โครงงาน

Mini Project

จัดทำโดย

6504062630022 นางสาวกิตติมา แพงไพรี

เสนอ

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2566

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทที่ 1 ที่มาและความสำคัญของโครงการ

โครงงานนี้จัดขึ้นเพื่อประยุกต์การเรียนรู้ในรายวิชา Object Oriented Programming ในการสร้างชิ้นงานที่มีความสำคัญและน่าสนใจ โดยผู้จัดทำได้ใช้ความชื่นชอบส่วนตัวที่เกี่ยวข้องกับแสงและสีของเมืองในตอนกลางคืนเพื่อสร้างความเข้าใจในบทเรียนอย่างลึกซึ้งมากยิ่งขึ้น

ผู้จัดทำได้นำเสนอเกมที่มีตัวละครที่สะท้อนตัวเอง ซึ่งเป็นทักษะที่สำคัญในการพัฒนาทักษะด้านเทคนิคและการแก้ปัญหา นอกจากนี้ โครงงานยังเป็นทางเลือกที่ดีในการฝึกฝนการทำงาน การวางแผน การจัดการเวลา และการดำเนินการโครงการที่เป็นที่น่าพอใจ

ผ่านการพัฒนาโครงงานนี้ ผู้จัดทำได้มีโอกาสประยุกต์การเรียนรู้ในการสร้างเกมและการใช้งานหลักการที่ได้เรียนรู้ในวิชา Object Oriented Programming ซึ่งเป็นพื้นฐานที่สำคัญในการพัฒนาซอฟต์แวร์ในปัจจุบัน ทั้งนี้ยังช่วยให้ผู้จัดทำมีความเข้าใจในการนำหลักการเหล่านี้ไปใช้ในการสร้างผลงานที่มีคุณภาพในอนาคตได้อีกด้วย

ประเภทโครงการ

เกม

ประโยชน์

1.ฝึกสมาธิ

2.ฝึกความอดทนและความพยายาม

3.เพื่อความบันเทิง

ขอบเขตของโครงการ

- **ตารางการทำงานเดือนกันยายน-พฤษจิกายน**

ลำดับ	รายการ	5 ก.ย. – 15 ก.ย.	16 ก.ย. – 30 ก.ย.	1 ต.ค. – 31 ต.ค.
1	หารูปจัดตัวละครและกราฟิกต่างๆ			
2	ศึกษาเอกสารและข้อมูลที่เกี่ยวข้อง			
3	ลงมือเขียนโปรแกรม			
4	จัดทำเอกสาร			
5	ตรวจสอบและแก้ไขข้อผิดพลาด			

เล็ท เกิร์ล อัพ

(Let's Girl Up!)

- **รายละเอียดเกม**

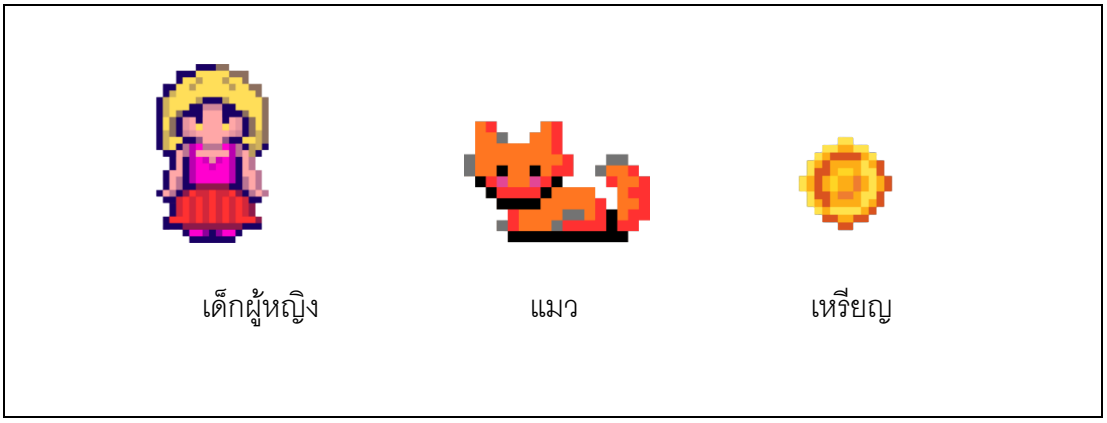
คุณจะต้องควบคุมตัวละครเด็กผู้หญิง ให้กระโดดผ่านสิ่งกรีดขวางต่างๆแล้วเก็บเหรียญ ถ้าหากเด็กผู้หญิงชนสิ่งกรีดขวาง พลังชีวิตของเด็กจะลดลง และถ้าชนสิ่งกรีดขวางจนกระทั่งพลังชีวิตหมดก็จะเป็นการจบเกม คุณจะได้เห็นความแตกต่างเล็กๆน้อยๆของฉากและสิ่งกรีดขวางในเกมที่จะทำให้เกมนี้น่าสนใจมากขึ้น

- **วิธีการเล่น**

เมื่อเริ่มเกม ถ้าเจอสิ่งกรีดขวาง กดปุ่ม Space bar หรือปุ่มลูกศรขึ้น เพื่อบังคับให้เด็กผู้หญิงให้กระโดดขึ้นเพื่อหลบสิ่งกรีดขวาง เมื่อเจอเหรียญให้ทำการเก็บโดยการชนกับเหรียญ

- **Storyboard**

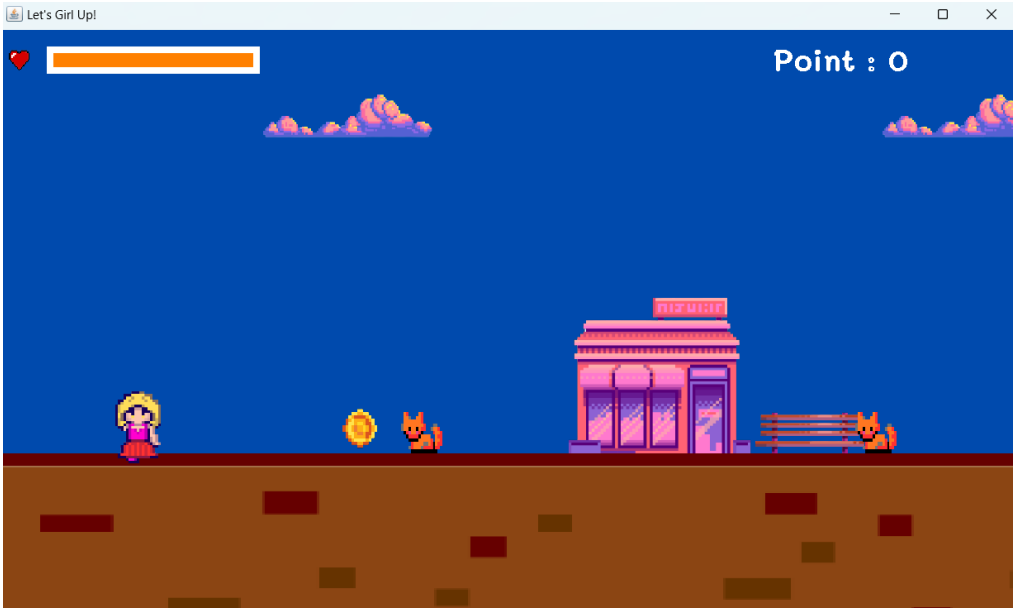
<u>ตัวละคร</u>



ฉาก

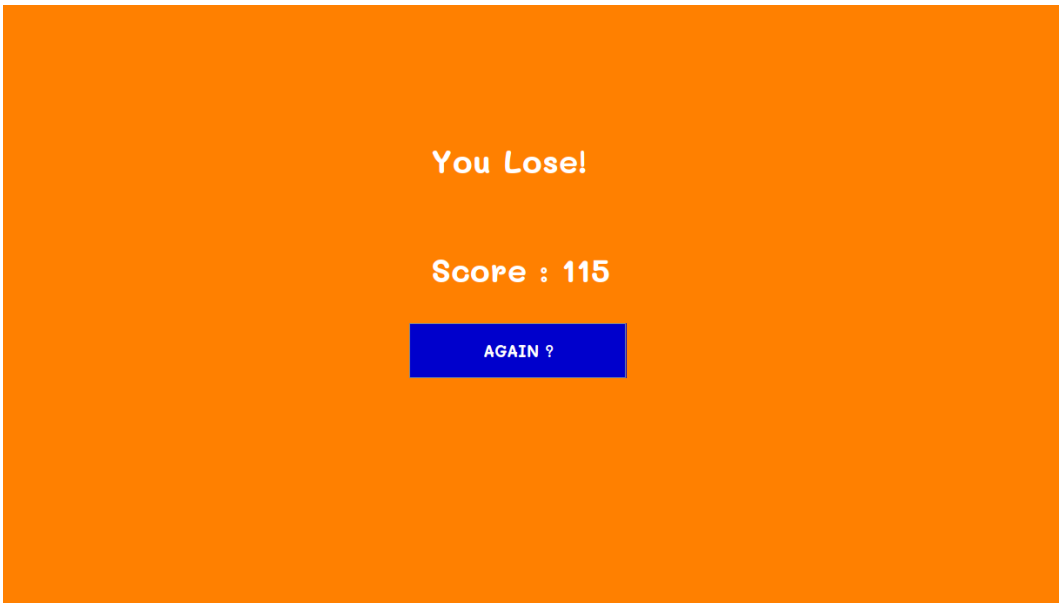
-เริ่มเกม

-เมื่อเริ่มเกมจะมีพลังชีวิตมาให้ และเริ่มนับคะแนนเมื่อเก็บเหรียญได้



-เมื่อเจอสิ่งกรีดขวางให้กดปุ่ม Space bar หรือปุ่มลูกศรขึ้นเพื่อกระโดดหลบสิ่งกรีดขวาง

-เมื่อชนสิ่งกรีดขวางจนพลังชีวิตหมดจะเป็นการจบเกม



บทที่ 2 ส่วนการพัฒนา

2.1) เนื้อเรื่องย่อ

ผู้เล่นจะได้รับบทเป็นเด็กผู้หญิงที่เดินเล่นในเมืองยามค่ำคืน และต้องเจอเข้ากับแมวส้ม เด็กหญิงจะต้องหลบหลีกจากแมวส้มและทำการเก็บเหรียญ จะต้องใช้ทักษะในการกระโดดเพื่อหลบหลีกจากแมวและรักษาพลังชีวิตของตัวละครให้มากที่สุด โดยมีเป้าหมายคือเก็บเหรียญเพื่อให้ได้คะแนนมากที่สุดเท่าที่ทำได้ เกมนี้จะมีกราฟิกที่น่ารักเพื่อให้ผู้เล่นได้รับประสบการณ์ที่น่าสนใจ

2.2) วิธีการเล่น

เมื่อเริ่มเกม ถ้าเจอสิ่งกรีดขวาง กดปุ่ม Space bar หรือปุ่มลูกศรขึ้น เพื่อบังคับให้เด็กผู้หญิงให้กระโดดขึ้นเพื่อหลบสิ่งกรีดขวาง เมื่อเจอเหรียญก็ทำการเก็บเหรียญด้วยการชนเหรียญ

2.3) คลาสไดอะแกรม

- **Game Class (คลาสเกม)**: คลาสนี้เป็นคลาสหลักที่ควบคุมการทำงานของเกม มีตัวแปรและเมธอดต่างๆ เช่น การวาดฉากหลัง, การตรวจสอบการชน, การจัดการการกระโดดของตัวละคร และการจัดการกับการกดแป้นพิมพ์

Game	
- speed: int	แทนความเร็วที่ถูกใช้ในเกม
- dogSize: int	แทนขนาดของตัวละคร
- waveHeight: int	แทนความสูงของสิ่งกรีดขวางในเกม
-pointHeight: int	แทนความสูงของเหรียญที่ใช้นับคะแนน
- base: int	แทนค่าฐานหรือพื้นที่ในเกม
- xStart: int	แทนตำแหน่งเริ่มต้นในแกน x ของเกม
- point: long	แทนคะแนนที่ผู้เล่นได้รับในเกม
- lastPress: long	แทนเวลาที่ผู้เล่นกดครั้งล่าสุด
- dog: Ghost	ตัวแปรที่เกี่ยวข้องกับตัวละคร
- waveSet: Wave[]	อาร์เรย์ของสิ่งกรีดขวางที่เกี่ยวข้องในเกม

-pointset: Point[] อาร์เรย์ของเหรียญที่ใช้นับคะแนนในเกม - envSet: Environment[] อาร์เรย์ของสิ่งแวดล้อมที่เกี่ยวข้องในเกม - building: Environment ตัวแปรที่เกี่ยวข้องกับสิ่งแวดล้อมอาคาร - moon: Environment ตัวแปรที่เกี่ยวข้องกับสิ่งแวดล้อมรถ - store: Environment ตัวแปรที่เกี่ยวข้องกับสิ่งแวดล้อมร้านค้า
+ Game() เมธอดที่ใช้สำหรับสร้างวัตถุ Game และกำหนดค่าเริ่มต้นที่จำเป็น + paint(Graphics): void เมธอดที่ใช้สำหรับการวาดรูปภาพและกราฟิกของเกมทั้งหมด + drawBackground(Graphics2D): void เมธอดที่ใช้ในการวาดพื้นหลังของเกม + drawDogHealth(Graphics2D): void เมธอดที่ใช้ในการวาดแสดงพลังชีวิตของตัวละคร + makeWave(int): Wave[] เมธอดที่ใช้ในการสร้างสิ่งกรีดขวางที่เกี่ยวข้องในเกม + makePoint(int): Point[] เมธอดที่ใช้ในการสร้างเหรียญที่ใช้นับคะแนน + makeEnv(int, int): Environment[] เมธอดที่ใช้ในการสร้างสิ่งแวดล้อมที่เกี่ยวข้องในเกม + drawWave(Wave, Graphics2D): void เมธอดที่ใช้ในการวาดสิ่งกรีดขวางในเกม + drawPoint(Point, Graphics2D): void เมธอดที่ใช้ในการวาดเหรียญที่ใช้นับคะแนน + keyPressed(KeyEvent): void เมธอดที่ใช้ในการจัดการกับการกดแป้นพิมพ์ที่เกิดขึ้นในขณะที่เล่นเกม + keyTyped(KeyEvent): void เมธอดที่ใช้ในการจัดการกับการพิมพ์ที่เกิดขึ้นในขณะที่เล่นเกม + keyReleased(KeyEvent): void เมธอดที่ใช้ในการจัดการกับการปล่อยแป้นพิมพ์ที่เกิดขึ้นในขณะที่เล่นเกม + main(String[]): void เมธอดที่ใช้เป็นจุดเริ่มต้นของโปรแกรม และสร้างวัตถุ Display เพื่อเริ่มการเล่นเกม

- **Ghost Class (คลาสตัวละคร)**: คลาสนี้เป็นคลาสที่จัดการกับตัวละครหลักของเกม มีตัวแปรเกี่ยวกับตำแหน่ง, รูปภาพ, และพลังชีวิตของตัวละคร รวมถึงเมธอดเกี่ยวกับการกระโดดของตัวละคร

Ghost
- x: int ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน x - y: int ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน y - health: int ค่าพลังชีวิตของตัวละคร - speed: int ค่าความเร็วของตัวละคร
+ Ghost() คอนสตรักเตอร์ที่สร้างตัวละครโดยกำหนดค่าเริ่มต้นให้กับตัวแปร

+ Ghost(int, int)	คอนสตรักเตอร์ที่สร้างตัวละครโดยกำหนดตำแหน่งเริ่มต้นให้กับตัวแปร
+ jump(JPanel): void	เมธอดที่ใช้ในการทำให้ตัวละครกระโดดขึ้นและลง
+ getImage():BufferedImage	เมธอดที่ใช้ในการโหลดภาพของตัวละคร

- **Environment Class (คลาสสิ่งแวดล้อม):** คลาสนี้เป็นคลาสที่จัดการกับสิ่งแวดล้อมในเกม เช่น อาคาร, ก้อนเมฆ, รถ มีตัวแปรที่เก็บตำแหน่งและประเภทของสิ่งแวดล้อม และเมธอดที่ใช้ในการดึงรูปภาพสิ่งแวดล้อม

Environment	
- x: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน x ของสิ่งแวดล้อม
- y: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน y ของสิ่งแวดล้อม
- game: Game	อ็อบเจกต์ที่เกี่ยวข้องกับเกมที่ใช้คลาส Environment นี้
- type: int	ประเภทของสิ่งแวดล้อม
- speed: int	ค่าความเร็วของสิ่งแวดล้อม
+ Environment(int, int, Game, int, int)	คอนสตรักเตอร์ที่ใช้ในการสร้างสิ่งแวดล้อม โดยกำหนดตำแหน่งเริ่มต้น ประเภท และความเร็ว ให้กับสิ่งแวดล้อม
+ getImage(): BufferedImage	เมธอดที่ใช้ในการโหลดภาพของสิ่งแวดล้อม

- **Wave Class (คลาสสิ่งกรีดขวาง)**: คลาสนี้เป็นคลาสที่จัดการกับสิ่งกรีดขวางที่เคลื่อนที่ในเกม มีตัวแปรเกี่ยวกับตำแหน่งและความเร็วของสิ่งกรีดขวาง รวมถึงเมธอดที่ใช้ในการดึงรูปภาพของสิ่งกรีดขวาง

Wave	
- x: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน x ของสิ่งกรีดขวาง
- y: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน y ของสิ่งกรีดขวาง
- speed: int	ค่าความเร็วของสิ่งกรีดขวาง
- game: Game	อ็อบเจกต์ที่เกี่ยวข้องกับเกมที่ใช้คลาส Wave นี้
+ Wave(int, int, int, Game)	คอนสตรักเตอร์ที่ใช้ในการสร้างสิ่งกรีดขวาง โดยกำหนดตำแหน่งเริ่มต้น และความเร็วให้กับสิ่งกรีดขวาง

+ getImage(): BufferedImage	เมธอดที่ใช้ในการโหลดภาพของสิ่งกรีดขวาง

- **Point Class (คลาสเหรียญ):** คลาสนี้เป็นคลาสที่จัดการกับเหรียญที่ใช้ในการนับคะแนน มีตัวแปรเกี่ยวกับตำแหน่งและความเร็วของเหรียญ รวมถึงเมธอดที่ใช้ในการดึงรูปภาพของเหรียญ

Point	
- x: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน x ของเหรียญ
- y: int	ตัวแปรที่ใช้เก็บตำแหน่งพิกัดในแกน y ของเหรียญ
- speed: int	ค่าความเร็วของเหรียญ
- game: Game	อ็อบเจกต์ที่เกี่ยวข้องกับเกมที่ใช้คลาส Point นี้
+ Point(int, int, int, Game)	คอนสตรักเตอร์ที่ใช้ในการสร้างเหรียญ โดยกำหนดตำแหน่งเริ่มต้น และความเร็วให้กับเหรียญ
+ getImage(): BufferedImage	เมธอดที่ใช้ในการโหลดภาพของเหรียญ

2.4) รูปแบบการพัฒนา Application / Applet

เป็นการพัฒนา Application ที่ใช้ภาษา Java โดยใช้ไลบรารี Swing ในการสร้างหน้าต่างและกราฟิกสำหรับเกม โดยโค้ดเกมประกอบด้วยหลายคลาสที่ทำหน้าที่ต่าง ๆ ดังนี้:

1. **Charactor.Ghost:** คลาสนี้เป็นคลาสที่เกี่ยวข้องกับตัวละครหลักในเกม มีเมธอดต่าง ๆ เช่น jump และ getImage ที่ใช้ในการควบคุมการกระโดดและการดึงรูปภาพของตัวละคร
2. **display.Game:** คลาสนี้เป็นหน้าต่างหลักของเกม มีเมธอดที่ใช้ในการวาดกราฟิกและการจัดการกับการกระทำต่าง ๆ ในเกม เช่นการวาดพื้นหลัง ตัวละคร การกระโดด และควบคุมการกดปุ่ม
3. **display.Display:** คลาสนี้ใช้สำหรับแสดงหน้าต่างของเกม และมีเมธอดที่ใช้ในการกำหนดการตั้งค่าหน้าต่าง เช่น การกำหนดชื่อและขนาดหน้าต่าง การแสดงผลผลการเล่น และการรีเซ็ตเกม

โดยทั้งสามคลาสนี้เป็นส่วนสำคัญในการพัฒนา Application แบบ Desktop ที่ใช้งานบนเครื่องคอมพิวเตอร์ และไม่ได้ใช้ Applet เนื่องจากไม่ได้นำโค้ดมาใช้งานบนเว็บไซต์ หากต้องการนำโค้ดไปใช้งานในรูปแบบ Applet จะต้องปรับแต่งโค้ดเพิ่มเติมเพื่อให้สามารถทำงานได้อย่างถูกต้องในบริบทของเว็บเบราว์เซอร์ได้อย่างเหมาะสม.

2.5) ส่วนของโปรแกรมที่เกี่ยวกับหลักการOOP

- **Constructor**

-มีการเรียกใช้ Constructor ของคลาส Game

```
39
40    public Game(){
41        this.setBounds(0,0,1000,600);
42        this.addKeyListener(this);
43        this.setLayout(null);
44        this.setFocusable(true);
45
46        playPanel = new Play(new ActionListener() {
47            @Override
48            public void actionPerformed(ActionEvent e) {
49                removePlayPanel();
50                gameStarted = true;
51            }
52        });
53        this.add(playPanel);
54    }
```

-มีการเรียกใช้ Constructor ของคลาส Ghost

```
18
19    public Ghost() {
20    }
21
22    public Ghost(int x,int y) {
23        this.x=x;
24        this.y=y;
25    }
26
```

-มีการเรียกใช้ Constructor ของคลาส Wave

```
17        public Wave(int x,int y,int speed,JPanel page) {
18            this.x = x;
19            this.y = y;
20            this.speed = speed;
21            this.move(page);
22        }
23
```

-มีการเรียกใช้ Constructor ของคลาส Point

```
    public Point(int x, int y, int speed, JPanel page) {
        super(x, y, speed, page);
        // TODO Auto-generated constructor stub
    }
```

-มีการเรียกใช้ Constructor ของคลาส Environment

```
    public Environment(int x,int y,JPanel page,int eType,int speed) {
        this.x = x;
        this.y = y;
        this.startX = x;
        this.speed = speed;
        this.eType = eType;
        this.move(page);
    }
```

-มีการเรียกใช้ Constructor ของคลาส Display

```

    public Display() {
        this.setting();
        this.getContentPane().add(new Game()); // เพิ่ม Game ลงใน JFrame

    }
```

-มีการเรียกใช้ Constructor ของคลาส Menu

```
    public Menu() {
        //----
    }

    public Menu(long point,ActionListener main) {
        try {
                this.point = point;
                this.setBackground(new Color(255, 128, 0));
                this.setBounds(0,0,1000,600);
                this.setFocusable(true);
                this.setLayout(null);

                EleLabel status = new EleLabel("You Lose!",30,400,100,200,100);
                status.setForeground(Color.white);

                EleLabel showPoint = new EleLabel("Score : "+this.point,30,400,200,200,100);
                showPoint.setForeground(Color.white);

                EleButton restart = new EleButton("AGAIN ?",15,380,300,200,50);
                restart.addActionListener(main);

                this.add(showPoint);
                this.add(status);
                this.add(restart);
            } catch (Exception e) {
                e.printStackTrace();
            }
```

-มีการเรียกใช้ Constructor ของคลาส Play

```
15
16      public Play() {
17          //----
18      }
19
20      public Play(ActionListener main) {
21          try {
22                  this.setBackground(new Color(255, 128, 0));
23                  this.setBounds(0,0,1000,600);
24                  this.setFocusable(true);
25                  this.setLayout(null);
26
27                  EleLabel status = new EleLabel("Let's Girl Up!",30,400,100,200,100);
28                  status.setForeground(Color.white);
29
30                  EleButton play = new EleButton("PLAY",15,380,300,200,50);
31                  play.addActionListener(main);
32
33                  this.add(status);
34                  this.add(play);
35          } catch (Exception e) {
36              e.printStackTrace();
37          }
38
39      }
40  }
```

- **Encapsulation**

-ประกาศตัวแปร jumpImage และ isJumping ในคลาส Ghost ในส่วนของ private

```
19
20      private BufferedImage jumpImage;
21      private boolean isJumping = false;
22
```

- **Composition**

- คลาส Game มีความสัมพันธ์แบบ Composition กับคลาส Ghost, Wave, Point และ Environment

```
26
27      private Ghost dog = new Ghost(100,base-50);
28      static Display display;
29
30  //  ------------------Wave SIze ----------------------------
31      private Wave[] waveSet = makeWave(4);
32  //-------------------Environment--------------------------
33      private Environment[] envSet = makeEnv(2,Environment.CLOUD);
34      private Environment building = new Environment(xStart-100,base-150,this,Environment.BUILDING,5);
35      private Environment moon = new Environment(xStart+600,base-150,this,Environment.MOON,8);
36  //-----------------Point Size-------------------------------
37      private Point[] pointset = makePoint(4);
38
```

-คลาส Display มีความสัมพันธ์แบบ Composition กับคลาส Game และ Menu

```
14
15●     public Display() {
16          this.setting();
17          this.getContentPane().add(new Game()); // เพิ่ม Game ลงใน JFrame
18
19      }
20
```

```
34
35●     public void endGame(long point) {
36          removeContent();
37          this.getContentPane().add(new Menu(point,this));
38      }
```

-คลาส Menu มีความสัมพันธ์แบบ Composition โดยการเพิ่ม showPoint, status, restart ลงใน Menu

```
36
37                      this.add(showPoint);
38                      this.add(status);
39                      this.add(restart);
40              } catch (Exception e) {
41                  e.printStackTrace();
```

-คลาส Play มีการใช้งานหลักการ Composition โดยการสร้างอ็อบเจกต์ EleLabel และ EleButton แล้ว นำมาใช้ในคลาส Play

```
EleLabel status = new EleLabel("Let's Girl Up!",30,400,100,200,100);
status.setForeground(Color.white);

EleButton play = new EleButton("PLAY",15,380,300,200,50);
play.addActionListener(main);

this.add(status);
this.add(play);
```

- **Polymorphism** -มีการใช้หลักการ Polymorphism โดยการใช้ this.getContentPane().add() ในการเพิ่มวัตถุของคลาส Game หรือ Menu ลงใน JFrame ของ Display

```
34
35●     public void endGame(long point) {
36          removeContent();
37          this.getContentPane().add(new Menu(point,this));
38      }
39
40●     @Override
41      public void actionPerformed(ActionEvent e) {
42          if(e.getActionCommand().equals("AGAIN ?")) {
43              removeContent();
44              Game game = new Game();
45              this.getContentPane().add(game);
46              game.requestFocus();
47          }
```

- **Inheritance**

-คลาส Game สืบทอดลักษณะจาก JPanel

```
15
16 public class Game extends JPanel implements KeyListener{
17
18     private static final long serialVersionUID = 1L;
19
```

```
139     @Override
140     public void keyPressed(KeyEvent e) {
141         if(System.currentTimeMillis() - lastPress > 600) {
142             if(e.getKeyCode()==32||e.getKeyCode()==38) {
143                     dog.jump(this);
144                     lastPress = System.currentTimeMillis();
145             }
146         }
147     }
148
149     @Override
150     public void keyTyped(KeyEvent e) {
151         //---
152     }
153
154     @Override
155     public void keyReleased(KeyEvent e) {
156         //---
157     }
158
159     public static void main(String[] arg) {
160          display = new Display();
161     }
162 }
```

-คลาส Point สืบทอดลักษณะจาก Wave

```
11
12 public class Point extends Wave{
13
14     public Point(int x, int y, int speed, JPanel page) {
15         super(x, y, speed, page);
16         // TODO Auto-generated constructor stub
17     }
18     public void move(JPanel page) {
19         this.timeMove = new Timer(speed,new ActionListener() {
```

-คลาส Display สืบทอดลักษณะจาก JFrame

```
public class Display extends JFrame implements ActionListener{

    private static final long serialVersionUID = 1L;
    private Dimension size = new Dimension(1000,600);

    public Display() {
        this.setting();
        this.getContentPane().add(new Game()); // เพิ่ม Game ลงใน JFrame

    }

    private void setting() {
        this.setTitle("Let's Girl Up!");
        this.setSize(size);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocation(280,100);
        this.setVisible(true);
    }
```

-คลาส Environment ได้ใช้การสืบทอดเนื่องจากมีการใช้งานเมธอด move และ stop ที่ถูกสืบทอดมาจากคลาสหลัก

```
        public void move(JPanel page) {
                    this.timeMove = new Timer(10,new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                            if(x+400<0) {
                                x = startX;
                            }
                            x -= speed;
                            page.repaint();
                    }
                });
                this.timeMove.start();
        }

        public void stop() {
            this.timeMove.stop();
        }
```

2.6) หน้าจอ GUI

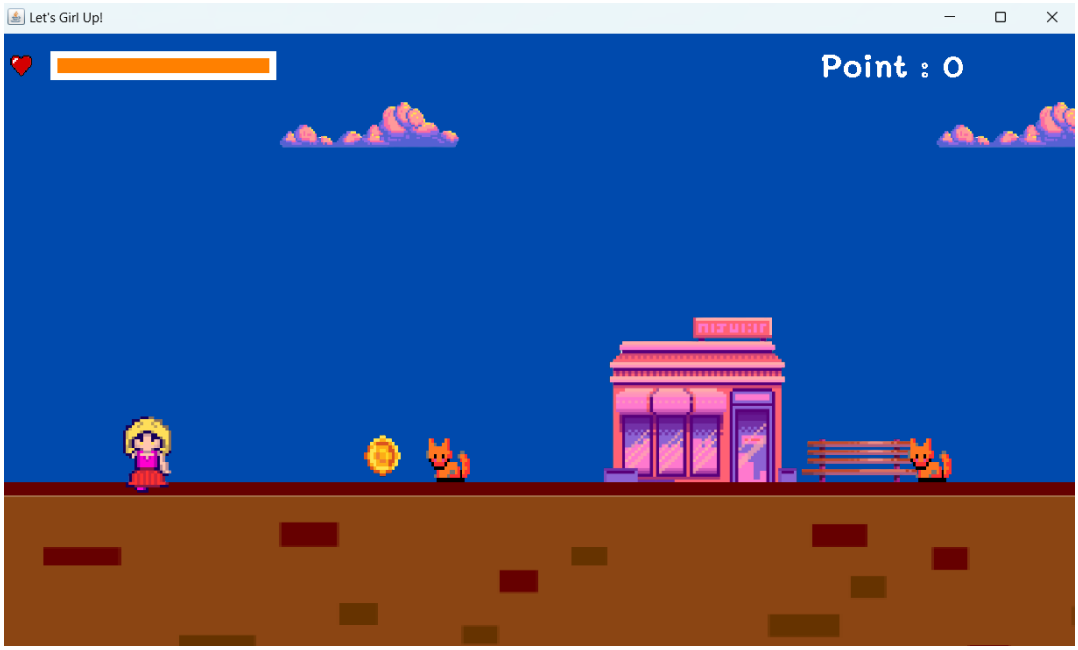
2.6.1)ส่วนประกอบของ GUI

1. JPanel (display.Game): สร้าง JPanel ที่ใช้เป็นพื้นที่การทำงานหลักของ GUI ซึ่งในที่นี้ถูกใช้เพื่อแสดงภาพ, สร้างการแสดงผล, และจัดการกับเหตุการณ์ต่าง ๆ เช่น KeyEvents
2. JFrame (display.Display): สร้าง JFrame ที่เป็นหน้าต่างหลักของแอปพลิเคชัน GUI โดยที่มีการกำหนดขนาด, ชื่อ, และตำแหน่งของหน้าต่าง
3. Graphics2D: สำหรับการวาดภาพและกราฟิกต่าง ๆ บน JPanel
4. KeyListener: ใช้ในการรับข้อมูลเกี่ยวกับการกดปุ่มบนแป้นพิมพ์เพื่อกระทำตามที่กำหนด
5. Event: มีเมธอดที่ใช้ในการตรวจสอบการชนของวัตถุในเกม
6. Charactor Package: มีคลาส Ghost และ Environment ที่ใช้ในการสร้างวัตถุต่าง ๆ ในเกม
7. Element Package: ประกอบด้วยเอกสารและองค์ประกอบที่ใช้ในการสร้างข้อความและภาพ

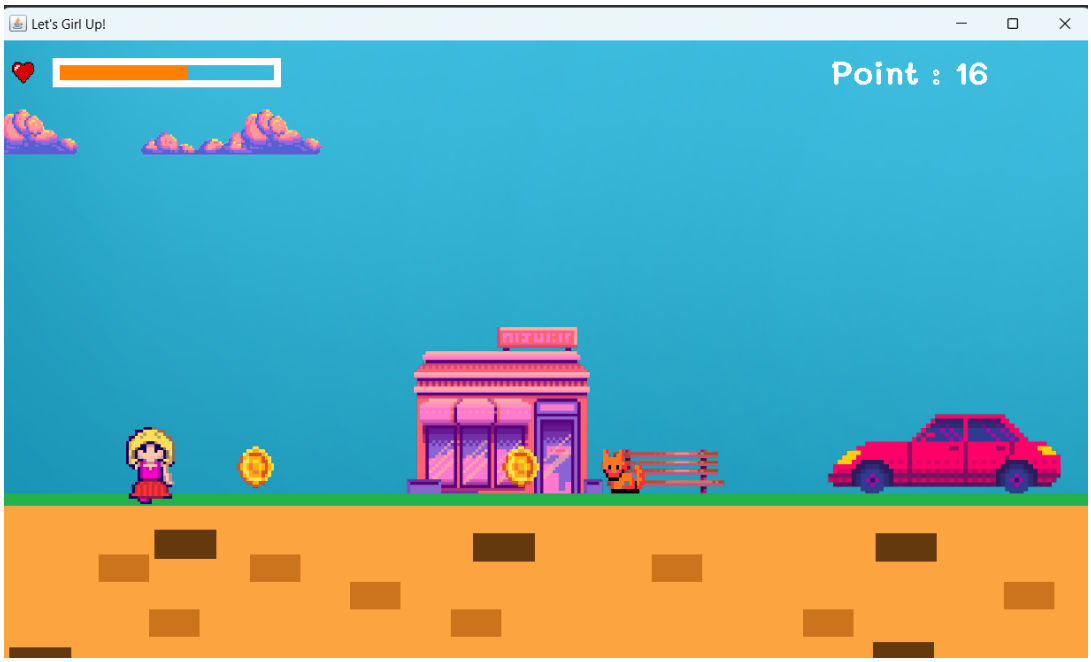
-หน้าจอ Play ก่อนเริ่มเกม จะมีปุ่ม PLAY เพื่อให้ผู้เล่นกดเริ่มเล่น

-เริ่มเกม มีหลอดพลังชีวิตที่มุมซ้ายบน มีการนับคะแนนที่มุมขวาบน

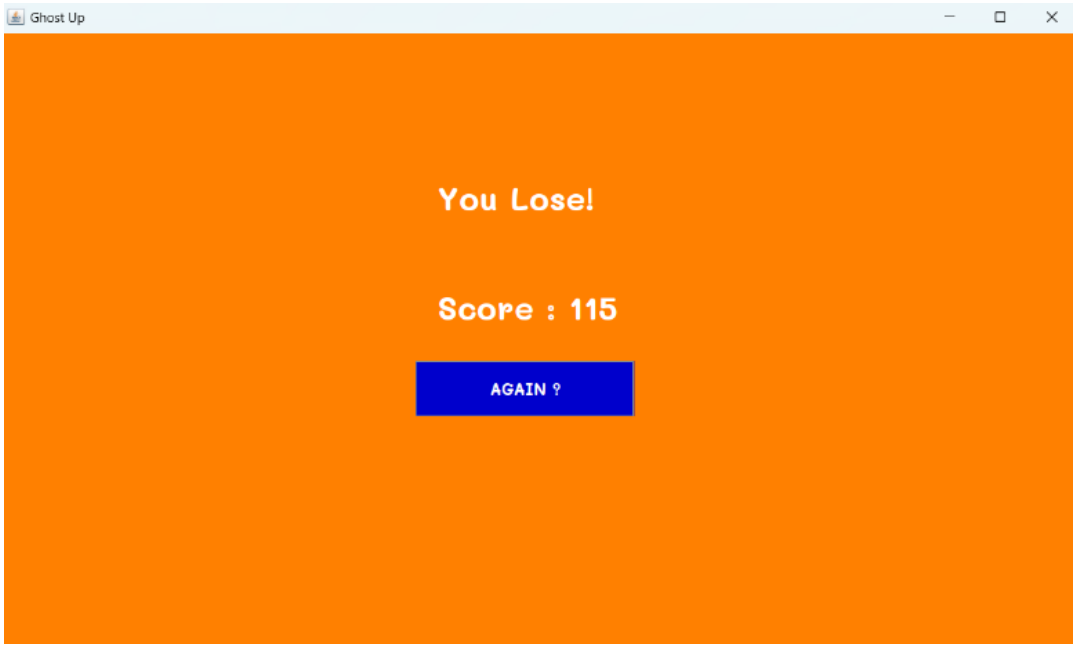
มีตัวละครเด็กผู้หญิง สิ่งกีดขวาง(แมว) เหรียญ และสิ่งแวะล้อม คือ ก้อนเมฆ อาคาร และรถ



-เมื่อเล่นได้คะแนนมากกว่า 10 ขึ้นไปจะทำการเปลี่ยนฉากหลัง



-จบเกมมีเมนูแสดงคะแนนที่เล่นได้ และมีปุ่ม AGAIN ? ให้กดเล่นอีกครั้ง



2.7) Event handling ที่มีในหน้าจอ

1. checkHit(Ghost dog, Wave wave, int dogSize, int waveHeight):

คือเมธอดที่ใช้ในการตรวจสอบว่าวัตถุทั้งสองมีการชนกันหรือไม่ โดยในที่นี้เป็นการตรวจสอบว่าวัตถุ dog (ตัวละคร) และ wave(วิ่งกรีดขวาง) มีพื้นที่ทับซ้อนกันหรือไม่ โดยเงื่อนไขที่ใช้คือ ถ้าตำแหน่ง x และ y ของ dog และ wave อยู่ในขอบเขตที่กำหนดไว้ และหากเงื่อนไขเหล่านี้เป็นจริงจะมีการส่งคืนค่า true ซึ่งแสดงว่ามีการชนเกิดขึ้น หากไม่เป็นจริงจะส่งคืนค่า false

```
1  package event;
2
3  import Charactor.*;
4
5  public class Event {
6          public static boolean checkHit(Ghost dog,Wave wave,int dogSize,int waveHeight){
7                  if(dog.x+dogSize>wave.x&&dog.x<wave.x) {
8                      if(dog.y+dogSize>=wave.y-waveHeight) {
9                          return true;
10                     }
11                 }
12                 return false;
13         }
14
15         public static void gameStop(Wave[] wave,Environment[] env) {
16
17         }
18
19 }
```

2. checPoint(Ghost dog, Point point, int dogSize, int pointHeight):

คือเมธอดที่ใช้ในการตรวจสอบว่าวัตถุทั้งสองมีการชนกันหรือไม่ โดยในที่นี้เป็นการตรวจสอบว่าวัตถุ dog (ตัวละคร) และ point(เหรียญ) มีพื้นที่ทับซ้อนกันหรือไม่ โดยเงื่อนไขที่ใช้คือ ถ้าตำแหน่ง x และ y ของ dog และ point อยู่ในขอบเขตที่กำหนดไว้ และหากเงื่อนไขเหล่านี้เป็นจริงจะมีการส่งคืนค่า true ซึ่งแสดงว่ามีการชนเกิดขึ้น หากไม่เป็นจริงจะส่งคืนค่า false

```
public static boolean checPoint(Ghost dog,Point point,int dogSize,int pointHeight){
    if(dog.x+dogSize>point.x&&dog.x<point.x) {
        if(dog.y+dogSize>=point.y-pointHeight) {
            return true;
        }
    }
    return false;
}
```

3.คลาส Game ใช้งาน Event Handling ผ่านการใช้ KeyListener

เป็นการจัดการกับการกระทำที่เกิดขึ้นจากการกดปุ่มบนแป้นพิมพ์ เช่น การกระโดดของตัวละครหลักของเกม dog จะถูกจัดการผ่านเมธอด keyPressed, keyTyped, และ keyReleased ที่ได้รับการโอเวอร์ไรด์จาก KeyListener ดังนั้น ส่วนนี้จะถือเป็นการจัดการเหตุการณ์ (Event Handling) ของการกระทำในเกม หากผู้ใช้กดปุ่มบนแป้นพิมพ์ เกมจะตอบสนองตามกับการกระทำนั้นๆ ที่ได้ถูกกำหนดไว้

```
139     @Override
140     public void keyPressed(KeyEvent e) {
141         if(System.currentTimeMillis() - lastPress > 600) {
142             if(e.getKeyCode()==32||e.getKeyCode()==38) {
143                     dog.jump(this);
144                     lastPress = System.currentTimeMillis();
145             }
146         }
147     }
148
149     @Override
150     public void keyTyped(KeyEvent e) {
151         //---
152     }
153
154     @Override
155     public void keyReleased(KeyEvent e) {
156         //---
157     }
158
159     public static void main(String[] arg) {
160          display = new Display();
161     }
162 }
```

2.8) อธิบายอัลกอริทึมที่สำคัญในโปรแกรม

เกมนี้เป็นเกมที่ให้ผู้เล่นควบคุมตัวละครซึ่งเป็นเด็กสาว โดยต้องกระโดดหลบสิ่งกรีดขวางซึ่งคือแมวส้ม และมีการนับคะแนน มีอัลกอริทึมที่สำคัญได้ดังนี้

1. **การวาดหน้าจอ:** โดยใช้ Java Swing เพื่อสร้าง GUI ของเกม ทำให้ผู้ใช้เห็นตัวละคร สิ่งกรีดขวาง และองค์ประกอบอื่น ๆ บนหน้าจอ
2. **การจัดการกับเหตุการณ์การกดแป้นพิมพ์:** โดยใช้ KeyListener เพื่อตรวจจับการกดปุ่มบนแป้นพิมพ์ ซึ่งในที่นี้ถูกใช้ในการทำให้ตัวละครกระโดดขึ้นและลง
3. **การคำนวณและเปลี่ยนแปลงคะแนน:** โดยเพิ่มคะแนนขึ้นทุกๆ ครั้งที่มีการชนกันของตัวละครกับเหรียญ และลดพลังชีวิตเมื่อตัวละครชนกับสิ่งกรีดขวาง
4. **การจัดการการกระทำของสิ่งกรีดขวางและการกระโดดหลบ:** โดยมีการตรวจสอบการชนของตัวละครกับสิ่งกรีดขวาง และหยุดเกมเมื่อตัวละครไม่มีพลังชีวิตเหลือ

โดยทั้งหมดนี้สร้างประสบการณ์เล่นเกมที่น่าสนใจและท้าทาย และโค้ดที่พัฒนาขึ้นมีการนำหลักการต่าง ๆ ของการเขียนโปรแกรมแบบ OOP มาประยุกต์ใช้ รวมถึงการใช้งาน Swing GUI การจัดการกับการกระทำจากผู้ใช้ การคำนวณคะแนน สถานะของตัวละครและสิ่งกรีดขวางในเกม

บทที่ 3 สรุป

3.1) ปัญหาที่พบระหว่างการพัฒนา

1. ปัญหาในการจัดการเหตุการณ์ (Event Handling) ไม่ถูกต้อง:

เกิดข้อผิดพลาดในการจัดการเหตุการณ์ต่าง ๆ ที่ส่งผลกระทบต่อพฤติกรรมของเกม เช่น การกระโดดของ ตัวละครหรือการตอบสนองต่อปุ่มกดที่ไม่ถูกต้องหรือช้าลง

2.ปัญหาการทำงานของภาพและการเคลื่อนไหว:

บางครั้งภาพอาจไม่แสดงผลได้อย่างถูกต้อง หรือการเคลื่อนไหวของวัตถุอาจไม่เหมือนกับที่คาดไว้ ซึ่งอาจเกิดจากปัญหาในการโหลดและแสดงภาพหรือการควบคุมการเคลื่อนไหวของวัตถุในเกม

3.ปัญหาเกี่ยวกับการทดสอบและตรวจสอบความถูกต้องของโปรแกรม:

การทดสอบไม่เพียงแค่ช่วยค้นพบข้อผิดพลาดและปัญหาต่าง ๆ แต่ยังช่วยให้ทราบถึงความเหมาะสมและประสิทธิภาพของโปรแกรม

3.2) จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

1. **การใช้งานตัวกระโดดแบบเรียลไทม์:**

   โปรแกรมนี้มีการปรับปรุงอัลกอริทึมการกระโดดที่ทำให้ตัวละครกระโดดได้โดยทันทีตามการกดปุ่ม โดยไม่มีความล่าช้าใด ๆ ซึ่งช่วยให้ประสบการณ์การเล่นเกมเป็นไปอย่างราบรื่นและน่าสนใจมากยิ่งขึ้น

2. **ระบบการเลือกรูปแบบพื้นหลังแบบไดนามิก:**

   โปรแกรมนี้มีระบบการเปลี่ยนแปลงรูปแบบพื้นหลังของเกมที่ขึ้นอยู่กับผลการเล่นของผู้เล่น โดยเมื่อผู้เล่นทำคะแนนมากขึ้น รูปแบบพื้นหลังจะเปลี่ยนไปตามไปด้วย ซึ่งทำให้ผู้เล่นรู้สึกถึงความท้าทายและความเปลี่ยนแปลงในการเล่น

3. **กราฟิกและภาพที่น่าทึ่ง (Stunning Graphics and Images)**: โปรแกรมนี้มีการออกแบบกราฟิกและภาพที่น่าทึ่งที่ช่วยให้ผู้เล่นได้รับประสบการณ์การเล่นที่น่าตื่นเต้นและน่าสนใจมากยิ่งขึ้น ทำให้ติดใจและติดตามเกมไปอย่างต่อเนื่อง

โดยรวมแล้ว โปรแกรมนี้มีความเป็นเอกลักษณ์ที่ชัดเจนที่ช่วยให้ผู้เล่นได้รับประสบการณ์การเล่นเกมที่ไม่เหมือนใครและน่าทึ่งอย่างแท้จริง