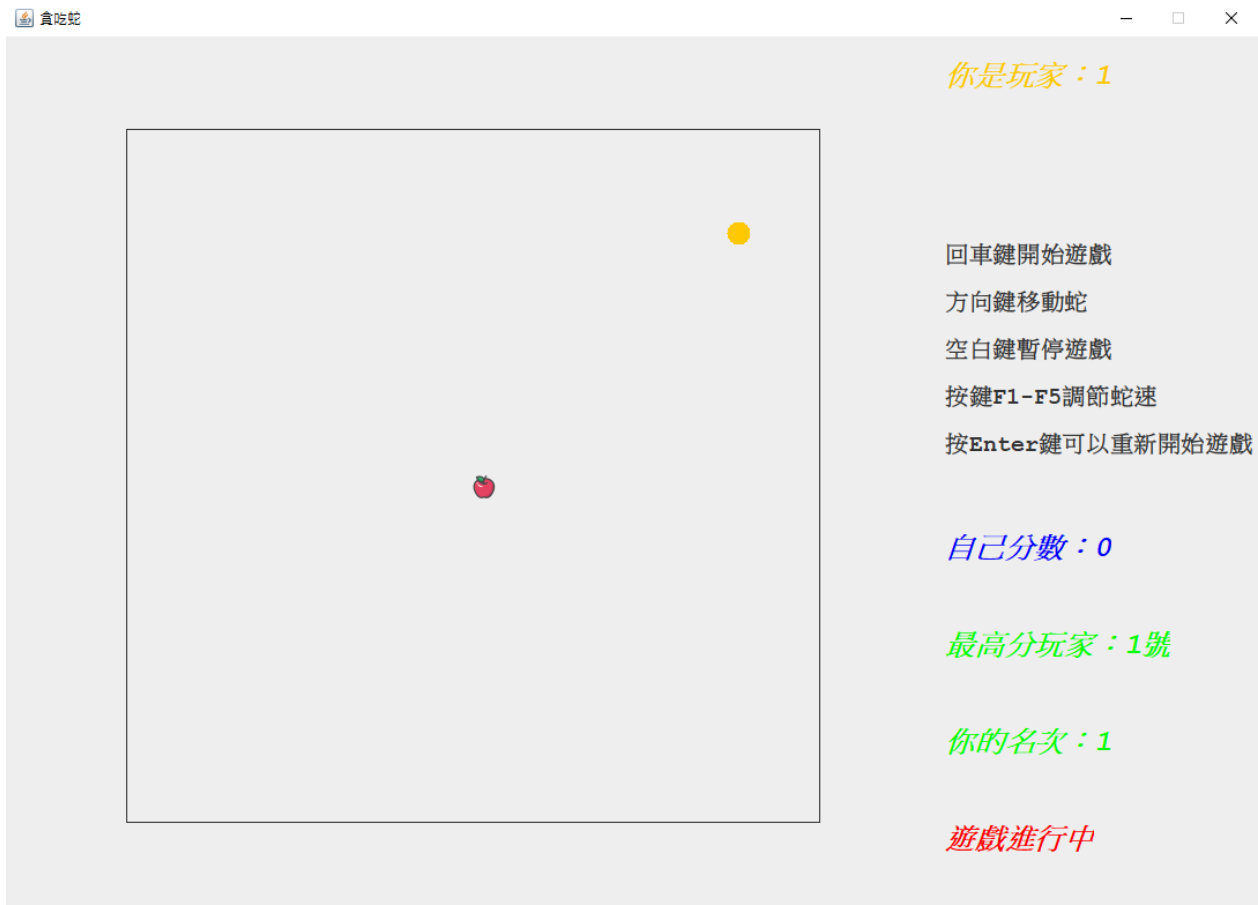


(一)遊戲規則與介紹(Client 端)

本次期末專題實作以 JAVA 單機版貪食蛇並配合課程內容與教學，將實作出多人連線版貪食蛇遊戲。

● 遊戲規則：

1. 遊戲介面左方遊玩區域，即玩家(貪食蛇)與目標物(蘋果)移動畫面；右方為遊戲資訊欄位，由上至下分別顯示，玩家編號、遊戲操作規則、我方玩家得分數、遊戲中最高分玩家、我方玩家在遊戲中當前名次以及遊戲狀態。(如圖一)



▲ 圖一、貪食蛇遊戲主介面

2. 遊玩區域介紹：

玩家進入伺服器時，玩家(貪食蛇本體-頭)將會隨機生成地點，範圍侷限在遊玩區域即方框內部；而目標(蘋果)初始將生成於中心位置，也就是當多人連線時若掌握遊戲技巧，將可以率先得分。

我方玩家的貪食蛇頭部顏色為黃橘色，而若有其他玩家進入，其他玩家在我方畫面上將會呈現其他顏色(隨機顏色)，以增加遊戲趣味性並易於分辨我方與對方玩家，如圖二(玩家1)與圖三(玩家3)。



▲ 圖二、玩家 1 的畫面



▲ 圖三、玩家 3 的畫面

- 2-1. 若玩家目標為吃到目標(蘋果)，並且可獲的分數，若玩家吃蘋果，蘋果將隨機生成位置，範圍一樣於方框中，吃取蘋果的玩家，蛇身將會增加，並所有玩家蛇身顏色均玩亮粉色，如圖四。



▲ 圖四、吃取蘋果後蛇身增加

1. 玩家可與其他玩家蛇身碰撞，但不可與自身碰撞，加上因所有玩家蛇身顏色相同，因此玩家須辨別是否為我方蛇身以免發生碰撞，具有一定的遊戲難度。而若與自身蛇身碰撞則將會結束遊戲，並於畫面上顯示 Game Over 字眼，遊戲右下方狀態資訊將會同步更新文字，顯示”你吃到自己了!”，圖五。



▲ 圖五、吃取蘋果後蛇身增加

2. 若玩家與牆壁發生碰撞，遊戲將會結束，並如上述第四點結果相似，於畫面上顯示 Game Over 字眼，而遊戲右下方狀態資訊將會同步更新文字，顯示”你穿越牆壁了!”，如圖六。



▲ 圖六、吃取蘋果後蛇身增加

3. 遊戲操作介紹：

玩家可利用鍵盤上的方向鍵(↑↓←→)進行貪食蛇的控制，對應方向如方向鍵方向，而本次遊戲設計上設有方向防護機制，即若貪食蛇當前方向為向上前進時，玩家按下方方向鍵下，貪食蛇則會忽略操作，反之如此，貪食蛇向左或右方向前進時同樣限制。簡單來說，玩家無法操作與貪食蛇相反方向的按鍵，若有相關操作，遊戲將忽略此次的無效控制。

1. 目標同時多玩家吃取問題：因考慮若有 2 位以上玩家同時間吃到蘋果時，誰將會得方，因此本次遊戲設計將設有速度調整模式(待會將詳細說明)，也就是當吃取蘋果的玩家速度較快則會優先獲取分數(其原理在於速度快的玩家於伺服器上有較快的更新速度，因此客戶端玩家將會較快取的位置的更新廣播，而判斷是否吃取蘋果為客戶端判斷，則將會先獲取吃取蘋果的條件，並向伺服器通知)；若當速度相同時，則為先入伺服器的玩家優先獲取分數。

藉由此特色，玩家可以調整速度，以吃取蘋果，有更豐富的遊玩方式，增加遊戲可玩性。

4. 遊戲資訊欄位：


此處提供玩家操作方式提示，而顯示相關遊戲數據。(由上至下說明)

- i. 玩家編號由伺服器分配，在玩家畫面上同步顯示，並利用與自己方的貪食蛇頭部相同的方式顯示顏色(黃橘色)。
- ii. 提供遊戲操作資訊
- iii. 自己分數：為目前遊戲進行時的得分，若玩家重新開始或進入遊戲，將會刷新分數，由 0 分開始計算
- iv. 最高分玩家：伺服器將會統計當前所有玩家目前遊玩得分數，並定時廣播給客戶端做更新，分數統計都就由伺服器端坐累加，因此可以避免遊戲相關安全性問題
- v. 你的名次：此欄位可以目前玩家在遊戲中的排名，讓玩家有想遊玩獲取高分

的衝動

- vi. 遊戲狀態說明欄位，會有”遊戲進行中”、”你撞穿牆壁了！”以及”你吃到自己了！”的文字提示

5. 遊戲特色功能：

- i. 玩家初次進入遊戲時，按下 Enter 鍵可開始進行遊戲
- ii. 遊戲進行時，玩家可使用 F1 至 F5 按鍵進行貪食蛇移動的速度的調整(共佑五種速度，F1 最慢、F5 最快)，並且所有玩家畫面亦同步更新，也就是說，都有人加速時，每個人的遊戲畫面可以看到有玩家加速
- iii. 若中途想暫停遊戲時，可以使用空白鍵進行暫停(蛇身會慢慢縮成一個點只剩頭)；若像繼續遊玩，則再次按下空白鍵，即可繼續遊玩(蛇身會慢慢變長到暫停前的長度)。
- iv. 若玩家 Game Over 時，可以再次按下，Enter 鍵重新開始遊戲(分數會歸零)
- v. 本次遊戲製作可以多人同時連線(可無上限玩家人數，伺服器效能好時)
- vi. 目標物使用蘋果圖示 

(二) 伺服器功能說明(Server 端)

- 設計特色：

- i. 利用 Executors.newCachedThreadPool()的方式進行伺服器的設計，能提供多人連線，並有效保持伺服器效能。

普通執行續使用.start()產生 Thread 時仍會有 overhead 的成本，因此若要處理的 task 越大量(此次為有多位玩家進入遊戲)那產出的 overhead 是很可觀的，又沒有善加管理這些 Thread 的話，就會拖累整體系統的效能。

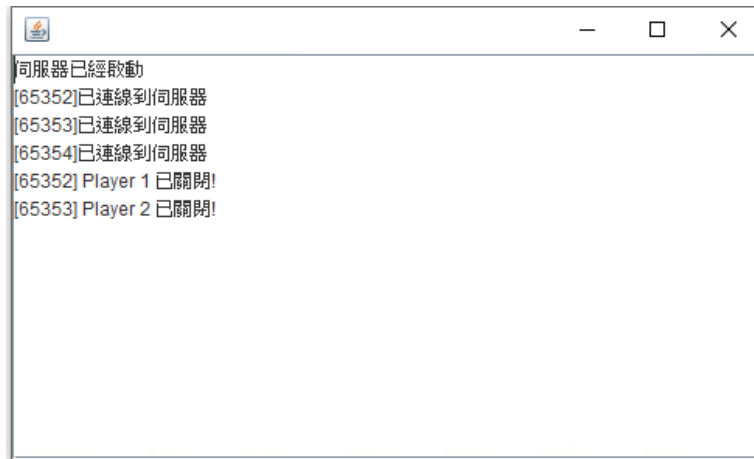
而 newCachedThreadPool 會建立一個可快取執行緒池，如果執行緒池長度超過處理需要，可靈活回收空閒執行緒，若無可回收，則新建執行緒，因此此次遊戲伺服器製作使用 newCachedThreadPool 並希望能達到多人(無上限)連線遊玩完的目標。

- ii. 使用 JAVA Socket 的 IO 方式進行控制信令的廣播，如發送位置資訊、成績資訊的，以字串的方式進行傳送。
- iii. 玩家出生點隨機分配，當玩家進入遊戲時將會隨機分配玩家位置，且若玩家重新開始遊戲時也將隨機分配位置，提高遊戲的挑戰程度。
- iv. 目標蘋果位置生成，如玩家出生點生成相同，採用隨機方式生成。
- v. 玩家人數的計數
- vi. 玩家貪食蛇位置更新廣播，利用伺服器回傳個玩家位置資訊，並同時廣播給所有玩家，讓每個 Client 端能依此資訊更新遊戲畫面，利用伺服器控制玩家貪食蛇位置，較符合多人連線遊戲的製作理念，可防止玩家惡意的篡改。
- vii. 伺服器更新玩家位置，是利用玩家回傳的方向，並於伺服器上進行座標的運算，再將結果發送給客戶端。
- viii. 本次伺服器端設計加入分數的通統計與排序利用 LinkedHashMap 給予每玩家分數計算的空間，採用 LinkedHashMap 目的為，能以字典方式儲存資料，一個鍵 key 對應一個值 value，在這對應玩家與該玩遊戲分數，而 LinkedHashMap 因為屬於

LinkedList 類型，具有有序的特性(普通 HashMap 是無序的)。因此藉由 LinkedHashMap 將這串資料進行在伺服器上排序，就可得出誰為分數最高玩家與每位玩家排名。

- 伺服器端 GUI 介面

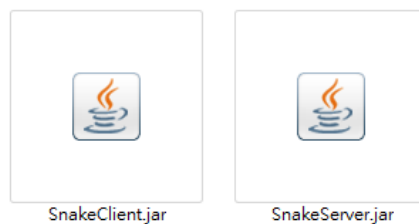
本次伺服器端將設有 GUI 提示介面，顯示玩家進入狀況並與離開狀況已作好伺服器管理，如圖七。



▲ 圖七、吃取蘋果後蛇身增加

(三) 遊戲打包

最後為了方便分享將遊戲打包成.jar 檔可供具有 JAVA 功能的電腦使用，如圖八。



▲ 圖八、吃取蘋果後蛇身增加

(四) 關鍵程式碼說明

- Client 端：

Main 類：

建立主函式，對進行初始化相關設定，如：建立版面與佈局、建立遊戲角色(貪食蛇)、開啟執行緒(Client 端亦使用 newCachedThreadPool 建立執行緒)、Socket 連接與初始化、Server 信令監聽器啟動等。

```

public class Main extends JFrame{
    /**
     * Create by ShengPo
     */
    private static final long serialVersionUID = 1L;
    public static GamePanel p1 ; //創建遊戲面板
    public static InformationPanel p2 ; //創建資訊面板
    public static ServerListener p3;
    Socket socket;
    private ExecutorService exec= null;
    public static Snake snakeSelf;

    public Main(){ //配置框架的佈局
        setLayout(new BorderLayout());

        try {
            socket = new Socket("127.0.0.1", 54321);
            p1 = new GamePanel(socket);
            p2 = new InformationPanel(socket);
            p3 = new ServerListener(socket);
            snakeSelf = new Snake();
            add(p1,BorderLayout.CENTER);
            add(p2,BorderLayout.EAST);
            exec = Executors.newCachedThreadPool();

            exec.execute(p1);
            // Thread.sleep(1000);
            exec.execute(p2);
            // Thread.sleep(1000);
            exec.execute(p3); //要發送初始位置 所以要先開始 蛇頭初始都一樣
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        JFrame frame = new Main(); //新建框架

        //配置框架
        frame.setTitle("貪吃蛇");
        frame.setSize(1100, 800);
        frame.setVisible(true);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);

    }
}

```

▲ 圖九、Main 類 程式碼片段

GamePanel 類：

負責遊戲區域的繪製、畫面更新、目標蘋果的渲染以及貪食蛇的繪製與相應遊戲狀態，呼叫後將會監聽玩家鍵盤方向鍵控制的狀態，並發送相應方向以文字方式傳送給伺服器端。

✎ 本次改動：

1. 鍵盤監聽器的信令發送

```
//註冊鍵盤監聽器
addKeyListener(new KeyAdapter(){
    public void keyPressed(KeyEvent e){
        int direction = Main.snakeSelf.getDirection();
        switch(e.getKeyCode()){
            case KeyEvent.VK_UP:
                if(!isStarted && !isPaused && !isCrushed()){ //優化isCrushed() 不要遍歷全部蛇
                    if(direction != Snake.DIRECTION_UP && direction != Snake.DIRECTION_DOWN){
                        Main.snakeSelf.setDirection(Snake.DIRECTION_UP);
                        // changeSnakeLocation();
                        try {
                            PrintWriter pw = new PrintWriter(Gsocket.getOutputStream(), true);
                            pw.println("Direction:UP");
                        } catch (IOException e1) {
                            e1.printStackTrace();
                        }
                    }
                }
            }
        }
    }
});
break;
```

▲ 圖十、鍵盤監聽器 程式碼片段

2. 目標採用蘋果圖片進行遊戲美化，利用 BufferedImage 載入圖檔

```
URL resource = getClass().getResource("apple.png");
try {
    image = ImageIO.read(resource);
    image = resize(image, 20, 20);
} catch (IOException e) {
    e.printStackTrace();
}
```

▲ 圖十一、圖片載入器實作

再利用撰寫 resize 函式將圖片重新縮放

```
private BufferedImage image;

public static BufferedImage resize(BufferedImage img, int newW, int newH) {
    Image tmp = img.getScaledInstance(newW, newH, Image.SCALE_SMOOTH);
    BufferedImage dimg = new BufferedImage(newW, newH, BufferedImage.TYPE_INT_ARGB);

    Graphics2D g2d = dimg.createGraphics();
    g2d.drawImage(tmp, 0, 0, null);
    g2d.dispose();

    return dimg;
}
```

▲ 圖十二、圖片 resize 函式

最後在繪製目標(蘋果)時，將圖片繪製在對應座標上

```
g.drawImage(image, dessert.getX(), dessert.getY(), this);
```

▲ 圖十三、繪製目標(蘋果)

2. 對方玩家貪食蛇隨機顏色生成

```
Random rand = new Random();
float r = rand.nextFloat();
float g = rand.nextFloat();
float b = rand.nextFloat();
Color randomColor = new Color(r, g, b);
private BufferedImage image;
```

▲ 圖十四、顏色隨機程式碼片段

3. 多位玩家的蛇頭繪製方式

遍利多位玩家，若為我方時給予預設顏色，若為對方給予隨機顏色。

蛇身部分則給予相同顏色(亮粉色)，並且可得知每位玩家目前蛇身長度的。

```
Snake snakeValue;
for (String snakeKey : snakes.getSnakes().keySet()) {
    snakeValue = snakes.getSnakes().get(snakeKey);

    //繪畫蛇身
    g.setColor(Color.MAGENTA);
    for(int i = 0; i < snakeValue.getBody().size(); i++){
        g.fillOval(snakeValue.getBody().get(i).getX(), snakeValue.getBody().get(i).getY(), PER_UNIT_LENGTH, PER_UNIT_LENGTH);
    }

    if (snakeKey.equals(ServerListener.Self_player)){
        g.setColor(Color.ORANGE);
        g.fillOval(snakeValue.getHead().getX(), snakeValue.getHead().getY(), PER_UNIT_LENGTH, PER_UNIT_LENGTH);
    }else{
        //繪畫蛇頭 //自動隨機玩家顏色 自己顏色固定 廣播??
        g.setColor(randomColor);
        g.fillOval(snakeValue.getHead().getX(), snakeValue.getHead().getY(), PER_UNIT_LENGTH, PER_UNIT_LENGTH);
    }
}
}
```

▲ 圖十五、蛇體繪製程式碼片段

4. 判斷是否自行碰撞(此部分較困難)

需要了解多位玩家當前更新貪食蛇座標位置時會呼叫 changeSnakeLocation() 函式進行遊戲畫面的更新(圖十六)，但只要有變動呼叫此函式，若沒有給予限制條件，則會使碰撞規則意外觸發導致錯誤的邏輯。因此首先得知是否為蘋果的座標更新，若否則以 FOR 迴圈遍利不同玩家的位置，在接續判斷是否為自己蛇體本身，若為是，則在判斷是否發生自撞狀態。而後根據改變的狀態(自撞狀態)，最相對應的畫面更新。

changeSnakeLocation() 函式使用 synchronized 進行同步處理，否則若無等待繪製完成，會產稱繪製衝突，畫面更新不完全。

```
//重新渲染畫面
repaint();
requestFocus();
```

▲ 圖十六、重新渲染畫面函式

```
public synchronized void changeSnakeLocation() { //在更新的位置重新渲染畫面
    PrintWriter pw;
    //int show;int yNow;
    Snake snakeValue;
    if (eggChange == false){
        for (String snakeKey : snakes.getSnakes().keySet()) {
            snakeValue = snakes.getSnakes().get(snakeKey);

            if (isEncountered(snakeValue)){
                //score++; //自Server端
                snakeValue.getBody().addFirst(new Dot(snakeValue.getHead_Pre_X(), snakeValue.getHead_Pre_Y()));
                snakeValue.getBody().addFirst(new Dot(snakeValue.getHead_Pre_X(), snakeValue.getHead_Pre_Y()));
                if (snakeKey.equals(ServerListener.Self_player)){
                    try {
                        pw = new PrintWriter(Gsocket.getOutputStream(), true);
                        //player = setPlayers.getPlayers().get("1");
                        pw.println("EGG:EAT:" + snakeKey);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                snakeValue.setHead_Pre_X(snakeValue.getHead().getX());
                snakeValue.setHead_Pre_Y(snakeValue.getHead().getY());
            }
            else{
                if (ServerListener.isMe == true && snakeKey.equals(ServerListener.Self_player)){
                    snakeValue.getBody().addFirst(new Dot(snakeValue.getHead_Pre_X(), snakeValue.getHead_Pre_Y()));
                    snakeValue.getBody().removeLast();
                    snakeValue.setHead_Pre_X(snakeValue.getHead().getX());
                    snakeValue.setHead_Pre_Y(snakeValue.getHead().getY());
                    ServerListener.isMe = false;
                }
                else if (snakeKey.equals(ServerListener.Self_player)){
                    snakeValue.getBody().addFirst(new Dot(snakeValue.getHead_Pre_X(), snakeValue.getHead_Pre_Y()));
                    snakeValue.getBody().removeLast();
                    snakeValue.setHead_Pre_X(snakeValue.getHead().getX());
                    snakeValue.setHead_Pre_Y(snakeValue.getHead().getY());
                }
            }
            if (snakeKey.equals(ServerListener.Self_player) && isPaused == true){
                boolean isCrushedByItself = false;
                for(int i = snakeValue.getBody().size()-1; i >= 1; i--){
                    if (snakeValue.getHead().getX() == snakeValue.getBody().get(i).getX()
                        && snakeValue.getHead().getY() == snakeValue.getBody().get(i).getY() && !isCrushedByItself){
                        isCrushedByItself = true;
                    }
                }
                if (isCrushedByItself){
                    System.out.println("CrushedByItself");
                    information = 2;
                }
            }
        }
    }
}
```

▲ 圖十七、changeSnakeLocation() 函式 程式碼片段

InformationPanel 類：

為遊戲資訊介面更新之類別。

✎ 本次改動：

1. 遊戲資訊版面數值更新

新增玩家遊玩的分、名次等相關資訊。

```
@Override
public void run(){//更新遊戲資訊
    while(true){
        string1 = "自己分數：" + Integer.toString(Main.p1.getScore());
        score.setText(string1);

        string_A = "最高分玩家：" + GamePanel.HighScore + "號";
        highScore.setText(string_A);

        string_B = "你的名次：" + GamePanel.No_self;
        No.setText(string_B);

        string_you = "你是玩家：" + ServerListener.Self_player;
        you.setText(string_you);

        switch(GamePanel.getInformation()){
            case 0:string2 = "遊戲進行中";break;
            case 1:string2 = "你撞穿牆壁了!";break;
            case 2:string2 = "你吃到自己了!";break;
            default:
        }
        show.setText(string2);
    }
}
```

▲ 圖十八、遊戲資訊更新程式碼片段

ServerListener 類：

Client 端將持續監聽 Server 所提供個更新資訊，並將所接收的資訊同步顯示於遊戲畫面上與相關的數值改動。Client-Server 通訊方式利用 Socket 進行文字通訊，並利用”：”來分割訊息，如：Server:Move:1:20:30，解析各欄位資訊可獲取相關的操作數值。

✎ 本次改動：

1. 新增數項遊戲資訊，如：其他玩家分數、我方名次、是否 Game Over 等

```

class ServerListener implements Runnable{

    boolean flag = false;
    static boolean flag2 = false;
    Socket Ssocket = null;
    static String player = "";
    static String Self_player = "";
    static boolean isMe = false;

    public ServerListener(Socket socket){
        this.Ssocket = socket;
    }

    @Override
    public void run() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(Ssocket.getInputStream()));
            String s1;
            while ((s1 = br.readLine()) != null) {
                System.out.println("cmd: " + s1);
                String control[] = s1.split(";");
                if(control[0].equals("Server")) {
                    if(control[1].equals("EGG")) {

                        System.out.println(control[1]+":"+control[2]+":"+control[3]);

                        Main.pl.getDessert().setX(Integer.parseInt(control[2]));
                        Main.pl.getDessert().setY(Integer.parseInt(control[3]));

                        GamePanel.eggChange = true;
                        Main.pl.changeSnakeLocation();
                    } else if(control[1].equals("Player")) {

                        if (flag=false){
                            Self_player = control[2];
                            // System.out.println(Self_player);
                            Main.pl.snakes.getSnakes().put(Self_player, Main.snakeSelf); //不加入自己的蛋 加入別人的蛋 自己用snakeSelf來運作
                            flag = true;
                        } else {
                            if(!control[2].equals(Self_player)){
                                player = control[2];
                                Main.pl.snakes.getSnakes().put(player, new Snake()); //不加入自己的蛋 加入別人的蛋 自己用snakeSelf來運作
                                System.out.println(control[1]+":"+player);
                            }
                        }
                    } else if(control[1].equals("Move")) {
                        System.out.println(control[1]+":"+control[2]+":"+control[3]);
                        System.out.println(control[1]+":"+control[2]+":"+control[4]);
                        Main.pl.snakes.getSnakes().get(control[2]).getHead().setX(Integer.parseInt(control[3]));
                        Main.pl.snakes.getSnakes().get(control[2]).getHead().setY(Integer.parseInt(control[4]));
                        if (control[2].equals(Self_player)){
                            isMe = true;
                        }
                        Main.pl.changeSnakeLocation();
                    } else if (control[1].equals("Score")){
                        System.out.println(control[1]+":"+control[2]+":"+control[3]);
                    } else if(control[1].equals("PlayersScore")) {
                        GamePanel.No_self = control[2];
                        GamePanel.HighScore = control[3];
                    } else if(control[1].equals("GAMEOVER")){
                        Main.pl.snakes.getSnakes().get(control[2]).getBody().clear();
                        System.out.println(control[1]+":"+control[2]+ " GAMEOVER");
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

▲ 圖十九、ServerListener 類 程式碼片段

Snake 類：

位玩家本身蛇體儲存相對應狀態與蛇頭座標。

✎ 本次改動：

1. 增加玩家貪食蛇-蛇頭上一座標點

```

public int getHead_Pre_X(){
    return head_Pre_X;
}

public int getHead_Pre_Y(){
    return head_Pre_Y;
}

public void setHead_Pre_X(int head_Pre_X){
    this.head_Pre_X = head_Pre_X;
}

public void setHead_Pre_Y(int head_Pre_Y){
    this.head_Pre_Y = head_Pre_Y;
}

```

▲ 圖二十、Snake 類 程式碼片段

Snakes 類：

新增 Snakes 類，為客戶端儲存多位玩家蛇體狀態。

👉 本次改動：

```
class Snakes{
    private Map<String, Snake> snakes = new LinkedHashMap<String, Snake>();
    public Snakes(){
    }
    public Map<String, Snake> getSnakes(){
        return snakes;
    }
}
```

▲ 圖二十一、Snakes 類

● Server 端：

Server 類：

進行初始化相關設定，如：等待玩家進入(計數玩家數量與對應連接的 port 號)，賦予初始遊戲數據並使用 newCachedThreadPool 建立執行緒，在伺服器上為每位玩家開啟相對應控制類別。

```
public class Server extends JFrame {
    /**
     * Create by ShengPo
     */
    private static final long serialVersionUID = 1L;
    private JTextArea txt = new JTextArea("伺服器已經啟動\r\n"); // 伺服器不會換行 \n\r 也一樣
    private ServerSocket serverSocket = null;
    ExecutorService exec = null;
    private Score sscore = new Score();
    private List<Socket> clients;
    Map<Integer, Integer> ClientScore = new LinkedHashMap<Integer, Integer>();
    PrintWriter pw;
    PrintWriter ppw;
    int EGG_X = 404;
    int EGG_Y = 380;

    public Server() throws IOException {
        txt.setLineWrap(true); // 設置自動換行功能
        setLayout(new BorderLayout());
        this.add(new JScrollPane(txt), BorderLayout.CENTER);
        setSize(500, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        try {
            clients = new ArrayList<Socket>();
            System.out.println("Server 啟動成功");
            int count = 1; // 計算使用者

            serverSocket = new ServerSocket(54321);
            exec = Executors.newCachedThreadPool();
            exec.execute(sscore);
            while (true) { // 當伺服器結束?
                Socket socket = null;
                socket = serverSocket.accept();
                System.out.println("Client " + socket.getPort() + " Enter");

                clients.add(socket);

                for (int i = clients.size() - 1; i >= 0; i--) {
                    ppw = new PrintWriter(clients.get(i).getOutputStream(), true);
                    for (int j = count; j > 0; j--) {
                        ppw.println("Server: Player: " + (j));
                        System.out.println("Server: Player: " + (j));
                    }
                }

                if (count == 1) {
                    pw = new PrintWriter(clients.get(count - 1).getOutputStream(), true);
                    pw.println("Server: EGG: " + EGG_X + " " + EGG_Y); // 發送初始位置 TODO 再隨機位置
                    pw.flush();
                } else {
                    pw = new PrintWriter(clients.get(count - 1).getOutputStream(), true);
                    pw.println("Server: EGG: " + EGG_X + " " + EGG_Y); // 發送初始位置 TODO 再隨機位置
                    pw.flush();
                }
                ppw.flush();
                ClientScore.put(count, 0);
                exec.execute(new ClientListener(socket, count)); // 用執行緒池啟動接收 Client 端資料 Thread
                count++;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

▲ 圖二十二、Server 類 程式碼片段

ClientListener 類：

每有玩家在 server 上也相對應的玩家狀態監聽器，玩家連線時會建立此類於一個執行緒，負責分配玩家位置、目標(蘋果)的位置、分數的增加控制以其玩家遊戲狀態，如：Game Over、Game Start、FIRST(Start)、STO 與速度調整 Speed。

👉 本次改動：

1. 新增玩家位置隨機生成、遊戲狀態控制信令擷取等功能。

```
@Override
public void run() {
    try {
        br = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        msg = "Player" + count + "進入遊戲室！";
        Broadcast(msg);
        while ( ( msg = br.readLine()) != null) { //接收Client端資訊
            msg = "Player" + count + ":" + msg;
            Broadcast(msg); //廣播，因為控制方向 Player1和2都要接收

            String control[] = msg.split(":");
            System.out.println("msg:"+msg);

            //Play吃到蛋
            if(control[1].equals("EGG") && control[2].equals("EAT")) {
                EGG_X = (404 - 300 + ((int)(Math.random() * 15 * 2)) * 20);
                EGG_Y = (380 - 300 + ((int)(Math.random() * 15 * 2)) * 20);
                String rand = "Server:EGG:" + EGG_X + ":" + EGG_Y;

                for (int i = clients.size() - 1; i >= 0; i--) {
                    PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                    pw.println(rand);
                    System.out.println(rand);
                }
            }
            if(control[3].equals(String.valueOf(count)) ) { //針對不同盤
                Play_Score++;
                ClientScore.put(count, Play_Score);

                for (int i = clients.size() - 1; i >= 0; i--) {
                    PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                    // pw.println("Server:Score:"+count+":"+ ClientScore.get(count));
                    pw.println("Server:Score:"+count+":"+ Play_Score);
                }
                System.out.println("Server:Score:"+count+":"+ Play_Score);
            }
        }
        if(control[1].equals("Speed") && control[2].equals(String.valueOf(count))) { //針對不同盤
            move.Speed = Integer.parseInt( control[3] );
            System.out.println("Server:Speed:"+count+":"+ Play_Score);
        }
        //方向
        if(control[1].equals("Direction")) {
            if(control[2].equals("UP")) {
                move.Play1_State = "UP";
            }else if(control[2].equals("DOWN")) {
                move.Play1_State = "DOWN";
            }else if(control[2].equals("LEFT")) {
                move.Play1_State = "LEFT";
            }else if(control[2].equals("RIGHT")) {
                move.Play1_State = "RIGHT";
            }
        }
        if(control[1].equals("GAMEOVER")&& control[2].equals(String.valueOf(count))) {
            System.out.println(control[0]+":"+control[1]);
            move.setHead(384, 360);
            move.StartGame = false;
            move.Step = 0;
            for (int i = clients.size() - 1; i >= 0; i--) {
                PrintWriter pw = new PrintWriter(clients.get(i).getOutoutStream(), true);
            }
        }
    }
}
```

```

} else if (control[1].equals("GAMESTART") && control[2].equals(String.valueOf(count))) {
    System.out.println(control[0] + ":" + control[1]);
    Play_Score = 0;

    exec.execute(move);
    move.Step = 20;
    move.Speed = 1000;
    move.StartGame = true;
    move.setHead((404 - 300 + ((int)(Math.random() * 15 * 2)) * 20), (380 - 300 + ((int)(Math.random() * 15 * 2)) * 20));
    move.Play1_State = "DOWN";
} else if (control[1].equals("FIRST") && control[2].equals(String.valueOf(count))) {
    System.out.println(control[0] + ":" + control[1]);
    if (flag == false) {
        Play_Score = 0;

        // exec.execute(move); //限制一直按Enter會一直制
        move.Step = 20;
        move.Speed = 1000;
        move.StartGame = true;
        move.Play1_State = "DOWN";
        flag = true;
    }
}

} else if (control[1].equals("STOP") && control[2].equals(String.valueOf(count))) {
    System.out.println(control[0] + ":" + control[1]);
    isPaused = !isPaused;
    if (isPaused == true) {
        move.Step = 0;
    } else {
        move.Step = 20;
    }
}
}

} catch (Exception e) {
    // socket.close();
    System.out.println("ClientListener "+socket.getPort()+"-count+" is close!"); // 將在GUI // 選擇player count 每個人用Linklist??
    txt.append("[*"+socket.getPort()+"] Player "+count+" 已關閉!\n");
    move.StartGame = false;
    System.out.println(e);
}
}

```

▲ 圖二十二、ClientListener 類 部分程式碼

Move 類：

基於網路遊戲安全性考量，設計遊 Server 控制玩家移動座標，並加入玩家可控制自己貪食蛇移動的速度。

✎ 本次改動：

1. 更改玩家移動信令規則，增加移動座標是屬於哪位玩家，如：
Server:Move:1:20:30，第三欄位為玩家編號，後面接續為 X 座標與 Y 座標。
2. 增加 Speed 控制位置更新速度已達到玩家貪食蛇加速目標。

```

public class Move implements Runnable{
    // private Socket socket;
    Socket ssocket;
    int ccount;
    int Speed = 1000;
    boolean StartGame = true;
    int Step = 0;
    // String Play1_State = "";
    String Play1_State = "DOWN"; //一開始預設方向
    int play1_X = 0; int play1_Y = 0; //可以控制初始位置
    public Move(Socket socket, int count){
        this.ssocket = socket;
        this.ccount = count;
    }

    public void setHead(int x, int y){
        this.play1_X = x;
        this.play1_Y = y;
    }

    @Override
    public void run() {
        //前面要設定StartGame變數 用public還是private?
        //改寫多人
        while(StartGame == true) {
            try {
                //玩家一
                if(Play1_State == "UP") {
                    play1_Y--Step;
                    for (int i = clients.size() - 1; i >= 0; i--) {
                        PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                        pw.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*1");
                        System.out.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*1");
                    }
                } else if(Play1_State == "DOWN") {
                    play1_Y++Step;
                    for (int i = clients.size() - 1; i >= 0; i--) {
                        PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                        pw.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*2");
                        System.out.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*2");
                    }
                } else if(Play1_State == "LEFT") {
                    play1_X--Step;
                    for (int i = clients.size() - 1; i >= 0; i--) {
                        PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                        pw.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*3");
                        System.out.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*3");
                    }
                } else if(Play1_State == "RIGHT") {
                    play1_X++Step;
                    for (int i = clients.size() - 1; i >= 0; i--) {
                        PrintWriter pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                        pw.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*4");
                        System.out.println("Server:Move:"+ccount+":"+ play1_X + ":" + play1_Y+"*4");
                    }
                }
            }
            Thread.sleep(Speed);
        }
    }
}

```

▲ 圖二十三、Move 類 部分程式碼

Score 類：

此類別負責統計玩家分數與分數排序，於每秒發送分數更新廣播，其中附帶玩家分數、排名與最高分玩家編號，並且此類為開啟 Server 時建立唯一執行緒。

✎ 本次改動：


```

public class Score implements Runnable{
    int ClientScoreVal;
    PrintWriter pw;
    int firstKey;
    public Score(){
    }

    @Override
    public void run() {

        while(true) {
            Set<Map.Entry<Integer, Integer>> companyFounderSet = ClientScore.entrySet();
            List<Map.Entry<Integer, Integer>> companyFounderListEntry = new ArrayList<Map.Entry<Integer, Integer>>(companyFounderSet);
            Collections.sort(companyFounderListEntry, new Comparator<Map.Entry<Integer, Integer>>() {
                @Override
                public int compare(Entry<Integer, Integer> es1,
                                   Entry<Integer, Integer> es2) {
                    return es2.getValue().compareTo(es1.getValue());
                }
            });
            ClientScore.clear();
            int count = 1;
            for(Map.Entry<Integer, Integer> map : companyFounderListEntry){
                if (count == 1) {
                    firstKey = map.getKey();
                }

                ClientScore.put(map.getKey(), map.getValue());
                count++;
            }
            Set<Integer> keys = ClientScore.keySet();
            List<Integer> listKeys = new ArrayList<Integer>(keys );

            try {

                for (int i = 0; i < clients.size(); i++) {
                    pw = new PrintWriter(clients.get(i).getOutputStream(), true);
                    pw.println("Server:PlayersScore:"+(listKeys.indexOf(i+1)+1)+"-"+firstKey);
                }
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

▲ 圖二十四、Score 類