Nombre Apellido: David Cusi   
Legajo: 42286775   
Email: davidcusi.estudiante@gmail.com

Profesor:Gonzalo Rodrigo  
Com: 03

Programacion 1  
trabajo practico

****

**Introducción**

En este trabajo práctico, desarrollamos un juego que simula el movimiento de un personaje, pep, en un entorno de plataformas, representado por varias islas. El objetivo del juego es mantener al personaje dentro de las plataformas; si el personaje cae del mundo, el juego termina. El proyecto involucra la implementación de clases orientadas a objetos que controlan el movimiento del personaje y la interacción con el entorno, proporcionando una experiencia interactiva al usuario.

**Descripción**

En el desarrollo del proyecto, implementamos las siguientes clases principales:

Clase Pep

La clase Pep representa al personaje principal del juego, incluyendo propiedades de su posición y métodos de movimiento.

Variables de instancia:

x: la posición en el eje X del personaje.

y: la posición en el eje Y del personaje.

ancho: el ancho del personaje, utilizado para calcular colisiones.

alto: la altura del personaje.

Métodos:

getX(): retorna la posición en el eje X.

getY(): retorna la posición en el eje Y.

getAncho(): retorna el ancho del personaje.

getAlto(): retorna la altura del personaje.

moverAbajo(): mueve al personaje hacia abajo en caso de que no esté dentro de los límites de una isla.

Clase Isla

La clase Isla representa las plataformas sobre las que puede posicionarse el personaje.

Variables de instancia:

x: la posición central en el eje X de la isla.

ancho: el ancho de la isla, utilizado para calcular sus límites en X.

Métodos:

getX(): retorna la posición en el eje X.

getAncho(): retorna el ancho de la isla.

Problemas Encontrados y Soluciones:

Problema de límites de colisión: Uno de los desafíos fue calcular correctamente los límites de las colisiones entre el personaje y la isla, ya que pep debe considerarse “dentro” de una isla si sus límites están dentro de los de la isla.

Solución: Se calcularon los límites izquierdo y derecho tanto del personaje como de la isla usando su posición x y ancho, lo que permitió simplificar la condición de colisión.

Condiciones anidadas: Para determinar si el personaje estaba dentro de una isla, surgieron varias condiciones anidadas, lo que complicó la lectura del código.

Solución: Simplificamos las condiciones intermedias y usamos variables temporales, lo que mejoró la claridad y la eficiencia.

Implementación

// Clase Pep

public class Pep {

    private int x;

    private int y;

    private int ancho;

    private int alto;

    public Pep(int x, int y, int ancho, int alto) {

        this.x = x;

        this.y = y;

        this.ancho = ancho;

        this.alto = alto;

    }

    public int getX() { return x; }

    public int getY() { return y; }

    public int getAncho() { return ancho; }

    public int getAlto() { return alto; }

    public void moverAbajo() {

        y += 1;  // Mueve al personaje hacia abajo en el eje Y

    }

}

// Clase Isla

public class Isla {

    private int x;

    private int ancho;

    public Isla(int x, int ancho) {

        this.x = x;

        this.ancho = ancho;

    }

    public int getX() { return x; }

    public int getAncho() { return ancho; }

}

// Ejemplo de lógica principal para verificar si Pep está dentro de una isla

if (pep != null) {

    Isla isla = this.islas[10];

    int anchoIsla = isla.getAncho();

    int xIsla = isla.getX();

    // Límites de la isla

    int limiteIzquierdoIsla = xIsla - anchoIsla;

    int limiteDerechoIsla = xIsla + anchoIsla;

    int anchoPep = pep.getAncho();

    int xPep = pep.getX();

    // Límites de pep

    int limiteIzquierdoPep = xPep - anchoPep;

    int limiteDerechoPep = xPep + anchoPep;

    // Verificar si pep está dentro de la isla

    if (limiteIzquierdoPep >= limiteIzquierdoIsla && limiteDerechoPep <= limiteDerechoIsla) {

        System.out.println("Dentro de la isla");

    } else if (pep.getY() - pep.getAlto() < this.entorno.alto()) {

        pep.moverAbajo();

    }

} else {

    System.out.println("Fin del juego: pepe cayó del mundo");

}

// Clase Pep

public class Pep {

    private int x;

    private int y;

    private int ancho;

    private int alto;

    public Pep(int x, int y, int ancho, int alto) {

        this.x = x;

        this.y = y;

        this.ancho = ancho;

        this.alto = alto;

    }

    public int getX() { return x; }

    public int getY() { return y; }

    public int getAncho() { return ancho; }

    public int getAlto() { return alto; }

    public void moverAbajo() {

        y += 1;  // Mueve al personaje hacia abajo en el eje Y

    }

}

// Clase Isla

public class Isla {

    private int x;

    private int ancho;

    public Isla(int x, int ancho) {

        this.x = x;

        this.ancho = ancho;

    }

    public int getX() { return x; }

    public int getAncho() { return ancho; }

}

// Ejemplo de lógica principal para verificar si Pep está dentro de una isla

if (pep != null) {

    Isla isla = this.islas[10];

    int anchoIsla = isla.getAncho();

    int xIsla = isla.getX();

    // Límites de la isla

    int limiteIzquierdoIsla = xIsla - anchoIsla;

    int limiteDerechoIsla = xIsla + anchoIsla;

    int anchoPep = pep.getAncho();

    int xPep = pep.getX();

    // Límites de pep

    int limiteIzquierdoPep = xPep - anchoPep;

    int limiteDerechoPep = xPep + anchoPep;

    // Verificar si pep está dentro de la isla

    if (limiteIzquierdoPep >= limiteIzquierdoIsla && limiteDerechoPep <= limiteDerechoIsla) {

        System.out.println("Dentro de la isla");

    } else if (pep.getY() - pep.getAlto() < this.entorno.alto()) {

        pep.moverAbajo();

    }

} else {

    System.out.println("Fin del juego: pepe cayó del mundo");

}

Conclusiones

En este proyecto, aprendimos cómo estructurar las clases para representar entidades de juego como personajes y plataformas, utilizando técnicas de programación orientada a objetos. Trabajamos en la lógica de colisiones. Además se vio errores comunes en el diseño de juegos, como los problemas de colisiones.