Licenciatura en Sistemas

Trabajos Práctico

“Vegan Bird”

Programación I

(2do semestre - 2019)

El trabajo consiste en la creación de un videojuego en el lenguaje de programación Java, el cual es un juego de plataformas en el que un pájaro deberá esquivar los diferentes obstáculos en pantalla, obtendrá puntos cada vez que consuma una verdura y perderá cuando consuma un alimento no vegano. Además, éste tiene como arma principal una pistola que transforma la comida no vegana en frutas.

**Desarrollo**

Para llevar a cabo la realización del juego utilizamos diferentes clases ya incluidas en el paquete base del Trabajo Práctico. La clase Entorno, Herramientas, Board e InterfaceJuego proveen la creación de una interfaz y métodos auxiliares. Como punto de partida optamos por lograr que el pájaro tenga gravedad, seguimos con la aparición de obstáculos en pantalla y su movimiento, luego logramos que el pájaro colisione con los obstáculos, a su vez agregamos el arma que transforma los alimentos y por último, que aparezcan los alimentos en pantalla, se muevan, y se obtenga diferentes puntos por consumirlos.

Creamos nuevas clases y modificamos otras, como por ejemplo la clase Bird, que tiene como variables de instancia dos enteros, un “x” y un “y”, que sirven para posicionarlo en pantalla, y una imagen que representa al pájaro. La clase Fondo, que tiene como variables de instancia tres imágenes diferentes para cada ventana y dos enteros “x” e “y”, que lo posicionan en pantalla. Las clases Carne y Fruta, en dónde ambas tienen coordenadas enteras “x” e “y”, un ancho, un alto y un valor propio al momento de que colisione el pájaro o el proyectil con ellos. La clase Arma que dibuja el arma que transforma el alimento, en donde sus variables de instancia son dos enteros x e y que lo posicionan en pantalla, una imagen, y un ángulo que va a servir para apuntar el arma durante el juego. Por otra parte, la clase Obstáculo, dibuja diferentes obstáculos en unas ciertas coordenadas “x” e “y”. Tiene como variables de instancia dos coordenadas enteras, un ancho, un alto y una imagen. La clase Puntaje, que almacena los puntos que el jugador obtiene durante el juego, tiene como variable de instancia un entero. La clase Proyectil, dibuja un proyectil en pantalla con el mismo ángulo y coordenadas que un objeto Arma, como variable de instancia posee dos coordenadas enteras “x” e “y”, el ángulo en donde se va a disparar y una imagen. Por otro lado, la clase Menu, crea una nueva ventana con dos opciones de dificultad, esta toma un nuevo ‘entorno’ y cuando se elige la dificultad esta ventana se cierra. La clase Juego2 es el juego original pero con un movimiento de obstáculos más rápido y por último, la clase Perdiste crea una ventana de “GameOver”, cuando el pájaro colisiona con algún obstáculo o se va fuera de los límites de la ventana. Esta misma toma un ‘entorno’ nuevo para crear la ventana y dentro tiene un método tick(), que va a funcionar siempre y cuando el jugador se quede en esa pantalla.

**Descripción de métodos por clases.**

Clase entorno:

* ***public void cerrar(boolean b)*** Logra que la ventana se esconda.
* ***public void terminar()*** Termina la ejecución del programa.

Clase Arma:

* ***public void dibujarArma()*** Dibuja un objeto Arma en un entorno específico.
* ***public void girar()*** Modifica el ángulo del objeto Arma.
* ***public void gravedadArma()*** Logra que el objeto sea afectado por la gravedad.
* ***public void moverArriba()*** Modifica la coordenada “y” del objeto
* ***public void dibujarRectanguloArma()*** *Dibuja un rectángulo alrededor del objeto en un entorno específico.*
* ***Rectangle dameRectangulo()*** *Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.*

Clase Bird:

* **public void dibujarse()** Dibuja un objeto Arma en un entorno específico.
* **public void gravedadBird()** Logra que el objeto sea afectado por la gravedad.
* **public void moverArriba()** Modifica la coordenada “y” del objeto.
* ***public void dibujarRectanguloBird()*** *Dibuja un rectángulo alrededor del objeto en un entorno específico.*
* ***Rectangle dameRectangulo()*** *Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.*

Clase Carne:

* ***public void dibujarseRectangulo()*** *Dibuja un rectángulo alrededor del objeto en un entorno específico.*
* **public void dibujarseImagen()** Dibuja un objeto Carne en un entorno específico.
* **public void transformar()** Cambia la imagen de un objeto carne, por otra pasada como parámetro.
* ***Rectangle dameRectanguloCarne()*** *Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.*
* ***public void moverCarne()*** Mueve el objeto modificando su coordenada x.

Clase Fondo:

* ***public void dibujarFondo()*** Dibuja la imagen en un entorno específico.
* ***public void dibujarFinal()*** Dibuja la imagen de GameOver.
* ***public void dibujarNombreJuego()*** Dibuja el nombre del juego en el menú.
* ***public void dibujarComienzo()*** Dibuja el fondo en el menú.

Clase Fruta:

* ***public void dibujarseRectangulo()*** *Dibuja un rectángulo alrededor del objeto en un entorno específico.*
* **public void dibujarseImagen()** Dibuja un objeto Fruta en un entorno específico.
* ***Rectangle dameRectanguloFruta()*** *Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.*
* ***public void moverFruta()*** Mueve el objeto modificando su coordenada “x”.

Clase Obstáculo:

* ***public void dibujarseRectangulo()*** *Dibuja un rectángulo alrededor del objeto en un entorno específico.*
* ***public void dibujarseImagenArriba()*** Dibuja la imagen del objeto en una posición diferente.
* ***public void dibujarseImagenAbajo()*** Dibuja la imagen del objeto en una posición diferente.
* ***Rectangle dameRectangulo()*** *Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.*
* ***public void moverObstaculo()*** Mueve el objeto modificando su coordenada x.

Clase Proyectil:

* ***public void dibujarseProyectil()*** Dibuja la imagen del objeto en una posición diferente.
* ***public void girar()*** Modifica el ángulo del objeto Proyectil.
* ***public void alLugar()*** Retoma su posición en las coordenadas como parámetros.
* ***public void dibujarseRectangulo()*** Dibuja un rectángulo alrededor del objeto en un entorno específico.
* ***public void moverAdelante()*** Logra que el proyectil se ‘dispare’.
* ***Rectangle proyectilRectangulo()*** Método auxiliar para controlar la colisión. Devuelve un objeto Rectangle con las mismas medidas que el objeto.

Clase Puntaje:

* ***public void sumarPuntos()*** Suma los puntos obtenidos durante el juego.
* ***public void mostrarPuntaje()*** Muestra el puntaje en la pantalla de GameOver.
* ***public void mostrarPuntaje()*** Muestra el puntaje durante la ejecución del juego.

Clase Juego/Juego2:

* ***public voidObjetosSinSuperposicion()*** Evita que se superpongan los objetos.
* ***void dibujarEnPantalla()*** Dibuja todos los objetos en pantalla.
* ***public void gameOver()*** Crea una nueva ventana cuando el pájaro colisione con un obstáculo o se vaya fuera de los límites.
* ***public void transformarAlimento\_Puntos()*** Se encarga de que el rayo transforme el alimento, y le suma puntos.
* ***public void colisionAlimentos\_Puntos()*** Controla que cuando el pájaro consume un objeto Fruta o Carne, se sumen o resten puntos. A su vez hace que el objeto consumido salga de pantalla.
* ***public void eventesBird\_Proyectil()*** Controla los movimientos del pájaro y la dirección de los proyectiles.
* ***public void objetosRandom()*** Cuando los objetos se van de pantalla, se crea otro modificando su coordenada y.
* ***public void obstaculosRandom()*** Cuando el obstáculo se va de pantalla, crea otro modificando su coordenada y.
* ***public void movimientosObjetos()*** Mueve objetos en pantalla.
* ***public void movimientoObstaculo()*** Mueve los obstáculos en pantalla.
* ***public boolean colisionPantalla()*** Indica cuando el objeto se va de la pantalla.
* ***public boolean colision()*** Indica cuando dos Rectangles se tocan.
* ***public void interseccionFruta()*** Trata de evitar que se superpongan objetos.
* ***public void contieneObstaculoFruta()*** Trata de evitar que se superpongan objetos.
* ***public void intersecaComidads()*** Trata de evitar que se superpongan objetos.
* ***private int randomRange()*** Método auxiliar que devuelve un número random en el rango solicitado.

**Invariantes de representación por métodos:**

Clase Arma:

* **int x** no se debe pasar de los límites de la pantalla.
* **int y** no se debe pasar de los límites de la pantalla.

Clase Bird:

* **int x** no se debe pasar de los límites de la pantalla.
* **int y** no se debe pasar de los límites de la pantalla.

Clase Carne:

* **int x** no se debe pasar de los límites de la pantalla.
* **int y** no se debe pasar de los límites de la pantalla.
* **int ancho** e **int alto** deben tener las mismas medidas de la imagen utilizada.
* **int valor** no debe valer otro entero que no sea -5.

Clase Fruta:

* **int x** no se debe pasar de los límites de la pantalla.
* **int y** no se debe pasar de los límites de la pantalla.
* **int ancho** e **int alto** deben tener las mismas medidas de la imagen utilizada.
* **int valor** no debe valer otro entero que no sea 5.

Clase Fondo:

* **int x** e **int y** no pueden pasarse de los límites de la pantalla.

Clase Obstáculo:

* **int x** no se debe pasar de los límites de la pantalla.
* **int y** no se debe pasar de los límites de la pantalla.
* **int ancho** e **int alto** deben tener las mismas medidas de la imagen utilizada.

Clase Proyectil:

* **int x** debe ser igual a las coordenada x del objeto Arma. Excepto cuando se ‘dispara’.
* **int y** debe ser igual a las coordenada x del objeto Arma. Excepto cuando se ‘dispara’.

Clase Puntaje:

* **int puntos** no puede ser menor a 0.

**Resumen de los problemas encontrados:**

El principal inconveniente que tuvimos durante la realización del trabajo fue con las colisiones entre objetos. Los parámetros que pasábamos no eran los correctos, los rectángulos estaban fuera de eje y esto no permitía la colisión entre ellos. Una vez resuelto este problema, no se generaba la colisión entre el pájaro y los obstáculos. Ante esto, la solución fue usar un método dentro de la clase Rectangle llamada intersects() que corrobora que dos rectángulos se intersequen. Dentro de cada clase creamos métodos que devuelven objetos Rectangle.

Otro inconveniente que tuvimos fue asignarle un valor a algún alimento ya que nosotros solo cambiabamos la imagen de una carne cuando un proyectil lo colisionaba, para resolverlo decidimos agregar una variable de instancia en la clase Fruta y en la clase Carne, para que la estos a la hora de colisionar con el proyectil, ademas de tomar la imagen de fruta, también toma el valor de la fruta que luego se le sumará a puntos

**Anexo:**

Método **tick()** con todas las funciones del juego resumidas.

