โครงงาน

เกม **Angel's Heart**

จัดทำโดย

นายวีรชิต พิทักษ์พรเกษม

เสนอ

ผู้ช่วยศาสตราจารย์สถิตย์ ประสมพันธ์

รายงานนี้เป็นส่วนหนึ่งของรายวิชา

Object-Oriented Programming
รหัสวิชา 040613204
ภาคเรียนที่ 1 ปีการศึกษา 2568

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

สารบัญ

เรื่อง	หน้า
- สารบัญ	1
บทที่ 1 บทนำ	
- ที่มาและความสำคัญของโปรเจ็ค	2
- ประเภทของโครงงาน	2
- ประโยชน์ของเกม	3
- ขอบเขตของโครงงาน	3
บทที่ 2 ส่วนการพัฒนา	
- เนื้อหาเกม	4
- วิธีการเล่น	4
- รูปแบบการพัฒนา Application	4
- Class Diagram	5
- องค์ประกอบของ Code	6
- โครงสร้างของ GUI	10
- Event handling ในหน้าจอ	14
- Algorithm ในโปรแกรม	16
บทที่ 3 สรุป	
- ปัญหาที่พบระหว่างการพัฒนา	26
- จุดเด่นของโปรแกรมที่ไม่เหมือนใคร	26
- คำแนะนำสำหรับผู้สอน	26

บทที่ 1
บทนำ

ที่มาและความสำคัญของโปรเจ็ค

- เกมนี้ถูกทำขึ้น เนื่องจากผู้สร้างเป็นคนที่ชอบเล่นเกมแนว Parkour ซึ่งเป็นเกมที่เราต้องวิ่งฝ่าสิ่งกีดขวางตามแต่ละจุด เพื่อไปถึงเส้นชัย

ซึ่งเกมนี้จะเพิ่มอุปสรรคอีกหลายอย่างในหนึ่งด่าน เพื่อให้เกมท้าทายขึ้น ผู้เล่นจะได้สัมผัสกับความยากในแต่ละด่านที่แตกต่างกัน และคะแนนรวมที่ผู้เล่นได้ ถ้ายิ่งเยอะจะบอกถึงทักษะของผู้เล่นที่เก่งมาก

โดยผู้สร้างนำแนวคิดการเขียนโปรแกรมแบบเชิงวัตถุมาพัฒนา เพื่อให้การจัดวางโค้ดสะดวกและง่ายต่อการเขียนเพิ่มเติมแก้ไขระหว่างทำ

ประเภทของโครงงาน

- เกม 2 มิติ สำหรับเล่นคนเดียว ใช้คีย์บอร์ดและเมาส์บังคับ

Github Link

https://github.com/KaoOat15120/AngelsHeart

ประโยชน์ของเกม

- ฝึกให้ผู้เล่นได้ใช้สมาธิใจจดใจจ่อ
- ฝึกให้ผู้เล่นหาจังหวะที่ดีในการกดนิ้วได้
- ฝึกให้ผู้เล่นบริหารเวลาตัวเองให้เร็วที่สุด
- ฝึกการคิดสถิติเรื่องจำนวนดาวและชีวิตที่เหลือ
- ฝึกความอดทนของผู้เล่นในการฝ่าอุปสรรค

ขอบเขตของโครงงาน

ลำดับ	รายการ	10-15 ส.ค.	16 ส.ค. – 20 ก.ย.	21 ก.ย. – 20 ต.ค.
1	คิดเนื้อหาเกม สิ่งของในด่าน วิธีการเล่น			
2	ออกแบบฉากและหารูปตัวละครกับพื้นหลัง			
3	เขียนโปรแกรมให้เกมทำงานถูกตามฟังก์ชั่น			
4	แก้ไขบัคในเกมและปรับการจัดวางโค้ด			
5	เพิ่มข้อความ เพลงในด่าน และภาพตกแต่ง			
6	เริ่มทำเอกสารเกมและอัพเดตหน้าต่างๆ			

บทที่ 2
ส่วนการพัฒนา

เนื้อหาเกม :

เกมนี้เรารับบทเป็นเทวดาที่มอบหัวใจ เพื่อต่อชีวิตให้คนป่วยตามเวลาที่กำหนด ในทุกๆ ด่านการที่เราจะลงจากสวรรค์ แล้วไปหาคนเจ็บได้ เราจะเจอกับผี 10 ตัวตามแต่ละความสูงที่คอยพ่นไฟออกมาเป็นช่วงๆ ให้เราต้องบินข้าม โดยไฟแต่ละสีก็มีเอฟเฟกต์เฉพาะที่ทำต่อเกมเป็นของตัวเอง หลังจากส่งหัวใจแล้ว เราก็จะต้องบินกลับสวรรค์ให้ได้เพื่อไปยังด่านต่อไป

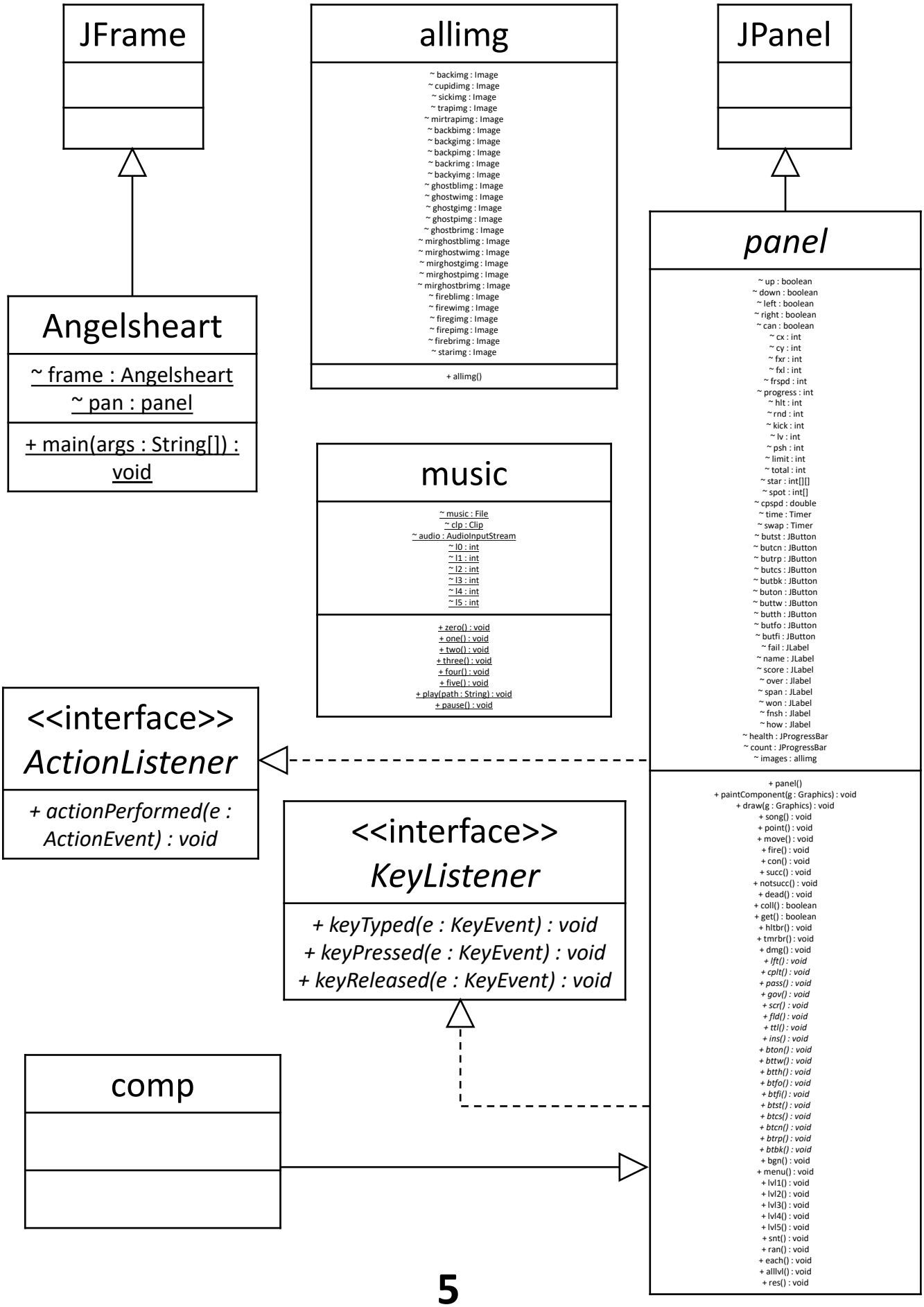
วิธีการเล่น :

- กดปุ่มลูกศรบนคีย์บอร์ด ในการบินขึ้นไปทางบน,ล่าง,ซ้าย, ขวา กดปุ่มค้างให้เคลื่อนไหวต่อไป และหยุดกดเพื่อหยุดเคลื่อนไหว

รูปแบบการพัฒนาแอป

- เขียนโค้ดด้วยภาษา Java บนซอฟต์แวร์ Apache NetBeans และใช้ OpenJDK เวอร์ชั่น 24.0.2

Class Diagram



องค์ประกอบของ **Code**

Constructor

```
class allimg{
   Image backimg,cupidimg,sickimg,trapimg,mirtrapimg;
   Image backbimg,backgimg,backpimg,backrimg,backyimg;
   Image ghostblimg,ghostwimg,ghostgimg,ghostpimg,ghostbrimg;
   Image mirghostblimg,mirghostwimg,mirghostgimg,mirghostpimg,mirghostbrimg;
   Image fireblimg,firewimg,firegimg,firepimg,firebrimg;
   allimg(){
      backimg = new ImageIcon("photos/back.png").getImage();
      backbimg = new ImageIcon("photos/backb.png").getImage();
      backgimg = new ImageIcon("photos/backg.png").getImage();
      backpimg = new ImageIcon("photos/backp.png").getImage();
      backrimg = new ImageIcon("photos/backr.png").getImage();
      backyimg = new ImageIcon("photos/backy.png").getImage();
      cupidimg = new ImageIcon("photos/cupid.png").getImage();
      cupidimg2 = new ImageIcon("photos/cupid2.png").getImage();
      ghostblimg = new ImageIcon("photos/ghostbl.png").getImage();
      ghostwimg = new ImageIcon("photos/ghostw.png").getImage();
      ghostgimg = new ImageIcon("photos/ghostg.png").getImage();
      ghostpimg = new ImageIcon("photos/ghostp.png").getImage();
      ghostbrimg = new ImageIcon("photos/ghostbr.png").getImage();
      mirghostblimg = new ImageIcon("photos/mirghostbl.png").getImage();
      mirghostwimg = new ImageIcon("photos/mirghostw.png").getImage();
      mirghostgimg = new ImageIcon("photos/mirghostg.png").getImage();
      mirghostpimg = new ImageIcon("photos/mirghostp.png").getImage();
      mirghostbrimg = new ImageIcon("photos/mirghostbr.png").getImage();
      fireblimg = new ImageIcon("photos/firebl.png").getImage();
      firewimg = new ImageIcon("photos/firew.png").getImage();
      firegimg = new ImageIcon("photos/fireg.png").getImage();
      firepimg = new ImageIcon("photos/firep.png").getImage();
      firebrimg = new ImageIcon("photos/firebr.png").getImage();
      sickimg = new ImageIcon("photos/sick.png").getImage();
      trapimg = new ImageIcon("photos/trap1.png").getImage();
      mirtrapimg = new ImageIcon("photos/mirtrap1.png").getImage();
      starimg = new ImageIcon("photos/star.png").getImage();
   }
}
```

- กำหนดให้ไฟล์ภาพใน package กับทุกตัวแปรรูปในคลาส allimg

Encapsulation

ทุก attribute ของคลาสต่างๆ จะกำหนดให้เข้าถึงได้แบบ default เพราะทุกคลาสในแพ็คเกจ angelsheart เดียวกันจะต้องใช้ข้อมูลของแต่ละคลาสร่วมกันหมด

Composition

```
abstract class panel extends JPanel {
   allimg images = new allimg();
   public void lvl1(Graphics g){
         g.drawImage(images.backbimg,0,0,300,615,null);
         g.drawImage(images.sickimg,125,550,50,50,null);
      }
```

- คลาส panel ต้องมีการใส่ทุกภาพที่กำหนดไว้ในคลาส allimg เลย ต้องประกาศวัตถุชนิด allimg ชื่อ images ให้สร้างจากคลาส allimg แล้วใช้วัตถุ images อ้างอิงถึงรูปต่างๆ เพื่อใส่กับเมธอดในแต่ละด่านได้

Polymorphism

```
public class Angelsheart extends JFrame{
   static panel pan;
   public static void main(String[] args) {
      pan = new comp();
   }
}

abstract class panel extends JPanel {
   abstract public void ttl();
}

class comp extends panel{
   public void ttl(){
      name = new JLabel();
      name.setBounds(10,30,265,150);
      name.setForeground(Color.decode("0xFFD700"));
      name.setFont(new Font("Jokerman",Font.BOLD,50));
      this.add(name);
   }
}
```

- คลาส Angelsheart ประกาศวัตถุชนิด panel ชื่อ pan ให้สร้างจากคลาส comp ทำให้เวลา pan เรียกเมธอด ttl ในคลาส panel จะเป็นการอ้างอิงถึงเมธอดชื่อเดียวกันที่ได้โอเวอร์ไรด์ไว้ในคลาส comp

Abstract

```
class panel implements KeyListener{
   public void keyPressed(KeyEvent e) {
      switch (e.getKeyCode()){
         case KeyEvent.VK_UP : up=true; break;
         case KeyEvent.VK_DOWN : down=true; break;
         case KeyEvent.VK_LEFT : left=true; break;
         case KeyEvent.VK_RIGHT : right=true; break;
      }
   }
}
```

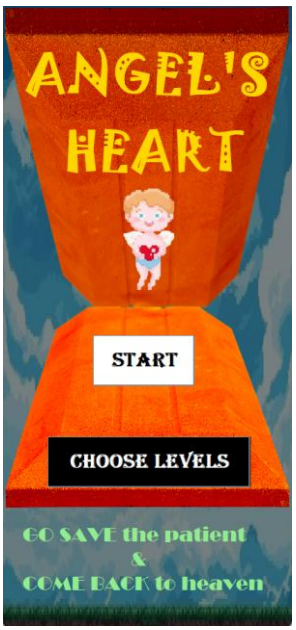
- คลาส panel โอเวอร์ไรด์เมธอด keyPressed จากอินเตอร์เฟส KeyListener และกำหนดคำสั่งให้ตรวจจับปุ่มลูกศรที่ถูกกด

Inheritance

```
public class Angelsheart extends JFrame{
   static Angelsheart frame;
   public static void main(String[] args) {
      frame = new Angelsheart();
      frame.setTitle("Angel's Heart");
      frame.setSize(300,650);
      frame.setVisible(true);
   }
}
```

- คลาส Angelsheart สืบทอดจากคลาส JFrame และเรียกใช้เมธอด setTitle, setSize, setVisible ของคลาส JFrame ได้

โครงสร้างของ GUI



หน้าแรก ประกอบด้วย

- ชื่อเกม
- รูปตัวละครหลักในเกม
- ปุ่ม START สำหรับเริ่มเล่นจากด่านแรกถึงด่านสุดท้าย
- ปุ่ม CHOOSE LEVELS สำหรับเลือกเล่นด่านไหนก็ได้
- ข้อความข้างล่าง บอกจุดหมายที่เราต้องไปในด่าน

หน้าเลือกด่าน ประกอบด้วย

- ปุ่มแสดงหมายเลขด่านต่างๆ จำนวน 5 ด่าน แต่ละปุ่มกดแล้วจะพาเราเข้าไปสู่ด่านนั้นๆ
- ปุ่ม BACK TO MENU สำหรับกดกลับไปยังหน้าแรก

ทุกด่านจะมีองค์ประกอบเหมือนกันได้แก่

- เทวดาที่เราเป็นผู้เล่น ทุกครั้งที่เริ่มด่านจะอยู่ตรงกลางข้างบนสุด
- หลอดเขียวข้างบน แสดงพลังชีวิตเรา เมื่อไฟโดนเราชีวิตในหลอดจะลดลง
- หลอดแดงข้างล่าง แสดงเวลาชีวิตของผู้ป่วย เวลาในหลอดจะลดลงเรื่อยๆ
- ผี 10 ตัวข้างซ้ายและขวาสลับกัน ทุกตัวจะคอยพ่นไฟออกมาให้เราได้หลบ
- ผู้ป่วยนอนอยู่ข้างล่างตรงกลาง เมื่อเราบินไปถึง เราจะส่งหัวใจให้ผู้ป่วย
- ทุก 10 จุดในด่านจะมีดาวสีรุ้งกระจายอยู่ เมื่อเก็บดาวเลือดเราจะเพิ่มขึ้น

ด่านที่ 1

- พื้นหลังสีน้ำเงิน
- ผีสีขาวพ่นไฟสีขาว

ด่านที่ 2

- พื้นหลังสีแดง
- ผีสีเทาพ่นไฟสีเทา

ด่านที่ 3

- พื้นหลังสีเขียว
- ผีสีน้ำตาลพ่นไฟสีน้ำตาล

ด่านที่ 4

- พื้นหลังสีเหลือง
- ผีสีชมพูพ่นไฟสีชมพู

ด่านที่ 5

- พื้นหลังสีม่วง
- ผีสีดำพ่นไฟสีดำ

เมื่อเราบินไปถึงผู้ป่วย หลอดเวลาและหัวใจในมือเทวดาจะหายไป เพราะเราส่งหัวใจให้ผู้ป่วยสำเร็จ

- หน้าจอ FAILED บอกแพ้ และจำนวนรอบที่เล่นได้อีกในด่านนั้น
- กดปุ่ม RETRY เพื่อเล่นด่านนั้นใหม่ เราจะเล่นด่านเดิมได้ 5 ครั้ง

- หน้าจอ GAME OVER บอกเกมจบ และคะแนนสะสมทั้งหมด
- เราจะเล่นด่านนั้นต่อไม่ได้ และต้องไปเริ่มเล่นใหม่ตั้งแต่ด่านแรก

- หน้าจอ PASSED แสดงคะแนนรวม เมื่อเราบินกลับถึงสวรรค์ โดยชีวิตในหลอดเขียวยังเหลือ และกดปุ่ม CONTINUE ไปด่านต่อไป

- หน้าจอ GAME DONE บอกคะแนนทั้งหมดที่เราเก็บมา เมื่อเราผ่านด่านที่ 5 เกมจะจบ และกดได้แค่ปุ่มไปหน้าแรก

Event Handling ในหน้าจอ

กดปุ่ม

Class panel

```
public void actionPerformed(ActionEvent e){
   if (e.getSource()==butcn){
         lv+=1;
         limit=5;
         each();
         butcn.setVisible(false);
         butbk.setVisible(false);
         score.setVisible(false);
         fnsh.setVisible(false);
      }
   }
```

```
public void each(){
      alllvl();
      hltbr();
      tmrbr();
      point();
      swap.start();
   }
```

```
public void alllvl(){
      can = true;
      cx=145;cy=20;fxr=300;fxl=0;
      cpspd=5;frspd=30;rnd=0;kick=1;
      spot = new int[] {50,100,150,200,250,300,350,400,450,500};
      butbk.setVisible(false);
      buton.setVisible(false);
      buttw.setVisible(false);
      butth.setVisible(false);
      butfo.setVisible(false);
      butfi.setVisible(false);
   }
```

- เมื่อเราผ่านด่าน เกมจะเรียกใช้เมธอด butcn ซึ่งจะวางปุ่ม CONTINUE บนหน้าจอ การคลิกปุ่มจะเพิ่มค่าในตัวแปร lvl เพื่อเปลี่ยนไปยังด่านต่อไป พร้อมกับเรียกใช้เมธอดอื่นๆ ที่จำเป็นกับทุกด่านด้วย เช่น ใส่ทั้งสองหลอดในหน้าจอ, กำหนดค่าตัวแปรใหม่, และปิดบังปุ่มต่างๆ ในหน้าที่แล้ว

กดและปล่อยปุ่มคีย์บอร์ด

Class Panel

```
@Override
   public void keyPressed(KeyEvent e) {
      switch (e.getKeyCode()){
         case KeyEvent.VK_UP : up=true; break;
         case KeyEvent.VK_DOWN : down=true;
break;
         case KeyEvent.VK_LEFT : left=true; break;
         case KeyEvent.VK_RIGHT : right=true; break;
      }
   }
```

```
@Override
   public void keyReleased(KeyEvent e) {
      switch (e.getKeyCode()){
         case KeyEvent.VK_UP : up=false; break;
         case KeyEvent.VK_DOWN : down=false; break;
         case KeyEvent.VK_LEFT : left=false; break;
         case KeyEvent.VK_RIGHT : right=false; break;
      }
   }
```

```
public void move(){
      if (can==true){
         if (up==true){
            if (cy>=20){
               cy-=cpspd;
            }
         }
         if (down==true){
            if (cy<=570){
               cy+=cpspd;
            }
         }
         if (left==true){
            if (cx>=5){
               cx-=cpspd;
            }
         }
         if (right==true){
            if (cx<=250){
               cx+=cpspd;
            }
         }
      }
   }
```

```
public void lvl1(Graphics g){
      g.drawImage(images.backbimg,0,0,300,615,null);
      g.drawImage(images.cupidimg,cx,cy,10,20,null);
      g.drawImage(images.sickimg,125,550,50,50,null);
   }
```

```
public void actionPerformed(ActionEvent e){
      move();
      }
   }
```

- เมื่อเรากดปุ่มลูกศรบนคีย์บอร์ดค้างไว้ เมธอด keyPressed จะถูกใช้และกำหนดตัวแปรที่ระบุจุดนั้นให้เป็น true จากนั้นเมธอด actionPerformed ที่ทำงานตลอดเวลาจะเรียกใช้เมธอด move เพื่อขยับตำแหน่งของตัวละครไปตามลูกศรที่เรากด แต่ถ้าเราปล่อยปุ่มลูกศรหรือไม่ได้กดเลย เมธอด keyReleased จะโดนใช้แทนและทำให้ตัวแปรทิศทางนั้นมีค่า false ซึ่งตัวละครจะหยุดเคลื่อนไหวทันที

Algorithm ในโปรแกรม

ไฟบินมาชนเรา

Class Panel

```
public boolean coll(){
      boolean[] b = new boolean[10];
      int d=0;
      for (int c=50;c<=500;c+=100){
         b[d] = (cy+25>c && cy<c+20);
         d+=2;
      }
      int e=1;
      for (int c=100;c<=500;c+=100){
         b[e] = (cy+25>c && cy<c+20);
         e+=2;
      }
      return ((cx+15>fxr && cx<fxr+15) &&
(b[0]||b[2]||b[4]||b[6]||b[8]))
            || ((cx+15>fxl && cx<fxl+15) &&
(b[1]||b[3]||b[5]||b[7]||b[9]));
   }
```

```
public void move(){
      if (can==true){
         if (coll()){
         }
      }
   }
```

- เมธอด coll จะตรวจจับว่า ตำแหน่งแนวนอน cx และแนวตั้ง cy ของเทวดาอยู่ทับกับตำแหน่งไฟของจุดใดจุดหนึ่งรึเปล่า หากเป็นจริงทั้งสองแนวจะคืนค่า true และเมธอด move จะรับค่าจากเมธอด coll มาใช้ดูเงื่อนไขเพื่อสร้างเอฟเฟต์ให้กับเกม

ปฏิกิริยาที่ไฟทำต่อเกม

Class panel

```
public void move(){
      if (can==true){
         if (coll()){
            if (lv==2){
               cpspd-=1;
            }
            if (lv==1){
               frspd+=5;
            }
            if (lv==5){
               cy-=25;
            }
            if (lv==3){
               cx+=psh;
               psh*=-1;
            }
            if (lv==4){
               frspd*=-1;
            }
         }
      }
   }
```

- ไฟแต่ละสีจะส่งเอฟเฟกต์ต่อเกมไม่เหมือนกัน เช่น ด่าน 1 เพิ่มความเร็วไฟ, ด่าน 2 ลดความเร็วเทวดา, ด่าน 3 ผลักเทวดาไปซ้ายขวา, ด่าน 4 ไฟวิ่งย้อนกลับ, ด่าน 5 ผลักเทวดาขึ้นไป

โดนไฟทำให้เลือดลด

Class panel

```
public void move(){
      if (can==true){
        if (coll()){
           dmg();
        }
     }
```

```
public void dmg(){
      hlt-=20;
      if (hlt<0){
         hlt=0;
      }
      health.setValue(hlt);
  }
```

```
public void hltbr(){
      health = new JProgressBar();
      health.setBounds(0,0,285,15);
      health.setValue(100);
      health.setForeground(Color.green);
      health.setBackground(Color.blue);
      add(health);
      hlt=100;
  }
```

- ทุกครั้งที่ตัวเราถูกไฟซน เมธอด dmg จะถูกเรียกใช้ โดยลดเลือดในตัวแปร hlt ลงทีละ 20% พร้อม กำหนดค่า hlt ที่เปลี่ยนใหม่ในหลอดเลือด health

เวลาค่อย ๆ ลดลง

Class panel

```
public void tmrbr(){
      progress = 100;
      count = new JProgressBar(0,100);
      count.setBounds(0,600,285,15);
      count.setValue(progress);
      count.setForeground(Color.red);
      count.setBackground(Color.yellow);
      add(count);

      time = new Timer(300,new ActionListener(){
         @Override
         public void actionPerformed(ActionEvent e){
            count.setValue(progress);
            if (progress<=0){
               count.setValue(0);
               time.stop();
            }
         }
      });
      time.start();
   }
```

```
public void move(){
      if (can==true){
         if (progress<=0){
            limit-=1;
            if (limit==0){
               dead();
            }
            else{
               notsucc();
            }
         }
      }
   }
```

- เมธอด tmrbr จะใส่หลอดเวลาบนหน้าจอ และลดเวลาในตัวแปร progress ลงครั้งละ 1 ทุกๆ 300 มิลลิวินาที เมื่อค่าในหลอดน้อยกว่าหรือเท่ากับ 0 เวลาจึงหยุดเดิน เราก็จะแพ้

ภารกิจสำเร็จ

Class panel

```
public void lvl1(Graphics g){
      sent(g);
   }
```

```
public void sent(Graphics g){
      if (rnd==0){
         g.drawImage(images.cupidimg,cx,cy,10,20,null);
      }
      else{
         g.drawImage(images.cupidimg2,cx,cy,10,20,null);
      }
   }
```

```
public void move(){
   if (cy>=540 && cx>=120 && cx<=170 &&
hlt>0 && progress>0){
            rnd=1;
            cpspd=5;
            frspd=25;
            kick=-1;
            hlt=100;
            health.setValue(hlt);
            count.setVisible(false);
            time.stop();
        }
   }
```

```
public void succ(){
      res();
      scr();
      btbk();
   }
```

```
public void move(){
  if (cy<=20 && hlt>0 && rnd==1){
         if (lv<=4){
            total+=limit;
            succ();
            btcn();
            pass();
         }
         else if (lv==5){
            total+=limit;
            succ();
            cplt();
         }
      }
  }
```

- เมื่อเราบินไปยังผู้ป่วย ค่าในตัวแปร rnd จะกลายเป็น 1 ทำให้เมธอด sent ที่ทำงานตลอดเวลาในด่านจะเปลี่ยนรูปเทวดาของเรา โดยความเร็วเทวดากับไฟ และเลือดในหลอดจะถูกรีเซ็ตใหม่หลอดเวลาจะหายไป ถ้าเรากลับมาถึงสวรรค์ได้ เมธอด succ จะแสดงคะแนนสะสมบนหน้าจอ คะแนนเก็บจะถูกบวกกับแต้มของด่านที่มี 5 แต้ม แต้มในด่านจะลดลงตามจำนวนการแพ้ของเรา

ภารกิจล้มเหลว

Class panel

```
public void move(){
      if (can==true){
         if (coll()){
            if (lv==4){
               frspd*=-1;
               dmg();
               con();
            }
         }
      }
   }
```

```
public void res(){
      health.setVisible(false);
      count.setVisible(false);
      time.stop();
      can = false;
   }
```

```
public void notsucc(){
      fld();
      lft();
      btrp();
      res();
      btbk();
   }
```

```
public void con(){
      if (hlt<=0){
         limit-=1;
         if (limit==0){
            dead();
         }
         else{
            notsucc();
         }
      }
   }
```

```
public void dead(){
      gov();
      scr();
      res();
      btbk();
   }
```

- เราจะแพ้ด่านเมื่อเลือดหรือเวลาหมด ตัวแปร limit ที่บอกจำนวนรอบที่เล่นได้ต่อจะถูกลบทีละ 1 จาก 5 โดยเมธอด res จะสั่งให้ทุกอย่างหยุดขยับและหลอดทั้งสองหายจากจอ ถ้าแต้มในด่านมากกว่า 0 เมธอด notsucc แสดงข้อความ FAILED แต่หากน้อยกว่า 0 เมธอด dead แสดงข้อความ GAME OVER

การวิ่งของไฟ

Class panel

```
public void fire(){
      if (can==true){
         fxr-=frspd;
         fxl+=frspd;
         if (fxr<0 && frspd>0){
            fxr=300;
         }
         if (fxl>300 && frspd>0){
            fxl=0;
         }
         if (fxr>300 && frspd<0){
            fxr=0;
         }
         if (fxl<0 && frspd<0){
            fxl=300;
         }
      }
   }
```

```
public void lvl1(Graphics g){
      for (int a=50;a<=500;a+=100){
         g.drawImage(images.firewimg,fxr,a,15,15,null);
      }
      for (int a=100;a<=500;a+=100){
         g.drawImage(images.firewimg,fxl,a,15,15,null);
      }
   }
```

```
public void actionPerformed(ActionEvent e){
      fire();
      }
   }
```

- ไฟในทุกด่านจะเคลื่อนที่ด้วยระยะทางที่กำหนดไว้ในตัวแปร frspd เมื่อไฟวิ่งถึงขอบจอ ตำแหน่งจะถูกรีเซ็ตกลับมาจุดเริ่มต้น และทำซ้ำๆ ต่อไป เมธอด drawImage จะเปลี่ยนจุดของไฟบนหน้าจอตลอด

เล่นเพลงวนไปเรื่อย ๆ

Class panel

```
public void song(){
      switch(lv){
         case 0 : music.zero(); break;
         case 1 : music.one(); break;
         case 2 : music.two(); break;
         case 3 : music.three(); break;
         case 4 : music.four(); break;
         case 5 : music.five(); break;
      }
   }
```

```
public void actionPerformed(ActionEvent e){
      song();
   }
```

Class music

```
static void one(){
      if (l0>0 || l2>0 || l3>0 || l4>0 || l5>0){
         pause();
      }
      l0=0;l2=0;l3=0;l4=0;l5=0;
      l1+=1;
      if (l1==1){
         play("tracks/level1.wav");
      }
   }
```

```
static void pause(){
      clp.stop();
      clp.close();
   }
```

```
static void play(String path){
      try{
         music = new File(path);
         audio =
AudioSystem.getAudioInputStream(music);
         clp = AudioSystem.getClip();
         clp.open(audio);

clp.loop(Clip.LOOP_CONTINUOUSLY);
         clp.start();
      }
      catch (Exception e){
         System.out.println(e);
      }
   }
```

- เมธอด song ของคลาส panel จะเลือกเมธอดเล่นเพลงประจำด่านของคลาส music ขึ้นอยู่กับด่านที่เกมแสดงตามค่าในตัวแปร lv ขณะนั้น
- ถ้า lv เป็น 1 จะใช้เมธอด one โดยตรวจสอบว่าด่านที่แล้ว ตามตัวแปร l0,l2,l3,l4,l5 ยังเล่นเพลงอยู่หรือไม่ ถ้าใช่ก็จะเรียกเมธอด pause เพื่อหยุดเพลงนั้นก่อน จากนั้นก็กำหนดทุกอันเป็น 0 ไม่ให้เพลงหยุดระหว่างด่าน แล้วเรียกเมธอด play ให้เล่นเพลงวนไปวนมา ต่อเมื่อค่า l1 มีค่าเท่ากับ 1 เพราะ l1 จะถูกบวกอยู่เรื่อยๆ เพื่อป้องกันการเล่นเพลงซ้อนกัน

เก็บดาวสีรุ้ง

Class panel

```
public void point(){
      Random r = new Random();
      for (int a=0;a<10;a++){
         int x = r.nextInt(30,240);
         star[a][0] = x;
         star[a][1] = spot[a];
      }
   }
```

```
public boolean get(int x, int y){
      if ((cx+15>x && cx<x+15)
&& (cy+25>y && cy<y+20)){
         return true;
      }
      return false;
   }
```

```
public void lvl1(Graphics g){
      ran(g);
   }
```

```
public void ran(Graphics g){
      for (int a=0;a<10;a++){

g.drawImage(images.starimg,star[a][0],star[a][1]
,15,15,null);
         boolean b = get(star[a][0],star[a][1]);
         if (b==true){
            if (hlt<100){
               star[a][1] = 700;
               spot[a] = 700;
               hlt+=10;
               health.setValue(hlt);
            }
         }
      }
   }
```

- เมธอด point จะสุ่มเลข 10 ตัว แล้วนำแต่ละเลขใส่ใน index[][0] ของอาร์เรย์ star กับให้ index[][1] เป็นความสูงต่างๆ ที่ผี 10 ตัวอยู่ เมธอด ran จะวางรูปดาวสีรุ้งตาม 10 ตำแหน่งใน star และเมื่อเราบินชนดาว ขณะที่เลือดในหลอดเขียวลดไปน้อยกว่า 100% เลือดเราจะเพิ่มขึ้นอีก 10% ส่วนดาวดวงนั้นที่เราไปโดนจะถูกย้ายพิกัดออกจากจอให้หายไป

ดาวย้ายตำแหน่ง

Class panel

```
panel(){
      swap = new Timer(3000,new ActionListener(){
         public void actionPerformed(ActionEvent e){
            point();
         }
      });
   }
```

```
if (e.getSource()==butrp){
         point();
         swap.start();
      }
```

```
public void res(){
      swap.stop();
   }
```

- swap เป็นตัวแปรชนิด Timer ที่จะเรียกเมธอด point ให้เปลี่ยนตำแหน่งของดาวสีรุ้งทุกๆ 3 วินาที โดยจะเริ่มจับเวลาเมื่อเรากดปุ่มอะไรก็แล้วแต่เพื่อเล่นด่านนั้นใหม่ เมธอด point จะกำหนดตำแหน่งรอบแรกของดาวก่อน แล้ว swap จะถูกใช้ไปเรื่อยๆ และจะหยุดทำงานตอนที่เราแพ้หรือผ่านในด่านนั้น

บทที่ 3 สรุป

ปัญหาที่พบระหว่างการพัฒนา

- หากสร้างวัตถุใหม่ แล้วลืมกำหนดพื้นที่ให้วัตถุ โปรแกรมจะคอมไพล์ไม่ได้ เพราะวัตถุถูกเรียกใช้เมื่อเปิดเกม
- ถ้าในเมธอดมีการเรียงคำสั่งไม่ถูกต้องตามลำดับการทำงานที่ต้องการ ผลลัพธ์ของเมธอดจะออกมาผิดพลาด
- ทุกรอบที่เกมเปลี่ยนหน้า ถ้าลืมปิดไม่ให้มองเห็นหลอดได้ หรือทำให้เวลาหลอดหยุดทำงาน หลอดจะทำงานติดขัด

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

- การชนะด่านไม่เน้นแค่ฝีมือ แต่ขึ้นอยู่กับโชคด้วย ถ้าดาวสีรุ้งเกิดใกล้ผู้เล่น โอกาสผ่านจะมากขึ้น
- เมื่อผู้เล่นทำภารกิจสำเร็จ จะต้องกลับมาจุดเริ่มต้นเพื่อผ่านด่านเหมือนกับเล่น 2 รอบในหนึ่งด่าน
- ทุกด่านยากหมด ไม่มีด่านง่าย และทุกครั้งที่เล่นด่านนั้นใหม่อาจจะยากหรือง่ายไม่เท่ากับคราวก่อน

คำแนะนำสำหรับผู้สอน

-