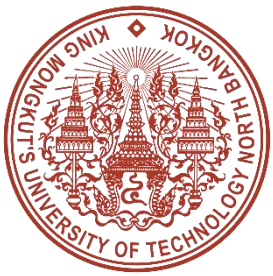


โครงงาน

Escape The Ghost

จัดทำโดย

6604062636135 นางสาว ฐาปณี ศิริพละ

เสนอ

ผู้ช่วยศาสตราจารย์สถิตย์ ประสมพันธ์

วิชา 040613204 Object-Oriented Programming

ภาคเรียนที่ 1 ปีการศึกษา 2567

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทที่ 1 บทนำ

1. ที่มาและความสำคัญ

โครงงานนี้จัดทำขึ้นเพื่อรายวิชา Object Oriented Programming โดยการนำเนื้อหาที่ได้เรียนมาประยุกต์ใช้ ผู้จัดทำได้ทำการสร้างโครงงานเกมนี้ขึ้น เพื่อสั่งสมความรู้และประสบการณ์ในการทำเกมและเรียนรู้ OOP โดยคิดระบบเกมมาให้สนุกสนานและมีกราฟฟิคที่สวยงาม หากมีความผิดพลาดประการใด ขออภัยมา ณ ที่นี้ด้วย

2. วัตถุประสงค์

2.1 เพื่อศึกษาการเรียนในภาคปฏิบัติของวิชา OOP

2.2 เพื่อสะสมประสบการณ์ในการทำเกมและการเขียนโปรแกรม JAVA แบบ OOP

3.ขอบเขตของโครงงาน

3.1 ใช้หลักการในการเขียนโปรแกรมแบบ OOP

3.2 ใช้ภาษา JAVA ในการเขียนโปรแกรม

4. ประโยชน์ของโครงงาน

4.1 ได้รับประสบการณ์ในการเขียนโปรแกรม JAVA แบบ OOP

4.2 ได้ความรู้ในการทำเกม สามารถนำประสบการณ์ไปใช้ในโปรเจคอื่นๆต่อได้

5. ขอบเขตของโครงการ

ตารางแผนการทำงานเดือนตุลาคม-พฤศจิกายน

ลำดับ	รายการ	6-19	20-26	27-3	4-6
1	จัดทำกราฟิก	✓			
2	ศึกษาวิธีการเขียนโปรแกรมและค้นคว้าเอกสารที่เกี่ยวข้อง	✓	✓		
3	ลงมือเขียนโปรแกรม		✓	✓	
4	จัดทำเอกสาร			✓	
5	ตรวจสอบและแก้ไขข้อผิดพลาด				✓

บทที่ 2 ส่วนการพัฒนา

1. เนื้อเรื่องย่อ

รายละเอียดเกมส์

เกมส์หนีผี เนื้อหาคือ เราจะเล่นเป็นเด็กหนุ่มชื่อ “หัวตั้ง” เป็นคนไปล่าท้าผีกับเพื่อนๆในป่า อาถรรพ์แห่งหนึ่ง แต่เขาก็พลัดหลงกับคนอื่นและเจอกับผีดุร้ายตัวเป็นๆในป่า มีแค่สิ่งเดียวที่เขาทำได้คือ หลบหนีกับผีนานาชนิดที่จะมาเผชิญเพื่อเอาชีวิตรอด โดยเขาจะต้องเก็บ “แต้มบุญ” เพื่อสะสมคะแนนบุญ จนสามารถผ่านม่านผีบังตาและออกจากป่าได้

วิธีการเล่นเกมส์

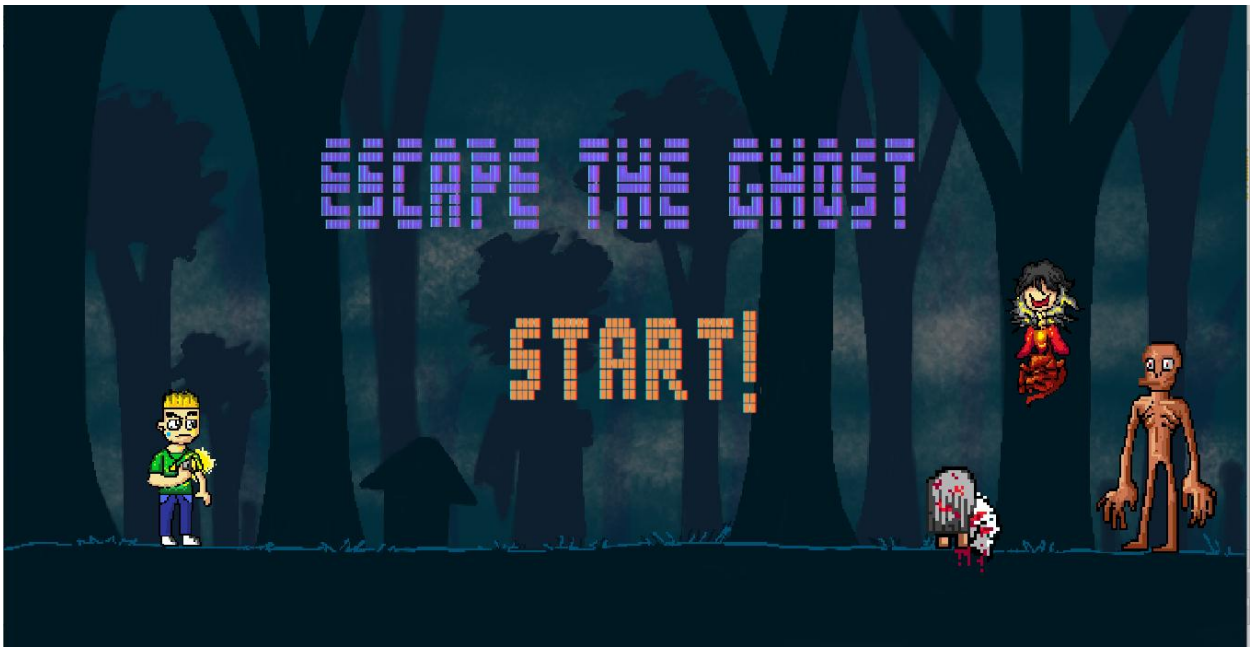
กดปุ่ม w เพื่อกระโดดหลบผี สามารถกระโดดสูง สูงสุด 2 ขั้น หรือ กดปุ่ม s เพื่อก้มหลบผีบางประเภท และกระโดดชนแต้มบุญเพื่อสะสมบุญให้ถึง8000 เพื่อชนะ

Storyboard

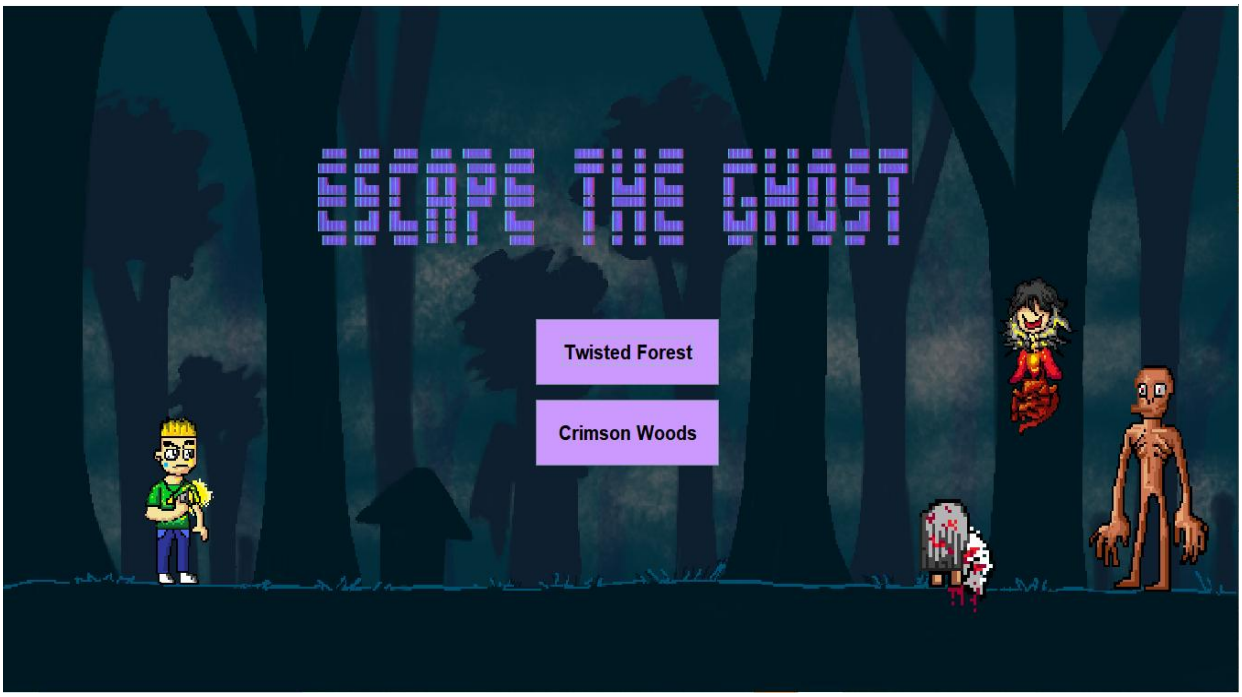
ตัวละคร

ฉาก

เริ่มเกม

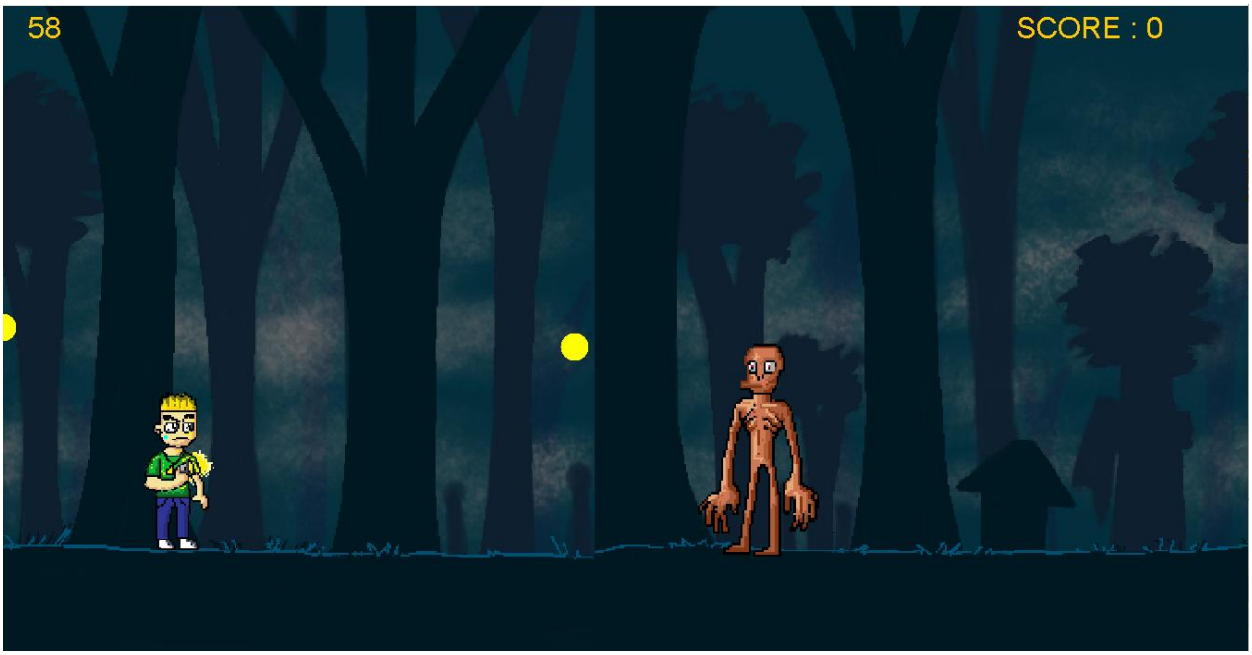


หน้าเมนู



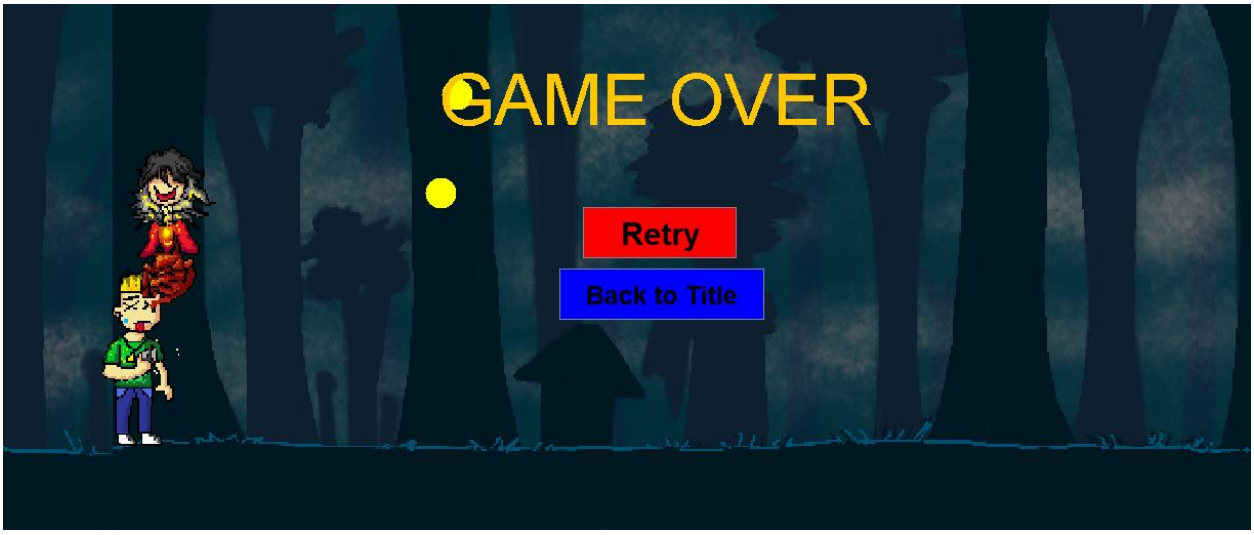
เริ่มเกม

-ฉาก Twisted Forest (ป่าคดเคี้ยว)



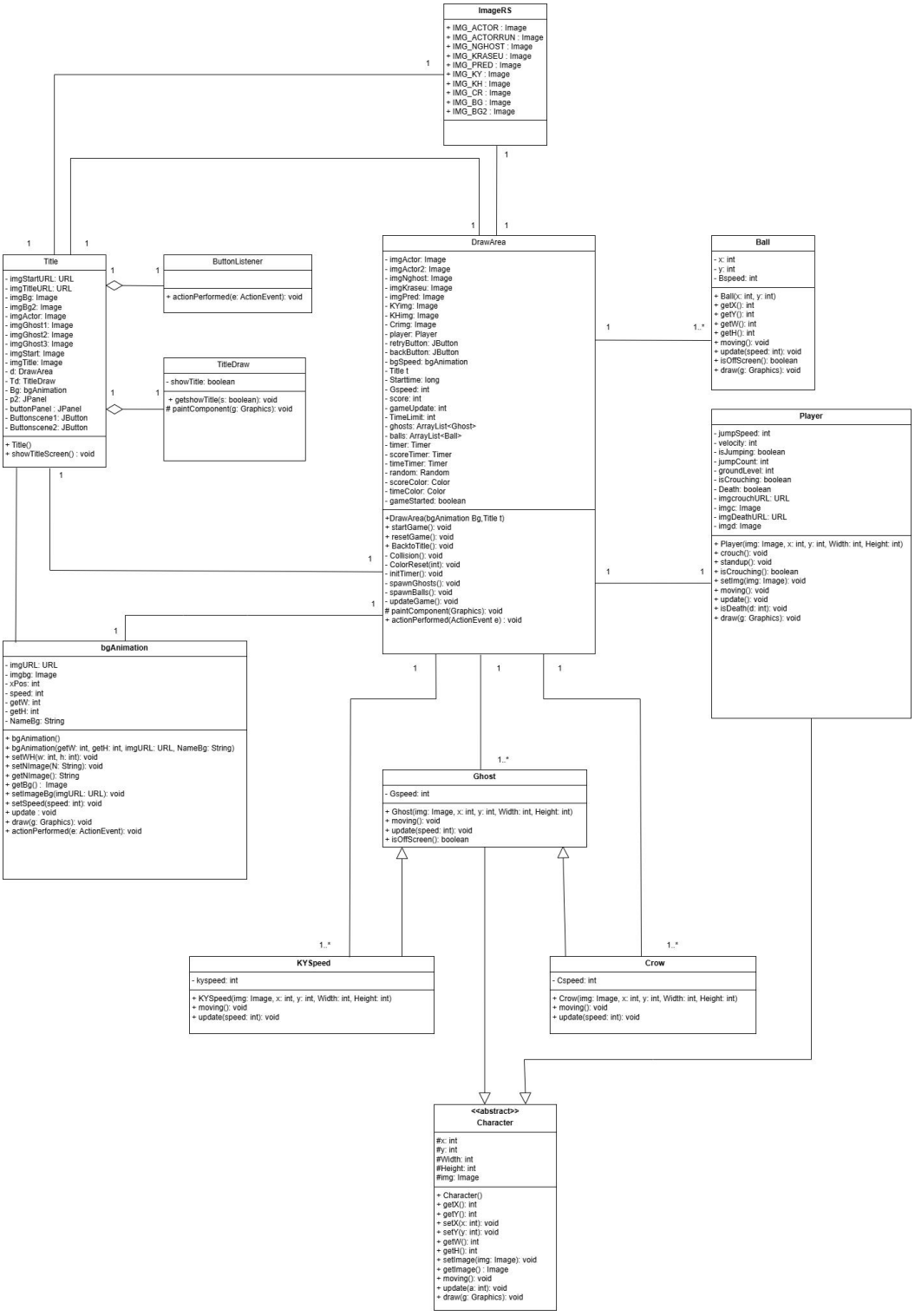
-ฉาก Crimson Woods (ป่าสีชาด)

เมื่อตาย



เมื่อชนะ

2. แผนภาพ Class diagram



1) คลาส Title เป็นคลาสสำหรับสร้างหน้าเกมเริ่มต้นในคลาสจะมี constructor Title() ที่จุJFrameเอาไว้จุComponents มี method showTitleScreen() สำหรับเรียกโชว์หน้าพื้นฐานเกม ในกรณีที่อยู่หน้าอื่นและกลับมา

2) คลาสButtonListener เป็นตัวดักจับเหตุการณ์ของปุ่มในp2ซึ่งมี Buttonscene1,Buttonscene2 เมื่อจับเหตุการณ์ไปแล้วจะแสดงหน้า DrawAreaและเริ่มเกม ตัวนี้เป็น innerclass ของ Titleและเป็นองค์ประกอบของTitle

3) คลาสTitleDraw เป็นคลาสที่มีการวาดตัวละครและองค์ประกอบในฉากหน้าstart โดย พารามิเตอร์ showTitle จะเป็นตัวกำหนดว่า จะวาดตัวละครและองค์ประกอบอย่างไร ตามเงื่อนไข ตัวนี้เป็น innerclass ของ Titleและเป็นองค์ประกอบของ Title

4) คลาสImageRS เป็นคลาสที่เอาไว้จุรูปภาพและURL ต่างๆ แบบstatic ทำให้ คลาสTitle,DrawArea สามารถดึงรูปมาใช้ได้ง่าย

5) คลาสDrawArea เป็นคลาสที่มีไว้สำหรับวาดรูปเมื่อเริ่มเกมหลังเลือกฉากในTitleเสร็จ ButtonListenerในTitleจะทำการใช้ method startGame() ในconstructorก็เรียกใช้methodต่างๆเพื่อดำเนินการเช่นinitTimer()เพื่อเริ่มกระบวนการต่างๆ ภายใน เช่น timer , เปลี่ยนค่าพารามิเตอร์ gameStarted เมื่อ gameStartedเป็นTrue จะทำการเรียกmethod spawnGhosts(), spawnBalls() มาใช้ได้ และทำการวาดรูปจากpaintComponent ได้ และกระบวนอื่นๆตามเงื่อนไข มีBacktoTitle()เพื่อกลับไปวาดหน้าTitle

6) คลาสBgAnimation มีไว้เพื่อเป็นตัวสำหรับรับBackgroundจากเงื่อนไขของButtonListnerของTitleที่จะมีการใช้method setImageBg()และตั้งชื่อเพื่อใช้ในDrawArea ด้วยmethod getNImage() และขยับฉากหลังด้วยเงื่อนไขจากในDrawAreaที่จะส่งความเร็วใน setSpeed(int speed) แล้วอัพเดทค่าใน update()

7) คลาสBall เป็นคลาสสำหรับวาดลูกบอลโดยใช้Draw(graphic g) ที่จะทำการชนกับPlayer โดยข้างในมี moving() เป็นตัวขยับแกนx และมี update(int speed) เป็นตัวรับค่าความเร็วมาจากเงื่อนไขของDrawArea มีisOffScreen() เพื่อลบบอลออกเมื่อออกจากหน้าจอแล้ว เช็คจากแกนx (x<-150) จะได้สามารถสร้างใหม่ได้ในDrawArea ตามเงื่อนไข

8) คลาสCharacter เป็นAbstract คลาสที่ยังไม่กำหนดเงื่อนไขใดๆใน moving()กับupdate(int a) เพื่อที่คลาสลูกจะได้ใช้เงื่อนไขที่ต่างกันต่อไป สามารถวาดรูปได้โดยdraw()และตั้งค่ากับรับค่าได้ตามmethod อื่นๆ

9) คลาสPlayer เป็นคลาสที่สืบทอดมาจากCharacter เพราะเป็นตัวละคร(Character) โดยในคลาสจะมีตัวหลักๆคือการควบคุมการกระโดดด้วยmoving()ที่Overrideมาจากcharacter เช็คทิศขึ้น(ใช้เฉพาะ แกน y)และupdate()คือตรวจสอบว่ากระโดดอยู่มั้ย ถ้ากระโดดให้ตกลงมาตามแรงโน้มถ่วงจนถึงพื้น,การเช็คการตาย isDeath(int d)

10) คลาสGhostเป็นคลาสที่สืบทอดมาจากCharacter เพราะเป็นตัวละคร(Character)มีการOverride method moving() เพื่อลบค่าแกนxให้ขยับด้วยพารามิเตอร์kyspeedที่ตั้งค่าใหม่ และOverride method update(int a)เพื่อตั้งค่าkyspeedที่ได้รับมาจากเงื่อนไขของDrawArea และมีmethod isOffScreen()เพื่อเช็คค่าแกนx ในกรณีน้อยกว่า-150คือออกจอ

11) คลาสCrow เป็นคลาสที่สืบทอดจากคลาสGhost เพราะเป็นผีอีกา(ในที่นี่ละคำว่าผีไว้ตอนเขียนคลาส)มีการOverriden method update(int a)เพราะตั้งค่าไม่เหมือนกัน แสดงผลเป็น*2ของความเร็วและmethod update()ในการคำนวณแกนxด้วย พารามิเตอร์Cspeed

12) คลาสKYSpeed เป็นคลาสที่สืบทอดจากคลาสGhost เพราะเป็นผีคุณยายสปีด(KYSpeed)มีการOverriden method update(int a)เพราะตั้งค่าไม่เหมือนกัน แสดงผลเป็น*3ของความเร็วและmethod update()ในการคำนวณแกนxด้วย พารามิเตอร์kyspeed

3. รูปแบบการพัฒนาโครงการ Application

มีการสร้างโครงสร้างพื้นฐานสำหรับเกม "Escape the Ghost" โดยใช้ Java Swing และ AWT เพื่อสร้าง GUI และจัดการกับการวาดภาพต่างๆ ภายในหน้าต่างเกม

3.1 การออกแบบและวางแผน

แนวคิดของเกม: เกมหลบหลีกผี โดยตัวละครผู้เล่นสามารถกระโดด 1ถึง2ครั้งหรือหมอบหลบ และสะสมคะแนน

โครงสร้างหลัก:

Title Screen: หน้าจอเริ่มเกมพร้อมปุ่มเลือกฉาก

Game Screen: หน้าจอเกมที่มีผู้เล่น, ผี, บอล, และพื้นหลังที่เปลี่ยนตามฉาก

Ending Screen: หน้าจอแสดงผลเมื่อจบเกม แพ้,ชนะ

3.2 การพัฒนาโครงสร้าง

<u>การแยกส่วนประกอบในแพ็กเกจ</u>

แพ็กเกจ: escapetheghost

ไฟล์หลัก:

Title.java: สำหรับหน้าจอเริ่มต้น

DrawArea.java: สำหรับพื้นที่วาดภาพในหน้าจอเกม

Player.java: คลาสผู้เล่น

Ghost.java: คลาสผี, ผีชนิดพิเศษ(Crow,KYSpeed)

Ball.java: คลาสลูกบอล

bgAnimation.java: คลาสสำหรับพื้นหลังแบบเคลื่อนไหว

ImageRS.java: สำหรับโหลดและจัดการรูปภาพ

การแบ่งหน้าที่แต่ละส่วน

Title.java:แสดง UI สำหรับหน้าจอเริ่มต้น,จัดการปุ่มเริ่มเกม และเลือกฉากสลับไปยังหน้าจอเกมเมื่อผู้ใช้กดปุ่ม

DrawArea.java:วาดผู้เล่น, ผี, บอล, และพื้นหลัง อัปเดตการเคลื่อนไหวของตัวละคร, ผี และไอเทมตรวจจับการชนและจัดการผลลัพธ์ของเกม

Ghost.java:เก็บข้อมูลศัตรู เช่น ตำแหน่ง, ขนาด, และการเคลื่อนไหว มีคลาสย่อยสำหรับผีที่มีพฤติกรรมเฉพาะ เช่น Crow และ KYSpeed

Player.java:จัดการตำแหน่งและสถานะของผู้เล่น เช่น การกระโดด, หมอบ, และตาย

3.3 การสร้างกลไกเกม

การเคลื่อนไหวของผู้เล่น:

-กระโดด1-2ครั้ง: กดwใช้ KeyListener เพื่อตรวจสอบจำนวนครั้งที่กระโดด

-หมอบ: ใช้ KeyListener จับการกดและปล่อยปุ่ม S

ผีและบอล: สุ่มการเกิดศัตรูโดยใช้ Random

การชน: ตรวจสอบการชนระหว่างผู้เล่นกับผีหรือบอลโดยใช้พื้นที่การชนด้วยขนาดรูปภาพ และตำแหน่งx y ของวัตถุ

ระบบคะแนน: ชนจากบอล +100 score ด้วยcollision()และแสดงคะแนนบนหน้าจอด้วย Graphics.drawString

ระบบเวลา:ใช้ตัวจับเวลา (Timer) เพื่อลดเวลาเล่น และเพิ่มความเร็วเกมทุก 10 วินาที

4. แนวคิดการเขียนโปรแกรมเชิงวัตถุ

4.1 Constructor

Class Title

```
public Title() {

    Bg = new bgAnimation(1366,710,imgBg,"bg.png");
    d = new DrawArea(Bg,this);
    setLayout(new BorderLayout());
    add(Td, BorderLayout.CENTER);

    buttonPanel = new JPanel(new GridBagLayout());
    buttonPanel.setOpaque(false);
    JButton startButton = new JButton(new ImageIcon(imgStart));
    startButton.setPreferredSize(new Dimension(280, 160));
    Image resizedImage = imgStart.getScaledInstance(280, 160, Image.SCALE_SMOOTH);
    startButton.setIcon(new ImageIcon(resizedImage));


    startButton.setContentAreaFilled(false); //remove fill
    startButton.setBorderPainted(false); //remove line
    startButton.setMargin(new Insets(0, 0, 0, 0)); // Remove extra space around the image
    buttonPanel.add(startButton);

    Td.setLayout(null);
    Td.add(buttonPanel);
    buttonPanel.setBounds((1366 - 280) / 2, (786 - 160) / 2, 280, 160);

    p2 = new JPanel(new GridLayout(2,1,0,15));
    Font font = new Font("Courier", Font.BOLD, 20);
    Buttonscene1.setFont(font);
```

```
    Buttonscene1.setForeground(Color.BLACK);
    Buttonscene1.setBackground(Color.decode("#CC99FF"));

    Buttonscene2.setFont(font);
    Buttonscene2.setForeground(Color.BLACK);
    Buttonscene2.setBackground(Color.decode("#CC99FF"));

    p2.add(Buttonscene1);
    p2.add(Buttonscene2);
    p2.setPreferredSize(new Dimension(200, 300));
    p2.setVisible(false);
    Td.add(p2);
    p2.setBounds((1366 - 200) / 2, (786 - 150) / 2, 200, 150);

    Buttonscene1.addActionListener(new ButtonListener());
    Buttonscene2.addActionListener(new ButtonListener());
    startButton.addActionListener(e -> {
        p2.setOpaque(false);
        p2.setVisible(true);
        Td.add(p2,BorderLayout.CENTER);
        buttonPanel.setVisible(false);
        Td.repaint();

    });

}
```

ประกอบด้วยการสร้างออบเจ็กต์ Bg จากคลาส BgAnimation และคลาส DrawArea โดยในคลาสDrawAreaจะใส่ตัว Title ลงไปด้วยเพื่อเอาใช้ในการที่จะสามารถควบคุมตัวTitle นี้ไปกับDrawAreaได้ เช่นเรียกใช้method showTitleScreen()

ส่วนที่เหลือจะเป็นComponentsต่างๆที่สร้างไว้สำหรับจุเข้า JFrame และมีActionListenerของปุ่มstartButton เอาไว้ดักจับเหตุการณ์ของstartButton

Class DrawArea

```
public DrawArea(bgAnimation Bg,Title t) {
    this.t = t ;
    this.player = new Player(imgActor,100, 410, 180, 180);
    this.Starttime = System.currentTimeMillis() ;
    this.ghosts = new ArrayList<>();
    this.balls = new ArrayList<>();
    this.bgSpeed = Bg;
    initTimer();
    setFocusable(true);
    this.retryButton = new JButton("Retry");
    Font font = new Font("Courier", Font.BOLD, 30);
    retryButton.setFont(font);
    retryButton.setForeground(Color.BLACK);
    retryButton.setBackground(Color.red);
    retryButton.setVisible(false);
    retryButton.addActionListener(e -> {
        resetGame();
    });
    this.setLayout(null);
    this.add(retryButton);
    this.backButton = new JButton("Back to Title");
    Font font2 = new Font("Courier", Font.BOLD, 25);
    backButton.setFont(font2);
    backButton.setForeground(Color.BLACK);
    backButton.setBackground(Color.blue);
    backButton.setVisible(false);
```

```
    backButton.addActionListener(e -> {
        BacktoTitle();
    });
    this.setLayout(null);
    this.add(backButton);

    addKeyListener(new KeyAdapter(){
          @Override
          public void keyPressed(KeyEvent e){

              if(e.getKeyCode()==KeyEvent.VK_W){
                  player.moving();
              }
              else if(e.getKeyCode()==KeyEvent.VK_S){
                 player.crouch();
          }
          }
           @Override
           public void keyReleased(KeyEvent e) {
              if (e.getKeyCode() == KeyEvent.VK_S) {
                  player.standup();
              }
           }
});
}
```

จะรับค่า ออบเจ็กต์ไทป์ bgAnimation กับ Title เข้ามาเพื่อดึงmethod() มาใช้งาน ตั้งค่าและควบคุม จากนั้นข้างในจะจัดการสร้างออบเจ็กต์ไทป์ที่สำคัญในDrawArea เช่น player สร้างArrayList ของไทป์ Ghost , Ball ตั้งค่าพารามิเตอร์และเรียกใช้method

ภายในเช่น initTimer()(timerเวลาเริ่มต้นและเรียกใช้methodเริ่มเกม)อื่นๆช่วงเริ่มเกมและสร้างComponentsต่างๆ รวมถึงมี AcionListenerในBackbutton และมีKeyListenerในการดักจับปุ่ม w s ในการควบคุมplayer

Class bgAnimation

```
public bgAnimation(){
    xPos = 0 ;
    speed = 0 ;
    NameBg = "" ;
    xPos = 0 ;
    W  = 0 ;
    H = 0 ;
    imgbg = ImageRS.IMG_BG ;
}

 public bgAnimation(int getW,int getH,Image Bg,String NameBg){
   this.W = getW ;
   this.H  = getH ;
   this.imgbg = Bg ;
   this.NameBg = NameBg ;
}
```

ในbgAnimation () จะกำหนดค่าdefaultไว้ในกรณีที่ไม่มี การใส่พารามิเตอร์เข้ามา พารามิเตอร์ที่เป็นint จะตั้งเป็น0 NameBg เป็นString ตั้งชื่อเป็น “” คือไม่มี และ imgBg จะใช้รูปจากในclass ImageRSไปก่อนกันnull

ส่วนbgAnimationที่มีพารามิเตอร์ จะรับค่าได้แก่ getW รับ ความกว้าง getH รับความสูง Bg รับBackground และNameBg รับ ชื่อของรูป

Class Ball

```
Ball(int x,int y){
    this.x = x ;
    this.y = y ;

}
```

รับแค่แกนxกับแกนy เนื่องจากลูกบอล ในmethod draw()จะใช้เป็นfillOvalในการวาดจึงต้องการแค่พิกัด ส่วนขนาดจะคงที่

Class Character

```
public Character(){}
```

เป็นDefaultเพื่อให้คลาสลูกสามารถกำหนดค่าได้เอง เมื่อ Character เป็นคลาส abstract และจะถูกสืบทอดโดยคลาสลูก

คอนสรัคเตอร์แบบ default ช่วยให้คลาสลูกไม่ต้องบังคับกำหนดค่าเบื้องต้นทันทีในคอนสตรัคเตอร์ของคลาสแม่

Class Player

```
public Player(Image img,int x,int y,int Width,int Height){
    this.img = img ;
    this.x = x ;
    this.y = y ;
    this.Width = Width ;
    this.Height = Height ;
}
```

เป็นคอนสตัคเตอร์แบบมีพารามิเตอร์ รับรูป(img) รับแกนx แกนy รับ ขนาด(Width)และความสูง(Height) ในการเซตค่าปัจจุบัน

Class Ghost

```
public Ghost(Image img,int x,int y,int Width,int Height){
    this.img = img ;
    this.x = x ;
    this.y = y ;
    this.Width = Width ;
    this.Height = Height ;
}
```

เหมือนกันกับ class Player แต่ว่าไม่ใช่แม่ลูกกัน จึงเขียนออกมาเป็นคอนสตัคเตอร์แบบมีพารามิเตอร์ รับรูป(img) รับแกนx แกนy รับ ขนาด(Width)และความสูง(Height) ในการเซตค่าปัจจุบัน

Class KYspeed,Crow

```
public KYSpeed(Image img, int x, int y, int Width, int Height) {
    super(img, x, y, Width, Height);
}

public Crow(Image img, int x, int y, int Width, int Height) {
    super(img, x, y, Width, Height);
}
```

ในคอนสตัคเตอร์ทั้งสอง เรียกใช้คอนสตรัคเตอร์ของคลาสแม่ (Ghost) เพื่อให้คลาสแม่รับผิดชอบการตั้งค่าคุณสมบัติเบื้องต้น

4.2 Encapsulation

```
protected int x = 0 ;
protected int y = 0 ;
protected int Width = 0 ;
protected int Height = 0 ;
protected Image img ;
```

ใน class Character มีการ protected จัดการให้พารามิเตอร์ภายในใช้ได้แต่คลาสลูกเท่านั้น ซึ่งคือ Player,Ghost โดยถ้าจะดึงค่าต้องใช้method getมาช่วย เช่น getX() หรือจะตั้งค่าต้องใช้ method set เช่น setX(int x)

```
private int jumpSpeed = 25;
private int velocity = 0;
private boolean isJumping = false;
private int jumpCount = 0;
private int groundLevel = 410;
private boolean isCrouching = false ;
private boolean Death = false ;

private URL imgcrouchURL = getClass().getResource("crouch.png");
private Image imgc = new ImageIcon(imgcrouchURL).getImage();
private URL imgDeathURL = getClass().getResource("death.png");
private Image imgd = new ImageIcon(imgDeathURL).getImage();
```

ใน class Player มีการตั้งprivateสำหรับพารามิเตอร์ซึ่งสามารถใช้งานได้ภายในคลาสเท่านั้น เช่น jumpSpeed(ความเร็วการกระโดดสูง) ,velocity ความเร็วการขึ้นและตก รูปสำหรับplayer

4.3 Composition

Class Title มี

```
private DrawArea d;

private TitleDraw Td = new TitleDraw();

private bgAnimation Bg;
```

DrawArea d: ใช้สำหรับการแสดงผลในDrawArea

TitleDraw Td: ใช้สำหรับการวาดและแสดงชื่อเกม

bgAnimation Bg: ใช้สำหรับการจัดการเคลื่อนไหวของพื้นหลัง

Class DrawArea มี

```
private Player player;

private Title t ;

private bgAnimation bgSpeed;

private ArrayList<Ghost> ghosts;

private ArrayList<Ball> balls;
```

Player player: เอาไว้สร้างออบเจ็กต์ผู้เล่น ควบคุมการเล่นและอัพเดตผลและวาดในmethod() paintComponent

Ball,ghost : เอาไว้สร้างออบเจ็กต์ในการเกิดผี , เกิดบอล โดยเก็บเป็นArrayListเพื่อสร้างหลายๆครั้ง รวมถึงวาดและเช็คการชน ใน DrawArea

Title t ,bgAnimation bgSpeed: เอาไว้เก็บออบเจ็กต์ที่จุค่ามาผ่านทางคอนสตัคเตอร์ของตัวDrawArea เพื่อนำmethodของออบเจ็กต์มาใช้

4.4 Polymorphism

```
private void spawnGhosts() {
   if (!gameStarted || !ghosts.isEmpty()) {
       return;
   }
   int ghostType;
   int NyPosition = 453;
   int yPosition = 0 ;
   int Gwidth = 0, Gheight = 0;
   Image ghostImg = null;
   Ghost newGhost;

   if (bgSpeed.getNImage().equals("bg")) {
       ghostType = random.nextInt(3);
   } else {
       ghostType = random.nextInt(7);
   }

   switch (ghostType) {
       case 0:
           ghostImg = imgNghost;
           yPosition = NyPosition;
           Gwidth = 180;
           Gheight = 180;
           break;
       case 1:
           ghostImg = imgKraseu;


           yPosition = NyPosition/2 + 60;
           Gwidth = 160;
           Gheight = 200;
           break;
       case 2:
           ghostImg = imgPred;
           yPosition = 355;
           Gwidth = 230;
           Gheight = 250;
           break;
       case 3:
           ghostImg = KHimg;
           yPosition = random.nextBoolean() ? 400 : NyPosition/2 + 60 ;
           Gwidth = 150;
           Gheight = 180;
           break;
       case 4:
           ghostImg = KYimg;
           yPosition = 425;
           Gwidth = 180;
           Gheight = 180;
           break;
       case 5:
           ghostImg = Crimg;
           yPosition = random.nextBoolean() ? 450 : 360 ;
           Gwidth = 100;
           Gheight = 100;
           break;


               break;
    }

    if (ghostImg != null) {
        switch (ghostType) {
            case 5:
                newGhost = new KYSpeed(ghostImg, getWidth(), yPosition, Gwidth, Gheight);
                break;
            case 6:
                newGhost = new Crow(ghostImg, getWidth(), yPosition, Gwidth, Gheight);
                break;
            default:
                newGhost = new Ghost(ghostImg, getWidth(), yPosition, Gwidth, Gheight);
                break;
        }
        ghosts.add(newGhost);
    }
}
```

ในแต่ละ case ของ switch ที่เลือกประเภทของ Ghost จะสร้างออบเจ็กต์ของ Ghost หรือคลาสย่อยที่สืบทอดจาก Ghost เช่น KYSpeed หรือ Crow โดยขึ้นอยู่กับค่าของ ghostType.

การสร้างออบเจ็กต์ Ghost (หรือคลาสย่อยของมัน) นั้นสามารถทำได้ด้วยคำสั่ง newGhost = new Ghost(ghostImg, getWidth(), yPosition, Gwidth, Gheight); หรือ newGhost = new KYSpeed(ghostImg, getWidth(), yPosition, Gwidth, Gheight); เป็นต้น

การที่สามารถสร้างออบเจ็กต์จากคลาสที่ต่างกันแต่มีชนิดเดียวกัน (Ghost) และสามารถใช้ฟังก์ชันเดียวกันในการเพิ่มเข้าไปใน ghosts (ซึ่งเป็น ArrayList<Ghost>) แสดงถึงการใช้ Polymorphism.

ใน class Crow

```
@Override
public void moving() {
    this.x -= this.Cspeed;
}
@Override
public void update(int speed){
    this.Cspeed = speed * 2 ;
}
```

```
for (int i = ghosts.size() - 1; i >= 0; i--) {
    Ghost ghost = ghosts.get(i);
    ghost.update(Gspeed);
    bgSpeed.setSpeed(Gspeed);
    ghost.moving();

    if(ghost.isOffScreen()) {
        ghosts.remove(i);
    }
}
```

ฟังก์ชัน moving() และ update(int speed) ถูกประกาศในคลาส **Character** แต่แต่ละคลาสที่สืบทอด (เช่น **Ghost**, **KYSpeed**, และ **Crow**) จะมีการ Override เมธอดเหล่านี้เพื่อให้มีพฤติกรรมที่แตกต่างกัน

และในรูปขวาแสดงถึง Dynamic Binding เมื่อเรียกใช้ ghost.update(Gspeed) หรือ ghost.moving(), methodที่ถูกเรียกจะเป็น **Override** ในคลาสลูก (เช่น KYSpeed หรือ Crow), แต่การเลือกเมธอดที่ถูกต้องจะเกิดขึ้นเมื่อโปรแกรมทำงาน (runtime)

4.5 Abstract

```
public abstract class Character {
    protected int x = 0 ;
    protected int y = 0 ;
    protected int Width = 0 ;
    protected int Height = 0 ;
    protected Image img ;
    public Character(){}

       public int getX(){
       return this.x ;
       }
        public int getY(){
            return this.y ;
       }
        public void setX(int x){
        this.x = x;
       }
        public void setY(int y){
        this.y = y;
       }
       public int getMW() {
            return img.getWidth(null);
       }

       public int getMH() {
           return img.getHeight(null);
       }
         public void setImage(Image img) {
           this.img = img ;
       }
       public Image getImage(){
            return this.img ;
       }

    public void moving(){}
    public void update(int a){}
    public void draw(Graphics g) {
       g.drawImage(img, x, y, Width,Height , null);
   }
}
```

ในที่นี้, Character เป็น **abstract class** ที่การกำหนดโครงสร้างพื้นฐานของคลาสต่าง ๆ ไว้ ประกาศเมธอด moving() และ update(int a) ซึ่งต้องการให้คลาสลูก implement เมธอดเหล่านี้เอง เช่น ในคลาส Ghost,Player, KYSpeed, และ Crow.

4.6 Inheritance

```
public class Ghost extends Character {
    private int Gspeed = 0;
    public Ghost(Image img,int x,int y,int Width,int Height){
        this.img = img ;
        this.x = x ;
        this.y = y ;
        this.Width = Width ;
        this.Height = Height ;
     }

    @Override
    public void moving() {
        this.x -= this.Gspeed; // Move left
    }
    @Override
    public void update(int speed){
        this.Gspeed = speed ;
    }

    public boolean isOffScreen() {
        return x < -150; // Ghost is off screen
    }
}
```

Class Ghost เป็นลูกของ Character สืบทอดคุณสมบัติและเมธอดจากคลาสCharacter

```
class KYSpeed extends Ghost{
    private int kyspeed = 0;
    public KYSpeed(Image img, int x, int y, int Width, int Height) {
        super(img, x, y, Width, Height);
    }
    @Override
    public void moving() {
        this.x -= this.kyspeed;

    }
    @Override
    public void update(int speed){
        this.kyspeed = speed * 3 ;
    }
}
```

Class KYSpeed สืบทอดจาก Class Ghost ในคอนสตัคเตอร์คลาสเรียกใช้คอนสตรัคเตอร์ของคลาสแม่ (Ghost) เพื่อให้คลาสแม่รับผิดชอบการตั้งค่าคุณสมบัติเบื้องต้น

5. GUI and Event handling

Class Title

```
public Title() {

    Bg = new bgAnimation(1366,710,imgBg,"bg.png");
    d = new DrawArea(Bg,this);
    setLayout(new BorderLayout());
    add(Td, BorderLayout.CENTER);

    buttonPanel = new JPanel(new GridBagLayout());
    buttonPanel.setOpaque(false);
    JButton startButton = new JButton(new ImageIcon(imgStart));
    startButton.setPreferredSize(new Dimension(280, 160));
    Image resizedImage = imgStart.getScaledInstance(280, 160, Image.SCALE_SMOOTH);
    startButton.setIcon(new ImageIcon(resizedImage));


    startButton.setContentAreaFilled(false); //remove fill
    startButton.setBorderPainted(false); //remove line
    startButton.setMargin(new Insets(0, 0, 0, 0)); // Remove extra space around the image
    buttonPanel.add(startButton);

    Td.setLayout(null);
    Td.add(buttonPanel);
    buttonPanel.setBounds((1366 - 280) / 2, (786 - 160) / 2, 280, 160);

    p2 = new JPanel(new GridLayout(2,1,0,15));
    Font font = new Font("Courier", Font.BOLD, 20);
    Buttonscene1.setFont(font);
    Buttonscene1.setForeground(Color.BLACK);
    Buttonscene1.setBackground(Color.decode("#CC99FF"));

    Buttonscene2.setFont(font);
    Buttonscene2.setForeground(Color.BLACK);
    Buttonscene2.setBackground(Color.decode("#CC99FF"));

    p2.add(Buttonscene1);
    p2.add(Buttonscene2);
    p2.setPreferredSize(new Dimension(200, 300));
    p2.setVisible(false);
    Td.add(p2);
    p2.setBounds((1366 - 200) / 2, (786 - 150) / 2, 200, 150);

    Buttonscene1.addActionListener(new ButtonListener());
    Buttonscene2.addActionListener(new ButtonListener());
    startButton.addActionListener(e -> {
        p2.setOpaque(false);
        p2.setVisible(true);
        Td.add(p2,BorderLayout.CENTER);
        buttonPanel.setVisible(false);
        Td.repaint();

    });
}
```

```
public void showTitleScreen() {
d.setVisible(false);
remove(d);
Td.setVisible(true);
Td.getshowTitle(false);
buttonPanel.setVisible(true);
p2.setVisible(false);
Bg.setNImage("bg");
Bg.setImageBg(imgBg);
}
public static void main(String[] args) {
    Title f = new Title();
    f.setSize(1366,750);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setLocationRelativeTo(null);
    f.setVisible(true);
}
```

```
class TitleDraw extends JPanel{
private boolean showTitle = false ;
public void getshowTitle(boolean s){
    this.showTitle = s ;
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
     g.drawImage(imgBg, 0, 0, getWidth(), getHeight(), this);
    if(!showTitle){
        g.drawImage(imgTitle, (getWidth() / 2) - 330, (getHeight() / 2) - 230, 650, 130, this);
        g.drawImage(imgGhost1, getWidth() - 400, 453, 180, 180, this); // normal g
        g.drawImage(imgGhost2, getWidth() - 300, 265, 160, 200, this); // kraseu
        g.drawImage(imgGhost3, getWidth() - 200, 355, 230, 250, this); // pred
        g.drawImage(imgActor, 100, 410, 180, 180, this); // Draw actor
    }
}
}
```

<u>ส่วนประกอบของ GUI</u>

ในคลาส Titleมีการสร้าง GUI จะมีส่วนประกอบหลักๆ ดังนี้:

JFrame (Title)

JFrame คือหน้าต่างหลักของโปรแกรมที่ใช้ในการแสดงผล UI.

Title เป็นคลาสที่สืบทอดมาจาก JFrame ซึ่งหมายความว่า Title เป็นหน้าต่างที่สามารถแสดงส่วนประกอบอื่น ๆ ภายในได้.

Image และ Background

มีการใช้ Image หลายตัว เช่น imgBg, imgBg2, imgActor, imgGhost1, imgGhost2, และ imgGhost3 สำหรับภาพพื้นหลังและตัวละครต่าง ๆ ที่ใช้แสดงใน GUI.

ImageIcon ใช้สำหรับแปลง URL ให้เป็น Image.

DrawArea (d)

DrawArea คือส่วนที่ใช้ในการวาดหรือแสดงผลกราฟิก เช่น การวาดพื้นหลัง (background) หรือภาพต่าง ๆ ที่เกี่ยวข้องกับเกม.

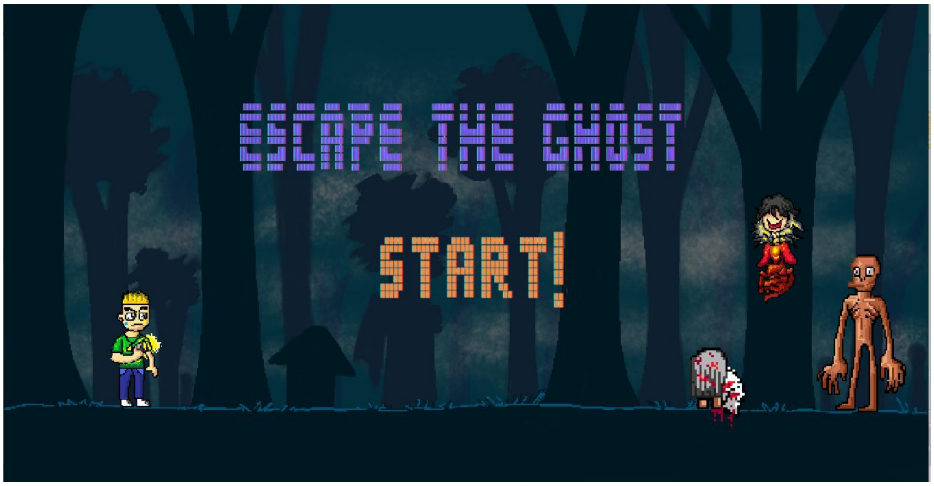
d เป็นตัวแปรที่เก็บอ็อบเจ็กต์ DrawArea.

TitleDraw (Td)

TitleDraw เป็น JPanel ที่ใช้สำหรับการวาดภาพในหน้าจอหลัก เช่น การวาดภาพตัวละคร, ภาพพื้นหลัง, และข้อความต่าง ๆ ในหน้าจอหลักของเกม.

ใช้ฟังก์ชัน paintComponent() เพื่อวาดภาพ.

ภายใน TitleDraw, เมื่อ showTitle เป็น false (เมื่อเกมเริ่ม), ระบบจะทำการวาดภาพต่าง ๆ เช่น ภาพตัวละคร (imgActor), ภาพผี (imgGhost1, imgGhost2, imgGhost3), และภาพของชื่อเกม (imgTitle).



JPanel (p2, buttonPanel)

buttonPanel: ใช้ในการวางปุ่มเริ่มเกมที่มีภาพพื้นหลัง.

p2: ใช้สำหรับแสดงปุ่มเพิ่มเติมที่ให้ผู้เล่นเลือกฉากที่ต้องการเล่น เช่น "Twisted Forest" และ "Crimson Woods".

JButton

JButton ใช้สร้างปุ่มที่สามารถคลิกได้.

ปุ่ม Buttonscene1 และ Buttonscene2 ใช้เพื่อเลือกฉากที่ผู้เล่นต้องการเล่น.

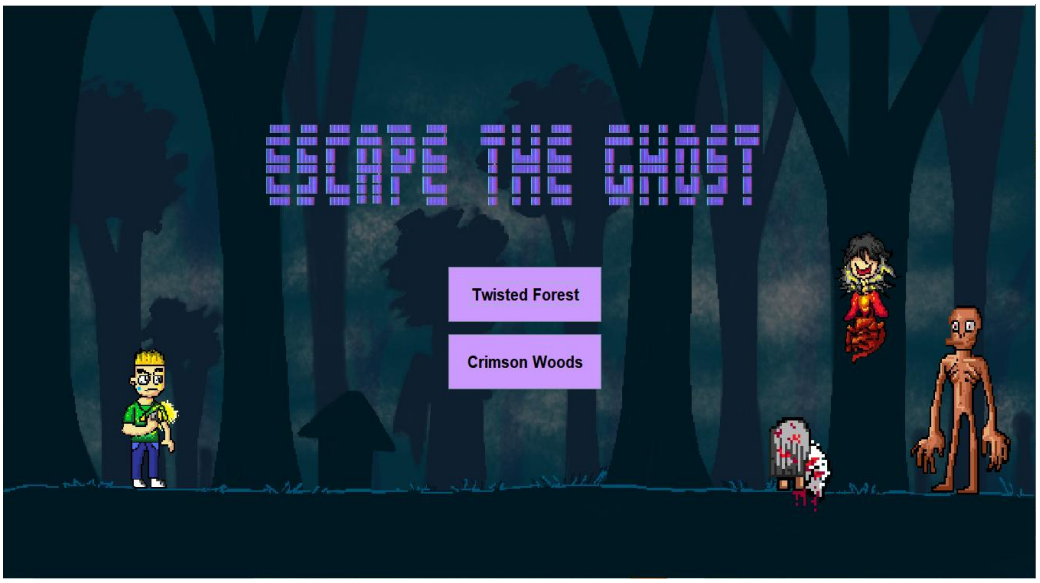
Event Handling

ในส่วนของ Event Handling จะมีการใช้ ActionListener เพื่อตอบสนองต่อการคลิกของปุ่มที่ผู้เล่นกด:

startButton.addActionListener()

เมื่อผู้เล่นคลิกปุ่ม startButton (ที่มีภาพเป็นไอคอน start.png), จะทำให้ p2 (ปุ่มเลือกฉาก) แสดงขึ้นมา และ buttonPanel หายไป (ไม่แสดง).

เรียก Td.repaint() เพื่อทำให้หน้าจอวาดใหม่.



ButtonListener

ในคลาส ButtonListener มี actionPerformed() ที่จะตรวจสอบว่า ActionEvent ที่เกิดขึ้นมาจากปุ่มไหน (เช่น "Twisted Forest" หรือ "Crimson Woods").

เมื่อผู้เล่นคลิกปุ่ม "Twisted Forest" หรือ "Crimson Woods", ระบบจะเปลี่ยนพื้นหลังของเกมไปยังฉากที่เลือก (ผ่านการตั้งค่าภาพพื้นหลังใหม่ใน Bg).

Class DrawArea

```
this.retryButton = new JButton("Retry");
Font font = new Font("Courier", Font.BOLD, 30);
retryButton.setFont(font);
retryButton.setForeground(Color.BLACK);
retryButton.setBackground(Color.red);
retryButton.setVisible(false);
retryButton.addActionListener(e -> {
    resetGame();
});
this.setLayout(null);
this.add(retryButton);
this.backButton = new JButton("Back to Title");
Font font2 = new Font("Courier", Font.BOLD, 25);
backButton.setFont(font2);
backButton.setForeground(Color.BLACK);
backButton.setBackground(Color.blue);
backButton.setVisible(false);
backButton.addActionListener(e -> {
    BacktoTitle();
});
this.setLayout(null);
this.add(backButton);
```

```
addKeyListener(new KeyAdapter(){
        @Override
        public void keyPressed(KeyEvent e){

            if(e.getKeyCode()==KeyEvent.VK_W){
                player.moving();
            }
            else if(e.getKeyCode()==KeyEvent.VK_S){
                player.crouch();
        }
        }
         @Override
          public void keyReleased(KeyEvent e) {
              if (e.getKeyCode() == KeyEvent.VK_S) {
                  player.standup();
              }
          }
});
}
```

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if(gameStarted){
        bgSpeed.draw(g);
        player.draw(g);
        for (Ball ball : balls) {
        ball.draw(g);
        }
        for (Ghost ghost : ghosts) {
        ghost.draw(g);
        }
        g.setColor(Color.orange);
        g.setFont(new Font("Courier", Font.PLAIN, 32));
        g.setColor(scoreColor);
        g.drawString("SCORE : " + String.valueOf(score), getWidth() - 250, 35);
        g.setColor(timeColor);
        g.setFont(new Font("Courier", Font.PLAIN, 32));
        g.drawString(String.valueOf(TimeLimit/60), 30, 35);

            if(gameUpdate>0){
                if(gameUpdate == 1){
                    g.setColor(Color.orange);
                    g.setFont(new Font("Courier",Font.PLAIN,70));
                    g.drawString("GAME OVER",getWidth()/2-190, getHeight()/2-80);
                }
                else{
                    g.setColor(Color.GREEN);
                    g.setFont(new Font("Courier",Font.PLAIN,100));
                    g.drawString("YOU WIN !",getWidth()/2-220, getHeight()/2-80);
                }
                timer.stop();
                retryButton.setBounds(getWidth() / 2 - 50, getHeight() / 2, 150, 50);
                retryButton.setVisible(true);
                backButton.setBounds(getWidth() / 2 - 73, getHeight() / 2 + 60, 200, 50);
                backButton.setVisible(true);

            }
            else{
                retryButton.setVisible(false);
                backButton.setVisible(false);
                }
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
    player.update();
    repaint();
    }
}
```

```
private void initTimer() {
    timer = new Timer(1000/60, e -> {
        spawnGhosts();
        spawnBalls();
        updateGame();
        repaint();
    });
    timer.start();
}
```

```
private void ColorReset(int a) {
    if (a == 1) {
        if (scoreTimer != null && scoreTimer.isRunning()) {
            scoreTimer.stop();
        }
        scoreTimer = new Timer(3000, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                scoreColor = Color.ORANGE;
                scoreTimer.stop();
            }
        });
        scoreTimer.start();
    } else if (a == 2) {
        if (timeTimer != null && timeTimer.isRunning()) {
            timeTimer.stop();
        }
        timeTimer = new Timer(3000, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                timeColor = Color.ORANGE;
                timeTimer.stop();
            }
        });
        timeTimer.start();
    }
```

ส่วนประกอบของ GUI

JPanel (DrawArea):

คลาส DrawArea ขยายจาก JPanel และใช้ในการแสดงผลกราฟิกต่าง ๆ ของเกม รวมถึงการจัดการกับ Event Handling และการจัดการเกมภายใน.

ปุ่ม (JButton):

retryButton: ปุ่มที่ใช้สำหรับเริ่มเกมใหม่ (เมื่อเกมจบ).

backButton: ปุ่มที่ใช้สำหรับกลับไปยังหน้าหลักของเกม.

ตัวแสดงผลของข้อมูล (score, TimeLimit):

คะแนน (score) และเวลาที่เหลือ (TimeLimit) จะถูกแสดงในกราฟิกของเกม.

การวาดในpaintComponent:

ตัวละคร (Player): วาดตัวละคร, ตัวผี (Ghost): วาดผี, ลูกบอล (Ball): วาดบอล, แอนิเมชันพื้นหลัง (bgAnimation):

วาดภาพพื้นหลัง

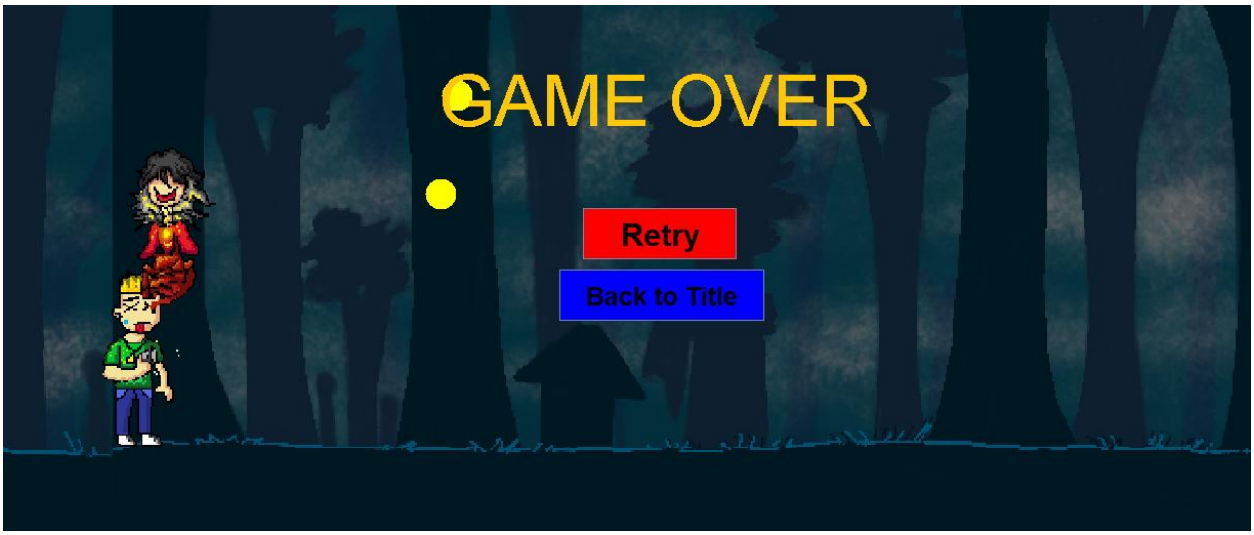
Event Handling

ActionListener:

ปุ่ม retryButton และ backButton มีการใช้ ActionListener เพื่อฟังเหตุการณ์เมื่อผู้ใช้คลิกปุ่ม:

เมื่อคลิกที่ปุ่ม retryButton จะรีเซ็ตเกม.

เมื่อคลิกที่ปุ่ม backButton จะพาผู้เล่นกลับไปที่หน้าหลักของเกม.



KeyListener:

ใน keyPressed และ keyReleased เราใช้ KeyListener เพื่อจัดการกับการกดปุ่ม:

เมื่อกดปุ่ม W จะทำให้ตัวละครกระโดด.

เมื่อกดปุ่ม S จะทำให้ตัวละครนั่ง (Crouch).

เมื่อปล่อยปุ่ม S จะทำให้ตัวละครยืนขึ้น (standup).

Timer:

timer: ใช้ในการอัพเดตสถานะของเกมทุก ๆ 1/60 วินาที เช่น การสร้างผีใหม่, การอัปเดตตำแหน่งของตัวละคร, และการเช็คการชน.

scoreTimer และ timeTimer: ใช้เพื่อปรับสีของคะแนนและเวลาเป็นสีขาวชั่วคราวเมื่อชนกับผีบางตัว.

ActionPerformed:

ใน actionPerformed จะมีการอัปเดตสถานะของตัวละครและวาดใหม่ทุกครั้งที่มีการกระทำ.

6. Algorithim ที่สำคัญ

Collision()

```
private void Collision(){
    int paddingX = 56 ;
    int paddingY = player.isCrouching() ? 60 : -30;

    Iterator<Ball> ballIterator = balls.iterator();
    while (ballIterator.hasNext()) {
    Ball ball = ballIterator.next();
    if (player.getX() < ball.getX() + ball.getW() &&
        player.getX() + player.getMW() > ball.getX() &&
        player.getY() < ball.getY() + ball.getH() &&
        player.getY() + player.getMH() > ball.getY()) {
        ballIterator.remove();
        score += 100;
    }
    }

    Iterator<Ghost> ghostIterator = ghosts.iterator();
    while (ghostIterator.hasNext()) {
    Ghost ghost = ghostIterator.next();

        if (player.getX() + paddingX < ghost.getX() + ghost.getMW() &&
            player.getX() - paddingX + player.getMW() > ghost.getX() &&
            player.getY() + paddingY < ghost.getY() + ghost.getMH() &&
            player.getY() - paddingY + player.getMH() > ghost.getY()) {
          if (ghost instanceof KYSpeed) {
                ghostIterator.remove();
                score = score - 200 ;
                scoreColor = Color.WHITE;
                ColorReset(1);
            } else if (ghost instanceof Crow) {
                ghostIterator.remove();
                TimeLimit = TimeLimit - 5*60;
                timeColor = Color.WHITE;
                ColorReset(2);
            } else if (ghost instanceof Ghost) {
                player.isDeath(1);
                gameUpdate = 1;
                break;
          }
          }
}
}
```

ขั้นตอนการทำงาน:

กำหนดค่าตัวแปร padding:

-paddingX = 56: ระยะห่างในแนวนอนที่ใช้ในการตรวจสอบการชน.

-paddingY = player.isCrouching() ? 60 : -30: ระยะห่างในแนวตั้งที่ใช้ในการตรวจสอบการชน. หากผู้เล่นอยู่ในท่าก้ม (isCrouching() เป็น true), ค่า paddingY จะเป็น 60, ถ้าไม่ใช่จะเป็น -30.

การตรวจสอบการชนกับลูกบอล (Ball):

-ใช้ Iterator เพื่อวนลูปผ่านลิสต์ของลูกบอล (balls).

-ตรวจสอบว่าตำแหน่งของผู้เล่น (โดยใช้ค่าตำแหน่ง getX() และ getY()) ตรงกับตำแหน่งของลูกบอล (โดยใช้ getX(), getY(), getW(), และ getH()).

-ถ้าผู้เล่นชนกับลูกบอล (มีการทับซ้อนของตำแหน่งระหว่างผู้เล่นและลูกบอล), ลูกบอลนั้นจะถูกลบออกจากลิสต์ (ballIterator.remove()) และจะเพิ่มคะแนนให้ผู้เล่น 100 คะแนน (score += 100).

การตรวจสอบการชนกับผี (Ghost):

-ใช้ Iterator เพื่อวนลูปผ่านลิสต์ของผี (ghosts).

-ตรวจสอบว่าตำแหน่งของผู้เล่น (โดยใช้ค่าตำแหน่ง getX() และ getY()) ตรงกับตำแหน่งของผี (โดยใช้ getX(), getY(), getMW(), และ getMH()).

-การตรวจสอบการชนจะมีการคำนึงถึงค่า paddingX และ paddingY ที่กำหนดในขั้นตอนก่อนหน้านี้.

การจัดการเมื่อเกิดการชนกับผี:

-หากผู้เล่นชนกับผีประเภท KYSpeed:

-ผีประเภทนี้จะถูกลบออกจากลิสต์ (ghostIterator.remove()).

-คะแนนของผู้เล่นจะลดลง 200 คะแนน (score = score - 200).

-เปลี่ยนสีของคะแนนให้เป็นสีขาว (scoreColor = Color.WHITE).

-เรียกฟังก์ชัน ColorReset(1) เพื่อตั้งค่าการรีเซ็ตสี.

-หากผู้เล่นชนกับผีประเภท Crow:

-ผีประเภทนี้จะถูกลบออกจากลิสต์ (ghostIterator.remove()).

-เวลาในเกมจะลดลง 5 หน่วย (5 * 60) (TimeLimit = TimeLimit - 5*60).

-เปลี่ยนสีของเวลาให้เป็นสีขาว (timeColor = Color.WHITE).

-เรียกฟังก์ชัน ColorReset(2) เพื่อตั้งค่าการรีเซ็ตสี.

-หากผู้เล่นชนกับผีประเภท Ghost:

-ฟังก์ชัน player.isDeath(1) จะถูกเรียกทำให้ผู้เล่นตาย

-ค่า gameUpdate = 1 ถูกตั้งค่าเพื่อเปลี่ยนสถานะของเกม.

-การจบการทำงาน: เมื่อพบการชนกับผีประเภท Ghost (ผู้เล่นตาย), ฟังก์ชันจะหยุดการทำงาน (break) เพื่อไม่ให้ตรวจสอบการชนกับผีอื่น ๆ ต่อไป.

บทที่ 3 สรุป

1. ปัญหาที่พบระหว่างการพัฒนา

-ระบบความคิดของตัวผู้ทำเกมและความเข้าใจในเนื้อหาOOPเมื่อแรกเริ่มนั้น ยังไม่แม่นเท่าที่ควร จึงทำให้การออกแบบเกมค่อนข้างไม่เป็นระเบียบ และใช้ทฤษฎีผิดๆถูกๆ

-ยังรู้จักเครื่องมือไม่กว้างพอ เพราะบางอันมีความเสถียรกว่าของที่เขียนปกติ

2. จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

-ระบบผีมีการเคลื่อนที่ความเร็วต่างกัน และ มีการแสดงผลที่ต่างกัน เช่น ลดคะแนน ลดเวลา

-ผีบางตัวแกนy ไม่เท่ากันทำให้กระโดดหรือก้มสนุก

3. คำแนะนำ

-ควรมีการบอกรายละเอียดที่ชัดเจนต้องแจ้งงานว่า ในตัวเกม ควรมีอะไรบ้าง เช่น ด่านมากกว่า 1 ด่าน เป็นต้น