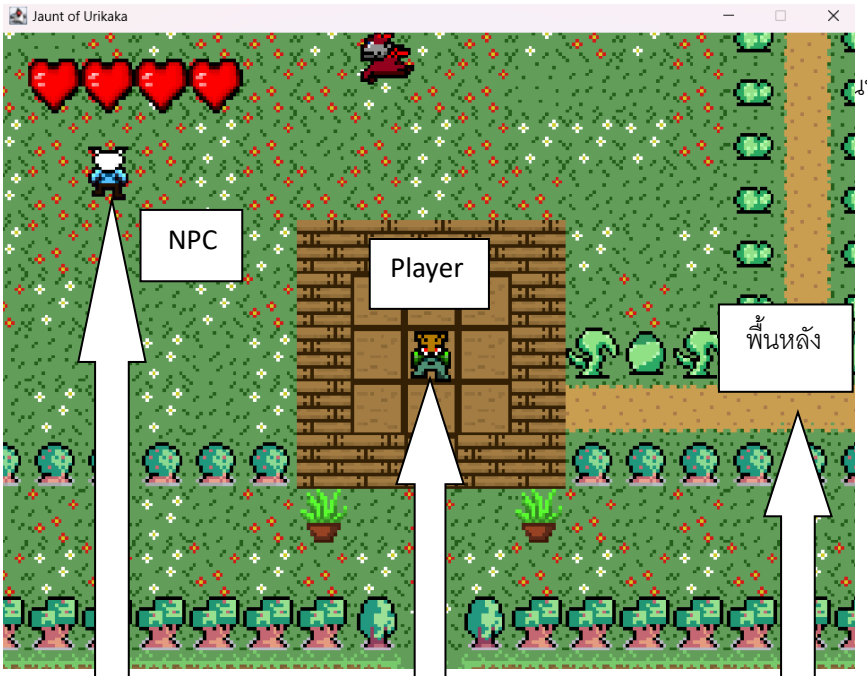
g2.drawString(text, x, y); > GAME START , QUIT

- หน้า PlayState

รูปหัวใจ เกิดจาก

```
public void drawPlayerLife() {

        // gp.player.life = 6 ;

        int x = gp.tileSize / 2;
        int y = gp.tileSize / 2;
        int i = 0;

        // max life
        while (i < gp.player.maxLife / 2) {
            g2.drawImage(heart_blank, x, y, null);
            i++;
            x += gp.tileSize;
        }

        // reset
        x = gp.tileSize / 2;
        y = gp.tileSize / 2;
        i = 0;
        // current life
        while (i < gp.player.life) {
            g2.drawImage(heart_half, x, y, null);
            i++;
            if (i < gp.player.life) {
                g2.drawImage(heart_full, x, y, null);

            }
            i++;
            x += gp.tileSize;
        }

    }
```

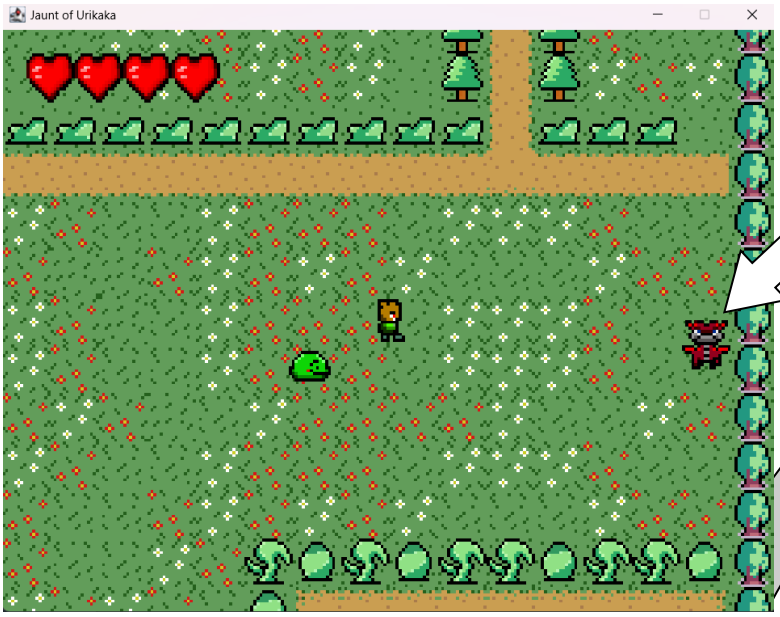





```java
tileM.draw(g2);

            entityList.add(player);

            // Add to entityList
            for (int i = 0; i < npc[1].length; i++) {
                if (npc[currentMap][i] != null) {
                    entityList.add(npc[currentMap][i]);
                }
            }

            for (int i = 0; i < obj[1].length; i++) {
                if (obj[currentMap][i] != null) {
                    entityList.add(obj[currentMap][i]);
                }
            }
            for (int i = 0; i < monster[1].length; i++) {
                if (monster[currentMap][i] != null) {
                    entityList.add(monster[currentMap][i]);
                }
            }

            // Sort
            Collections.sort(entityList, new Comparator<Entity>() {

                @Override
                public int compare(Entity e1, Entity e2) {
                    int result = Integer.compare(e1.worldY, e2.worldY);
                    return result;
                }

            });

        }
        // draw entitys
        for (int i = 0; i < entityList.size(); i++) {
            entityList.get(i).draw(g2);
        }
```

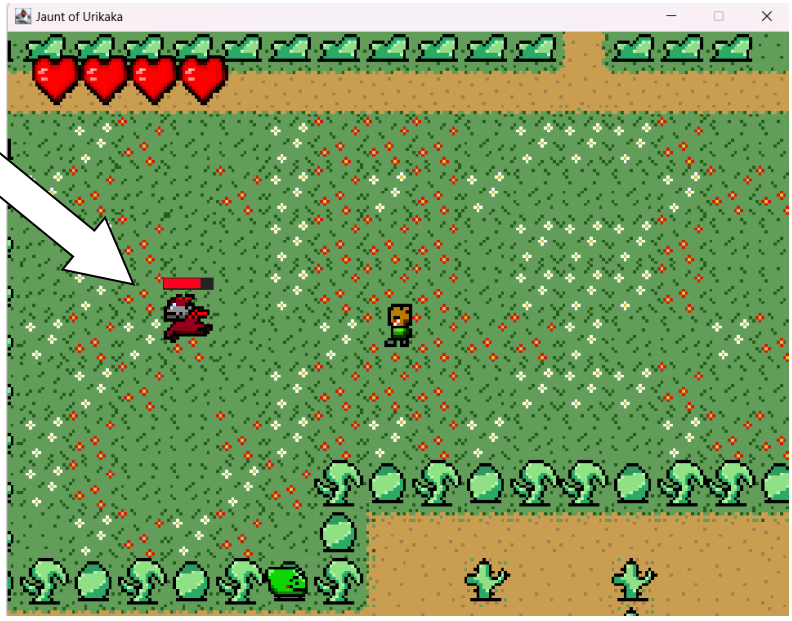
Event handling

- HP BAR



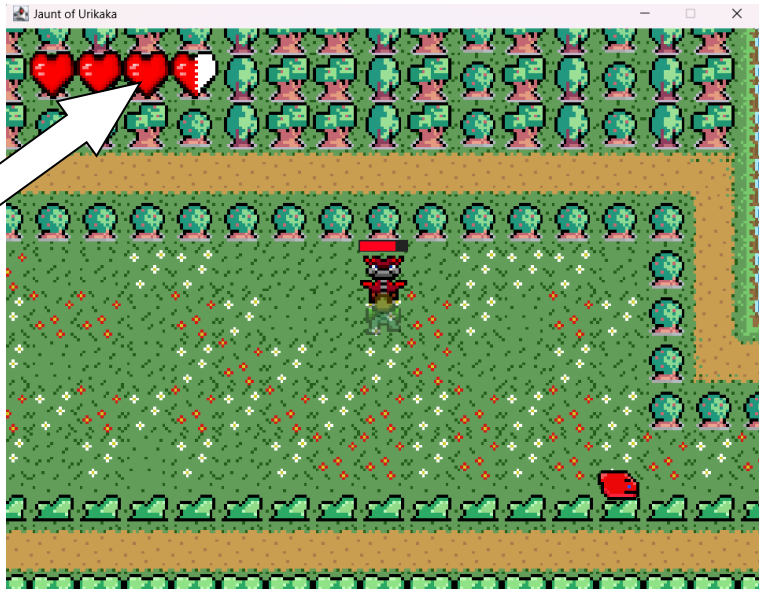
Monster ถ้ายังไม่โจมตีมัน หลอดเลือดก็จะไม่ขึ้น

แต่เมื่อโจมตี จะมีหลอดเลือดขึ้นมา และพฤติกรรมการตอบสนองต่อการโจมตีก็จะไม่เหมือนกัน แล้วแต่ว่าจะเป็นตัวไหน

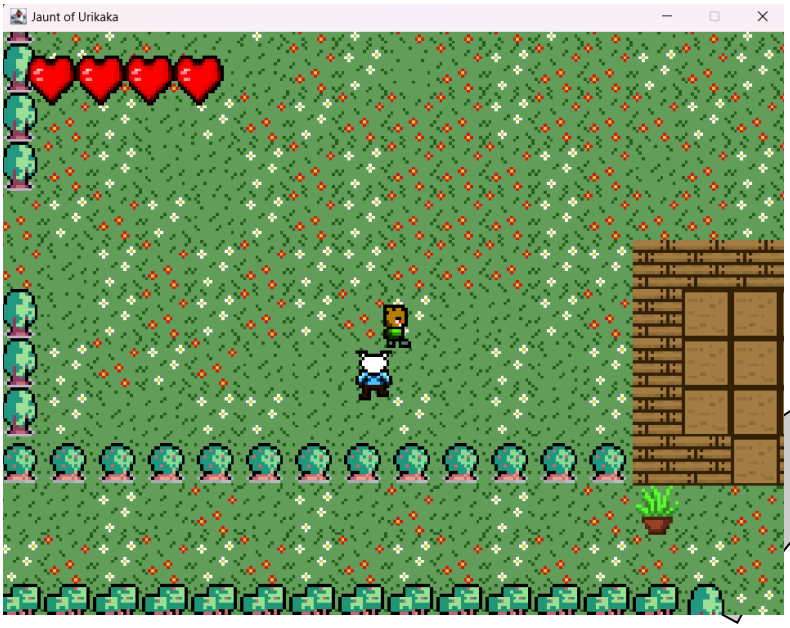


- Damage Player

เมื่อถูก Monster โจมตี เลือดจะลดตามพลังโจมตีของMonster นั้นๆ และ จะมีสถานะ คงกระพันเป็นเวลาหนึ่ง ซึ่งตัวจะกระพริบ



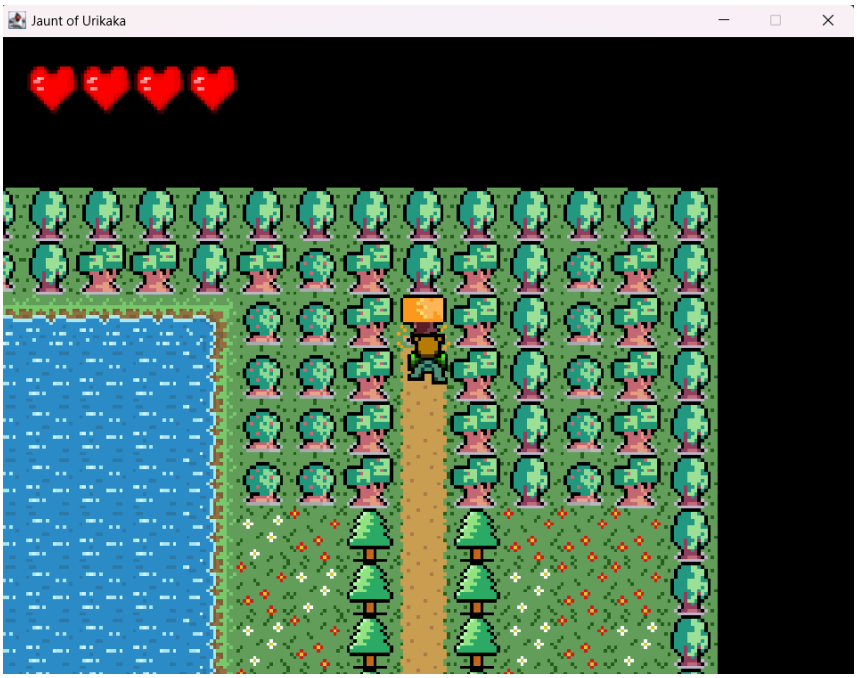
- Talking to NPC



เมื่อเข้าไปใกล้ NPC จะยังไม่มีอะไรเกิดขึ้น

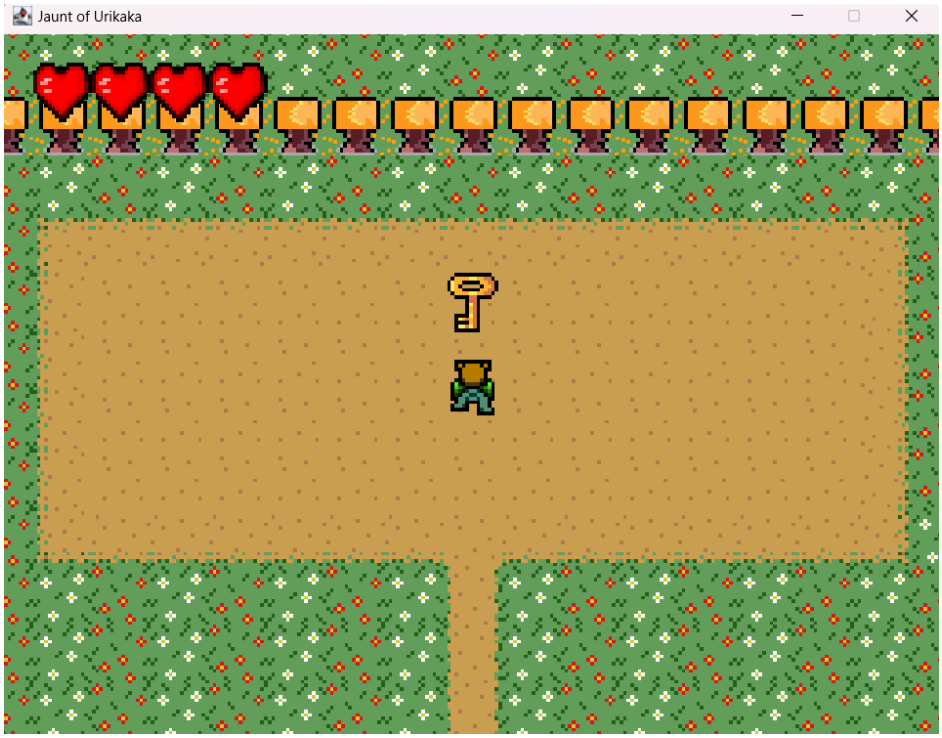
เมื่อเข้าไปใกล้ NPC และกด ENTER จะเกิดเป็นข้อความปรากฏขึ้นบนหน้าจอ

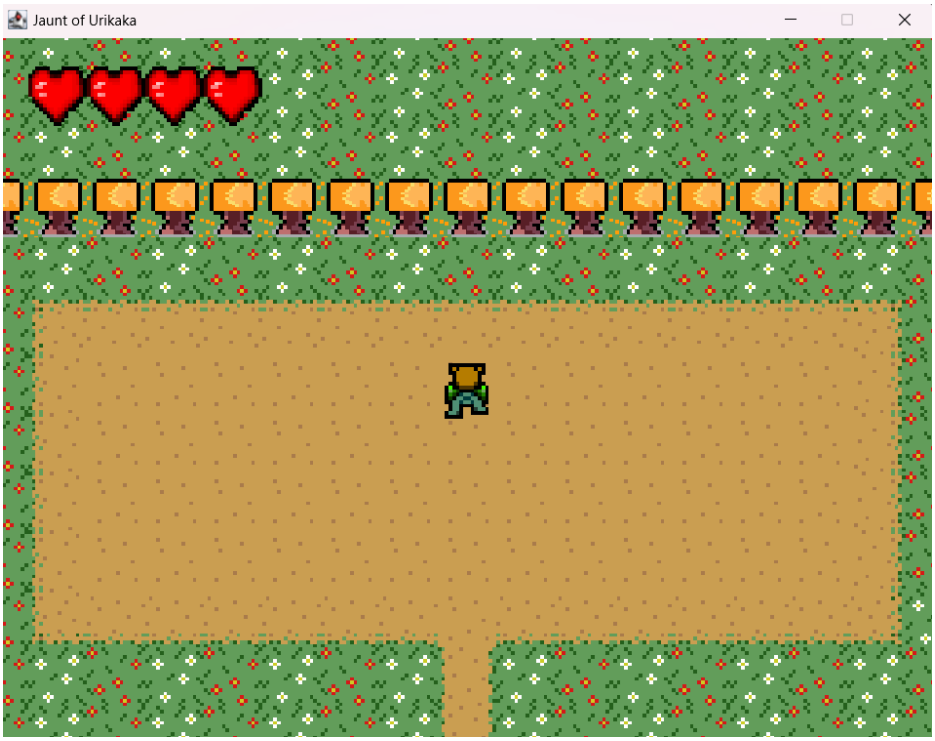
- Teleport with ENTER



เมื่อ กด ENTER ที่จุดนี้จะวาปไปอีกที่หนึ่ง

- Pick items



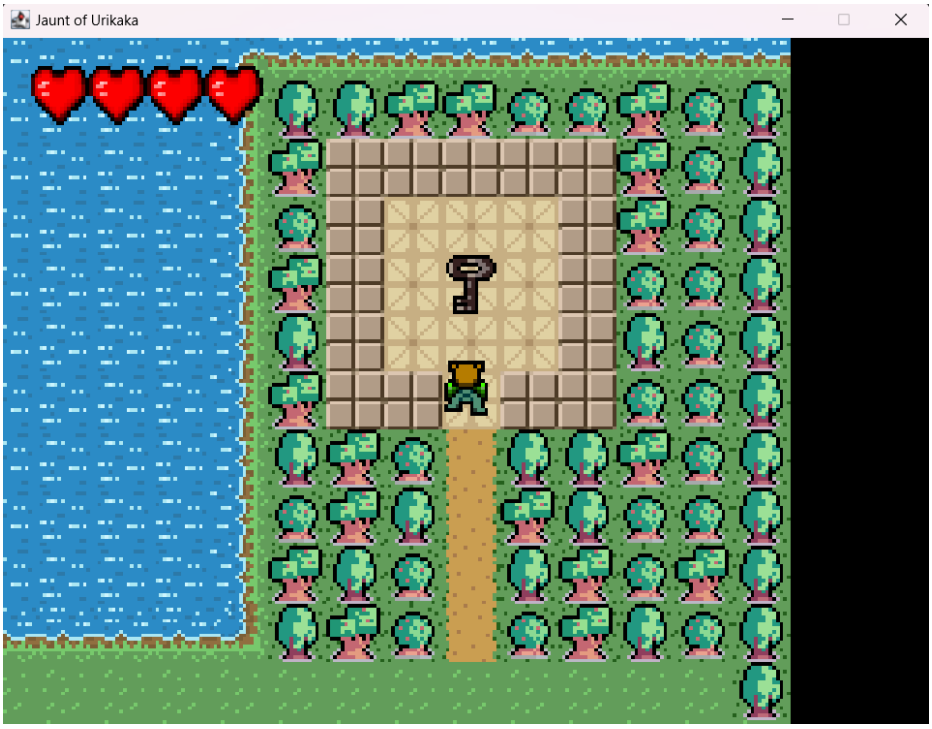


ถ้าไม่เดินเข้าไปใกล้ items จะไม่ทำการเก็บ

แต่ถ้าเดินเข้าไปที่จุดนั้นๆ จะทำการเก็บ items อัตโนมัติ

- Opened Door

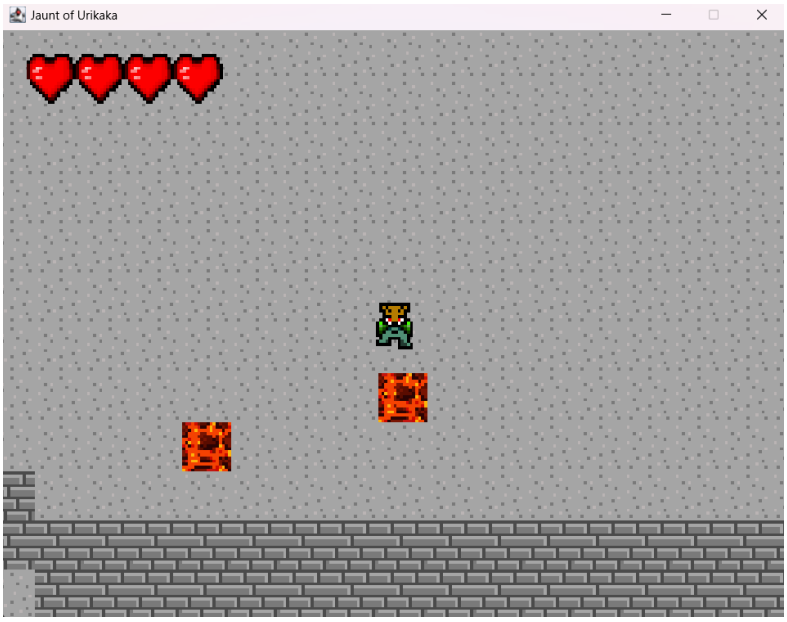
-



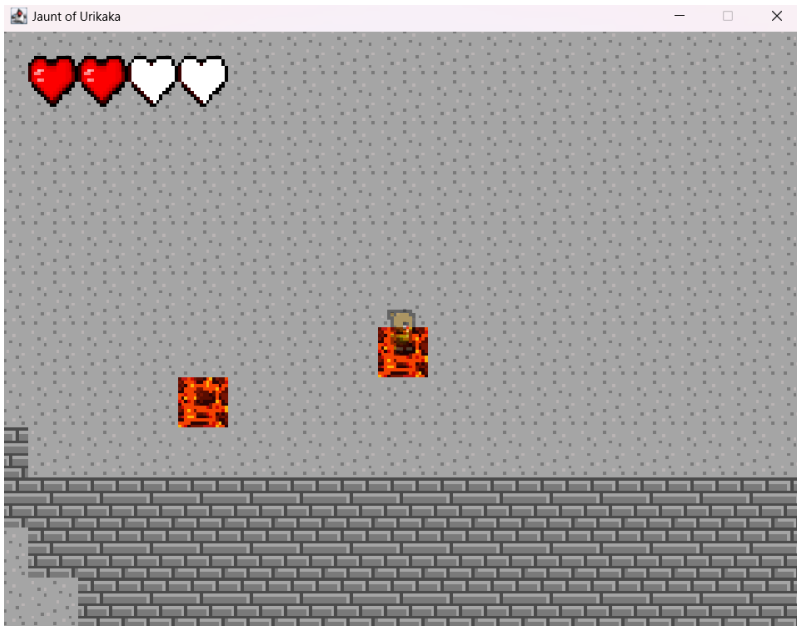
ถ้าเดินเข้าไปใกล้ประตู และ ถ้าไม่มีกุญแจของประตูนั้นๆ ประตูจะไม่สามารถเปิดออกได้

แต่ถ้ามีกุญแจของประตูนั้นๆ แล้วถ้าเดินไปใกล้ ประตูจเปิดออก

- Lava block



ถ้าโดน บล็อคลาวา HP จะลด



- End Game

เมื่อเข้าใกล้ สมบัติของราชาปีศาจ จะยังไม่มีอะไรเกิดขึ้น แต่เมื่อ กด ENTER ข้างๆ กับ สมบัติ

เกมจะจบลง และ จะมีข้อความขึ้นมาว่า

“ YOU WIN “

อัลกอริทึมที่สำคัญในโปรแกรม

```java
public void setupGame() {

        aSetter.setObject();
        aSetter.setNPC();
        aSetter.setMonster();
        gameState = titleState;

    }
```

```java
public void startGameThread() {
        gameThread = new Thread(this);
        gameThread.start();
    }
```

- อัลกอริทึมการเริ่มเกม (ในเมธอด setupGame และ startGameThread): ตั้งค่าเกมและเริ่มเทรดเกม

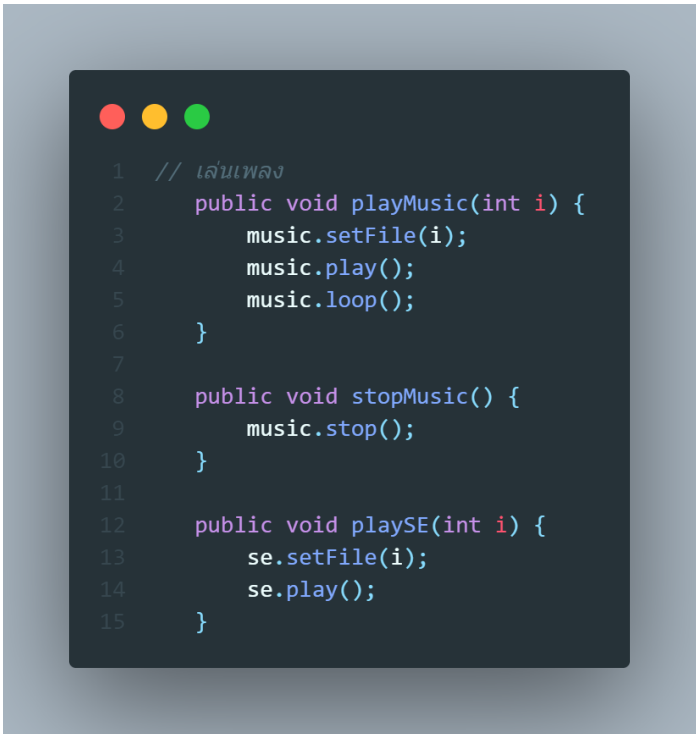
```java
public void run() {

        double drawInterval = 1000000000 / FPS; // 1 second = 1000000000 nanosecond
        double delta = 0;
        long lasttime = System.nanoTime();
        long currentTime;
        long timer = 0;
        int drawCount = 0;

        while (gameThread != null) {

            currentTime = System.nanoTime();

            delta += (currentTime - lasttime) / drawInterval;
            timer += (currentTime - lasttime);
            lasttime = currentTime;

            if (delta >= 1) {
                // เพิ่มตัวละคร
                update();
                // วาดตัวละคร
                repaint();
                delta--;
                drawCount++;
            }

            if (timer >= 1000000000) {
                System.out.println("FPS : " + drawCount);
                drawCount = 0;
                timer = 0;
            }

        }
    }
```

```java
public void update() {
        if (gameState == playState) {
            // อัพเดทตัวละคร
            player.update();

            // NPC
            for (int i = 0; i < npc[1].length; i++) {
                if (npc[currentMap][i] != null) {
                    npc[currentMap][i].update();
                }
            }
            for (int i = 0; i < monster[1].length; i++) {
                if (monster[currentMap][i] != null) {
                    if (monster[currentMap][i].alive == true && monster[currentMap][i].dying == false) {
                        monster[currentMap][i].update();
                    }
                    if (monster[currentMap][i].alive == false) {
                        monster[currentMap][i] = null;
                    }

                }
            }

        }
        if (gameState == pauseState) {
            // do nothing
        }
        if (gameState == gameFinished) {
            ui.drawGameFinishedScreen();
            // gameThread = null ;
        }

    }
```

- อัลกอริทึมการอัปเดตสถานะเกม (ในเมธอด run และ update): ควบคุมการอัปเดตสถานะของเกมและวัตถุต่าง ๆ ในเกม

```java
public void paintComponent(Graphics g) {

        super.paintComponent(g);

        Graphics2D g2 = (Graphics2D) g;
        // debug
        long drawStart = 0;
        if (keyH.showDebugText == true) {
            drawStart = System.nanoTime();
        }

        // วาดฉากหลัง
        if (gameState == titleState) {

            ui.draw(g2);
            csManager.draw(g2);


        } else {
            // วาดแผนที่
            tileM.draw(g2);

            entityList.add(player);

            // Add to entityList
            for (int i = 0; i < npc[1].length; i++) {
                if (npc[currentMap][i] != null) {
                    entityList.add(npc[currentMap][i]);
                }
            }

            for (int i = 0; i < obj[1].length; i++) {
                if (obj[currentMap][i] != null) {
                    entityList.add(obj[currentMap][i]);
                }
            }
            for (int i = 0; i < monster[1].length; i++) {
                if (monster[currentMap][i] != null) {
                    entityList.add(monster[currentMap][i]);
                }
            }

            // Sort
            Collections.sort(entityList, new Comparator<Entity>() {

                @Override
                public int compare(Entity e1, Entity e2) {
                    int result = Integer.compare(e1.worldY, e2.worldY);
                    return result;
                }

            });

        }
        // draw entitys
        for (int i = 0; i < entityList.size(); i++) {
            entityList.get(i).draw(g2);
        }
        // Empty entityList
        entityList.clear();

        // วาด UI
        ui.draw(g2);
        if (keyH.showDebugText == true) {
            long drawEnd = System.nanoTime();
            long passed = drawEnd - drawStart;

            g2.setFont(new Font("Arial", Font.PLAIN, 20));
            g2.setColor(Color.white);

            int x = 10;
            int y = 400;
            int lineHeight = 20;

            g2.drawString("WorldX" + player.worldX, x, y);
            y += lineHeight;
            g2.drawString("WorldX" + player.worldY, x, y);
            y += lineHeight;
            g2.drawString("Col" + (player.worldX + player.solidArea.x) / tileSize, x, y);
            y += lineHeight;
            g2.drawString("Row" + (player.worldY + player.solidArea.y) / tileSize, x, y);
            y += lineHeight;
            g2.drawString("Draw Time :  " + passed, x, y);
            y += lineHeight;
            g2.drawString("GodMode" + keyH.godModeOn, x, y);
            y += lineHeight;
            g2.drawString("MAP" + currentMap, x, y);

        }
```

- อัลกอริทึมการวาดกราฟิก (ในเมธอด **paintComponent**): ควบคุมการวาดกราฟิกของเกม



```
// เล่นเพลง
    public void playMusic(int i) {
        music.setFile(i);
        music.play();
        music.loop();
    }

    public void stopMusic() {
        music.stop();
    }

    public void playSE(int i) {
        se.setFile(i);
        se.play();
    }
```

- อัลกอริทึมการจัดการเสียง (ในเมธอด **playMusic, stopMusic,** และ **playSE**): ควบคุมการเล่นและหยุดเสียงในเกม

```
public void endGame() {

        // กำหนดสถานะของเกมว่าจบแล้ว
        gameState = gameFinished;
        stopMusic();

    }
```

- อัลกอริทึมการจัดการการสิ้นสุดเกม (ในเมธอด **endGame**): จัดการสถานะเมื่อเกมสิ้นสุด

บทที่ 3 สรุป

ปัญหาที่พบระหว่างการพัฒนา

- การทำงานดำเนินไปอย่างล่าช้าไปเป็นไปตามกำหนด
- ชิ้นงานออกมาไม่ค่อยเป็นที่พอใจ เนื่องจากมีความรู้ที่ไม่เพียงพอ

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

- เป็นเกมที่ออกแบบมาแบบไม่ซ้ำใคร ไม่ซ้ำในทีนี้หมายถึงเนื้อเรื่อง และ การออกแบบภายในเกม

คำแนะนำสำหรับผู้สอนที่อยากให้อธิบาย หรือที่เรียนแล้วไม่เข้าใจ หรืออยากให้เพิ่มสำหรับน้อง ๆ รุ่นต่อไป

- อยากให้อาจารย์ผู้สอน ช่วยตามคนที่ตามเนื้อหาไม่ทัน คนที่ไม่ทันในที่นี้หมายถึงคนที่อ่อน Coding หรือ ขาดเรียน บางทีอยากจะสอบถามแต่ก็อาจจะไม่กล้าถามอาจารย์ ครับ
- อยากให้คะแนน Project มากกว่านี้ด้วยครับ