รายงาน

เรื่อง การสร้างเกมส์

วิชา 040613204 Object Oriented Programming

ชื่อโปรเจ็ค Rocket Shooting Game

(เกมส์ยิงจรวด)

นำเสนอ

อาจารย์ สถิตย์ ประสมพันธ์

จัดทำโดย

นาย อติกานต์ ทนุพันธ์ รหัสนักศึกษา 6604062636739

ภาควิชาการวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทที่ 1

ที่มาและความสำคัญของโปรเจ็ค

ในปัจจุบันเกมคอมพิวเตอร์และเกมมือถือมีการพัฒนาไปอย่างรวดเร็ว และได้รับความนิยมในกลุ่มผู้เล่นทุกวัย เกมต่าง ๆ ไม่เพียงแต่เป็นเครื่องมือสำหรับความบันเทิงเท่านั้น แต่ยังเป็นเครื่องมือในการฝึกทักษะทางความคิด การตัดสินใจ และการทำงานร่วมกับผู้อื่น เกมยิงจรวดที่ผู้พัฒนาสร้างขึ้นในโปรเจ็คนี้จึงมีเป้าหมายในการสร้างประสบการณ์การเล่นที่น่าสนใจ ท้าทาย และสนุกสนาน รวมทั้งใช้เป็นเครื่องมือในการฝึกฝนทักษะการเขียนโปรแกรมในภาษา Java โดยใช้แนวคิดและเทคนิคการพัฒนาเกมต่าง ๆ เช่น การสร้างกราฟิก การจัดการเหตุการณ์ (Event Handling) การใช้คลาสและออบเจ็กต์ (Object-Oriented Programming) รวมถึงการพัฒนาส่วนต่าง ๆ ของเกมในเชิงทฤษฎีและปฏิบัติ

โปรเจ็คนี้มีความสำคัญเนื่องจากสามารถช่วยให้ผู้พัฒนาฝึกทักษะในการเขียนโปรแกรมและการพัฒนาเกม รวมทั้งทำความเข้าใจหลักการออกแบบโปรแกรมและแนวทางการพัฒนาเกมในทางปฏิบัติ โดยเฉพาะอย่างยิ่งการใช้งานกราฟิกในเกม การจัดการกับเหตุการณ์ต่าง ๆ เช่น การเคลื่อนที่ของตัวละคร การยิงจรวด การคำนวณคะแนน และการเปลี่ยนแปลงฉากหลัง ซึ่งทั้งหมดนี้ช่วยให้ผู้พัฒนาสามารถเข้าใจการทำงานของโปรแกรมในมุมมองที่ชัดเจนและลึกซึ้งยิ่งขึ้น

ประเภทของโครงการ

ในกรณีของโปรเจ็คเกมยิงจรวดนี้ จะเป็นโครงการที่จัดอยู่ในประเภท **"โครงการพัฒนาเกม"** หรือ **"เกมแอปพลิเคชัน"** ซึ่งเป็นการพัฒนาโปรแกรมที่ใช้ในกิจกรรมบันเทิงเพื่อให้ผู้เล่นมีประสบการณ์การเล่นที่สนุกสนาน โดยจะใช้ทักษะในการพัฒนาโปรแกรม, การออกแบบกราฟิก, การจัดการกับเหตุการณ์ในเกม, และการประมวลผลข้อมูลต่าง ๆ ในระบบเกม

ประเภทของโครงการนี้สามารถแบ่งออกเป็น:

1. **โครงการพัฒนาเกมคอมพิวเตอร์ (Computer Game Development)**: การพัฒนาเกมที่สามารถเล่นบนคอมพิวเตอร์หรืออุปกรณ์ที่มีระบบปฏิบัติการ โดยในโปรเจ็คนี้ใช้ภาษา Java เป็นเครื่องมือในการพัฒนา
2. **โครงการพัฒนาซอฟต์แวร์**: เนื่องจากเป็นการพัฒนาโปรแกรมที่มีลักษณะเป็นแอปพลิเคชันเพื่อความบันเทิงและฝึกทักษะการเขียนโปรแกรม

3. **โครงการการศึกษาผ่านการพัฒนาเกม (Educational Game Development)**: การพัฒนาเกมที่ไม่เพียงแต่สร้างความบันเทิง แต่ยังช่วยให้ผู้พัฒนาสามารถฝึกทักษะการเขียนโปรแกรม การจัดการข้อมูล และการใช้แนวคิดต่าง ๆ ในการพัฒนาโปรแกรม

โดยสรุป โครงการนี้จะเน้นไปที่การพัฒนาเกมแบบ 2D โดยใช้ Java ซึ่งจะช่วยในการฝึกทักษะและความเข้าใจในการพัฒนาโปรแกรมเกมในแนวทางที่มีประสิทธิภาพ

ประโยชน์ของโครงการ

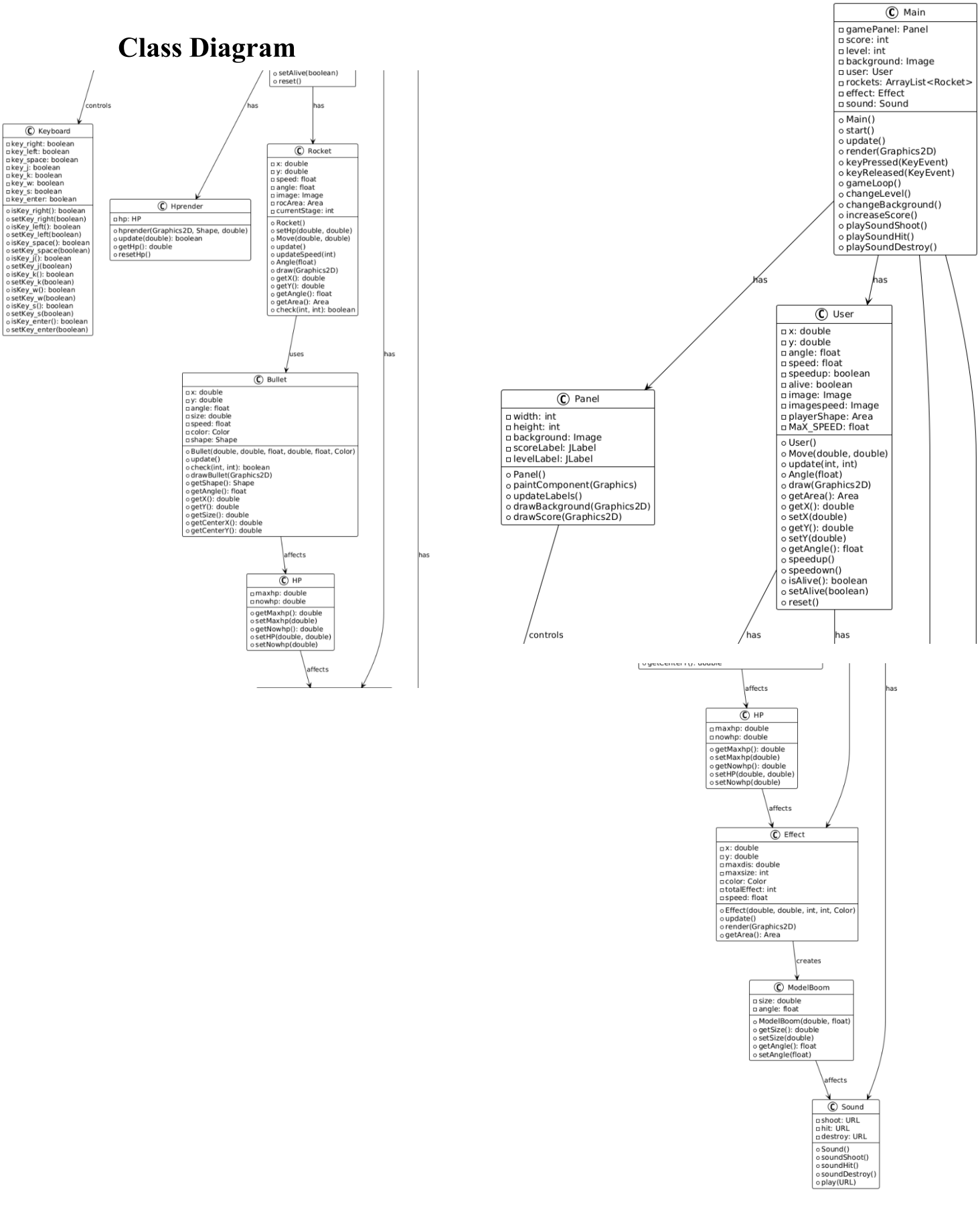
1. ฝึกทักษะการเขียนโปรแกรม
2. การพัฒนาแนวคิดในการออกแบบเกม
3. พัฒนาองค์ความรู้ด้าน GUI
4. ฝึกทักษะการวิเคราะห์และการแก้ปัญหา
5. การพัฒนาเกมเพื่อความบันเทิง
6. การพัฒนาโค้ดที่มีประสิทธิภาพ

ขอบเขตของโครงการ

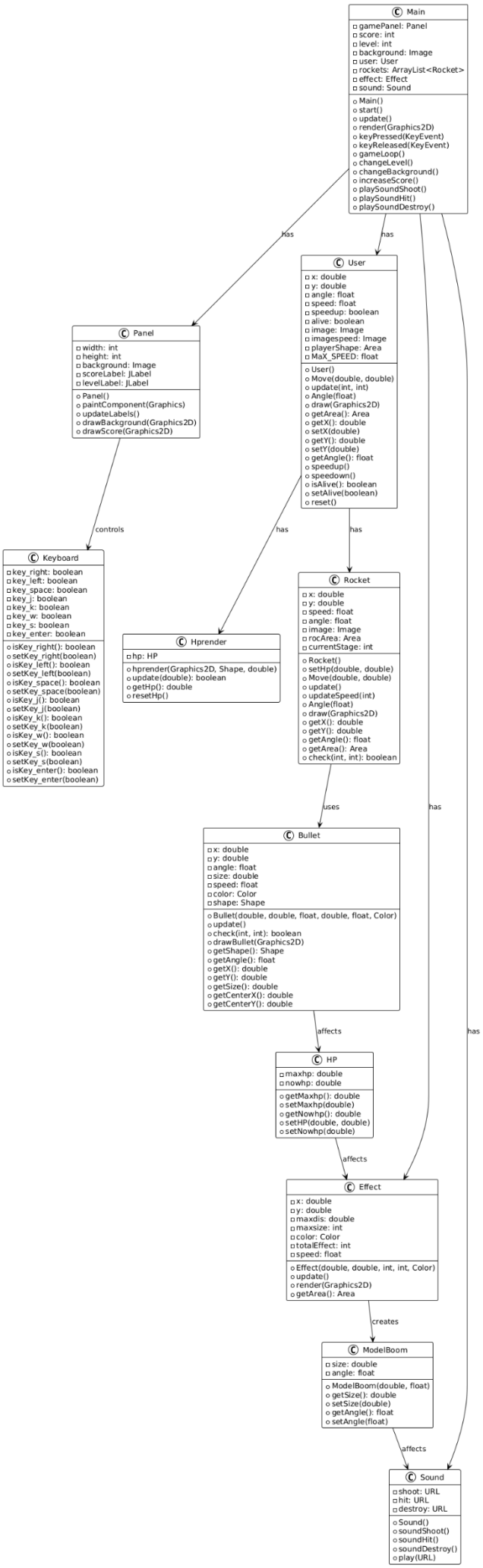
1. การออกแบบและพัฒนาเกมยิงจรวด
2. การสร้างกราฟิกในเกม
3. การพัฒนา GUI (Graphical User Interface)
4. การใช้งานการควบคุมผ่านแป้นพิมพ์
5. การพัฒนาเกมให้สามารถทำงานได้ในระบบคอมพิวเตอร์ทั่วไป
6. การทดสอบและปรับปรุงเกม
7. การเพิ่มฟีเจอร์และปรับปรุงระบบในอนาคต

บทที่ 2

Class Diagram



Class Diagram



อธิบาย Class Diagram นี้

1. Main Class

ฟังก์ชันหลัก: คลาสนี้เป็นตัวหลักในการควบคุมการทำงานของเกมทั้งหมด ซึ่งรวมถึงการเริ่มต้นเกม (start()), การอัปเดตสถานะเกม (update()), การแสดงผล (render(Graphics2D)), การจับการกดปุ่ม (keyPressed(KeyEvent)), การควบคุมการเล่นเสียง (playSoundShoot(), playSoundHit(), playSoundDestroy()), และการเปลี่ยนแปลงระดับ (changeLevel(), increaseScore()).

ความสัมพันธ์: คลาส Main มีความสัมพันธ์กับคลาสอื่นๆ เช่น Panel, User, Effect, และ Sound ซึ่งช่วยควบคุมการแสดงผล, การเคลื่อนที่ของผู้เล่น, การแสดงผลเอฟเฟกต์, และเสียงในเกม.

2. Panel Class

ฟังก์ชันหลัก: คลาสนี้จัดการการแสดงผลบนหน้าจอ โดยประกอบด้วยการวาดพื้นหลัง, การแสดงคะแนน, การอัปเดตข้อความต่างๆ ในเกม เช่น คะแนน และระดับ.

ความสัมพันธ์: คลาส Panel เชื่อมโยงกับ Keyboard ซึ่งทำให้สามารถควบคุมการเคลื่อนที่ของผู้เล่นจากการกดปุ่มได้.

3. Keyboard Class

ฟังก์ชันหลัก: คลาสนี้ทำหน้าที่เก็บสถานะของปุ่มต่างๆ ที่ผู้เล่นกดในเกม เช่น ปุ่มขวา, ซ้าย, พื้นที่, และปุ่มอื่นๆ.

ความสัมพันธ์: คลาส Keyboard ถูกใช้งานใน Panel เพื่อให้สามารถตรวจจับการกดปุ่มและควบคุมผู้เล่นได้.

4. User Class

ฟังก์ชันหลัก: คลาสนี้แทนผู้เล่นในเกม ซึ่งประกอบด้วยคุณสมบัติของผู้เล่น เช่น ตำแหน่ง, ความเร็ว, มุม, การเคลื่อนที่, การแสดงผลภาพ, และสถานะชีวิตของผู้เล่น (alive).

ความสัมพันธ์: คลาส User มีความสัมพันธ์กับ Hprender ที่แสดงค่า HP ของผู้เล่น และ Rocket ที่เป็นอาวุธของผู้เล่น.

5. Hprender Class

ฟังก์ชันหลัก: คลาสนี้มีหน้าที่แสดงผลและอัปเดตสถานะ HP ของผู้เล่น ซึ่งจะช่วยให้สามารถติดตามสุขภาพของผู้เล่นในเกม.

ความสัมพันธ์: คลาส Hprender ใช้ข้อมูลจากคลาส HP ซึ่งเก็บค่า HP สูงสุด (maxhp) และ HP ปัจจุบัน (nowhp).

6. Rocket Class

ฟังก์ชันหลัก: คลาสนี้แทนที่กระสุนในเกม ซึ่งจะมีการเคลื่อนที่ตามมุมและความเร็วที่กำหนดไว้ รวมถึงสามารถตรวจสอบว่ากระสุนยังอยู่ในพื้นที่ที่สามารถแสดงผลได้หรือไม่ (check()).

ความสัมพันธ์: คลาส Rocket ใช้คลาส Bullet สำหรับการสร้างกระสุนที่ยิงจากผู้เล่นและพุ่งไปยังทิศทางที่กำหนด.

7. Bullet Class

ฟังก์ชันหลัก: คลาสนี้แทนที่กระสุนในเกม ซึ่งมีคุณสมบัติของการเคลื่อนที่, ขนาด, สี, และการตรวจสอบว่ากระสุนได้ออกนอกขอบเขตของหน้าจอหรือไม่.

ความสัมพันธ์: คลาส Bullet ส่งผลกระทบต่อคลาส HP เมื่อกระสุนชนกับศัตรู หรือเมื่อยิงใส่ผู้เล่น.

8. HP Class

ฟังก์ชันหลัก: คลาสนี้แทนที่การจัดการ HP ของผู้เล่นและศัตรู โดยมีการเก็บค่า HP สูงสุดและ HP ปัจจุบันที่สามารถเปลี่ยนแปลงได้เมื่อเกิดการโจมตี.

ความสัมพันธ์: คลาส HP ใช้คลาส Effect เพื่อแสดงผลเมื่อ HP ลดลง หรือเกิดการเปลี่ยนแปลงในเกม.

9. Effect Class

ฟังก์ชันหลัก: คลาสนี้จัดการเอฟเฟกต์ต่างๆ ในเกม เช่น การแสดงผลเมื่อเกิดการชน หรือการโจมตี รวมถึงการควบคุมความเร็วและขนาดของเอฟเฟกต์.

ความสัมพันธ์: คลาส Effect ใช้คลาส ModelBoom ในการสร้างการระเบิดหรือผลกระทบจากการโจมตี.

10. ModelBoom Class

ฟังก์ชันหลัก: คลาสนี้สร้างเอฟเฟกต์ระเบิดหรือการกระทำที่เกิดจากการโจมตี เช่น การแสดงผลระเบิดในเกม.

ความสัมพันธ์: คลาส ModelBoom จะส่งผลกระทบต่อคลาส Sound เพื่อเล่นเสียงที่เกี่ยวข้องกับการทำลาย.

11. Sound Class

ฟังก์ชันหลัก: คลาสนี้ใช้ในการเล่นเสียงต่างๆ ในเกม เช่น เสียงยิง, เสียงชน, และเสียงการทำลาย.

ความสัมพันธ์: คลาส Sound ถูกใช้ในหลายๆ คลาสเช่น Main และ ModelBoom เพื่อเล่นเสียงที่เหมาะสมกับเหตุการณ์ในเกม.

ความสัมพันธ์ระหว่างคลาสต่างๆ:

ความสัมพันธ์ has: ตัวอย่างเช่น, Main มี Panel, User, Effect, และ Sound ซึ่งหมายความว่า Main ใช้งานคลาสเหล่านี้ในการควบคุมการทำงานของเกม.

ความสัมพันธ์ controls: Keyboard ควบคุมการกดปุ่มที่ใช้ในการควบคุมผู้เล่น.

ความสัมพันธ์ uses: เช่น Rocket ใช้ Bullet เพื่อยิงกระสุน.

ความสัมพันธ์ affects: เช่น Bullet ส่งผลกระทบต่อ HP เมื่อกระทบกับเป้าหมาย.

อธิบายส่วนของโปรแกรม

1. Constructor

ตัวอย่าง: ในคลาส Bullet, User, Rocket, Sound มีการใช้ constructor เพื่อกำหนดค่าพารามิเตอร์เริ่มต้นให้กับอ็อบเจ็กต์ที่สร้างขึ้น

```
public Bullet(double x, double y, float angle, double size, float speed, Color color) {
    x += User.playersize / 2 - (size / 2);
    y += User.playersize / 2 - (size / 2);
    this.x = x;
    this.y = y;
    this.angle = angle;
    this.size = size;
    this.speed = speed;
    this.color = color;
    shape = new Ellipse2D.Double(0, 0, size, size);
}
```

```
public User() {
    super(new HP(400,400));
    this.image = new ImageIcon(getClass().getResource("/gameimage/rocket1.png")).getImage();
    this.imagespeed = new ImageIcon(getClass().getResource("/gameimage/rocket2.png")).getImage();
    Path2D p = new Path2D.Double();
    p.moveTo(0, 15);
    p.lineTo(20, 5);
    p.lineTo(playersize+15,playersize/2);
    p.lineTo(20, playersize-5);
    p.lineTo(0, playersize-15);
    playerShape = new Area(p);
}
```

```
public Rocket() {
    super(new HP(100, 100));
    this.image = new ImageIcon(getClass().getResource("/gameimage/enermy1.png")).getImage();
    Path2D p = new Path2D.Double();
    p.moveTo(0, ROCKET_SIZE / 2);
    p.lineTo(15, 10);
    p.lineTo(ROCKET_SIZE - 5, 13);
    p.lineTo(ROCKET_SIZE + 10, ROCKET_SIZE / 2);
    p.lineTo(ROCKET_SIZE - 5, ROCKET_SIZE - 13);
    p.lineTo(15, ROCKET_SIZE - 10);
    rocArea = new Area(p);
}
```

```
public Sound() {
    this.shoot = this.getClass().getClassLoader().getResource("gameobject/sound/shoot.wav");
    this.hit = this.getClass().getClassLoader().getResource("gameobject/sound/hit.wav");
    this.destroy = this.getClass().getClassLoader().getResource("gameobject/sound/destroy.wav");
}
```

2. Encapsulation

ตัวอย่าง:ในหลายคลาส เช่น Rocket, User, Sound มีการใช้ private fields เพื่อปกป้องข้อมูลจากการเข้าถึงโดยตรงจากภายนอก และใช้ getter และ setter ในการเข้าถึงข้อมูล

```
// Getter และ Setter
public double getX() { return x; }
public void setX(double x) { this.x = x; }

public double getY() { return y; }
public void setY(double y) { this.y = y; }

public float getSpeed() { return speed; }
public void setSpeed(float speed) { this.speed = speed; }

public boolean isAlive() { return alive; }
public void setAlive(boolean alive) { this.alive = alive; }

public Image getImage() { return image; }
public void setImage(Image image) { this.image = image; }
```

```
// Getter และ Setter
public double getX() { return x; }
public void setX(double x) { this.x = x; }

public double getY() { return y; }
public void setY(double y) { this.y = y; }

public float getSpeed() { return speed; }
public void setSpeed(float speed) { this.speed = speed; }

public float getAngle() { return angle; }
public void setAngle(float angle) { this.angle = angle; }

public Image getImage() { return image; }
public void setImage(Image image) { this.image = image; }
```

```
private double x, y;
private float speed;
public double getX() { return x; }
public void setX(double x) { this.x = x; }
public double getY() { return y; }
public void setY(double y) { this.y = y; }
```

การใช้งาน: การใช้ getter และ setter ช่วยให้สามารถจัดการข้อมูลได้อย่างปลอดภัย โดยไม่สามารถเข้าถึงหรือแก้ไขข้อมูลโดยตรง

3. Composition

ที่เห็นในโค้ดนี้คือ Main เก็บอ็อบเจ็กต์ของ Panel เป็นส่วนหนึ่งในตัวเอง ซึ่งช่วยในการจัดการและแสดงผลกราฟิกส์ของเกม.

Panel panel = new Panel();

ในการสร้างตัวแปร panel ซึ่งเป็นอ็อบเจ็กต์ของคลาส Panel, คลาส Main เก็บคลาส Panel ไว้ในตัวเอง โดยคลาส Main เรียกใช้งาน Panel ในการแสดงผล (UI) ของเกม, โดยการแสดงผลจะเกิดขึ้นใน digi() ซึ่งกำหนด layout และขนาดของหน้าต่างเกม.

panel.start();

ใน windowOpened ของ WindowAdapter, เมื่อหน้าต่างถูกเปิด, ฟังก์ชัน start() ของ Panel จะถูกเรียกใช้งานจาก Main นี่คือการที่คลาส Main ใช้คลาส Panel เพื่อเริ่มการทำงานบางอย่าง เช่น การเริ่มเล่นเกม.

```
Panel panel = new Panel(); // Composition: Main มี Panel
add(panel);

addWindowListener(new WindowAdapter() {
    @Override
    public void windowOpened(WindowEvent e) {
        panel.start(); // Composition: เรียกใช้ method ของ Panel
```

4. Polymorphism

ตัวอย่าง:ในคลาสต่าง ๆ เช่น Rocket และ Bullet มีเมธอดที่มีชื่อเดียวกัน (เช่น update(), draw()) แต่ทำงานแตกต่างกันตามชนิดของอ็อบเจ็กต์

ของ rocket

```
public void update() {
    x += Math.cos(Math.toRadians(angle)) * speed;
    y += Math.sin(Math.toRadians(angle)) * speed;
}
```

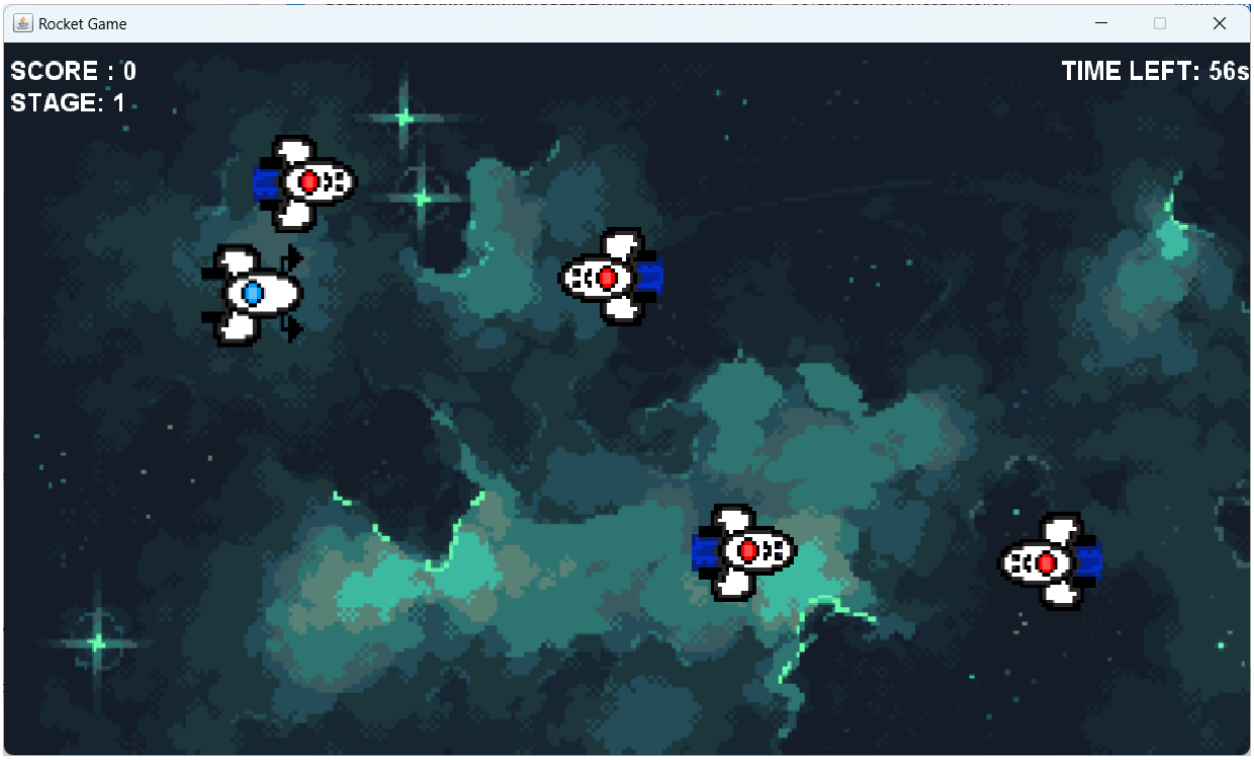
```
public void update() {
    x += Math.cos(Math.toRadians(angle)) * speed;
    y += Math.sin(Math.toRadians(angle)) * speed;

}
```

5 .Inheritance

คลาส Rocket และ User สืบทอด คลาส Hprender ซึ่งทำให้ Rocket ได้รับคุณสมบัติ (properties) และพฤติกรรม (methods) จาก Hprender

```
public class Rocket extends Hprender {
```

```
public class User extends Hprender{
```



GUI ของผมประกอบไปด้วย JFrame และ Panel และ WindowListener

การจัดการเหตุการณ์ (Event Handling)

1. **การควบคุมการกดปุ่ม (KeyEvent):**
    - **เมื่อมีการกดปุ่มบนแป้นพิมพ์, KeyAdapter จะตรวจจับและตั้งค่าผลลัพธ์ลงในตัวแปร key ซึ่งเป็นตัวควบคุมการเคลื่อนที่และการยิงกระสุนของผู้เล่น**
        - **ตัวอย่าง: การกดปุ่ม A หรือ D จะทำให้ผู้เล่นเคลื่อนที่ไปทางซ้ายหรือขวา**
        - **การกดปุ่ม Space จะเพิ่มความเร็วของยาน**
        - **การกดปุ่ม K และ J จะยิงกระสุนที่มีลักษณะต่างๆ (ปืนกล, กระสุนระเบิด)**
2. **การควบคุมการหยุดเกมและเริ่มเกมใหม่:**
    - **เมื่อกด Enter เมื่อเกมจบจะรีเซ็ตเกมใหม่**
    - **การกด Esc จะหยุดเกมชั่วคราว (Pause)**

อัลกอริธึมที่สำคัญในโปรแกรม

1. **การสุ่มภาพพื้นหลัง (Randomizing Background):**
    - **โปรแกรมจะเลือกภาพพื้นหลังแบบสุ่มจากรายการที่เตรียมไว้ โดยการใช้คลาส Random เพื่อสุ่มชื่อไฟล์ภาพพื้นหลัง และโหลดภาพพื้นหลังใหม่ๆ โดยใช้ ImageIO.read().**
2. **การสร้างและอัพเดตอ็อบเจกต์ในเกม:**
    - **โปรแกรมจะสร้างอ็อบเจกต์ต่างๆ เช่น Rockers (ยานศัตรู), Bullet (กระสุน), และ Effect (เอฟเฟกต์) โดยใช้วิธีการ addRocket() เพื่อเพิ่มยานศัตรูลงในเกมทุกๆ 20 คะแนน**
    - **ทุกๆ 3 วินาทีจะสร้างยานศัตรูใหม่เพิ่มขึ้นตามคะแนน**
3. **การตรวจสอบการชน (Collision Detection):**

- **การตรวจสอบการชนระหว่างกระสุน (Bullet) กับยาน (Rocket) หรือผู้เล่น (User) จะถูกตรวจสอบโดยใช้ Area.intersect() ซึ่งจะคำนวณพื้นที่ที่ซ้อนทับกันระหว่างวัตถุต่างๆ เพื่อดูว่ามีการชนหรือไม่**
- **เมื่อกระสุนชนกับยาน, คะแนนจะเพิ่มขึ้นและยานจะถูกลบออกจากเกม**

4. **การอัพเดตการเคลื่อนที่และการยิง:**
    - **ผู้เล่นสามารถเคลื่อนที่ไปทางซ้ายหรือขวาโดยการปรับมุมของยาน (angle) และสามารถยิงกระสุนในทิศทางที่ต้องการได้**
    - **กระสุนจะมีเวลาการยิงที่แตกต่างกันไปตามปุ่มที่กด (K สำหรับยิงกระสุนกล, J สำหรับยิงกระสุนเบา)**
    - **กระสุนแต่ละลูกจะได้รับการอัพเดตทุกครั้งที่มีการเปลี่ยนแปลงและจะถูกลบออกเมื่อออกจากกรอบของหน้าจอ**
5. **การจัดการเวลา:**
    - **โปรแกรมใช้ Timer เพื่อลดเวลาในเกมทุกๆ 1 วินาที และเมื่อเวลาหมด เกมจะจบและรีเซ็ต**
    - **หากผู้เล่นทำคะแนนถึง 100 คะแนน, เกมจะยุติด้วยข้อความว่า "You Win!!"**

บทที่ 3 - สรุป

ปัญหาที่พบระหว่างการพัฒนา

-กำหนด เวลา ตอนที่ เวลาหมดแล้ว Game Over ยากมาก แต่พยามทำ

-ตรวจจับการชนก็ยากบางครั้งมันตรวจจับไม่ใหญ่พอ ทำให้ชนแล้วไม่เกิดอะไร

-สุ่มด่านอาจจะได้ด่านซ้ำ

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

-คิดว่าเกมตัวเองมีเสียงเอฟเฟคเวลาสังหาร ยิง และ ตอนฝ่ายศัตรูโดนยิง

-มีการกด Esc เพื่อหยุด แต่ความจริงมันไม่ใช่หยุดแบบหยุดนิ่งแค่ทำให้มันหยุดช้าลงเพื่อให้มีเวลาคิด

- **รายละเอียดเกมส์**

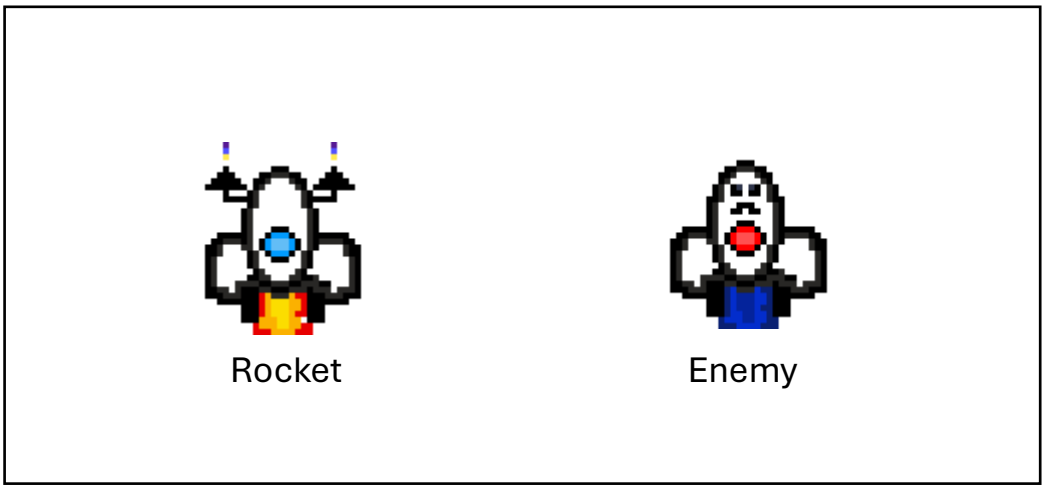
เกมส์ยิงจรวดเมื่อเข้าของจรวดเริ่มเกมจะต้องทำการหลบหลีก บน ล่าง ซ้าย และขวาเพื่อป้องการการโดนชนจากจรวดต่างๆ และเจ้าของจรวดสามารถยิงทำลายจรวดที่มาจากอีกฝั่งด้วยเช่นเดียวกันเล่นไปเรื่อยๆจนกว่า HP จะหมดจึงถือว่าจบเกมส์หรือ Game Over นั้นเอง

- **วิธีการเล่น**

ใช้เม้าเพื่อทำการเล็งไปในทิศทางที่ต้องการจะยิงจรวดออกไปและกดA D และ SPACE BARเพื่อทำการเดินหน้า ถอยหลัง ขึ้นบน และ ลงล่าง

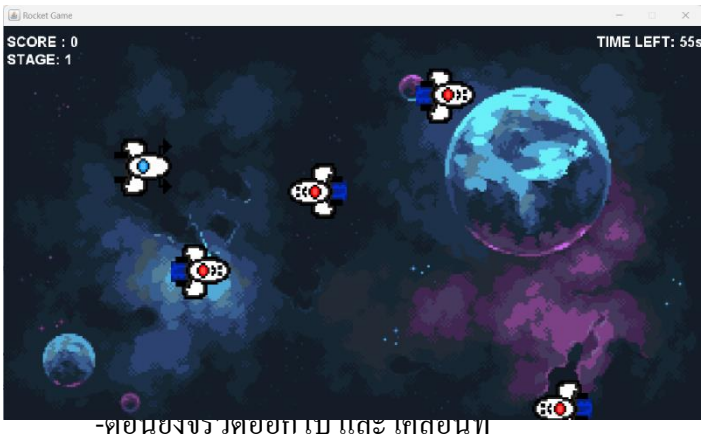
- **Story Board**

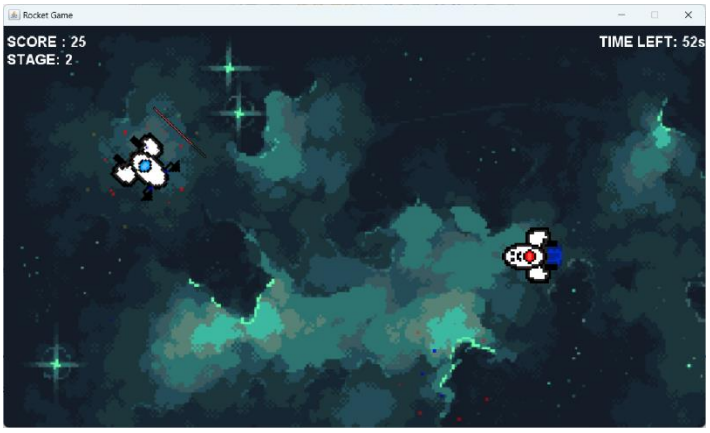
<u>ตัวละคร</u>



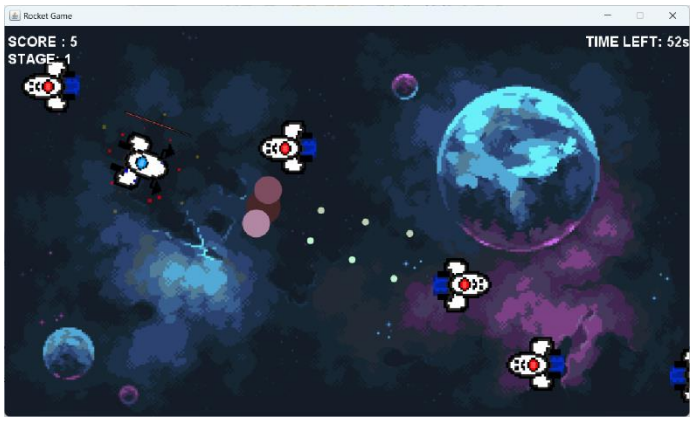
ฉาก

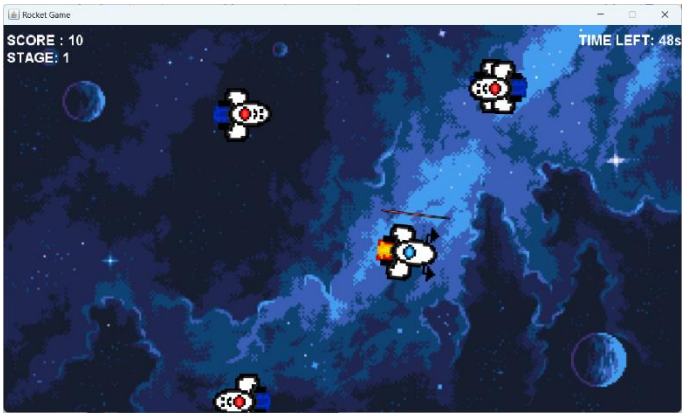
-เริ่มเกมและเข้าด่าน(ด่านจะสุ่ม)และเปลี่ยนไปในแต่ละstage



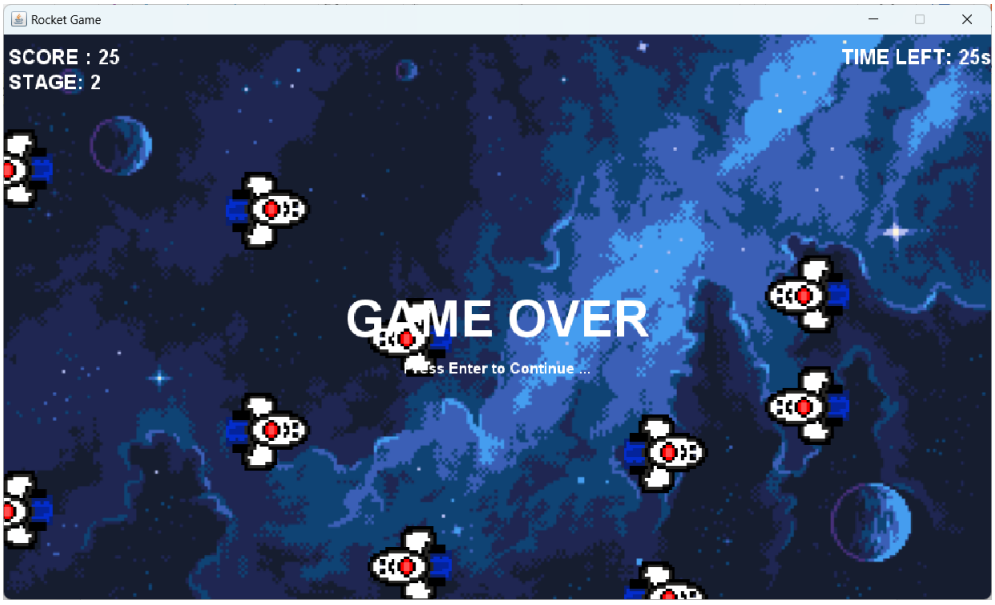


-ตอนยังจรวดออกไป และ โดนอีก





-ตอนจบเกม หรือ Game Over



- **ตารางแผนการทำงานในเดือนกันยายน-พฤศจิกายน**

ลำดับ	รายการ	14-20	21-23	24-30
1	ดีไซน์ตัวละครและฉากต่างๆ			
2	ศึกษาค้นหาข้อมูล			
3	เริ่มลงมือเขียนโปรแกรม			
4	จัดทำเอกสาร			
5	ตรวจสอบความผิดพลาดและแก้ไข			