



โครงการ

Mini Project

จัดทำโดย

6504062620051 ธนพ แสงจันทร์

เสนอ

ผู้ช่วยศาสตราจารย์ สกิต ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2566

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เกี่ยวกับโครงการ

ชื่อโปรเจค: Jaunt of Urikaka

นำเสนอโดย: นาย ธนพ แสงจันทร์

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

บทที่ 1 ที่มาและความสำคัญของโปรเจ็ค

โครงการนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดย การนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบเว็บ ผู้จัดทำได้สร้างเว็บนี้ขึ้นมา

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			

บทที่ 2 ส่วนการพัฒนา

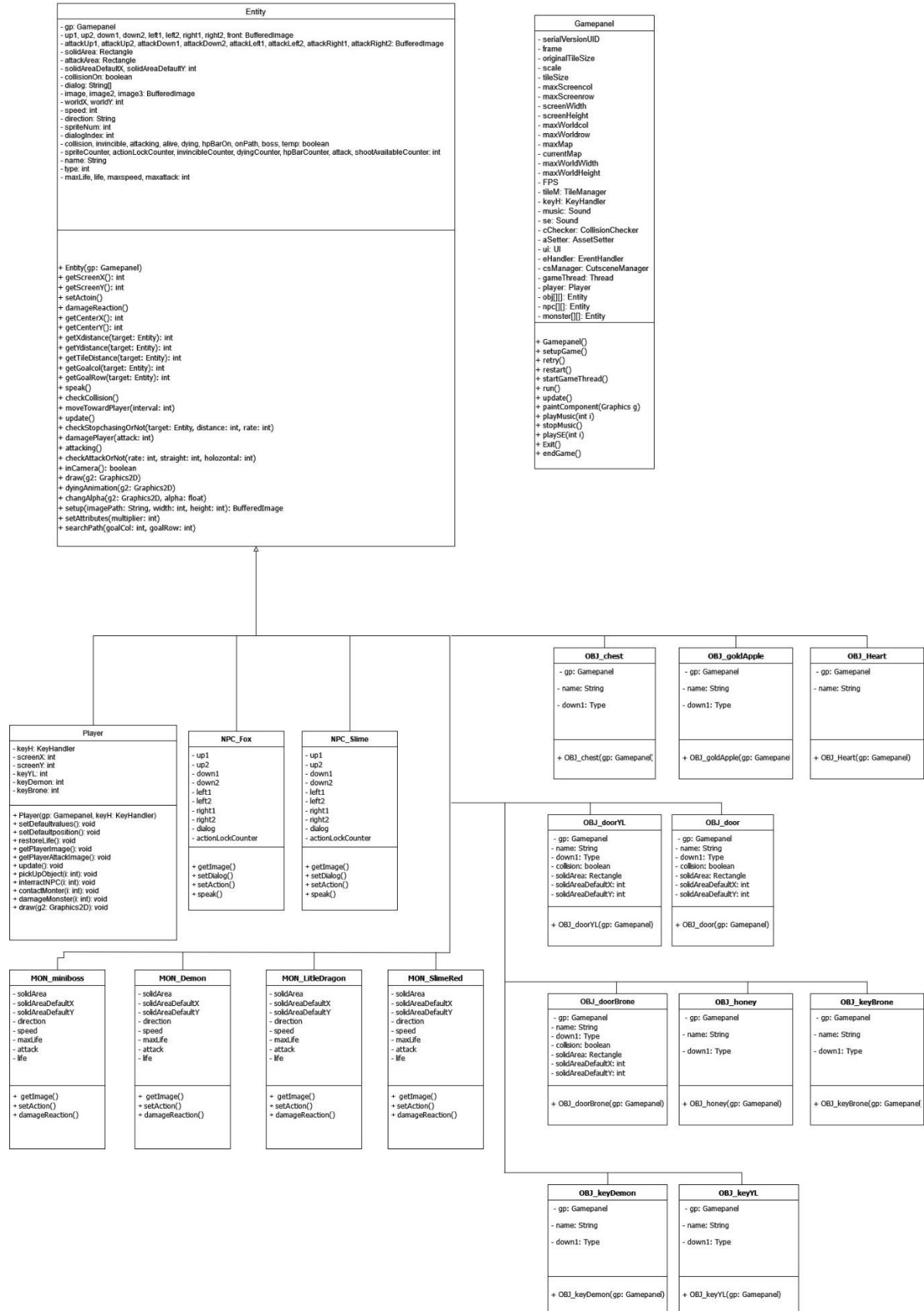
เนื้อเรื่องย่อ

- เจ้าหนูอุริก้าได้ตื่นขึ้นมาในกระท่อมแห่งหนึ่ง 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_LittleDragon , 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

```
1 public Gamepanel() {  
2  
3     this.setPreferredSize(new DimensionUIResource(screenWidth, screenHeight));  
4     this.setBackground(Color.black);  
5     this.setDoubleBuffered(true);  
6     this.addKeyListener(keyH);  
7     this.setFocusable(true);  
8 }
```

- **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



```
1  public Player(Gamepanel gp, KeyHandler keyH) {  
2  
3      super(gp);  
4      this.keyH = keyH;  
5  
6      screenX = gp.screenWidth / 2 - (gp.tileSize / 2);  
7      screenY = gp.screenHeight / 2 - (gp.tileSize / 2);  
8  
9      solidArea = new Rectangle();  
10     solidArea.x = 8;  
11     solidArea.y = 16;  
12     solidAreaDefaultX = solidArea.x;  
13     solidAreaDefaultY = solidArea.y;  
14     solidArea.width = 32;  
15     solidArea.height = 32;  
16  
17     attackArea.width = 36;  
18     attackArea.height = 36;  
19  
20     setDefaultvalues();  
21     getPlayerImage();  
22     getPlayerAttackImage();  
23 }
```

- **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`


```

1 public UI(Gamepanel gp) {
2     this.gp = gp;
3
4     try {
5         InputStream is = getClass().getResourceAsStream("/res/font/x12y16pxMaruMonica.ttf");
6         maruMoica = Font.createFont(Font.TRUETYPE_FONT, is);
7         is = getClass().getResourceAsStream("/res/font/Purisa Bold.ttf");
8         purisaB = Font.createFont(Font.TRUETYPE_FONT, is);
9     } catch (FontFormatException e) {
10        e.printStackTrace();
11    } catch (IOException e) {
12        e.printStackTrace();
13    }
14
15    // heart
16    Entity heart = new OBJ_Heart(gp);
17    heart_full = heart.image;
18    heart_half = heart.image2;
19    heart_blank = heart.image3;
20 } // 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

```

1  public TileManager(Gamepanel gp) {
2      this.gp = gp;
3
4      tile = new Tile[100];
5
6      mapTileNum = new int[gp.maxMap][gp.maxWorldcol][gp.maxWorldrow];
7
8      getTileImage();
9
10     loadMap("/res/maps/worldmap01.txt",0);
11     loadMap("/res/maps/worldmap02.txt",1);
12     loadMap("/res/maps/worldmap03.txt",2);
13
14 }

```

- **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

```
1  TileManager tileM = new TileManager(this);
2      public KeyHandler keyH = new KeyHandler(this);
3      Sound music = new Sound();
4      Sound se = new Sound(); // sound effect
5      public CollisionChecker cChecker = new CollisionChecker(this);
6      public AssetSetter aSetter = new AssetSetter(this);
7      public UI ui = new UI(this);
8      public EventHandler eHandler = new EventHandler(this);
9      public CutsceneManager csManager = new CutsceneManager(this);
10     public Player player = new Player(this, keyH);
```

- Composition ใน class Gamepanel

➤ Polymorphism

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

```

1 for (int i = 0; i < npc[1].length; i++) {
2     if (npc[currentMap][i] != null) {
3         npc[currentMap][i].update();
4     }
5 }

```

```

1 for (int i = 0; i < entityList.size(); i++) {
2     entityList.get(i).draw(g2);
3 }

```

- 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

```

1 public class Gamepanel extends JPanel implements Runnable

```

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

คลาส Player นี่เป็นคลาสที่สืบทอดมาจาก Entity และสามารถมีคุณสมบัติและเมธอดเพิ่มเติมที่เฉพาะเจาะจงสำหรับ Player ได้

```

1 public class Player extends Entity

```

```

1 public class MON_miniboss extends Entity

```

คลาส MON_miniboss นี่เป็นคลาสที่สืบทอดมาจาก Entity และสามารถมีคุณสมบัติและเมธอดเพิ่มเติมที่เฉพาะเจาะจงสำหรับ MON_miniboss ได้

GUI ประกอบด้วย Component อะไรบ้าง

- หน้า TitleState



GUI หน้านี้เกิดจาก

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

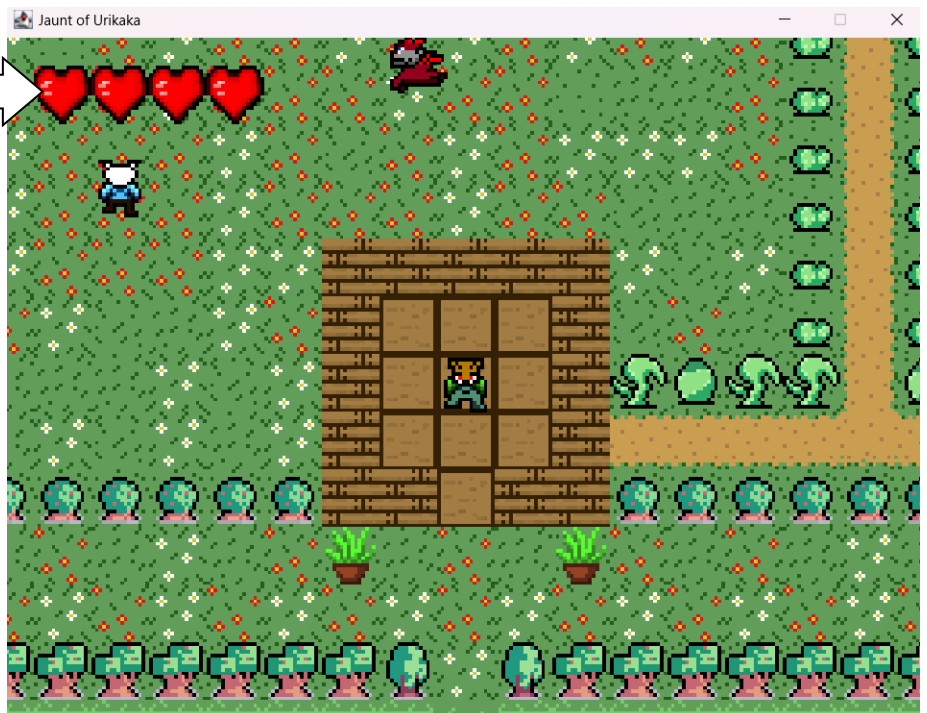
`g2.drawImage(gp.player.front, x, y, gp.tileSize * 2, gp.tileSize * 2, null); > รูป ตัวละคร`

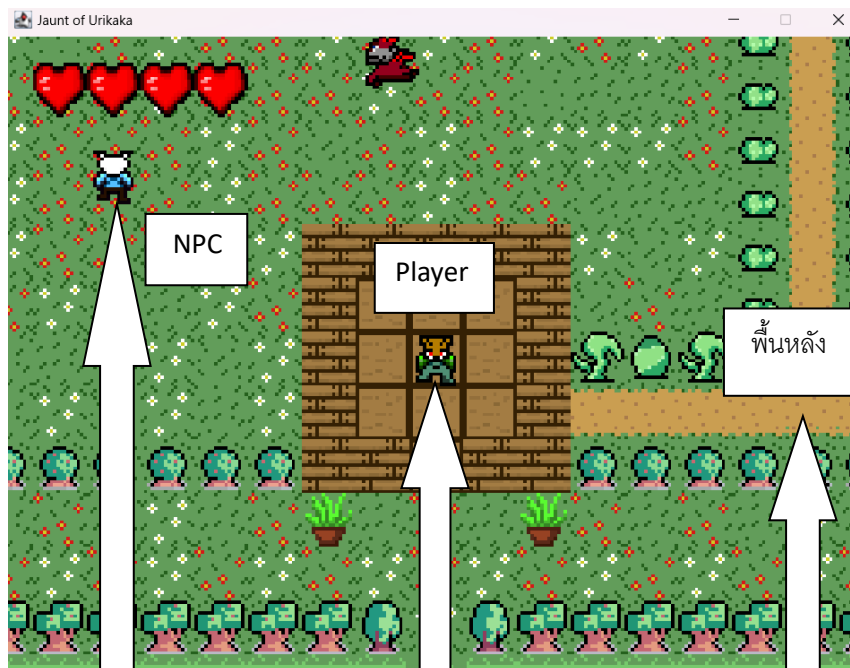
`g2.drawString(text, x, y); > GAME START , QUIT`

- หน้า PlayState

รูปหัวใจ เกิดจาก

```
1 public void drawPlayerLife() {  
2  
3     // gp.player.life = 6 ;  
4  
5     int x = gp.tileSize / 2;  
6     int y = gp.tileSize / 2;  
7     int i = 0;  
8  
9     // max life  
10    while (i < gp.player.maxlife / 2) {  
11        g2.drawImage(heart_blank, x, y, null);  
12        i++;  
13        x += gp.tileSize;  
14    }  
15  
16    // reset  
17    x = gp.tileSize / 2;  
18    y = gp.tileSize / 2;  
19    i = 0;  
20  
21    // current life  
22    while (i < gp.player.life) {  
23        g2.drawImage(heart_half, x, y, null);  
24        i++;  
25        if (i < gp.player.life) {  
26            g2.drawImage(heart_full, x, y, null);  
27        }  
28        i++;  
29        x += gp.tileSize;  
30    }  
31 }  
32 }
```

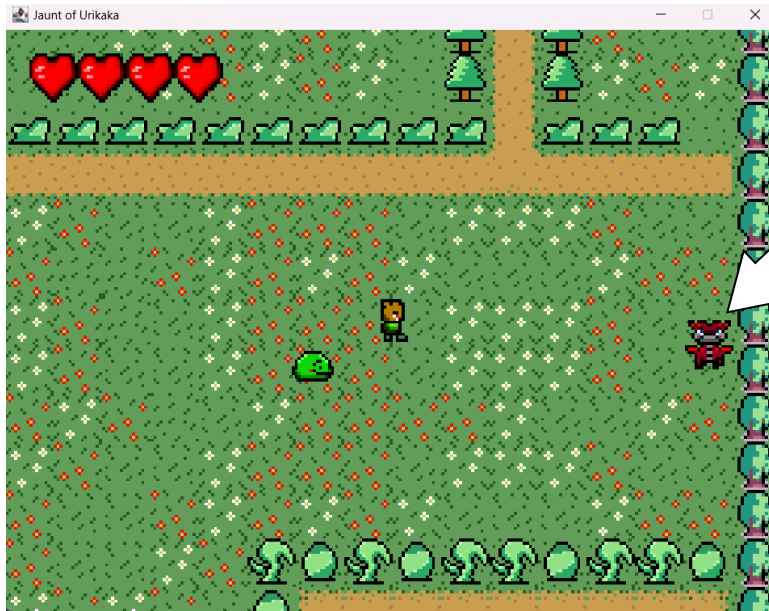




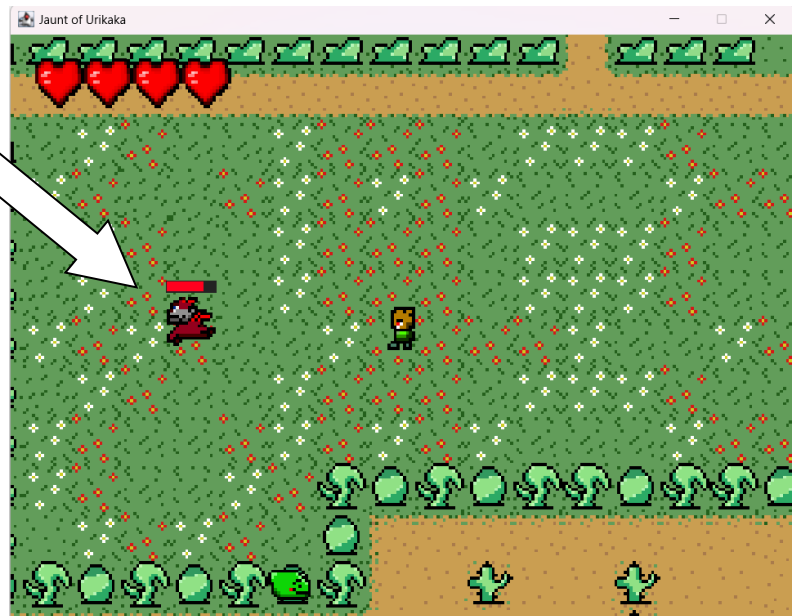
```
1  tileM.draw(g2);
2
3      entityList.add(player);
4
5      // Add to entityList
6      for (int i = 0; i < npc[1].length; i++) {
7          if (npc[currentMap][i] != null) {
8              entityList.add(npc[currentMap][i]);
9          }
10     }
11
12     for (int i = 0; i < obj[1].length; i++) {
13         if (obj[currentMap][i] != null) {
14             entityList.add(obj[currentMap][i]);
15         }
16     }
17     for (int i = 0; i < monster[1].length; i++) {
18         if (monster[currentMap][i] != null) {
19             entityList.add(monster[currentMap][i]);
20         }
21     }
22
23     // Sort
24     Collections.sort(entityList, new Comparator<Entity>() {
25
26         @Override
27         public int compare(Entity e1, Entity e2) {
28             int result = Integer.compare(e1.worldY, e2.worldY);
29             return result;
30         }
31     });
32
33 }
34
35 // draw entitys
36 for (int i = 0; i < entityList.size(); i++) {
37     entityList.get(i).draw(g2);
38 }
```

Event handling

- HP BAR

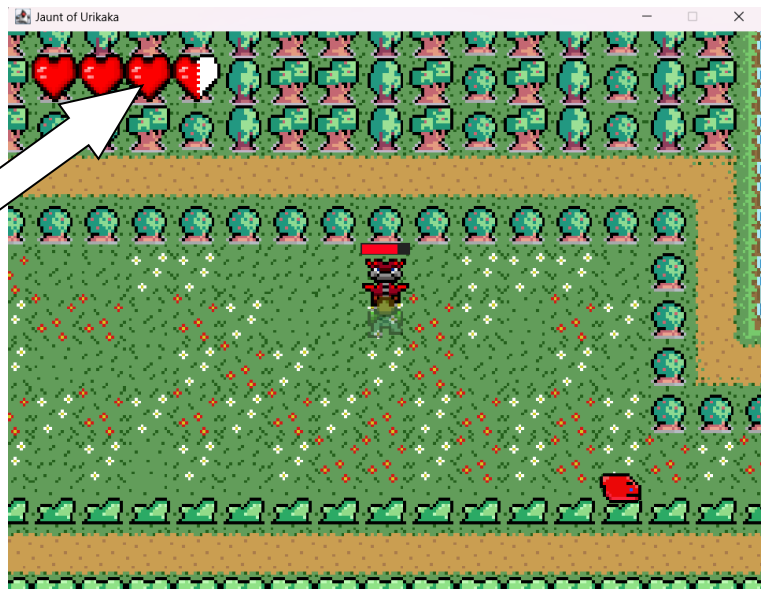


แต่เมื่อโจมตี จะมีหลอดเลือดขึ้นมา และ
พฤติกรรมตอบสนองต่อการโจมตีก็จะไม่
เหมือนกัน แล้วแต่ว่าจะเป็นตัวไหน



- Damage Player

เมื่อถูก Monster โจมตี เลือดจะลดตามพลังโจมตีของ Monster นั้นๆ และ จะมีสถานะ คงกระพันเป็นเวลาหนึ่ง ซึ่งตัวจะกระพริบ



- Talking to NPC

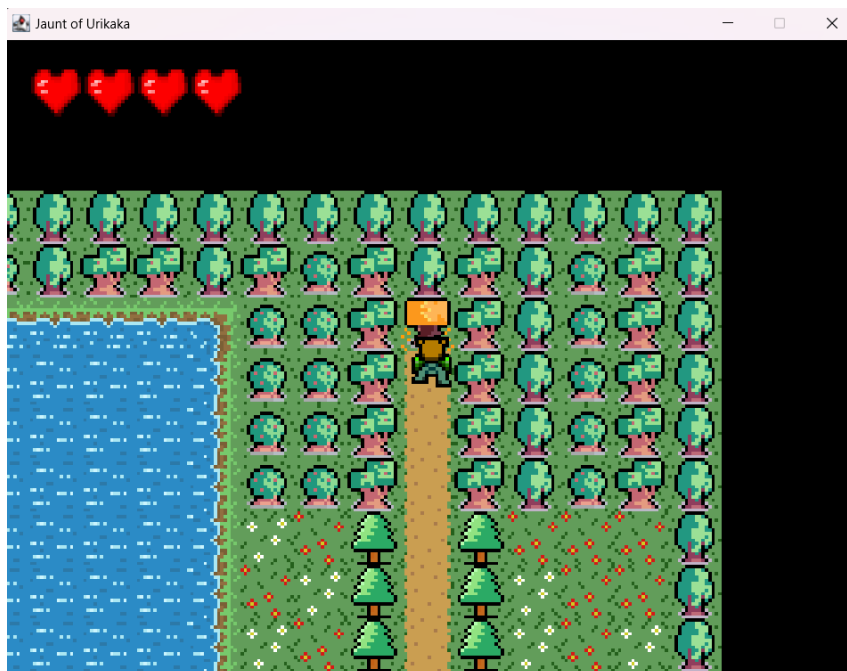


เมื่อเข้าไปใกล้ NPC จะยังไม่มีอะไรเกิดขึ้น

เมื่อเข้าไปใกล้ NPC และกด ENTER จะเกิดเป็น
ข้อความปรากฏขึ้นบนหน้าจอ

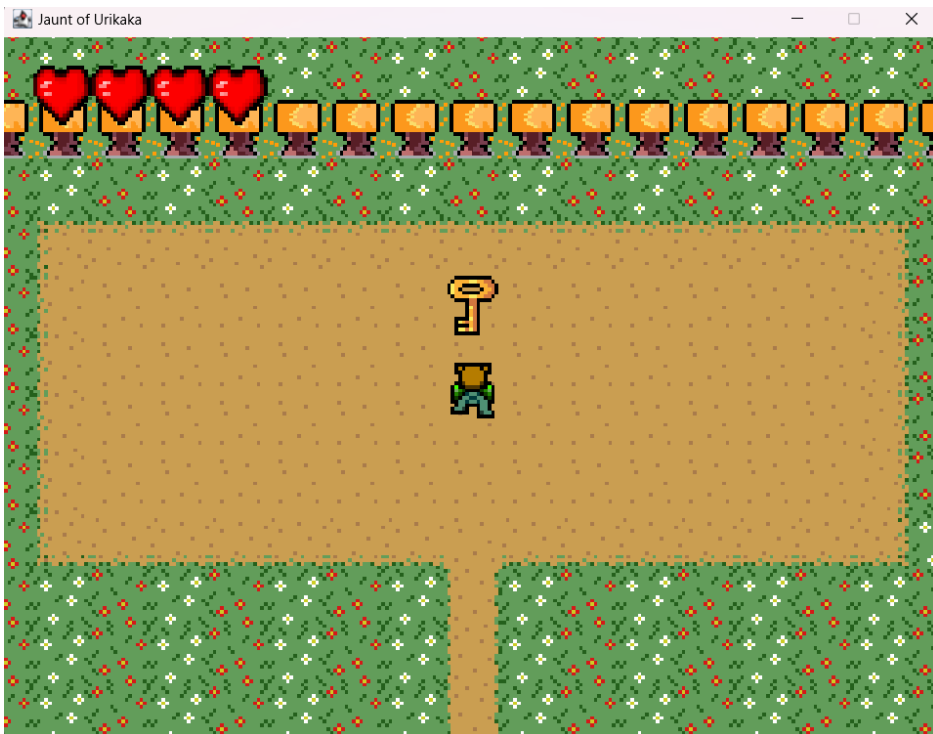


- Teleport with ENTER



เมื่อ กด ENTER ที่จุดนี้จะวarpไปอีก
ที่หนึ่ง

- Pick items



ถ้าไม่เดินเข้าไปใกล้ items จะไม่ทำการเก็บ
แต่ถ้าเดินเข้าไปที่จุดนั้นๆ จะทำการเก็บ
items อัตโนมัติ



- Opened Door

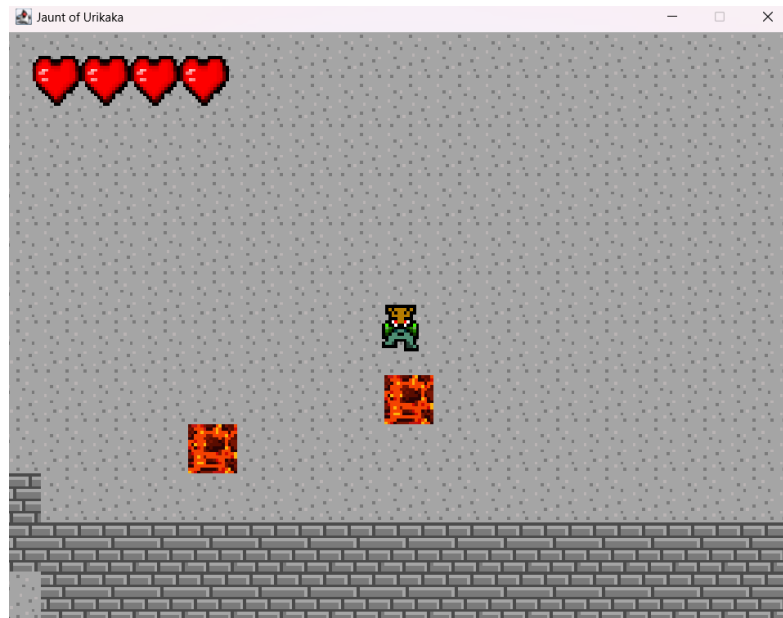


ถ้าเดินเข้าไปใกล้ประตู และ ถ้าไม่มี
กุญแจของประตูนั้นๆ ประตูจะไม่สามารถเปิดออกได้

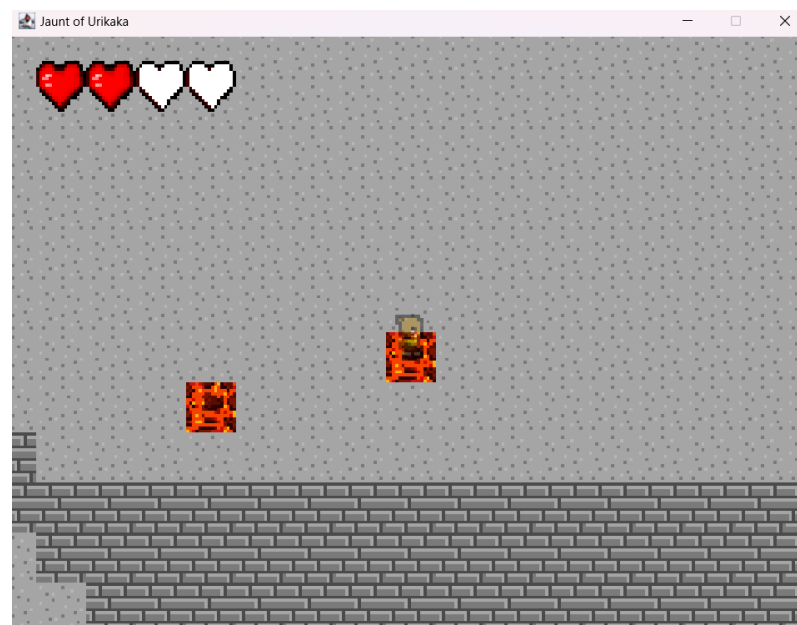
แต่ถ้ามีกุญแจของประตูนั้นๆ แล้วถ้า
เดินไปใกล้ ประตูจะเปิดออก



- Lava block



ถ้าโดน บล็อกลาวา HP จะลด



อัลกอริทึมที่สำคัญในโปรแกรม

```
1 public void setupGame() {  
2  
3     aSetter.setObject();  
4     aSetter.setNPC();  
5     aSetter.setMonster();  
6     gameState = titleState;  
7  
8 }
```

```
1 public void startGameThread() {  
2     gameThread = new Thread(this);  
3     gameThread.start();  
4 }
```

- อัลกอริทึมการเริ่มเกม (ในเมธอด setupGame และ startGameThread): ตั้งค่าเกมและเริ่มเทรดเกม

```
1 public void run() {  
2  
3     double drawInterval = 1000000000 / FPS; // 1 second = 1000000000 nanosecond  
4     double delta = 0;  
5     long lastTime = System.nanoTime();  
6     long currentTime;  
7     long timer = 0;  
8     int drawCount = 0;  
9  
10    while (gameThread != null) {  
11  
12        currentTime = System.nanoTime();  
13  
14        delta += (currentTime - lastTime) / drawInterval;  
15        timer += (currentTime - lastTime);  
16        lastTime = currentTime;  
17  
18        if (delta >= 1) {  
19            // เพิ่มตัวเลข  
20            update();  
21            // วาดตัวเลข  
22            repaint();  
23            delta--;  
24            drawCount++;  
25        }  
26  
27        if (timer >= 1000000000) {  
28            System.out.println("FPS : " + drawCount);  
29            drawCount = 0;  
30            timer = 0;  
31        }  
32    }  
33 }
```

```
1 public void update() {  
2     if (gameState == playState) {  
3         // อัปเดตตัวเลข  
4         player.update();  
5  
6         // NPC  
7         for (int i = 0; i < npc[1].length; i++) {  
8             if (npc[currentMap][i] != null) {  
9                 npc[currentMap][i].update();  
10            }  
11        }  
12        for (int i = 0; i < monster[1].length; i++) {  
13            if (monster[currentMap][i] != null) {  
14                if (monster[currentMap][i].alive == true && monster[currentMap][i].dying == false) {  
15                    monster[currentMap][i].update();  
16                }  
17                if (monster[currentMap][i].alive == false) {  
18                    monster[currentMap][i] = null;  
19                }  
20            }  
21        }  
22    }  
23  
24    if (gameState == pauseState) {  
25        // do nothing  
26    }  
27  
28    if (gameState == gameFinished) {  
29        ui.drawGameFinishedScreen();  
30        // gameThread = null ;  
31    }  
32  
33 }
```

- อัลกอริทึมการอัปเดตสถานะเกม (ในเมธอด run และ update): ควบคุมการอัปเดตสถานะของเกมและวัตถุต่างๆ ในเกม

```

1  public void paintComponent(Graphics g) {
2
3      super.paintComponent(g);
4
5      Graphics2D g2 = (Graphics2D) g;
6      // debug
7      long drawStart = 0;
8      if (keyH.showDebugText == true) {
9          drawStart = System.nanoTime();
10     }
11
12     // วาดฉากหลัง
13     if (gameState == titleState) {
14
15         ui.draw(g2);
16         csManager.draw(g2);
17
18     } else {
19         // วาดแผนที่
20         tileM.draw(g2);
21
22         entityList.add(player);
23
24         // Add to entityList
25         for (int i = 0; i < npc[1].length; i++) {
26             if (npc[currentMap][i] != null) {
27                 entityList.add(npc[currentMap][i]);
28             }
29         }
30
31         for (int i = 0; i < obj[1].length; i++) {
32             if (obj[currentMap][i] != null) {
33                 entityList.add(obj[currentMap][i]);
34             }
35         }
36
37         for (int i = 0; i < monster[1].length; i++) {
38             if (monster[currentMap][i] != null) {
39                 entityList.add(monster[currentMap][i]);
40             }
41         }
42
43         // Sort
44         Collections.sort(entityList, new Comparator<Entity>() {
45
46             @Override
47             public int compare(Entity e1, Entity e2) {
48                 int result = Integer.compare(e1.worldY, e2.worldY);
49                 return result;
50             }
51
52         });
53
54     }
55     // draw entitys
56     for (int i = 0; i < entityList.size(); i++) {
57         entityList.get(i).draw(g2);
58     }
59     // Empty entityList
60     entityList.clear();
61
62     // วาด UI
63     ui.draw(g2);
64     if (keyH.showDebugText == true) {
65         long drawEnd = System.nanoTime();
66         long passed = drawEnd - drawStart;
67
68         g2.setFont(new Font("Arial", Font.PLAIN, 20));
69         g2.setColor(Color.white);
70
71         int x = 10;
72         int y = 400;
73         int lineHeight = 20;
74
75         g2.drawString("WorldX" + player.worldX, x, y);
76         y += lineHeight;
77         g2.drawString("WorldY" + player.worldY, x, y);
78         y += lineHeight;
79         g2.drawString("Col" + (player.worldX + player.solidArea.x) / tileSize, x, y);
80         y += lineHeight;
81         g2.drawString("Row" + (player.worldY + player.solidArea.y) / tileSize, x, y);
82         y += lineHeight;
83         g2.drawString("Draw Time : " + passed, x, y);
84         y += lineHeight;
85         g2.drawString("GodMode" + keyH.godModeOn, x, y);
86         y += lineHeight;
87         g2.drawString("MAP" + currentMap, x, y);
88
89     }
90
91     g2.dispose();
92
93 }

```

- อัลกอริทึมการวาดกราฟิก (ในเมธอด `paintComponent`): ควบคุมการวาดกราฟิกของเกม

```
1 // เล่นเพลง
2 public void playMusic(int i) {
3     music.setFile(i);
4     music.play();
5     music.loop();
6 }
7
8 public void stopMusic() {
9     music.stop();
10 }
11
12 public void playSE(int i) {
13     se.setFile(i);
14     se.play();
15 }
```

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

```
1 public void endGame() {
2
3     // กำหนดสถานะของเกมว่าจบแล้ว
4     gameState = gameFinished;
5     stopMusic();
6
7 }
```

- 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 บางทีอาจจะสอบถามแต่ก็อาจจะไม่กล้าถามอาจารย์ ครับ
- อยากให้คะแนน Project มากกว่านี้ด้วยครับ