โครงงาน

Monster Hunter

จัดทำโดย

6604062636241 ธนโชติ สุนทรกำจรพานิช

เสนอ

ผู้ช่วยศาสตราจารย์ สถิตย์ ประสมพันธ์

รายงานนี้เป็นส่วนหนึ่งของวิชา 040613204 Object-Oriented Programming

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ภาคเรียนที่ 1 ปีการศึกษา 2567

บทที่ 1 บทนำ

ที่มาและความสำคัญ

โครงงานนี้จัดขึ้นเพื่อวัดความรู้ในการเรียนวิชา Object Oriented Programming โดยการนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบของเกมโดยใช้แนวคิดการเขียนโปรแกรมแบบเชิงวัตถุ

ประเภทของโครงการ

เกม 2 มิติ

ประโยชน์

1.ฝึกความแม่นยำ

2.ฝึกไหวพริบ

3.นำเนื้อหาที่เรียนมาประยุกต์ใช้

ขอบเขตของโครงการ

ลำดับ	รายการ	18-20 ก.ย.	21-23 ก.ย.	24-30 ก.ย.	1-8 ต.ค
1	หารูปตัวละครและทำกราฟิกต่างๆ				
2	ศึกษาเอกสารและข้อมูลที่เกี่ยวข้อง				
3	ลงมือเขียนโปรแกรม				
4	จัดทำเอกสาร				
5	ตรวจสอบและแก้ไขข้อผิดพลาด				

บทที่ 2 ส่วนการพัฒนา

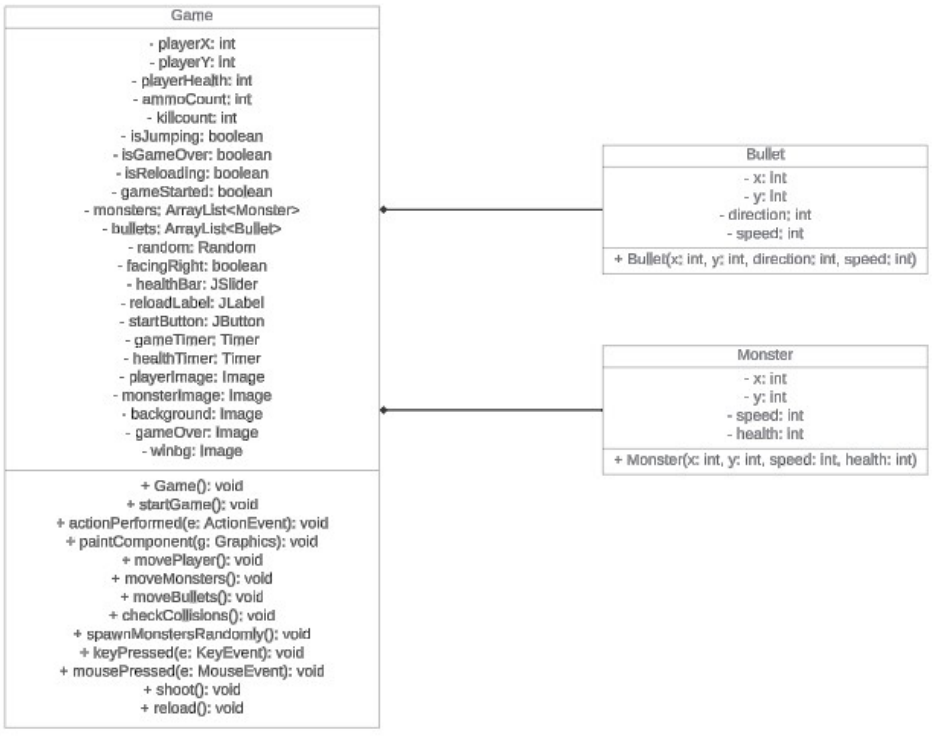
เนื้อเรื่องย่อ

ผู้เล่นสวมบทบาทเป็นตัวละครผู้ที่มีความสามารถและได้รับเลือกให้ปกป้องมนุษยชาติและต้องเผชิญหน้ากับมอนสเตอร์โดยการผจญภัยไปในเมืองที่ถูกทำลาย

วิธีการเล่น

ใช้ปุ่ม A,W,D เพื่อใช้ในทำให้ตัวละครเดินไป ซ้าย กระโดด และ ขวา ตามลำดับ ใช้ปุ่มคลิกซ้ายของเมาส์เพื่อทำการยิงปืนและปุ่ม R เพื่อ Reload กระสุนปืน โดยต้องยิงมอนเตอร์ให้หมดจึงจะชนะเกม

คลาสไดอะแกรม



รูปแบบการพัฒนา

-ภาษาที่ใช้ Java พัฒนาแบบ Java Application

Constructor

Constructor ของคลาสนี้จะทำการกำหนดค่าพื้นฐานของสถานะ

เกม

Liste

```
public Game() {
    playerImage = new ImageIcon(getClass().getResource("MainChar.png")).getImage();
    monsterImage = new ImageIcon(getClass().getResource("Monster.png")).getImage();
    background = new ImageIcon(getClass().getResource("BG.jpg")).getImage();
    gameOver = new ImageIcon(getClass().getResource("gameOver.png")).getImage();
    winbg = new ImageIcon(getClass().getResource("win.png")).getImage();
    setFocusable(true);
    setPreferredSize(new Dimension(800, 600));
    setLayout(null);
    addKeyListener(this);
    addMouseListener(this);

    healthBar=new JSlider(0,200,playerHealth);
    healthBar.setPreferredSize(new Dimension(200,50));
    healthBar.setBorder(BorderFactory.createLineBorder(Color.black));
    healthBar.setBounds(10, 10, 200, 30);
    healthBar.setForeground(Color.RED);
    healthBar.setVisible(false);
    healthBar.setEnabled(false);
    add(healthBar);

    reloadLabel=new JLabel("Reloading",JLabel.CENTER);
    reloadLabel.setFont(new Font(null,Font.PLAIN,30));
    reloadLabel.setForeground(Color.white);
    reloadLabel.setVisible(false);
    reloadLabel.setBounds(0,0,1000,1000);
    add(reloadLabel);

    startButton = new JButton("Start Game");
    startButton.setBounds(350, 300, 100, 50);
    startButton.addActionListener(e->startGame());
    add(startButton);

    gameTimer=new Timer(30,this);
    healthTimer=new Timer(500,this);

    spawnMonstersRandomly();
}
```

-กำหนด Image ต่าง ๆในเกม

-ตั้ง Focusable และ preferred size

-เพิ่ม KeyListener MouseListenter สำหรับคีย์บอร์ดและเมาส์

-สร้าง healthBar และ reloadLabel ทำการกำหนดค่าต่าง ๆ

-สร้าง startButton เป็นปุ่มเริ่มเกม

-เริ่ม gameTimer สำหรับ Timer หลักของเกมและ healthTimer สำหรับ Timer ของเลือดผู้ เล่น

-เรียกใช้ spawnMonstersRandomly() เพื่อสุ่มจุดเกิดและสร้างมอนสเตอร์ครั้งแรกในเกม

Encapsulation

Encapsulation ทำให้ข้อมูลอยู่ภายในคลาสและเข้าถึงได้ผ่านเมธอด โดยใช้ตัวแปร private เพื่อไม่ให้เข้าถึงได้จากภายนอก

```
public class Game extends JPanel implements ActionListener,KeyListener,MouseListener {
    private int playerX=200,playerY=520,playerHealth=200,ammoCount=20,killcount=0;
    private boolean isJumping=false,isGameOver=false,isReloading=false,gameStarted=false;
    private ArrayList<Monster>monsters=new ArrayList<>();
    private ArrayList<Bullet>bullets=new ArrayList<>();
    private Random random=new Random();
    private boolean facingRight=true;
    private JSlider healthBar;
    private JLabel reloadLabel;
    private JButton startButton;
    private Timer gameTimer;
    private Timer healthTimer;
    private Image playerImage;
    private Image monsterImage;
    private Image background;
    private Image gameOver;
    private Image winbg;
```

-Data Field ต่าง ๆ playerX, playerY, playerHealth, ammoCount ที่เป็น private

```
private void movePlayer() {
    if (isJumping) {
        playerY-=10;
        if (playerY<=450) {
            isJumping=false;
        }
    } else if (playerY<520) {
        playerY+=10;
    }
}

private void moveMonsters() {
    for (Monster monster:monsters) {
        if (monster.x<playerX) {
            monster.x+=monster.speed;
        } else {
            monster.x-=monster.speed;
        }

        if (Math.abs(monster.x-playerX)<30&&Math.abs(monster.y-playerY)<30) {
            playerHealth-=1;
            if (playerHealth<=0) {
                isGameOver=true;
            }
        }
    }
}
```

```
private void moveBullets() {
    for (int i=0;i<bullets.size();i++) {
        Bullet bullet=bullets.get(i);
        if (bullet.direction==-1) {
            bullet.x-=20;
        }
        bullet.x+=bullet.direction*bullet.speed;
        if (bullet.x<playerX-300||bullet.x>playerX+300) {
            bullets.remove(i);
            i--;
        }
    }
}

private void checkCollisions() {
    for (int i=0;i<monsters.size();i++) {
        Monster monster=monsters.get(i);
        for (int j=0;j<bullets.size();j++) {
            Bullet bullet=bullets.get(j);
            if (bullet.x>=monster.x-20&&bullet.y>=monster.y-20) {
                monster.health-=10;
                bullets.remove(j);
                if (monster.health<=0) {
                    killcount++;
                    monsters.remove(monster);
                    break;
                }
            }
        }
    }
    if (monsters.isEmpty()) {
        if(killcount>=10) {
            isGameOver=true;
        }
        spawnMonstersRandomly();
    }
}
```

```
private void spawnMonstersRandomly() {
    int count=random.nextInt(3)+1;
    for (int i=0;i<count;i++) {
        int x=Math.max(getWidth()==0?1000:getWidth(),getWidth()+(random.nextInt(100)+random.nextInt(100)));
        int y=520;
        monsters.add(new Monster(x,y,2,100));
    }
}
```

```
private void shoot() {
    if (ammoCount>0) {
        int direction=facingRight ? 1:-1;
        bullets.add(new Bullet(playerX+25,playerY+25,direction,40));
        ammoCount--;
    } else {
        reloadLabel.setVisible(true);
        isReloading=true;
        Thread thread=new Thread(()->{
            ammoCount=20;
            reloadLabel.setVisible(false);
            isReloading=false;
        });
        thread.start();
    }
}

private void reload() {
    if (ammoCount<20) {
        reloadLabel.setVisible(true);
        isReloading=true;
        Timer reloadTimer=new Timer(2000,e->{
            reloadLabel.setLocation(playerX-520,playerY-520);
            ammoCount=20;
            reloadLabel.setVisible(false);
            isReloading=false;
        });
        reloadTimer.setRepeats(false);
        reloadTimer.start();
    }
}
```

-การเข้าถึงและการเปลี่ยนค่าของตัวแปรที่ทำผ่านเมธอด เช่น

movePlayer() moveMonsters()

moveBullets()checkCollisions()

spawnMonstersRandomly() shoot() reload()

Composition

Composition มีให้เห็นโดย Object ของคลาสอื่น ๆ เป็น Field ของคลาสของเกมนี้

```
public class Game extends JPanel implements ActionListener,KeyListener,MouseListener {
    private int playerX=200,playerY=520,playerHealth=200,ammoCount=20,killcount=0;
    private boolean isJumping=false,isGameOver=false,isReloading=false,gameStarted=false;
    private ArrayList<Monster>monsters=new ArrayList<>();
    private ArrayList<Bullet>bullets=new ArrayList<>();
    private Random random=new Random();
    private boolean facingRight=true;
    private JSlider healthBar;
    private JLabel reloadLabel;
```

-ArrayList<Monster> และ ArrayList<Bullet> ใช้สำหรับการเก็บมอนสเตอร์และกระสุน

-JSlider (healthBar) และ JLabel (reloadLabel) เป็นส่วนประกอบของ GUI เมื่อ JFrame ถูกปิดจะทำให้ JSlider (healthBar) และ JLabel (reloadLabel) หายไปด้วย

Polymorphism

โค้ดในเกมนี้ไม่ได้มีการใช้งาน Polymorphism

Abstract

โค้ดใน

เมธอด

Inherit

คลาสนี้

```
public cla
```

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    startButton.setLocation(getWidth()/2-50,getHeight()/2);
    if (playerY<getHeight()/2+100&&check) {
        g.drawImage(background,0,0,getWidth(),playerY+100,this);
        temp=playerY+100;
        check=false;
    } else {
        if(check) {
            temp=getHeight();
        }
        g.drawImage(background,0,0,getWidth(),temp,this);
    }
    if(playerX>getWidth()-20) {
        playerX=getWidth()-20;
    }
    if (!gameStarted) {
        g.setColor(Color.white);
        g.setFont(new Font(null,Font.PLAIN,30));
        g.drawString("Monster Hunter",getWidth()/2-100,getHeight()/2-100);
        return;
    }
    if (facingRight){
        g.drawImage(playerImage,playerX,playerY,50,50,this);
    } else {
        g.drawImage(playerImage,playerX+20,playerY,-50,50,this);
    }

    for (Monster monster:monsters) {
        g.drawImage(monsterImage,monster.x,monster.y,40,50,this);
    }

    g.setColor(Color.BLACK);
```

-การ Inheritance จาก JPanel ของในคลาสนี้และทำการ override เมธอด paintComponent(Graphics g) เพื่อทำการวาด

ส่วนประกอบของ GUI

GUI ประกอบด้วยส่วนประกอบดังนี้

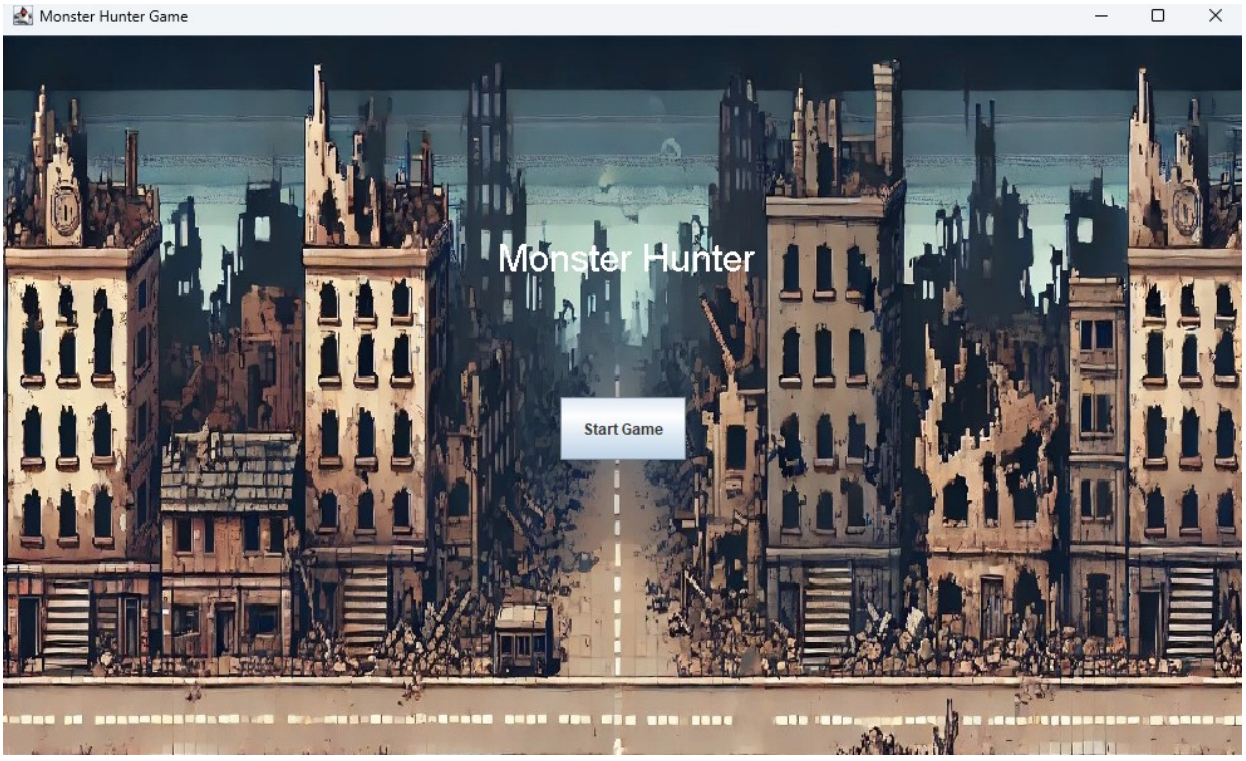
-JPanel เป็น panel ที่กำหนดเองของเกมนี้

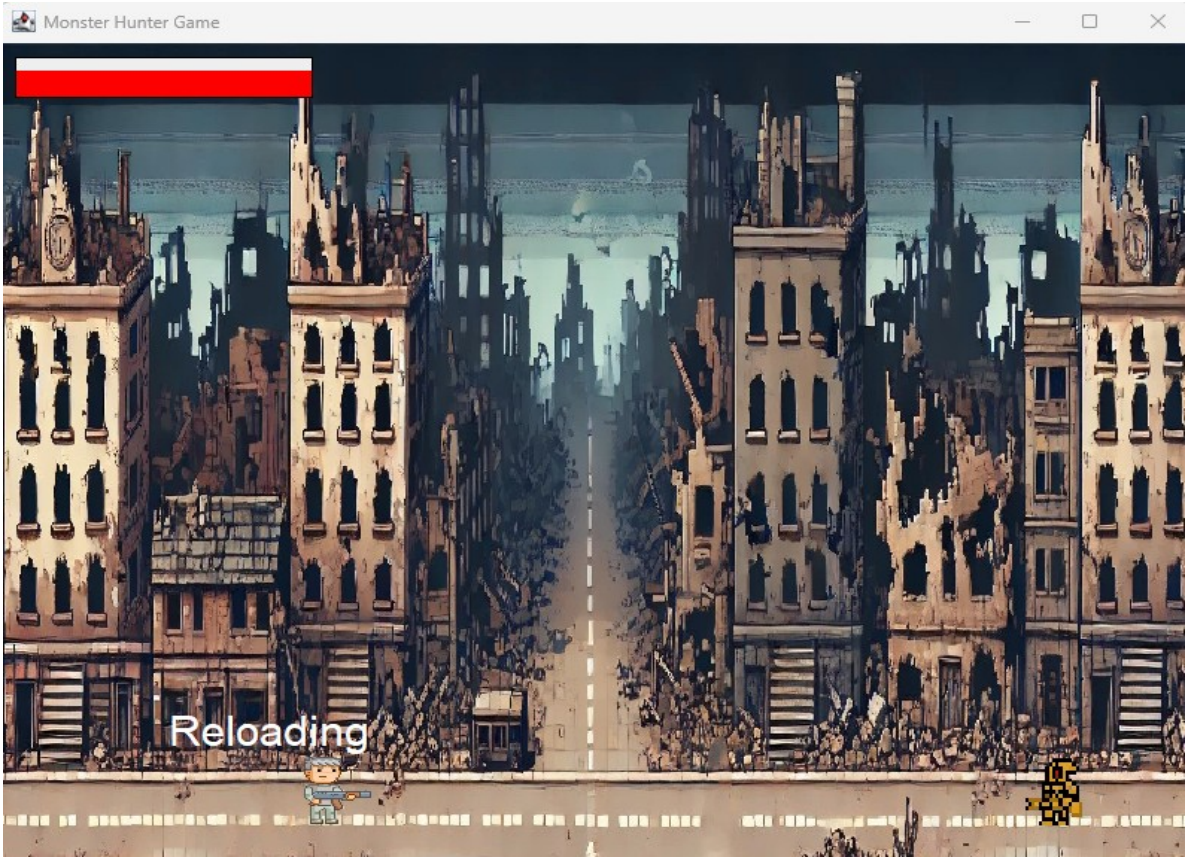
-JFrame เป็น container สำหรับเกมนี้

-JSlider (healthBar) แสดงสถานะเลือดปัจจุบันของผู้เล่น

-JLabel (reloadLabel) แสดงข้อความ “Reload” เมื่อมีการกดปุ่มรีโหลด

-JButton (Start Game) เป็นปุ่มไว้ทำการเริ่มเกม




Monster Hunter Game
Reloading

Event Handling

จัดการกับเหตุการณ์ถูกใช้ผ่าน Listener หลาย Listener ดังนี้

-KeyListener: จัดการปุ่มบนคีย์บอร์ดเพื่อให้ผู้เล่นเคลื่อนที่ปุ่ม W, A, D และ Reload R

```
@Override
public void keyPressed(KeyEvent e) {
    if (gameStarted) {
        if (e.getKeyCode()==KeyEvent.VK_W&&!isJumping) {
            isJumping=true;
        } else if (e.getKeyCode()==KeyEvent.VK_D) {
            playerX+=10;
            facingRight=true;
        } else if (e.getKeyCode()==KeyEvent.VK_A) {
            playerX-=10;
            facingRight=false;
            if (playerX<=0) {
                playerX=0;
            }
        } else if (e.getKeyCode()==KeyEvent.VK_R) {
            reload();
        }
    }
}
```

-MouseListener: ควบคุมการคลิกเมาส์เพื่อยิงกระสุนปืน

```
@Override
public void mousePressed(MouseEvent e) {
    if (SwingUtilities.isLeftMouseButton(e)&&!isReloading&&gameStarted) {
        shoot();
    }
}
```

-ActionListener: ใช้เพื่อดักจับว่าตอนนี้อยู่ที่หน้าก่อนเริ่มเกมไหมถ้าไม่จะทำการเริ่มเกมเลยและเป็นเมธอดที่จัดการเกมทั้งหมด

```
@Override
public void actionPerformed(ActionEvent e) {
    if (gameStarted&&!isGameOver) {
        movePlayer();
        moveMonsters();
        moveBullets();
        checkCollisions();
        repaint();
    }
}
```

อัลกอริธึมที่สำคัญ

-การเคลื่อนที่ของผู้เล่น:

```
private void movePlayer() {
    if (isJumping) {
        playerY-=10;
        if (playerY<=450) {
            isJumping=false;
        }
    } else if (playerY<520) {
        playerY+=10;
    }
}
```

-เมธอด movePlayer() จะจัดการการกระโดดตามสถานะ isJumping

-การเคลื่อนที่และการชนของมอนสเตอร์:

```
private void moveMonsters() {
    for (Monster monster:monsters) {
        if (monster.x<playerX) {
            monster.x+=monster.speed;
        } else {
            monster.x-=monster.speed;
        }

        if (Math.abs(monster.x-playerX)<30&&Math.abs(monster.y-playerY)<30) {
            playerHealth-=1;
            if (playerHealth<=0) {
                isGameOver=true;
            }
        }
    }
}
```

-เมธอด moveMonsters() จะเคลื่อนที่มอนสเตอร์ไปหาผู้เล่นและตรวจสอบว่าถ้ามอนสเตอร์เข้าใกล้ผู้เล่นน้อยกว่า 30 พิกเซลจะทำการลดเลือดผู้เล่นทีละ 1 และเมื่อเลือดผู้เล่นหมดเกมจะแพ้ทันที

-การเคลื่อนที่ของกระสุนและการชน:

```
private void moveBullets() {
    for (int i=0;i<bullets.size();i++) {
        Bullet bullet=bullets.get(i);
        if (bullet.direction==-1) {
            bullet.x-=20;
        }
        bullet.x+=bullet.direction*bullet.speed;
        if (bullet.x<playerX-300||bullet.x>playerX+300) {
            bullets.remove(i);
            i--;
        }
    }
}
```

-เมธอด moveBullets() จะทำการขยับกระสุนและลบกระสุนหลังห่างจากผู้เล่นไป 300 พิกเซล

```
private void checkCollisions() {
    for (int i=0;i<monsters.size();i++) {
        Monster monster=monsters.get(i);
        for (int j=0;j<bullets.size();j++) {
            Bullet bullet=bullets.get(j);
            if (bullet.x>=monster.x-20&&bullet.y>=monster.y-20) {
                monster.health-=10;
                bullets.remove(j);
                if (monster.health<=0) {
                    killcount++;
                    monsters.remove(monster);
                    break;
                }
            }
        }
    }
    if (monsters.isEmpty()) {
        if(killcount>=10) {
            isGameOver=true;
        }
        spawnMonstersRandomly();
    }
}
```

-เมธอด checkCollisions() จะตรวจสอบว่ากระสุนถูกมอนสเตอร์หรือไม่ โดยถ้ากระสุนโดนจะลดเลือดมอนสเตอร์ทีละ 10 และทำการลบมอนสเตอร์ออกหากเลือดหมด

-การสร้างมอนสเตอร์:

```
private void spawnMonstersRandomly() {
    int count=random.nextInt(3)+1;
    for (int i=0;i<count;i++) {
        int x=Math.max(getWidth()==0?1000:getWidth(),getWidth()+(random.nextInt(100)+random.nextInt(100)));
        int y=520;
        monsters.add(new Monster(x,y,2,100));
    }
}
```

-เมธอด spawnMonstersRandomly() จะสร้างมอนสเตอร์ในตำแหน่งสุ่มด้านนอกจอ เพื่อเพิ่มความท้าทายในเกมและในเมธอดยังมีการสุ่มจำนวนการเกิดของมอนเตอร์ในแต่ละครั้งด้วย

บทที่ 3 สรุป

ปัญหาที่พบระหว่างการพัฒนา

-กระสุนยิงไปแล้วเลือดมอนสเตอร์ไม่ลด

-เมื่อถูกมอนสเตอร์ตีจะไม่สามารถตีมอนสเตอร์กลับได้

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

-การเกิดของมอนสเตอร์แบบสุ่มและจำนวนการเกิดแบบสุ่ม

-มีพื้นหลังที่สมจริง

คำแนะนำสำหรับผู้สอนที่อยากให้อธิบาย

-