Trabajo Práctico “Selva Mono Capuchino”

Materia: Programación I

Grupo 6:

Nombre: Ormeño Martin Felipe
Legajo: 44.505.023
Contacto:

Nombre: Paez Rocco Nicolas –
Legajo: 44.171.874
Contacto:

Nombre: Aquino Gaspar Abel –
Legajo: 44.213.587
Contacto:

Objetivo: Diseñar, coordinar e implementar lo visto en clases en el desarrollo del juego propuesto por los docentes.

Introducción:

El proyecto consistía en crear una aplicación que tiene como protagonista a un mono en una selva, manteniendo un plano de dos dimensiones, el cual, junto al mono, siempre se mantenía estático, pero que, al mover los distintos objetos, creaba la ilusión de desplazamiento por parte del mono. Los distintos objetos que observamos en el escenario representan una clase.

Descripción:

Para la creación de los distintos objetos que podemos observar en el escenario, se crearon clases, las cuales tienen sus respectivas variables de instancia.
A continuación, se nombrarán las clases y una breve explicación (no se explicarán dos veces la misma variable o método implementados):

Arbol :

Con esta clase creamos los árboles, se toman 9 variables de instancia (de las cuales 7 comparte con las clases Tigre, Serpiente, Saco, Piedra, Fruta y Basura).

- Las variables “x” e “y” son de tipo double y representan la coordenada del objeto en la pantalla.
- Las variables enteras “al” y “an” permite definir el tamaño del hitbox del objeto.
- La variable “escala” como su nombre bien lo expresa, es usada para controlar el tamaño del objeto.
- Para la variable “observer” se importó una clase externa y es utilizada para

- Al igual que con la anterior variable de instancia, para esta también se importa una clase y nos sirve para insertarle una imagen al objeto.

- Variable “rama”, es de tipo Rama y representa si un árbol tiene el objeto rama o no.

Esta clase tiene un constructor y dos métodos.

- El constructor recibe como parámetros una coordenada “x” y una variable de tipo “Suelo” (más adelante veremos de qué se trata), crea los árboles con una probabilidad de que tengan una rama.

- public void dibujarse (Entorno entorno), no hay mucho para explicar, el método dibuja la imagen dentro del entorno, utiliza variables de instancia para determinar el tamaño a dibujar.

```
public void dibujarse(Entorno entorno) {
        entorno.dibujarImagen(arbol, x, y, 0, escala);
}
```

- public void moverse(), este método decrementa en 4 la posición del x, esto da la ilusión de que el mono corre y va dejando atrás los objetos del escenario (el árbol en este caso).

```
public void moverse() {
        this.x -= 4;
}
```

Basura / Piedra / Saco / Serpiente / Tigre / Fruta:

Con estas clases se crean los sacos de bolsa de basura, las piedras que lanza el mono y los depredadores que el mono se puede encontrar en el juego.

Todas las variables de instancia que contienen las clases fueron explicadas en la sección “Arbol” (esto porque como anticipamos, varias clases comparten variables de instancia).

Los métodos que tienen dichas clases son los mismos que los de “Arbol”, por lo tanto, no se volverán a detallar. Cabe destacar que, en el caso del tigre, el método “moverse” decrementa más rápido la variable “x” debido a que el tigre al igual que el mono, corre, por lo que debemos tomarlo en cuenta para la ilusión de contraposición.

Corazon:

Para añadirle un poco de versatilidad al juego, se propuso que este tenga un contador de vidas, por lo cual al perder tres veces se termine el juego. Para llevarlo a la pantalla se implementó la clase “Corazon”.

Además de las variables de instancias que comparte con las anteriores clases (x, y, escala, al, an, y observer), se implementó una de tipo booleana y dos imágenes para diferenciar las vidas que le quedan al jugador.

Su constructor lleva como parámetro un entero para definir el espacio que se quiera ver en la pantalla entre los corazones.

Tiene solo un método y es el public void dibujarse(Entorno entorno), el cual depende de la variable “lastimado”, si vale true, el corazón se dibujará rojo. Por otro lado, si es false, el corazón se dibujará negro (indicando que no se cuenta con esa vida).

```
public void dibujarse(Entorno entorno) {
        if(!lastimado) {
            entorno.dibujarImagen(corazonImg, x, y, 0, escala);
        }else {
            entorno.dibujarImagen(corazonLasImg, x, y, 0, escala);
        }
    }
```

Suelo:

Sus variables de instancias son las vistas hasta ahora, “y, x, an, al, superficie, escala, pisoimg y observer” con las cuales se posiciona el objeto en pantalla, se define un tamaño y se le coloca una imagen.

Solamente tiene el método public void dibujarse(Entorno entorno) con el cual se dibuja en pantalla el suelo que podemos ver debajo del mono.

Rama:

Uno de los requisitos que debían tener los arboles es que algunos tenían que tener una rama para que el mono pueda saltar sobre ella. Con la clase Rama se logró implementar esto, la clase tiene catorce variables de instancia, algunas anteriormente explicadas, otras nuevas, tales como:

- Para que la rama tenga un tamaño se optó por una variable donde indica el inicio de coordenada de la rama “double principio”.
  Por lo tanto, debería tener otra variable que indique donde termina, tal es el caso de “double fin”
- Para que las ramas no tengan siempre una serpiente encima, se implementó la variable “random”, para que haya una probabilidad de aparición.
- Como uno de los requisitos lo pedía, las ramas debían tener un depredador, para eso usamos la variable de tipo Serpiente “serpiente”.
- Se optó por añadir frutas que el mono puede agarrar en las ramas, para eso se usa una lista de tipo frutas que contiene las bananas, esto lo hacemos con la variable “frutas”

Sus métodos son los mismos que vimos en otras clases, “dibujarse(Entorno entorno)” y “public void moverse()”.

Mono:

Como no podía ser de otra manera, nuestro protagonista es un mono, el cual se crea mediante la clase “Mono” cuyas variables de instancia ya vimos a excepción de una:

- Para que el mono no tenga una infinita cantidad de piedras, se decidió que la variable “piedras” sea una lista de tipo Piedra, que contenga la cantidad de piedras que se quiera.

Dentro de la clase se pueden observar cuatro métodos, omitiremos el método dibujarse:

- Para que el mono salte, se usa el método “saltar()” que decrementa la coordenada y del mono.

```
public void saltar() {
        y -= 20;
}
```

- Como todo lo que sube debe caer, tiene el método “caer()” que va incrementando la coordenada.

```
public void caer(Entorno entorno) {
        this.y += 6;
        if(entorno.estaPresionada(entorno.TECLA_ABAJO)) {
                this.y += 10;
        }

}
```

- Por último, encontramos el método “lanzar()” el cual añade un objeto piedra a la variable de instancia “piedras”.

```
public void lanzar() {
        piedras.add(new Piedra(this.x, this.y));
}
```

Juego:

Dentro de la clase Juego, podemos encontrar el método “intersecan()” que devuelve un booleano para indicar si dos objetos se intersecan o no.

```
public boolean intersecan(double x1, double x2, double y1, double y2, int al1, int al2, int an1, int an2) {
        boolean intersecY = false;
        double distanciaY = Math.max(y1, y2) - Math.min(y1, y2);
        double sumaAlt = (al1/2) + (al2/2);
        if(distanciaY - sumaAlt < 0) {
```

```
                intersecY = true;
        }

        boolean intersecX = false;
        double distanciaX = Math.max(x1, x2) - Math.min(x1, x2);
        double sumaAnch = (an1/2) + (an2/2);
        if(distanciaX - sumaAnch < 0) {
                intersecX = true;
        }

        if(intersecX && intersecY) {
                return true;
        }else {
                return false;
        }

    }
```

Código fuente:

```
package juego;


import java.awt.Color;
import java.awt.Image;
import java.util.ArrayList;
import javax.sound.sampled.Clip;

import entorno.Entorno;
import entorno.Herramientas;
import entorno.InterfaceJuego;

public class Juego extends InterfaceJuego {
    // El objeto Entorno que controla el tiempo y otros
    private Entorno entorno;
    Mono mono;
    Suelo piso;
    Saco saco;
    Basura basura;
    ArrayList<Arbol> arboles;
    ArrayList<Piedra> piedras;
    ArrayList<Tigre> tigres;
    ArrayList<Basura> basuras;
    ArrayList<Corazon> corazones;
    int tiempo_sal;
    int tiempo_disparo;
```

```
	int pos_inicial;
	int separacion;
	int cantPiedras;
	int puntaje;
	int puntos_rama;
	int cont;
	int record;
	int vidas;
	int obtenerVida;
	Arbol a;
	java.awt.Image fondo;
	Image gameOver;
	Clip musica;

//	Clip musica;
	// Variables y métodos propios de cada grupo
	// ...

	Juego() {
			// Inicializa el objeto entorno
			this.entorno = new Entorno(this, "Selva Mono Capuchino - Grupo 6 - v1", 800, 600);
			// Inicializar lo que haga falta para el juego
			// ...
			fondo = Herramientas.cargarImagen("fondo2.gif");
			gameOver = Herramientas.cargarImagen("gameOver.png");
			tiempo_disparo = 0;
			tiempo_sal = 0;
			saco = new Saco();
			mono = new Mono();
			piso = new Suelo();
			basura = new Basura();
			a = new Arbol(1200, piso);
			arboles = new ArrayList<Arbol>();
			tigres = new ArrayList<Tigre>();
			basuras = new ArrayList<Basura>();
			corazones = new ArrayList<Corazon>();
			corazones.add(new Corazon(0));
			corazones.add(new Corazon(20));
			corazones.add(new Corazon(40));
			pos_inicial = 800;
			 separacion = 230;
			  cantPiedras = 0;
			vidas = 3;
			puntaje = 0;
			obtenerVida = 0;
			cont = 0;
			record = 0;


			// Inicia el juego!
			this.entorno.iniciar();

	}

	/**
	 * Durante el juego, el método tick() será ejecutado en cada instante y por lo
	 * tanto es el método más importante de esta clase. Aquí se debe actualizar el
	 * estado interno del juego para simular el paso del tiempo (ver el enunciado
	 * del TP para mayor detalle).
	 */
	public void tick() {
			entorno.dibujarImagen(fondo, 400, 240, 0, 1);
			entorno.cambiarFont("", 20, Color.white);
			entorno.escribirTexto("Puntaje:"+puntaje, 680, 70);
			entorno.escribirTexto("Piedras:"+cantPiedras, 680, 50);
			entorno.escribirTexto("Record: "+record, 680, 90);
```

```
//Crear corazones
if(obtenerVida >= 150 && vidas <= 3) {
	for(Corazon c : corazones) {
		if(c.lastimado == true) {
			c.lastimado = false;
			vidas += 1;
			obtenerVida = 0;
			break;
		}
	}
}

//dibujar corazones
if(corazones != null) {
	for(Corazon c : corazones) {
		c.dibujarse(entorno);

	}
}

//Crear arboles
if(arboles != null && arboles.size() != 1000) {
	arboles.add(new Arbol(pos_inicial += separacion, piso));
}

// Salto
if(mono != null && entorno.estaPresionada(entorno.TECLA_ARRIBA) && tiempo_sal < 15) {
	tiempo_sal += 1;
	mono.saltar();
}
// Caer
if(mono != null && mono.y < 435) {
	mono.caer(entorno);
}else {
	tiempo_sal = 0;
	cont = 0;
}

//Lanzar piedra
if(mono != null && entorno.sePresiono(entorno.TECLA_ESPACIO) && cantPiedras > 0) {
	mono.lanzar();
	cantPiedras -= 1;
}

//Dibujar piso
piso.dibujarse(entorno);

if(arboles != null) {
	//Recorro los arboles
	for(Arbol ar : arboles) {
		if(ar != null) {
			//Dibujar arboles
			ar.dibujarse(entorno);
			ar.moverse();
			//dibujar ramas
			if(ar.rama != null) {
				ar.rama.dibujarse(entorno);
				ar.rama.moverse();
				//dibujar frutas
				if(ar.rama.frutas != null) {
					for(Fruta f : ar.rama.frutas) {
						if(f!=null) {
							f.dibujarse(entorno);
							f.moverse();
						}
```

```
                                        }
                                }
                                //dibujar serpientes
                                if(ar.rama.serpiente != null) {
                                        ar.rama.serpiente.dibujarse(entorno);
                                        ar.rama.serpiente.moverse();
                                }
                        }
                        if(mono != null) {
                                //Que el mono pueda subirse a una rama y sume puntos por ello
                                if(ar.rama != null && mono.y < ar.rama.superficie &&
intersecan(mono.x, ar.rama.x, mono.y, ar.rama.y, mono.al, ar.rama.al, mono.an, ar.rama.an)) {
                                        mono.y = ar.rama.y - mono.al/2 - ar.rama.al/2;
                                        tiempo_sal = 0;
                                        cont += 1;
                                        if(cont == 1 || !ar.equals(a)) {
                                                puntaje += 5;
                                                obtenerVida += 5;
                                                a = ar;
                                        }
                                }
                                //Que el mono no atraviese la rama desde abajo
                                if(ar.rama != null && ar.rama != null &&
entorno.estaPresionada(entorno.TECLA_ARRIBA) && mono.cabeza > ar.rama.bottom && intersecan(mono.x, ar.rama.x,
mono.y, ar.rama.y, mono.al, ar.rama.al, mono.an, ar.rama.an)) {
                                        mono.y = ar.rama.y + mono.al/2 + ar.rama.al/2;
                                }
                                //Frutas
                                if(ar.rama != null && ar.rama.frutas != null) {
                                        for(Fruta f : ar.rama.frutas) {
                                                //Agarrar frutas
                                                if(f!=null && intersecan(mono.x, f.x,
mono.y, f.y, mono.al, f.al, mono.an, f.an)) {

        ar.rama.frutas.set(ar.rama.frutas.indexOf(f), null);
                                                        puntaje += 10;
                                                        obtenerVida += 10;
                                                }
                                                //Eliminar frutas fuera de la pantalla
                                                if(f!=null && f.x < 0-f.an/2) {

        ar.rama.frutas.set(ar.rama.frutas.indexOf(f), null);
                                                }
                                        }
                                }
                                //Matar serpientes
                                if(mono.piedras != null && ar.rama != null && ar.rama.serpiente
!= null) {
                                        for(int i = 0 ; i<mono.piedras.size() ; i++) {
                                                if(ar.rama.serpiente != null &&
intersecan(ar.rama.serpiente.x, mono.piedras.get(i).x, ar.rama.serpiente.y, mono.piedras.get(i).y, ar.rama.serpiente.al,
mono.piedras.get(i).al, ar.rama.serpiente.an, mono.piedras.get(i).an)) {
                                                        ar.rama.serpiente = null;
                                                        mono.piedras.remove(i);
                                                }
                                        }
                                }
                                //Que la serpiente mate al mono
                                if(ar.rama != null && ar.rama.serpiente != null &&
intersecan(mono.x, ar.rama.serpiente.x, mono.y, ar.rama.serpiente.y, mono.al, ar.rama.serpiente.al, mono.an,
ar.rama.serpiente.an)) {
                                        mono = null;
                                        puntaje = 0;
                                        obtenerVida = 0;
                                        for(int i = corazones.size()-1 ; i>-1 ; i--) {
                                                if(corazones.get(i).lastimado == false) {
```

```
                                                                                corazones.get(i).lastimado =
true;
                                                                                vidas -= 1;
                                                                                break;
                                                                        }
                                                                }
                                                        }
                                                }
                                                //Eliminar arboles fuera de pantalla
                                                if(ar.x < 0 - ar.an/2) {
                                                        arboles.set(arboles.indexOf(ar), null);
                                                }
                                        }
                                }
                        }
                        //Dibujar basura
                        if(basuras != null && (int) (Math.random()*400) == 50 && basuras.size() != 200) {
                                basuras.add(new Basura());
                        }
                        if(basuras != null) {
                                for(Basura b : basuras) {
                                        if(b != null) {
                                                b.moverse();
                                                b.dibujarse(entorno);

                                                if(mono != null) {
                                                        if(intersecan(mono.x, b.x, mono.y, b.y, mono.al, b.al, mono.an, b.an)){
                                                                basuras.set(basuras.indexOf(b), null);
                                                                puntaje -= 5;
                                                                obtenerVida -= 5;
                                                        }
                                                }
                                                if(b != null && b.x < 0 - b.an/2) {
                                                        basuras.set(basuras.indexOf(b), null);
                                                }
                                        }
                                }
                        }

                        //Crear tigres
                        if(tigres != null && (int) (Math.random()*250) == 100 && tigres.size() != 1000) {
                                        tigres.add(new Tigre());
                        }

                        if(tigres != null) {
                                for(Tigre t : tigres) {
                                        if(t != null) {
                                                //Dibujar tigre y moverlo
                                                t.dibujarse(entorno);
                                                t.moverse();

                                                for(Tigre t_ : tigres) {
                                                        //Separar los tigres para que no se superpongan
                                                        if(t_ != null && !t_.equals(t) && intersecan(t.x, t_.x, t.y, t_.y, t.al,
t_.al, t.an, t_.an)) {
                                                                t_.x += 20;
                                                        }
                                                }
                                                //Que el mono mate al tigre
                                                if(mono != null) {
                                                        if(mono.piedras != null) {
                                                                for(int i = 0; i<mono.piedras.size(); i++) {
                                                                        if(tigres.contains(t) && mono.piedras.get(i)
!= null && intersecan(mono.piedras.get(i).x, t.x, mono.piedras.get(i).y, t.y, mono.piedras.get(i).al, t.al,
mono.piedras.get(i).an, t.an)) {
                                                                                mono.piedras.remove(i);
```

```
                                                                                tigres.set(tigres.indexOf(t), null);
                                                                        }
                                                                }
                                                        }
                                                        //Que el tigre mate al mono
                                                        if(intersecan(mono.x, t.x, mono.y, t.y, mono.al, t.al, mono.an,
t.an)) {
                                                                mono = null;
                                                                puntaje = 0;
                                                                obtenerVida = 0;
                                                                for(int i = corazones.size()-1 ; i>-1 ; i--) {
                                                                        if(corazones.get(i).lastimado == false) {
                                                                                vidas -= 1;
                                                                                corazones.get(i).lastimado =
true;
                                                                                break;
                                                                        }
                                                                }

                                                        }
                                                }
                                                //eliminar tigre cuando sale de pantalla
                                                if(t.x < 0 - t.an/2 ) {
                                                        tigres.set(tigres.indexOf(t), null);
                                                }
                                        }
                                }
                        }

                        //Dibujar saco
                        if(saco != null) {
                                saco.dibujarse(entorno);
                                saco.moverse();
                                if(saco.x < 0 - saco.an/2) {
                                        saco = null;
                                }
                        }
                        //Pickear saco
                        if(saco != null && mono != null && intersecan(mono.x, saco.x, mono.y, saco.y, mono.al, saco.al, mono.an,
saco.an)) {
                                saco = null;
                                cantPiedras += 10;
                        }
                        //Generar saco
                        if(saco == null && cantPiedras <= 3  && (int) (Math.random()*450) == 50) {
                                saco = new Saco();
                        }
                        //dibujar piedras
                        if(mono != null && mono.piedras != null) {
                                for(Piedra p : mono.piedras) {
                                        if(p!= null) {
                                                p.dibujarse(entorno);
                                                p.moverse();
                                        }
                                }
                        }

                        //Dibujar mono
                        if(mono != null) {
                                mono.dibujarse(entorno);
                        }
                        //Respawn
                        if(mono == null && entorno.estaPresionada(entorno.TECLA_ESPACIO) && vidas > 0) {
                                mono = new Mono();
                        }
```

```
		//Eliminar piedras que salen del mapa
		if(mono != null) {
			for(int i = 0 ; i < mono.piedras.size() ; i++) {
				if(mono.piedras.get(i).x > 800) {
					mono.piedras.remove(i);
					}
				}
		}

		if(mono == null && puntaje > record) {
			record = puntaje;
		}

		if(arboles != null && tigres != null && basuras != null && arboles.get(arboles.size()-1) == null || vidas == 0){
			mono = null;
			arboles = null;
			tigres = null;
			piedras = null;
			basuras = null;
			entorno.cambiarFont("", 50, Color.white);
//			entorno.escribirTexto("R para volver a jugar", 170, 400);
			entorno.dibujarImagen(gameOver, 400, 200, 0, 1.5);
		}

	}
	//Interseccion de Bloques
	public boolean intersecan(double x1, double x2, double y1, double y2, int al1, int al2, int an1, int an2) {
		boolean intersecY = false;
		double distanciaY = Math.max(y1, y2) - Math.min(y1, y2);
		double sumaAlt = (al1/2) + (al2/2);
		if(distanciaY - sumaAlt < 0) {
			intersecY = true;
		}

		boolean intersecX = false;
		double distanciaX = Math.max(x1, x2) - Math.min(x1, x2);
		double sumaAnch = (an1/2) + (an2/2);
		if(distanciaX - sumaAnch < 0) {
			intersecX = true;
		}

		if(intersecX && intersecY) {
			return true;
		}else {
			return false;
		}

	}

	@SuppressWarnings("unused")
	public static void main(String[] args) {
		Juego juego = new Juego();
	}


}
```

Conclusiones:

Una de las dudas que más se generó en uno de los participantes fue el abordaje de tantos problemas para resolver, a medida que se fue resolviendo el trabajo y con el paso de los días, comprendió que era mejor separar el proyecto grande en pequeñas funcionalidades, que a medida que se iban implementando, creaban el juego interconectándose.

Es importante mantener las buenas prácticas y nombres claros para las variables o métodos, porque permite que alguien que no fue parte del trabajo, leerlo (siempre que tenga los conocimientos correspondientes claro está), así como también comentar las secciones para diferenciar lo que hace cada fragmento de código.

Si tuviésemos que trabajar nuevamente en un proyecto así, sin dudas, usaríamos la herramienta GitLab, esto porque al no usarlo, se complicó algo las implementaciones a archivos con versiones “anteriores” o funcionalidades que un integrante tenía y otro no.

Podemos decir que, se cumplió con el objetivo de desarrollar el juego propuesto por los docentes.