Mini Project

เกมกระต่ายเก็บแครอท (Bunny Bunny)

จัดทำโดย

นางสาวจารุพร ธนจุติพร 6604062636089

เสนอ

ผู้ช่วยศาสตราจารย์สถิตย์ ประสมพันธ์

โครงงานนี้เป็นส่วนหนึ่งของวิชา Object Oriented Programming รหัสวิชา 040613204

คณะวิทยาศาสตร์ประยุกต์ ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ภาคเรียนที่ 1 ปีการศึกษา 2567

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

โครงงานนี้จัดทำขึ้นเพื่อวัดผลการเรียนในรายวิชา Object Oriented Programming โดยการนำเนื้อหาในบทเรียนมาประยุกต์สร้างเป็นชิ้นงานในรูปแบบเกม "Bunny Bunny" ซึ่งเป็นเกมเล่นคนเดียวที่ออกแบบมาเพื่อมอบความบันเทิงและฝึกทักษะของผู้เล่น โดยผู้เล่นจะต้องควบคุมกระต่ายให้วิ่งเก็บแครอทเพื่อสะสมแต้ม ในขณะเดียวกันต้องหลบหลีกระเบิดไปพร้อมๆกัน

1.2 ประเภทของโครงงาน

เป็นการพัฒนาเกมประเภท Action Puzzle ซึ่งผสมผสานการใช้ทักษะการตอบสนองอย่างรวดเร็วกับการคำนวณและเลือกเส้นทางเพื่อหลีกเลี่ยงการเก็บระเบิด เกมเป็นรูปแบบ Single-player โดยผู้เล่นจะควบคุมตัวละครกระต่ายเพื่อเก็บแครอทและหลบระเบิด ผู้เล่นต้องใช้ทักษะความเร็วในการตัดสินใจและการควบคุมตัวละครเพื่อสะสมคะแนนให้ได้มากที่สุด

1.3 ประโยชน์ของโครงงาน

1.3.1 เพื่อนำเนื้อหาที่เรียนมาใช้งานจริง

1.3.1 เพื่อฝึกความรวดเร็วในการตัดสินใจ

1.3.2 เพื่อความสนุกและลดความเครียด

1.3.3 เพื่อเสริมสร้างสมาธิ

1.4 ขอบเขตของโครงงาน

ลำดับ	รายการ	1-5 ต.ค. 67	6-20 ต.ค. 67	21-30 ต.ค. 67
1	ออกแบบกราฟิกและตัวละคร			
2	ศึกษาเอกสารและข้อมูลที่เกี่ยวข้อง			
3	เขียนโปรแกรม			
4	จัดทำเอกสาร			
5	ตรวจสอบข้อผิดพลาด			

1.5 Source Code

https://github.com/6604062636089/BunnyBunny.git

บทที่ 2

การพัฒนา

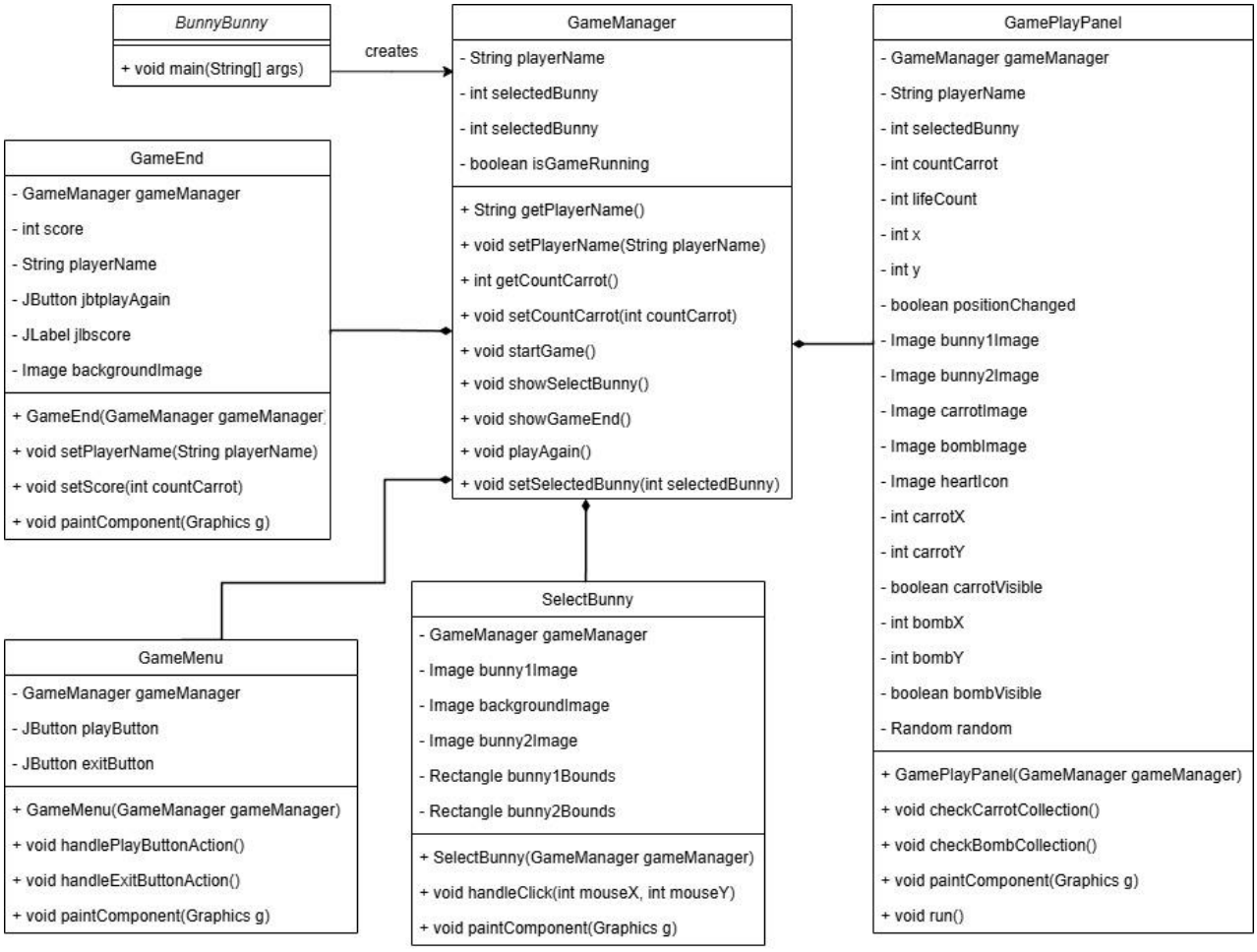
2.1 รายละเอียดเกม

เกมนี้กระต่ายจะต้องวิ่งเก็บแครอทเพื่อสะสมแต้ม โดยทุกครั้งที่เก็บแครอทได้ จะได้รับ 1 แต้ม เป้าหมายคือเก็บให้ได้มากที่สุด แต่หากเผลอไปเก็บระเบิด จะเสียหัวใจครั้งละ 1 ดวง โดยเกมจะจบลงเมื่อสูญเสียหัวใจครบ 3 ดวง หรือเวลาหมดลง ความท้าทายคือการวิ่งเก็บแครอทให้มากที่สุดโดยไม่เก็บระเบิด

2.2 วิธีการเล่น

ผู้เล่นสามารถควบคุมกระต่ายโดยกดปุ่ม A เพื่อให้กระต่ายเดินไปทางซ้าย และกดปุ่ม D เพื่อให้กระต่ายเดินไปทางขวา กระต่ายจะต้องเคลื่อนไหวให้สัมผัสกับแครอทที่ตกลงมาจากด้านบนเพื่อเก็บคะแนน

2.3 Class Diagram



2.3.1 Class BunnyBunny (Main) : เป็นจุดเริ่มต้นของโปรแกรม เริ่มต้นแสดงหน้าจอเมนูเกม

2.3.2 Class GameManager : เป็นตัวจัดการหลักของเกม ใช้ควบคุมการทำงานต่างๆ การเริ่มเกม การเลือกกระต่าย การนับคะแนน และการแสดงผลต่างๆ

2.3.3 Class StartMenu : แสดงเมนูเริ่มต้นของเกม มีให้กรอกชื่อผู้เล่น

2.3.4 Class SelectBunny : แสดงหน้าจอให้ผู้เล่นเลือกกระต่ายที่จะใช้ในการเล่นเกม

2.3.5 Class GamePlayPanel : แสดงหน้าจอขณะเล่นเกม การต่าย แครอท ระเบิด หัวใจ และการนับคะแนน

2.3.6 Class GameEnd : แสดงหน้าจอเมื่อจบเกม โดยแสดงคะแนนสุดท้าย และมีให้กดเล่นอีกครั้ง

2.4 รูปแบบการพัฒนา

เกม Bunny Bunny เป็นแบบ Java Application

2.5 Constructor

2.5.1 Class GameManager

```
public GameManager() {
    cardLayout = new CardLayout();
    setLayout(cardLayout);

    //create Page
    startMenu = new StartMenu(this);
    selectBunny = new SelectBunny(this);
    gamePlayPanel = new GamePlayPanel(this);
    gameEnd = new GameEnd(this);

    add(startMenu, "StartMenu");
    add(selectBunny, "SelectBunny");
    add(gamePlayPanel, "GamePlay");
    add(gameEnd, "Game Over");

    showStartMenu();
}
```

- สร้างตัวแปร CardLayout เพื่อจัดการการแสดงผลของแต่ละหน้าจอ

- สร้างหน้าต่างๆ (Panels) คือ StartMenu, SelectBunny, GamePlayPanel, และ GameEnd

- เพิ่ม Panels ที่สร้างขึ้นไปใน GameManager โดยใช้ CardLayout สำหรับการสลับแสดงหน้าต่าง

- เรียกใช้ showStartMenu() เพื่อแสดงหน้าเริ่มต้น

2.5.2 Class StartMenu

```
public StartMenu(GameManager gameManager){
    setLayout(null);
    jtfInputName.setBounds(680, 300, 250, 40);//(x, y, width, height)
    jbtStart.setBounds(730, 400, 160, 49);

    Border roundedBorder = new LineBorder(Color.BLACK, 3, true);
    jtfInputName.setHorizontalAlignment(JTextField.CENTER);
    jtfInputName.setBorder(roundedBorder);
    jtfInputName.setOpaque(false);//no bg

    ImageIcon startIcon = new ImageIcon(getClass().getResource("jbtStart.png"));
    jbtStart.setIcon(startIcon);
    jbtStart.setHorizontalAlignment(SwingConstants.CENTER);
    jbtStart.setVerticalAlignment(SwingConstants.CENTER);
    jbtStart.setContentAreaFilled(false); //no bg


     jbtStart.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String playerName = jtfInputName.getText().trim();
            if(!playerName.isEmpty()) {
                gameManager.setPlayerName(playerName);
                gameManager.showSelectBunny(); //call Page SelectBunny
            } else {
                JOptionPane.showMessageDialog(StartMenu.this, "Please enter your name.");
            }
        }
    });
    add(jtfInputName);
    add(jbtStart);
    try {
        backgroundImage = ImageIO.read(getClass().getResource("Bghome.png"));
    } catch (IOException ex) {
        System.out.println("Error loading background image: " + ex.getMessage());
    }
}
```

- ตั้งค่า Layout โดยใช้ null layout เพื่อให้สามารถจัดตำแหน่งคอมโพเนนต์ได้เอง

- ตั้งค่า JTextField สำหรับกรอกชื่อผู้เล่น

- ตั้งค่า JButton (Start) พร้อมการตั้งค่าภาพ Icon

- เพิ่ม ActionListener ให้กับปุ่มเริ่ม (Start) เพื่ออ่านชื่อผู้เล่นจาก JTextField และส่งข้อมูลไปยัง GameManager

- โหลดภาพพื้นหลัง โดยใช้ ImageIO.read() และจับข้อผิดพลาดหากไม่สามารถโหลดได้

2.5.3 Class SelectBunny

```
public SelectBunny(GameManager gameManager) {

    this.gameManager = gameManager;

    backgroundImage = new ImageIcon(getClass().getResource("BgSelectBunny.png")).getImage();
    bunny1Image = new ImageIcon(getClass().getResource("bunny1.png")).getImage();
    bunny2Image = new ImageIcon(getClass().getResource("bunny2.png")).getImage();

    bunny1Bounds = new Rectangle(200, 150, 300, 600);
    bunny2Bounds = new Rectangle(680, 175, 300, 550);

    addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            handleClick(e.getX(), e.getY());
        }
    });

    setLayout(null);
}
```

- รับ GameManager เพื่อให้สามารถเชื่อมต่อกับ GameManager และใช้ฟังก์ชันจากมันได้

- โหลดภาพพื้นหลังและภาพกระต่ายสองตัว

- กำหนดตำแหน่งและขนาดของพื้นที่ที่กระต่ายแต่ละตัวจะอยู่บนหน้าจอ โดยใช้ Rectangle สำหรับการตรวจจับการคลิก

- เพิ่ม MouseListener โดยใช้ MouseAdapter เพื่อเพิ่มการตรวจจับเหตุการณ์การคลิกที่ตำแหน่งต่างๆ ของกระต่าย

- ตั้งค่า Layout โดยใช้ null layout เพื่อให้สามารถจัดตำแหน่งคอมโพเนนต์ได้เอง

2.5.4 Class GamePlayPanel

```
public GamePlayPanel(GameManager gameManager) {
    this.gameManager = gameManager;
    this.playerName = gameManager.getPlayerName();//call player name from GameManager
    setLayout(null);

    // Set focusable and request focus
    setFocusable(true);

    backgroundImage = new ImageIcon(getClass().getResource("BgGame.png")).getImage();
    startIcon = new ImageIcon(getClass().getResource("StartIcon.png")).getImage();
    startIconBounds = new Rectangle(400, 200, 400, 250);
    bunny1Image = new ImageIcon(getClass().getResource("bunny1.png")).getImage();
    bunny2Image = new ImageIcon(getClass().getResource("bunny2.png")).getImage();
    heartIcon = new ImageIcon(getClass().getResource("heart1.png")).getImage();
    carrotImage = new ImageIcon(getClass().getResource("carrot.png")).getImage();
    bombImage = new ImageIcon(getClass().getResource("bomb.png")).getImage();

    jlbplayName.setText(playerName);
    jlbplayName.setBounds(980, 3, 1000, 100);
    jlbplayName.setForeground(Color.BLACK);
    add(jlbplayName);

    // Display countdown timer
    countdownLabel.setText("Time Left: " + formatTime(countdownTime));
    countdownLabel.setBounds(10, 65, 200, 50);
    countdownLabel.setFont(new Font("Arial", Font.BOLD, 20));
    countdownLabel.setForeground(Color.RED);
    add(countdownLabel);

    // Timer to handle countdown
    countdownTimer = new Timer(1000, e -> {
        countdownTime--;
        countdownLabel.setText("Time Left: " + formatTime(countdownTime));
        if (countdownTime <= 0) {
            countdownTimer.stop();
            gameManager.setCountCarrot(countCarrot);
            gameManager.showGameEnd();
            running = false; // Stop the threads
            lifeCount -= 1;
            System.out.println("Time Out!");
        }
    });

    // Generate initial carrot position
    generateCarrot();

    // Start thread for falling carrot
    Thread carrotThread = new Thread(this);
    Thread bombThread = new Thread(new BombFallingRunnable());
```

```
    addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {
            positionChanged = false; // Reset flag on new key press
            if (e.getKeyChar() == 'a' || e.getKeyChar() == 'A') {
                x -= moveAmount;
                positionChanged = true;
            } else if (e.getKeyChar() == 'd' || e.getKeyChar() == 'D') {
                x += moveAmount;
                positionChanged = true; // Mark position as changed
            }
            x = Math.max(0, Math.min(x, getWidth() - 135));
            if (positionChanged) {
                //System.out.println("Bunny moved to x: " + x + ", y: " + y);
            }
            checkCarrotCollection(); //Check for carrot collection
            repaint();
        }
    });

    addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (clickCounter == 0) {
                requestFocusInWindow(); //Request focus on mouse click
                clickCounter += 1;
                startIcon = null;
                countdownTimer.start(); //Start countdown timer
                carrotThread.start();
                bombThread.start();
            }
        }
    });

    addFocusListener(new FocusAdapter() {
        @Override
        public void focusGained(FocusEvent e) {
            System.out.println("GamePlayPanel has focus");
        }

        @Override
        public void focusLost(FocusEvent e) {
            System.out.println("GamePlayPanel has lost focus");
        }
    });
}
```

- รับ GameManager และ playerName จากข้อมูล GameManager

- ตั้งค่า Layout และ focus ให้ panel สามารถรับ focus ได้

- โหลดภาพพื้นหลัง กระต่าย ระเบิด Icon ต่างๆ ที่ใช้ในเกม

- แสดงชื่อของผู้เล่นและเวลาบนหน้าจอ

- ใช้ Timer สำหรับจับเวลาในเกม และแสดงเวลาที่เหลือ

- สร้างและจัดการกับ Thread สำหรับการตกของแครอทและระเบิด

- เพิ่ม KeyListener และ MouseListener เพื่อควบคุมการเคลื่อนที่ของกระต่ายด้วยปุ่มคีย์และเริ่มเกมเมื่อคลิกเมาส์

2.5.5 Class GameEnd

```
public GameEnd(GameManager gameManager) {
    setLayout(null);
    backgroundImage = new ImageIcon(getClass().getResource("BgEndGame.png")).getImage();

    System.out.println("final score");

    this.gameManager = gameManager;

    jbtplayAgain.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(!playerName.isEmpty()) {
                //gameManager.showSelectBunny(); // call Page SelectBunny
                  gameManager.playAgain();
            } else {
                //JOptionPane.showMessageDialog(StartMenu.this, "Please enter your name.");
            }
        }
    });
    jbtplayAgain.setBounds(540, 520, 100, 50);
    add(jbtplayAgain);

    jlbscore.setFont(new Font("Arial", Font.BOLD, 50));
    jlbscore.setForeground(Color.BLACK);


    add(jlbscore);
}
```

- ตั้งค่า Layout โดยใช้ null layout เพื่อให้สามารถจัดตำแหน่งคอมโพเนนต์ได้เอง

- โหลดภาพพื้นหลังในการแสดงผลบนหน้าจอเมื่อเกมจบ

- รับ GameManager เพื่อให้สามารถเชื่อมต่อกับ GameManager และใช้ฟังก์ชันจากมันได้

- เพิ่ม ActionListener ให้กับปุ่มเล่นอีกครั้ง (Play Again) เพื่อให้สามารถเริ่มเกมใหม่ได้เมื่อคลิก

- กำหนดฟอนต์และสีสำหรับการแสดงคะแนนของผู้เล่น

2.6 Encapsulation

การใช้ Encapsulation โดยการตั้งค่า attribute เป็น private และการใช้งาน getter/setter methods จะช่วยเพิ่มความปลอดภัยในการจัดการข้อมูลภายในคลาส รวมถึงช่วยให้โค้ดมีความยืดหยุ่นและสามารถปรับปรุงได้ง่ายในอนาคต ตัวอย่าง code จาก Class GameManager

```
private String playerName;
private int selectedBunny;
private int countCarrot;
```

```
public int getSelectedBunny() {
    return selectedBunny;
}

public void setSelectedBunny(int selectedBunny) {
    this.selectedBunny = selectedBunny;
}

public String getPlayerName() {
    return playerName;
}

public void setPlayerName(String playerName) {
    this.playerName = playerName;
}

public void setCountCarrot(int countCarrot) {
    this.countCarrot = countCarrot;
}

public int getCountCarrot() {
    return countCarrot;
}
```

2.7 Composition

เป็นการรวมคลาสต่างๆ คือ StartMenu SelectBunny GamePlayPanel และ GameEnd เข้าไว้ในตัวมันเอง โดยที่ GameManager ควบคุมการทำงานและแสดงผลของ Class เหล่านี้ คลาสที่ถูกประกอบเข้าไปนั้นสามารถมีความรับผิดชอบหรือหน้าที่ของตัวเองที่ไม่เกี่ยวข้องกับคลาสอื่น ๆ แต่การมีอยู่ภายใน Composition ของ GameManager ทำให้มันสามารถทำงานร่วมกันได้ ตัวอย่าง code จาก Class GameManager

```
private StartMenu startMenu;
private SelectBunny selectBunny;
private GamePlayPanel gamePlayPanel;
private GameEnd gameEnd;
```

```
startMenu = new StartMenu(this);
selectBunny = new SelectBunny(this);
gamePlayPanel = new GamePlayPanel(this);
gameEnd = new GameEnd(this);

add(startMenu, "StartMenu");
add(selectBunny, "SelectBunny");
add(gamePlayPanel, "GamePlay");
add(gameEnd, "Game Over");
```

- Class GameManager มีสมาชิกภายในเป็น objects ของคลาสอื่น คือ StartMenu SelectBunny GamePlayPanel และ GameEnd

- แต่ละ Class ถูกสร้างขึ้นใน constructor ของ GameManager() และถูกเพิ่มเข้าไปใน layout ของ GameManager

- Composition ช่วยให้ GameManager สามารถควบคุมหน้าจอต่างๆ ของเกมได้ โดยการแสดง panel ที่แตกต่างกัน คือ เมนูเริ่มต้น (startMenu) เลือกกระต่าย (selectBunny) การเล่นเกม (gamePlayPanel) และจบเกม (gameEnd)

2.8 Polymorphism & Inheritance

ไม่มีการใช้งาน Polymorphism แบบชัดเจน ทั้งในลักษณะของการ Overload หรือ Override methods โดยตรง แต่ยังมีบางจุดที่อาจจะเกี่ยวข้องกับ Polymorphism คือ Event Handling - Polymorphism ผ่าน Interfaces ได้ ตัวอย่าง code จาก Class StartMenu

```
jbtStart.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String playerName = jtfInputName.getText().trim();
        if(!playerName.isEmpty()) {
            gameManager.setPlayerName(playerName);
            gameManager.showSelectBunny(); //call Page SelectBunny
        } else {
            JOptionPane.showMessageDialog(StartMenu.this, "Please enter your name.");
        }
    }
});
```

```
public class GameEnd extends JPanel
```

Class StartMenu SelectBunny GamePlayPanel และ GameEnd สืบทอดจาก JPanel เพื่อให้สามารถใช้งานคุณสมบัติของ JPanel ในการสร้าง UI ได้ ได้ ตัวอย่าง code จาก Class GameEnd

2.9 Abstract

ไม่มีการใช้งาน abstract class หรือ abstract method

2.10 Timmer & Thread

ใน Class GamePlayPanel มีการใช้ Timer ในการจับเวลา countdown ซึ่งจะทำการอัพเดตทุกๆ 1 วินาที (1000 ms) โดยจะลดค่าเวลาลงและแสดงใน UI

```
countdownTimer = new Timer(1000, e -> {
    countdownTime--;
    countdownLabel.setText("Time Left: " + formatTime(countdownTime));
    if (countdownTime <= 0) {
        countdownTimer.stop();
        gameManager.setCountCarrot(countCarrot);
        gameManager.showGameEnd();
        running = false; // Stop the threads
        lifeCount -= 1;
        System.out.println("Time Out!");
    }
});
```

ใน Class GamePlayPanel มีการใช้ Thread สำหรับการทำงานที่เกี่ยวข้องกับการตกของแครอทและระเบิด

```
Thread carrotThread = new Thread(this);
Thread bombThread = new Thread(new BombFallingRunnable());

carrotThread.start();
bombThread.start();
```

- Thread จะถูกสร้างขึ้นและรันโดยใช้ start() เพื่อให้การทำงานในฟังก์ชัน run() ที่ implement ใน Runnable

- carrotThread ใช้ GamePlayPanel เองในการ implement Runnable (implements Runnable), ดังนั้นจะใช้ method run() ที่ถูก implement ไว้ใน GamePlayPanel

- bombThread ใช้ BombFallingRunnable ซึ่งเป็นคลาสอื่นที่ implement Runnable และทำการตกของระเบิด

2.11 Interface

ใน Class GamePlayPanel ได้ implement interface Runnable เพื่อให้สามารถใช้กับ Thread ในการทำงานร่วมกับการเคลื่อนที่ของแครอทและระเบิด

```
@Override
public void run() {
    while (running) {
        if (carrotVisible) {
            carrotY += 5; //move carrot
            if (carrotY > getHeight()) {
                generateCarrot(); //regenerate carrot if it goes off screen
            }
            checkCarrotCollection(); //check if the carrot is collected
            repaint(); //update the panel with new carrot position
        }
        try {
            Thread.sleep(30); //control carrot movement speed
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

- Class GamePlayPanel ใช้ interface Runnable เพื่อทำการเคลื่อนที่ของแครอท

```
private class BombFallingRunnable implements Runnable {

    @Override
    public void run() {
        while (running) {
            try {
                Thread.sleep(2000); //wait for 2s
                generateBomb(); //regenerate a new bomb every 2 seconds
                while (bombVisible) {
                    bombY += 5; //move bomb
                    if (bombY > getHeight()) {
                        bombVisible = false; //hide bomb if it goes off screen
                    }
                    checkBombCollection(); //check if the bomb is collected
                    repaint(); //update the panel with new bomb position
                    Thread.sleep(30); //control bomb movement speed
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

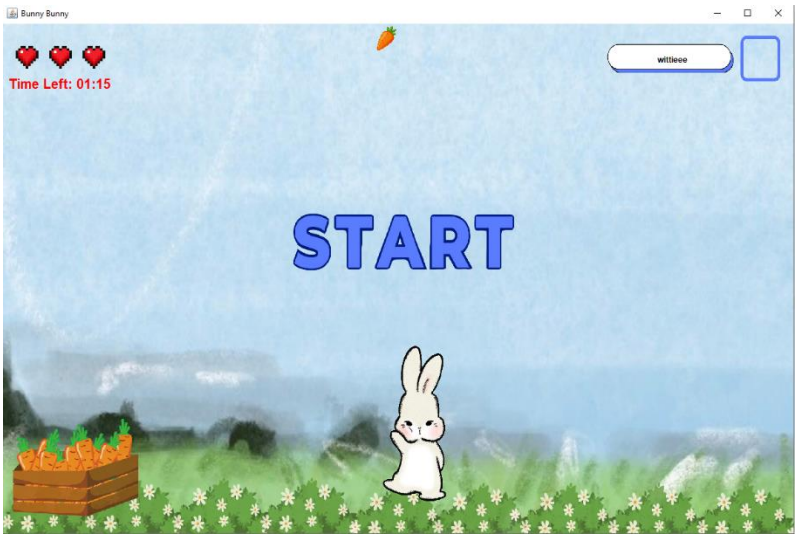
- BombFallingRunnable เป็น class ย่อยที่ implement Runnable เพื่อจัดการกับการเคลื่อนที่ของระเบิด

2.12 GUI

2.12.1 หน้า StartMenu ประกอบด้วย พื้นหลัง ช่องว่างสำหรับกรอกชื่อผู้เล่น และปุ่มสำหรับกดเพื่อไปหน้าถัดไป

2.12.2 หน้า SelectBunny ประกอบด้วย พื้นหลัง และกระต่าย 2 ตัว โดยผู้เล่นสามารถเลือกได้

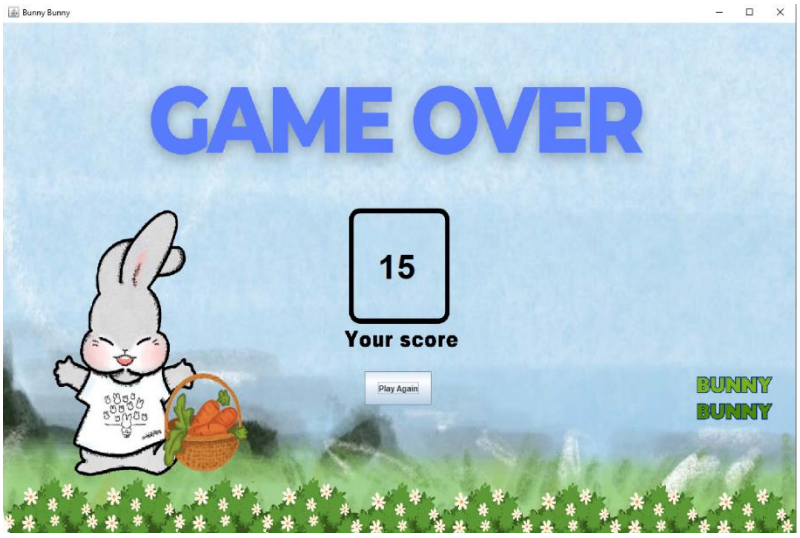
2.12.3 หน้า GamePlayPanel เป็นหน้าของเกม เมื่อกด “START” เกมจะเล่นทันที



2.12.4 กระต่ายวิ่งเก็บแครอท และหลีกเลี่ยงระเบิด มีชื่อของผู้เล่นที่กรอกไว้ในตอนแรกแสดงอยู่ เมื่อเก็บแครอทได้คะแนนจะแสดงในกรอบทางขวาบน

2.12.5 เมื่อโดนระเบิด หัวใจทางซ้ายบนจะหายไป 1 ดวง เกมจะจบลงเมื่อเสียหัวใจครบ 3 ดวง หรือหมดเวลาของเกม (75 วินาที)

2.12.6 หน้า GameEnd หรือ GameOver จะแสดงเมื่อเกมจบลง แสดงคะแนนที่ได้ และมีปุ่ม “Play Again” เมื่อกดปุ่ม “Play Again” จะแสดงหน้า SelectBunny เพื่อเลือกกระต่ายและเล่นเกมอีกครั้ง



2.13 Event handling

2.13.1 ActionListener

```
jbtStart.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String playerName = jtfInputName.getText().trim();
        if(!playerName.isEmpty()) {
            gameManager.setPlayerName(playerName);
            gameManager.showSelectBunny(); //call Page SelectBunny
        } else {
            JOptionPane.showMessageDialog(StartMenu.this, "Please enter your name.");
        }
    }
});
```

- ActionListener เพื่อรับเหตุการณ์การกดปุ่ม jbtStart

- เมื่อปุ่ม jbtStart ถูกคลิก จะตรวจสอบว่าผู้ใช้ได้กรอกชื่อหรือไม่ หากกรอกชื่อแล้วจะเรียก method ใน GameManager เพื่อเปลี่ยนไปยังหน้าเลือกตัวละคร

2.13.2 KeyListener

```
addKeyListener(new KeyAdapter() {
    @Override
    public void keyPressed(KeyEvent e) {
        positionChanged = false; // Reset flag on new key press
        if (e.getKeyChar() == 'a' || e.getKeyChar() == 'A') {
            x -= moveAmount;
            positionChanged = true;
        } else if (e.getKeyChar() == 'd' || e.getKeyChar() == 'D') {
            x += moveAmount;
            positionChanged = true; // Mark position as changed
        }
        x = Math.max(0, Math.min(x, getWidth() - 135));
        if (positionChanged) {
            //System.out.println("Bunny moved to x: " + x + ", y: " + y);
        }
        checkCarrotCollection(); //Check for carrot collection
        repaint();
    }
});
```

- KeyListener เพื่อรับเหตุการณ์การกดแป้นพิมพ์

- เมื่อผู้ใช้กดปุ่ม A หรือ D ตัวละครจะเลื่อนไปซ้ายหรือขวาตามลำดับ แล้วจึงเรียก repaint() เพื่ออัพเดตหน้าจอ

2.13.3 MouseListener

```
addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (clickCounter == 0) {
            requestFocusInWindow(); //Request focus on mouse click
            clickCounter += 1;
            startIcon = null;
            countdownTimer.start(); //Start countdown timer
            carrotThread.start();
            bombThread.start();
        }
    }
});
```

- MouseListener เพื่อรับเหตุการณ์การคลิกเมาส์ครั้งแรกเพื่อเริ่มเกม

- การคลิกจะทำให้เริ่ม countdownTimer และ Thread สำหรับ carrot และ bomb

2.14 Algorithm

2.14.1 การเก็บแครอท

```
private void checkCarrotCollection() {
    Rectangle bunnyRect = new Rectangle(x, y, 80, 180);
    Rectangle carrotRect = new Rectangle(carrotX, carrotY, 25, 25);
    if (bunnyRect.intersects(carrotRect)) {
        countCarrot += 1;
        System.out.println("Carrot collected!");
        System.out.println("Your scores : " + countCarrot);
        carrotVisible = false; // Hide carrot after collection
        setCarrotCountText(countCarrot);
        generateCarrot(); // Generate a new carrot
    }
}
```

ตรวจสอบว่ากระต่ายได้เก็บ carrot แล้วหรือยัง โดยใช้การสร้าง Rectangle สำหรับกำหนดขอบเขตของกระต่ายและ carrot ซึ่งทำให้สามารถตรวจจับการชน (collision) ระหว่างกระต่ายและ carrot ได้ เมื่อชนกัน (intersects) จะเพิ่มคะแนน (countCarrot) ซ่อน carrot ปัจจุบัน และสร้าง carrot ใหม่

2.14.2 การชนกับระเบิด

```
private void checkBombCollection() {
    Rectangle bunnyRect = new Rectangle(x, y, 80, 180);
    Rectangle bombRect = new Rectangle(bombX, bombY, 25, 25);
    if (bunnyRect.intersects(bombRect)) {
        if (lifeCount != 1) {
            lifeCount -= 1;
            bombVisible = false;
            System.out.println("life count : " + lifeCount);
        } else {
            gameManager.setCountCarrot(countCarrot);
            gameManager.showGameEnd();
            running = false; // Stop the threads
            System.out.println("Bomb collected! Game Over!");
        }
    }
}
```

ทำงานคล้ายกับการเก็บ carrot โดยเมื่อชนกับระเบิด จะลดจำนวนหัวใจ 1 ดวง และตรวจสอบว่าจำนวนหัวใจเหลือเท่าไร หากหัวใจเหลือ 0 ระบบจะเรียกให้เกมจบลงโดยบันทึกคะแนนสุดท้ายไว้

2.14.3 การนับถอยหลัง

```
countdownTimer = new Timer(1000, e -> {
    countdownTime--;
    countdownLabel.setText("Time Left: " + formatTime(countdownTime));
    if (countdownTime <= 0) {
        countdownTimer.stop();
        gameManager.setCountCarrot(countCarrot);
        gameManager.showGameEnd();
        running = false; // Stop the threads
        lifeCount -= 1;
        System.out.println("Time Out!");
    }
});
```

ใช้ Timer สำหรับการนับถอยหลังเวลาของเกม โดยจะลดค่าของ countdownTime ลงทุกๆ วินาที เมื่อ countdownTime เป็น 0 จะหยุด timer บันทึกคะแนนสุดท้าย และสิ้นสุดเกม

บทที่ 3

สรุป

3.1 ปัญหาที่พบระหว่างการพัฒนา

3.1.1 การตรวจจับการชนของแครอท ระเบิดกับกระต่ายอาจไม่แม่นยำ แก้ไขโดยการปรับระยะห่างระหว่างวัตถุให้เหมาะสมขึ้น

3.1.2 การเคลื่อนที่ของแครอทและระเบิดไม่ลื่นไหล แก้ไขโดยปรับเวลา sleep ของ Thread ให้เหมะสม

3.1.3 เมื่อแครอทและระเบิดอยู่ในตำแหน่งใกล้กัน กระต่ายอาจชนทั้งแครอทและระเบิดพร้อมกัน ซึ่งอาจทำให้เกิดข้อผิดพลาด แก้ไขโดยเพิ่มลำดับของการตรวจจับหรือหรือปรับความถี่ในการเกิดแครอทและระเบิดให้เหมาะสมไม่ทับซ้อนกัน

3.2 จุดเด่นของโปรแกรม

3.2.1 การออกแบบ UI ที่ดึงดูดสายตา ใช้ภาพพื้นหลังที่เป็นเอกลักษณ์ของแต่ละหน้าจอ รวมถึงตัวละครกระต่าย แครอท ระเบิดที่น่ารักมีเรื่องราว ช่วยเพิ่มความสนุกและทำให้ผู้เล่นเพลิดเพลินกับการเล่น

3.2.2 การแสดงคะแนนและชื่อผู้เล่นแบบเรียลไทม์ ผู้เล่นสามารถเห็นความคืบหน้าในการเล่นได้ตลอดเวลา และกระตุ้นให้ผู้เล่นพยายามทำคะแนนให้สูงขึ้น

3.2.3 เล่นเกมที่เข้าใจง่ายและเหมาะสำหรับทุกเพศทุกวัย ไม่ซับซ้อน

3.3 คำแนะนำ

-