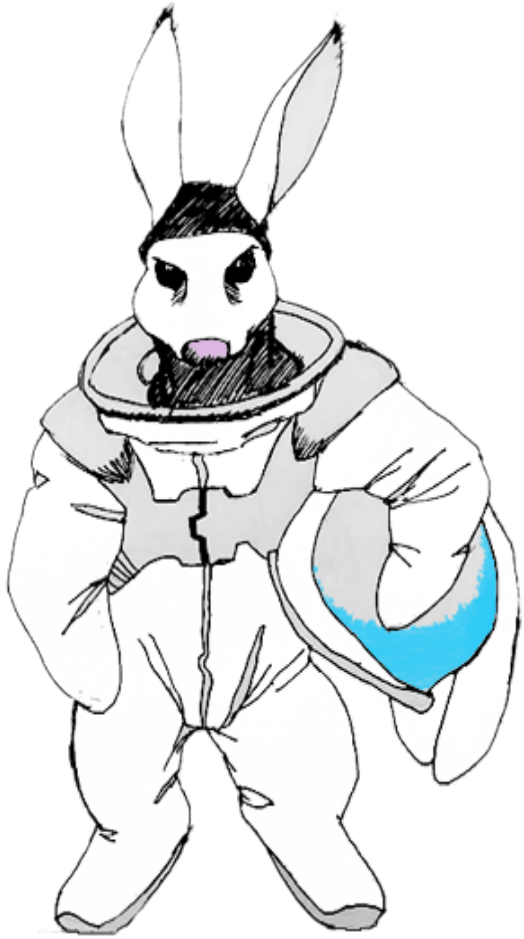
Boss Rabbits Rabber



Integrantes:

Luis Alberto Rosa - Leg: 30659937/2020 –

mail: rosa.luis.alberto@gmail.com

1.1.1.1

Alan Echabarri – Leg: 40093680/2019 –

mail: alanechabarri@gmail.com

Rodrigo Galarza – Leg: 38392571/2015 –

mail: nahuel_n.n@hotmail.com

Lugar de Elaboración: *Universidad Nacional General Sarmiento*

Fecha de Elaboración: *30 de noviembre de 2020*

Indice de contenidos

Introducción

En este programa se buscó diseñar (a través del lenguaje de programación "Java") un juego de características similares a los juegos de plataforma, en el cual el jugador toma el control de un conejo espacial que debe cruzar las diferentes avenidas sin ser atropellado por los diferentes vehículos que allí transitan. Otra dificultad a la que el jugador debe enfrentarse es que, nuestro personaje no debe tocar el borde inferior de la pantalla, y para agregar más dificultad, se implementó que a medida que pasa el tiempo, el borde inferior de la pantalla vaya avanzando.

El jugador, cuenta con las teclas direccionales (←, ↑, →) del teclado para moverse hacia la izquierda, adelante y derecha, y la tecla "espacio" para lanzar un kameha, que destruye a un vehículo con el que haga colisión. El kameha, se carga al realizar 10 pasos hacia adelante. Al estar cargado, se podrá visualizar sobre la interface ubicada en el borde de superior, un indicador que pardea en color amarillo indicando que el poder esta listo para lanzarse.

El juego, también cuenta con una tabla de puntajes en la parte superior en la que se muestran:

- **Tiempo** transcurrido entre el comienzo de la partida hasta la muerte del personaje
- **Saltos** realizados (solo se cuentan los saltos hacia adelante)
- **Kamehames impactados** en vehículos, los cuales sumarán 5 puntos adicionales

(se genera en esta misma tabla el indicador de Kamehame listo para lanzar)

- **Puntaje** acumulado hasta el momento

El puntaje irá aumentado a medida que transcurra el tiempo, otras formas de aumentar los puntos son:

realizando la mayor cantidad de saltos hacia adelante, por cada salto hacia adelante se suma 1 punto

destruyendo vehículos con el Kamehame, por cada vehículo destruido se suman 5 puntos.

La tabla de puntaje se puede visualizar al finalizar el juego a pantalla completa o durante el juego en la interface ubicada en la parte superior de la pantalla.

Se aclara que, al morir el personaje, todos los diferentes puntajes vuelven a "cero" y a medida que transcurre el tiempo la velocidad de los vehículos y el desplazamiento vertical de la pantalla se incrementa.

Importante: Para facilitar la lectura del documento, las clases mencionadas serán puestas en negritas y subrayadas, por ejemplo **Clase Ejemplo**, los métodos serán mencionados con negrita, por ejemplo **metodoEjemplo()**, y las variables serán mencionadas subrayadas, ejemplo variableEjemplo.

Descripción

-Clase Juego:

Es la clase principal en donde se ejecuta el juego en sí. Allí se encuentra el método **juego()**, en donde se llama a la **Clase Entorno**, el cual crea el entorno del juego. Además, se inicializan los objetos: “calle”, “calle2” y “calle3”, que hacen referencia a los segmentos de rutas y veredas por las cuales transitan el conejo y los autos; “conejo”, que crea a nuestro personaje; “i”, que refiere a la clase interface (tabla de puntos, estados, etc.); y, por último, los objetos “transito derecho” y “tránsito izquierdo”, que crean los diferentes vehículos de las diferentes manos y calles respectivamente. Y también, se inician el entorno y la música del juego en modo loop.

Métodos generales:

- **Juego()**, en donde se llama a la **Clase Entorno**, el cual crea el entorno del juego.

Métodos generales en función del tiempo:

- **tick()**, es el método más importante del juego, ya que, se ejecuta a cada instante y es el encargado de actualizar el estado interno del juego para simular el paso del tiempo e ir redibujando en pantalla los distintos elementos y sus cambios. Allí, se dibujan las calles; los vehículos (ya sean, los que figuran al comenzar la partida, los que colisionan con los bordes y reaparecen más arriba o del borde opuesto, o los que son destruidos por el kameha); el conejo, el cual, además de dibujarse se redibuja si se hacen diferentes acciones, como saltar adelante o hacia los lados al apretar las diferentes teclas, el movimiento constante hacia abajo por el paso del tiempo, o si muere por colisionar con un vehículo o el borde inferior. Como el conejo también están los objetos “kameha”, que se dibuja si se presiona la barra espaciadora avanzando hacia arriba, y “i” que se redibuja constantemente actualizando los puntajes, cantidad de saltos, tiempo, etc. “i”, también se encarga de mostrar los mensajes si se gana o pierde y la tabla final de puntuaciones.También en este método se define la variable int aceleración que toma el valor del tiempo actual, utilizando su geter (*getTiempo())* el cual va ser pasado como parámetro en otros métodos de desplazamientos, como por ejemplo *bajar().*

- **main();** es en donde se llama al método “Juego” para que se ejecute.

Observaciones:

- Originalmente se utilizaba el audio completo de la cancion Tarzan Boy, de Baltimora, asi como multiples elementos graficos sin optimizar. Dichos archivos fueron revisados y adecuados a los parametros reales de resolucion, y en el caso del audio, se genero un loop con un segmento de la cancion, con bajo bitrate. Con estas medidas, se logro reducir el tamaño total del proyecto a menos de un sexto del espacio utilizado originalmente.

-Clase Conejo:

Es la clase encargada de moldear a nuestro personaje, definiendo las variables: “alto”, “ancho”, “x” (posición en x del objeto), “y” (posición en y del objeto), conejito (animación del personaje) y estaVivo (estado de salud del personaje). Se emplean los “Getters” de las variables: alto, ancho, “x”, “y” y estaVivo

Métodos generales:

- **dibujar(Entorno entorno);** encargado de dibujar la imagen del personaje;
- **muere()**; que modifica a la variable estaVivo a falso
- **revivir(double x, double y)**; encargado de modificar de nuevo estaVivo a verdadero y ubicar al conejo en la posición inicial;
- **lanzar()**, se encarga de la creación de un objeto tipo “Kamehameha” sobre la posición del conejo
- **colisiónInf()**, que verifica si el conejo choca con el borde inferior
- **colisiónConVehiculo(Vehículo[]a)** que recibe un array de Vehículo y retorna “true” si el personaje choca con algún vehículo; y *colisionConBordes*(Entorno entorno), que no permite que el conejo traspase los bordes laterales ni el superior.

Métodos de desplazamiento:

- **bajar(int a)**, el cual hace que el personaje descienda sobre el eje “y” en cada instante de tiempo. Dicho desplazamiento es sincronizado al resto de elementos del juego que deben moverse de manera sincronizadamente gracias a la variable compartida aceleración.
- **saltarArriba()/Derecha()/Izquierda()**, hace que el personaje avance una cantidad determinada de espacio (el salto está definido en 50 píxeles al ser este el tamaño del sprite del conejo). Se implementó también sobre saltarArriba() una restricción que verifica que haya espacio disponible en la parte superior, para evitar que se vaya de pantalla o se desfase del scrolling del resto de elementos.

Observaciones:

- Quizás una dificultad de esta clase fue pensar el método del que el conejo colisiona con el vehículo, pero implementando lo aprendido con el código del pong pudimos superar ese desafío. Por otra parte, originalmente, el tamaño del conejo había sido definido en 40x40px, siendo definido posteriormente en 50x50px, con un hitbox de 48x48, para eliminar cualquier desfase minimo que pudiera haber al implementarse los desplazamientos sincronizados con el resto de elementos del juego. No se verifico ningún inconveniente asociado a colisiones luego de definir estos valores tal lo mencionado.

-Clase Kamehameha:

En esta clase se definen las diferentes variables para el objeto “kamehameha”, utilizado en el juego para destruir coches. Las variables que se definen son: “x” e “y” (la posición del objeto las coordenadas x e y), tamano (tamaño del objeto) y bolaDeFuego (imagen del objeto).

Se implementan los métodos:

- **colisionConCoche**(Vehículo[]a) toma un array de Vehículos y retorna el elemento de Vehículo que fue impactado por el kameha.
- **dibujar** (Entorno entorno) dibuja la imagen kameha(bolaDeFuego) en el entorno en un punto x, y.
- **desplazarArriba**() hace que el objeto se desplace hacia el borde de arriba -3 px.

Observaciones:

- Con esta clase tuvimos principalmente el problema de que se dibujaba como una ráfaga, y no como una esfera que se desplaza. En un principio lo desplazábamos con un ciclo while que vaya aumentando su posición de "y" pero aunando en el asunto y consultando con la profesora pudimos ver que la forma correcta era que se dibujara y desplazará en el tick, de manera similar a los vehículos.
- Otra dificultad fue cuando la kameha colisionaba con el vehículo y lo volvíamos, null que lo explicaremos en el apartado de vehículo.

-Clase FondoSegmentado:

Es la encargada de manejar el fondo de pantalla del juego. Al ser el mismo de scroll vertical constante, se deben redibujar los segmentos que quedan fuera de los 800x600 píxeles definidos en el entorno en la parte superior nuevamente para que vuelvan a scrollear hacia abajo. A tal efecto se definen las variables x e y (la posición del objeto las coordenadas x e y), segmento1 (imagen de la ruta y la vereda). También cuenta con Getters de las variables “x” y “y”, además de los métodos.

Metodos generales:

- **dibujarabc(Entorno entorno)**, dibuja el segmento1 en las coordenadas “x” e “y” al comienzo del juego.

Métodos de desplazamiento:

- **bajar(int a)**, que permite que los segmentos se mueven +”a” px hacia abajo en relación con la pantalla
- **redibujar()**, condiciona a que el “y” de los segmentos no superen los 750 px en y, en caso de hacerlo, redibujara los mismos en la parte superior de la pantalla para permitir re scrolling.

Observaciones:

Al ser un juego de scrolling vertical constante, si se utilizara un fondo entero, debería ser extremadamente alto, para poder mantenerse en movimiento descendente durante el transcurso de todo el juego. Dicha forma de encarar el problema, derivaría en un uso excesivo de memoria, así como en archivos de gran tamaño, volviéndolo pesado más allá de valores viables. Por esta razón, solo se utilizan 3 segmentos pequeños que son redibujados. Al mismo tiempo, esta manera de encararlo, permite escalabilidad futura, ya que se podrían dibujar alternadamente segmentos diferentes, dándole mayor variedad al juego sin incrementar notoriamente el peso del mismo.

-Clase Vehículo:

La clase "Vehículo" representa el vehículo y gestiona las propiedades del mismo dentro del juego. Se utilizan las variables:

- "alto" y "ancho":para dar el tamaño del objeto
- "x"e "y" :la posición del objeto las coordenadas x e y,
- autoDer : imagen de los autos que avanzan hacia la derecha
- autoIzq : imagen de autos que avanzan hacia la izquierda.

También, se utilizan los Getters de las variables: alto, ancho, "x" y "y", y los métodos:

Métodos generales:

- **dibujarAutosDerecha**(Entorno entorno) y **dibujarAutosIzquierda**(Entorno entorno): dibujan a los autos de mano derecha e izquierda respectivamente.

Métodos de desplazamiento:

- **desplazarIzqVelocidad**(double velocidad) y **desplazarDerVelocidad**(double velocidad):

aumenta la velocidad de los autos de la respectiva mano dependiendo el double ingresado.

- **colisionConBordesIzquierdo**(Entorno entorno, Vehiculo[]a)

  y
- **colisionConBordesDerecho**(Entorno entorno, Vehículo[]a): reciben una variable tipo Entorno y un array de Vehículo, dependiendo para qué lado se dirijan los vehículos recibidos van a reaparecer del lado opuesto de la pantalla.
- **colisionConBordeInferior**(Entorno entorno, Vehiculo[]a) : si el vehículo recibido supera el borde inferior +25px, hace que el vehículo aparezca en el borde superior dando la sensación de un loop de autos infinitos;
- **bajar**(int a): hace que los vehículos desciendan en la pantalla dependiendo el parámetro recibido.

Observaciones:

- Esta fue una de las clases más trabajadas, ya que al querer implementar el método *colisiónConBorde()*, primero debíamos hacer que el método corresponda a una dirección del tránsito (izquierda, derecha), el cual lo resolvimos creando dos métodos diferentes para cada dirección de Tránsito. El siguiente problema con el que nos topamos, fue que, al reaparecer en el borde de la pantalla, los vehículos se superponían, pero pensándolo entre los tres y con ayuda de la profesora, resolvimos que lo ideal sería que el método se fije primero en el arreglo de vehículos, si hay otro coche adelante suyo, y dependiendo de la distancia, que el coche al que se le usa el método, aparezca más o menos cerca (pero fuera pantalla) del borde correspondiente.
- Otra dificultad fue cuando, los elementos de Vehiculo se volvían null al colisionar con kameha. Para poder resolver este problema, condicionamos la creación de los vehículos, para que luego de crearlo se fije si algún elemento fuera null y cree otro objeto en la misma posición del objeto anterior en la coordenada "y", y alguna posición de "x".

-Clase Interface:

La última clase que se implementó fue la de Interface, que se encarga básicamente de gestionar y mostrar en pantalla la tabla de puntajes y las imágenes de Win, Game Over, y el indicador de que el Kamehame esta listo para utilizarse. Para ello se utilizaron las siguientes variables:

- kamehamehaImpactado :cantidad de kamehas colisionado con vehículos
- puntos :cantidad de puntos acumulado
- cantSaltos :saltos realizados hacia adelante
- tiempo :tiempo transcurrido desde que se empieza la partida
- puntajes :interface gráfica que muestra los stats durante la partida
- cargado : gif que muestra cuando el kameha está listo para utilizarse
- win:imagen que se muestra al ganar
- gameOver:imagen que se muestra al perder el juego
- tablaPuntajeFinal :imagen de la tabla de puntos al finalizar una partida
- estaTablaDePuntaje:boolean que indica que si se llamó a tablaPuntajeFinal

- energiaDeKamehameha :indica la carga del kameha

- seTerminoElJuego:boolean que indica si se acaba el juego

- BotonOnOff :sirve para verificar si el juego termina

Además se usaron Getters para las variables estaTablaDePuntaje, puntos, tiempo, seTerminoElJuego y BotonOnOff.

En cuanto a los métodos utilizados son:

- **mostrarPantalla**(Entorno entorno), que dibuja las variables, puntajes, kamehamehaImpactado, cantSaltos, tiempo y puntos en pantalla y les da un estilo de fuente.

- **kamehameListo**() que permite consultar la energía cargada para utilizar el kameha.

- **cargado**(Entorno entorno) que dibuja la variable "cargado" cuando el kameha está listo para utilizarse.

- **confirmarImpacto**()confirma la colisión del kameha con un auto.

- **agotarEnergia**() resetea a 0 "energiaDeKamehameha" una vez lanzado.

- **aumentarSaltos**() aumenta el contador de saltos.

- **aumentarEnergia**() aumenta en +1 a "energiaDeKamehameha".

- **aumentarTiempo**() aumenta la variable "tiempo" en +1.

- **aumentarPuntos**() calcula la sumatoria de puntos.

- **resetearPuntajes**()resetea los diferentes stats a 0.

- **gameOver**(Entorno entorno)básicamente dibuja el cartel de Game Over cuando se pierde.

- **dibujarGanar**(Entorno entorno) que dibuje you win si se gana.

- **dibujarTablaDePuntaje**(Entorno entorno) que dibuja la tabla de puntaje final.

- **mostrarTablaDePuntaje**(), muestra la tabla de puntaje.

- **ocultarTablaDePuntaje**(), oculta la tabla de puntaje.

- **Prender**(), **Apagar**(), **SeTerminoElJuegoOn**() y **SeTerminoElJuegoOff**(): son métodos que se implementaron en último momento para verificar que el juego haya terminado y se puedan dibujar los diferentes carteles sin que el juego continúe de fondo.

Observaciones:

- Durante el desarrollo, se verifico que los valores asociados no debian solamente acumularese, si no tambien generar “estados”, por lo que se fueron definiendo diferentes booleanos para determinar por ejemplo, si ya fueron mostradas las pantallas de game over y si el juego efectivamente concluyo, para dejar de contar el tiempo y mostrar el puntaje final, asi como disparar diversos eventos asociados.

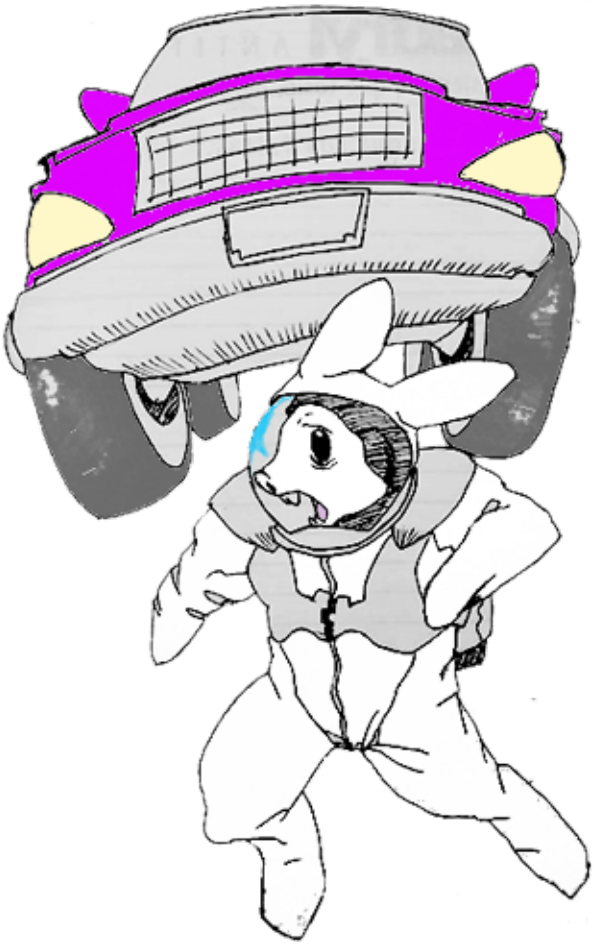
Conclusión

En conclusión, a pesar de que con un poco más de tiempo podríamos haber implementado más elementos para hacer más completo al juego, creemos que este trabajo nos sirvió mucho como simulacro para aplicar los conocimientos adquiridos durante la cursada, también a cómo trabajar en grupo para desarrollar el mismo. Además, aprendimos a pensar los objetos como objetos de la vida real y no tanto como un programa sin correlacion con elementos fisicos.

Individualmente, nos complementamos muy bien. A pesar de que se trabajó el código por separado, en ningún momento ninguno pisó a otro en cuanto al trabajo que veníamos realizando, es más, cuando surgieron problemas organizamos reuniones y lo debatimos entre todos hasta resolverlos, y si no podíamos resolverlo, se lo consultamos a la profesora, asi como tambien consultamos en sitios online o intentamos solventar por nuestros medios las dificultades que aparecieron.

Incluso en los elementos que no se llegaron a desarrolar en todo su potencial, se logro un enfoque en el desarrollo del juego que permitiria facilmente agregarlos a futuro de ser necesario, teniendo en cuenta cuestiones como la posibilidad de desarrollo futuro sobre el codigo basico trabajado.

Más allá de lo dicho en el primer renglón de este apartado, nos sentimos muy satisfechos por lo logrado en este proyecto y las enseñanzas obtenidas de las diferentes dificultades ya descritas en el apartado Descripción.



Implementacion

Se adjunta la totalidad del codigo implementado, para facilitar la lectura del mismo, las anotaciones se encuentran en color azul (ej. "`// El objeto Entorno que controla el tiempo y otros`"), y se utiliza el remarcado en azul para separar las clases en segmentos menores (ej "`\\variables de instancia`").

Clase Juego

```
package juego;

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;

	private Conejo conejo;			//personaje

	private Interface i;			//interface del juego

	private Kamehameha kamehameha;	//poder que lanza el personaje

	private Vehiculo [] transitoDerecho;	//autos que avanzan hacia la derecha

	private Vehiculo [] transitoIzquierdo;	//autos que avanzan hacia la izquierda

	private int cantVehiculosPorCalle =7;	//cant vehículos en las calles

	private FondoSegmentado calle;			//segmento de calle

	private FondoSegmentado calle2;	//segmento de calle

	private FondoSegmentado calle3;	//segmento de calle

	public Juego() {

		// Inicializa el objeto entorno

		this.entorno = new Entorno(this, "Boss Rabbit Rabber", 800, 600);


		// Inicializar lo que haga falta para el juego

		calle=new FondoSegmentado(400,250);	//calles

		calle2=new FondoSegmentado(400,550);	//calles

		calle3=new FondoSegmentado(400,-50);	//calles

		conejo = new Conejo (entorno.ancho()/2,entorno.alto()-225,50,50);  //personaje

		i= new Interface (1);	//interface grafica
```

```
//-----*Creacion de vehiculos*-----------------------------

        //Autos q van para la derecha --------------->>
        transitoDerecho = new Vehiculo[cantVehiculosPorCalle];
        double xDerecho = entorno.ancho() - 750;
        double yDerecho = entorno.alto()/2+275.0; //1er auto se crea en el carril 1
        for(int i=0; i < transitoDerecho.length; i++) {
                if(i == 1 || i == 4 || i == 7) {
                        transitoDerecho[i]= new Vehiculo(xDerecho, yDerecho);
                        xDerecho += 100.0;
                        yDerecho =entorno.alto()/2 + 125; // Carril 4 de abajo para arriba 325 en y >
                }else if(i == 2 || i == 5 || i == 8) {
                        transitoDerecho[i]= new Vehiculo(xDerecho, yDerecho);
                        xDerecho += 150.0;
                        yDerecho = entorno.alto()/2-175.0;  //Carril 8 de abajo para arriba 125 en y >
                }else if(i == 3 || i == 6 || i == 9){
                        transitoDerecho[i]= new Vehiculo(xDerecho,yDerecho);
                        xDerecho += 100.0;
                        yDerecho = entorno.alto()/2-125.0; // Carril 7 de abajo para arriba 175 en y >
                }else{
                        transitoDerecho[i]= new Vehiculo(xDerecho, yDerecho);
                        xDerecho += 150.0;
                        yDerecho = entorno.alto()/2+275.0; //Carril 1 de abajo para arriba 475 en y >
                }
        }
        //Autos q van para la izquierda  <<----------
        transitoIzquierdo = new Vehiculo[cantVehiculosPorCalle];
        double xIzquierdo = entorno.ancho() - 50;
        double yIzquierdo = entorno.alto()/2+175.0; //1er auto crea en el carril 8
        for(int i=0; i < transitoIzquierdo.length; i++) {
                if(i == 1 || i == 4 || i == 7) {
                        transitoIzquierdo[i]= new Vehiculo(xIzquierdo, yIzquierdo);
                        xIzquierdo += 100.0;
                        yIzquierdo =entorno.alto()/2-25.0;//Carril 5 de abajo para arriba   275 en y <
```

```
            }else if(i == 2 || i == 5 || i == 8) {
                  transitoIzquierdo[i]= new Vehiculo(xIzquierdo, yIzquierdo);
                  xIzquierdo += 150.0;
                  yIzquierdo = entorno.alto()/2-75.0;//Carril 6 de abajo para arriba 225 en y <
            }else if(i == 3 || i == 6 || i == 9){
                  transitoIzquierdo[i]= new Vehiculo(xIzquierdo, yIzquierdo);
                  xIzquierdo += 100.0;
                  yIzquierdo = entorno.alto()/2+225.0;//Carril 2 de abajo para arriba 425 en y <

            }else{
                  transitoIzquierdo[i]= new Vehiculo(xIzquierdo, yIzquierdo);
                  xIzquierdo += 150.0;
                  yIzquierdo = entorno.alto()/2+175.0;//Carril 3 de abajo para arriba 375 en y <

            }
      }
//-----------------------------------------------------------------
      // Inicia el juego!

      this.entorno.iniciar();
      Herramientas.loop("tarzan_boy.wav"); //cancion de juego
}
/*
 * 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() {
      // Procesamiento de un instante de tiempo
      //Dibujar en pantalla y scroll de calle
      int aceleracion=i.getTiempo(); // la aceleracion se va incrementado al pasar del tiempo
      calle.dibujarabc(entorno);
      calle2.dibujarabc(entorno);
```

```
		calle3.dibujarabc(entorno);
		calle.bajar((aceleracion/2100000000)+1);
		calle2.bajar((aceleracion/2100000000)+1);
		calle3.bajar((aceleracion/2100000000)+1);
		calle.redibujar();
		calle2.redibujar();
		calle3.redibujar();
		//-----------------------------------------------------------


		//----------*Dibujar Propiedades de los Vehiculos*----------------------------


		//----------->>>---Autos que se desplazan hacia la derecha-------->>>
		for(int i=0; i < cantVehiculosPorCalle; i++) {
			if (transitoDerecho[i] != null) {
				transitoDerecho[i].dibujarAutosDerecha(entorno);


				if (i == 1 || i == 4 || i == 7) {
					transitoDerecho[i].desplazarDerVelocidad((aceleracion/120000000)+1);
				} else if (i == 2 || i == 5 || i == 8) {
					transitoDerecho[i].desplazarDerVelocidad((aceleracion/120000000)+3);
				} else if (i == 3 || i == 6 || i == 9) {
					transitoDerecho[i].desplazarDerVelocidad((aceleracion/120000000)+4);
				} else {
					transitoDerecho[i].desplazarDerVelocidad((aceleracion/120000000)+1);
				}
				// transitoDerecho[i].desplazarDerecha();
				transitoDerecho[i].colicionConBordesDerecho(entorno, transitoDerecho);
				transitoDerecho[i].bajar((aceleracion/120000000)+1);
				transitoDerecho[i].colicionConBordeInferior(entorno, transitoDerecho);
			}else {
			    if(i>4) {
			    transitoDerecho[i]=new Vehiculo(-300,transitoDerecho[i-3].getY());
//si se rompio el coche crea otro fijándose en la posicion en y del coche anterior del mismo carril
			    }else if(i==1||i==2||i==3){
```

```
                        transitoDerecho[i]=new Vehiculo(-300,transitoDerecho[i+3].getY());
                    }else{
                    transitoDerecho[i]=new Vehiculo(-300,transitoDerecho[3].getY()+50);
//si la posicion es 0 toma la posicion de y
                    //del transito 3,+50
                    }
                }
            }
                //<<<-----Autos que se desplazan hacia la izquierda---<<<--------
            for (int i=0; i < cantVehiculosPorCalle; i++) {
                if(transitoIzquierdo[i]!=null) {
                    transitoIzquierdo[i].dibujarAutosIzquierda(entorno);

                            if (i == 1 || i == 4 || i == 7) {
      transitoIzquierdo[i].desplazarIzqVelocidad((aceleracion/120000000)+1);
                            } else if (i == 2 || i == 5 || i == 8) {
      transitoIzquierdo[i].desplazarIzqVelocidad((aceleracion/120000000)+3);
                            } else if (i == 3 || i == 6 || i == 9) {
      transitoIzquierdo[i].desplazarIzqVelocidad((aceleracion/120000000)+4);
                            } else {
      transitoIzquierdo[i].desplazarIzqVelocidad((aceleracion/120000000)+1);
                            }

                    //transitoIzquierdo[i].desplazarIzquierda();
                    transitoIzquierdo[i].colicionConBordesIzquierdo(entorno,transitoIzquierdo);
                    transitoIzquierdo[i].bajar((aceleracion/120000000)+1);
                    transitoIzquierdo[i].colicionConBordeInferior(entorno, transitoIzquierdo);
                } else {
                    if (i > 4) {
                            transitoIzquierdo[i] = new Vehiculo(entorno.ancho()+300,
transitoIzquierdo[i - 3].getY());
                        } else if (i == 1 || i == 2 || i == 3) {
```

```
                                transitoIzquierdo[i] = new Vehiculo(entorno.ancho()+300,
transitoIzquierdo[i + 3].getY());
                            } else {
                                transitoIzquierdo[i] = new Vehiculo(entorno.ancho()+300,
transitoIzquierdo[3].getY() + 50);
                            }
                        }
                }

        if(conejo.getEstaVivo() && !i.getSeTerminoElJuego()) {
                        //---------*Dibujar Propiedades del conejo*------------------------------
                        {
                        conejo.dibujar(entorno);
                        conejo.colicionConBordes(entorno);
                        conejo.bajar((aceleracion/120000000)+1);

                        // Dibujar interface del juego (puntos, tiempo, etc) se debe ejecutar al final para
no ser tapado por el resto
                        i.mostrarEnPantalla(entorno);
                        i.aumentarTiempo();
                        i.aumentarPuntos();
                        }
                        //movimientos y acciones del conejo
        if(entorno.sePresiono(entorno.TECLA_ARRIBA) && conejo.getY() >=100 &&!i.getSeTerminoElJuego()) {
//el && verifica casilla libre arriba
                            conejo.saltarArriba();
                            i.aumentarEnergia();
                            i.aumentarSaltos();
        }
                        if(entorno.sePresiono(entorno.TECLA_DERECHA) && !i.getSeTerminoElJuego()) {
//el conejo se desplaza a la derecha
                            conejo.saltarDerecha();
                        }
                        if(entorno.sePresiono(entorno.TECLA_IZQUIERDA) && !i.getSeTerminoElJuego()) {
 //el conejo se desplaza a la izquierda
```

```
                        conejo.saltarIzquierda();

                    }

        //lanzar kameha si se presiona la tecla ESPACIO

        if(entorno.sePresiono(entorno.TECLA_ESPACIO) && i.kamehameListo() >= 10 && !
i.getSeTerminoElJuego()) {

                        kamehameha= conejo.lanzar();

                        i.agotarEnergia();

            }

                    if(i.kamehameListo()>= 10) {

                        i.cargado(entorno);

                    }

                    //si el kameha es creado, se dibuja y se lanza hacia arriba

                    if(kamehameha != null) {

                        kamehameha.desplazarArriba();

                        kamehameha.dibujar(entorno);

                    for(int e=0;e<transitoIzquierdo.length;e++){

                        if(kamehameha!=null && transitoIzquierdo[e]!=null &&
kamehameha.colicionConCoche(transitoIzquierdo) == e) {

                                kamehameha=null;

                                transitoIzquierdo[e] = null;

                                i.confirmarImpacto();//se verifica si el kameha impacta con algun
autoIzquierdo y lo destruye

                    }

                    }

            for(int e=0;e<transitoDerecho.length;e++){

             if(kamehameha!=null && transitoDerecho[e]!=null &&
kamehameha.colicionConCoche(transitoDerecho) == e) {

                            kamehameha=null;

                            transitoDerecho[e] = null;

                    i.confirmarImpacto();    //se verifica si el kameha impacta con algún  autoDerecho
y lo destruye

                     }

                    }

            }

            //--------------Muerte del conejo, si choca con el piso o lo choca un
auto----------------------
```

```
if(conejo.colicionConVehiculo(transitoDerecho) ||conejo.colicionInf() ||
conejo.colicionConVehiculo(transitoIzquierdo))

              {

              conejo.muere();

              }

       }

              //mientras el conejo no este vivo...

                     //---------------Game Over--------------------------------------


              if(!conejo.getEstaVivo() && !i.getEstaTablaDePuntaje() &&!i.getSeTerminoElJuego()) {

// se fija si el conejo no esta vivo y si la tabla de puntajes no esta activada

                           i.gameOver(entorno);

              }

       //----------------reaparicion del conejo despues del game over

                     if(entorno.sePresiono(entorno.TECLA_ENTER)&&i.getEstaTablaDePuntaje()){

                       //revivir conejo despues de la tabla de puntaje

                           i.resetearPuntajes();

                           if(calle.getY()<=300 &&calle.getY()>=0 ) { //alinear con vereda 1

                                  conejo.revivir(calle.getX(),calle.getY()+125);

                                  i.Apagar();

                           }else {

                                  if(calle2.getY()<=300 &&calle2.getY()>=0 ) {

                                                                        //alinear con vereda 2

                                        conejo.revivir(calle2.getX(),calle2.getY()+125);;

                                        i.Apagar();

                                        }else if(calle3.getY()<=300 &&calle3.getY()>=0 ){
                                                                              //alinear con vereda 3

                                               conejo.revivir(calle3.getX(),calle3.getY()+125);

                                               i.Apagar();

                                  }

                           }

                           i.ocultarTablaDePuntaje();

                     }

              //-------------------Tabla De puntajes final-------------------------
```

```
                    if(entorno.sePresiono(entorno.TECLA_ENTER)
                                        && !conejo.getEstaVivo() && !i.getSeTerminoElJuego()) {
//si se presiona enter y el conejo esta muerto tablaDePuntaje == true
                            i.mostrarTablaDePuntaje();
                            i.SeTerminoElJuegoOff();
                    }
                    if(i.getEstaTablaDePuntaje()) {
                            i.dibujarTablaDePuntaje(entorno);
                    }
                    if(entorno.sePresiono(entorno.TECLA_ENTER)
                                        && i.getSeTerminoElJuego()) {
//si se presiona enter y el conejo esta muerto tablaDePuntaje == true
                        i.mostrarTablaDePuntaje();
                        i.Prender();
                        i.SeTerminoElJuegoOff();
                    }
            //------------ You Win si el conejo si se llegan a los 1000 puntos
                    if(i.getPuntos() >= 1000 && !i.getBotonOnOff()) {
                            i.dibujarGano(entorno);
                            i.SeTerminoElJuegoOn();
                    }
        }

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

Clase Conejo

```
package juego;

import java.awt.Image;
```

```java
import entorno.Entorno;
import entorno.Herramientas;

public class Conejo {
	\\Variables de instancia
	 private double alto ;
	 private double ancho;
	 private double x;
	 private double y;
	 private Image conejito;
	 private boolean estaVivo;
	 \\getters
	public double getAlto() {
		return alto;
	}
	public double getAncho() {
		return ancho;
	}
	public double getX() {
		return x;
	}
	public double getY() {
		return y;
	}
	public boolean getEstaVivo() {
		return estaVivo;
	}

	public Conejo(double x, double y, double alto,double ancho ) {
		 this.x = x;
		 this.y = y;
		 this.alto = alto;
		 this.ancho = ancho;
```

```
        this.conejito=Herramientas.cargarImagen("conejete.gif");\\carga de imagen
        this.estaVivo = true;
 }

 public void dibujar(Entorno entorno) {
              entorno.dibujarImagen(conejito, x, y, 0,1);
}
  public void bajar(int a) { //Gravedad del conejo
          this.y+= a;
}
  public void saltarArriba(){
        this.y=y-this.alto;
}
  public void saltarIzquierda(){
        this.x=x-this.ancho;
}
  public void saltarDerecha(){
        this.x=x+this.ancho;
 }

  public void muere() { // modifica el estado del conejo si muere
        estaVivo = false;
}
  public void revivir(double x, double y) { //revive al conejo
        estaVivo = true;
        this.x=x;
        this.y=y;
  }



 public Kamehameha lanzar(){ // Crea el kameha
       Kamehameha bolita= new Kamehameha (this.x, this.y);
       return bolita;
```

```
    }

    public boolean colicionInf() {//colicion del conejo con el borde de abajo
            if(this.y>600)
            {
                    return true;
            }
          return false;
    }
                \\colision con vehículos, revisa si el conejo choca con un vehículo del arreglo y devuelve true, de lo contrario false.
     public boolean colicionConVehiculo(Vehiculo[] a) {
            for(int c=0;c<a.length;c++) {
                if(a[c]==null) {
                   return false;
                }
                if(this.x+this.ancho/2-1>a[c].getX()-a[c].getAncho()/2
                   &&this. -this.ancho/2+1<a[c].getX()+a[c].getAncho()/2

                   && this.y + this.alto/2 -2> a[c].getY() - a[c].getAlto()/2
                   &&this.y - this.alto/2 +2< a[c].getY() + a[c].getAlto()/2) {
                       return true;
                  }
            }
               return false;
          }

\\colision con bordes, no permite que el conejo pase del borde.
     public void colicionConBordes(Entorno entorno) {

            if(x > entorno.ancho()-25) { //borde derecho
                    x -= 25;
```

```
                }
                if(x < 25) { // borde izquierdo
                        x +=25;
                }
                if(y<25) { // borde de arriba
                        y =25;
                }
        }
}
```

Clase FondoSegmentado

```
package juego;

import java.awt.Image;
import entorno.Entorno;
import entorno.Herramientas;

public class FondoSegmentado { // clase para el fondo de pantalla
        private double x;
        private double y;
    private Image segmento1;
    //private Image segmento2;

        public double getX() {
                return x;
        }

        public double getY() {
                return y;
        }

        public FondoSegmentado(double x, double y) {
```

```
            this.y = y;
            this.x = x;
\\carga de imagen    this.segmento1=Herramientas.cargarImagen("segmento1.gif");
      }
\\se dibuja en el entorno
      public void dibujarabc(Entorno entorno) {
            entorno.dibujarImagen(segmento1, x, y, 0,1);
      }

      /*public void dibujar2(Entorno entorno) {

            entorno.dibujarImagen(segmento2, x, y, 0,1);
      }*/
      \\se desplaza hacia abajo
      public void bajar(int a ) {
            this.y+= a;
      }
      public void redibujar() {
            if(this.y>=750)
            {
                  this.y=-150;
            }
      }
}
```

Clase Interface

```
package juego;
import java.awt.Color;
import java.awt.Image;
import entorno.Entorno;
import entorno.Herramientas;
public class Interface {
```

```
\\cvariables de instancia
	private int kamehamehaImpactado;
	private int puntos; //variable de la cant de puntos
	private int cantSaltos;
	private int tiempo;
	private Image puntajes;// interface de puntaje
	private Image cargado;
	private Image win; // imagen de gano
	private Image gameOver;
	private Image tablaPuntajeFinal;
	private boolean estaTablaDePuntaje;
	private int energiaDeKamehameha;
	private boolean seTerminoElJuego;
	private boolean BotonOnOff; // boton que sirve para verificar si el juego termina
	\\getters:
	public int getTiempo() {
		return tiempo;
	}
	public boolean getEstaTablaDePuntaje() {
		return estaTablaDePuntaje;
	}
	public int getPuntos() {
		return puntos;
	}
	public boolean getSeTerminoElJuego() {
		  return seTerminoElJuego;
	}
	public boolean getBotonOnOff(){ // boton que sirve para verificar si el juego termina
		return this.BotonOnOff;
	}
	public void Prender() { // boton que sirve para verificar si el juego termina
		this.BotonOnOff = true;
	}
	public void Apagar() { // boton que sirve para verificar si el juego termina
```

```
        this.BotonOnOff = false;
    }
    public void SeTerminoElJuegoOn() { // boton que sirve para verificar si el juego termina
         this.seTerminoElJuego = true;
    }
    public void SeTerminoElJuegoOff() { // boton que sirve para verificar si el juego termina
        this.seTerminoElJuego = false;
    }
    public Interface(int energiaDeKamehameha) {
        this.kamehamehaImpactado = 0;
        this.cantSaltos = 0;
        this.tiempo = 0;
        this.puntos = 0;
\\carga de imagenes:        this.puntajes=Herramientas.cargarImagen("puntajes.png");
        this.cargado=Herramientas.cargarImagen("cargado.gif");
        this.energiaDeKamehameha=energiaDeKamehameha;
        this.win = Herramientas.cargarImagen("win.png");
        this.gameOver = Herramientas.cargarImagen("gameOver.png");
        this.tablaPuntajeFinal = Herramientas.cargarImagen("tablaPuntos.png");
        this.estaTablaDePuntaje = false;
    }
\\dibuja los puntos ,saltos y kameha disponibles como usados en el entorno
    public void mostrarEnPantalla(Entorno entorno) {
        entorno.dibujarImagen(puntajes, 400, 20, 0,1);
        entorno.cambiarFont("Impact", 18, Color.BLACK); //ver Ravie,Consolas,Impact
        entorno.escribirTexto(""+kamehamehaImpactado, entorno.ancho()-270, entorno.alto()-572);
        entorno.escribirTexto(""+cantSaltos, entorno.ancho()-490, entorno.alto()-572);
        entorno.escribirTexto(""+tiempo/50, entorno.ancho()-685, entorno.alto()-572);
        entorno.escribirTexto(""+puntos, entorno.ancho()-60, entorno.alto()-572);
    }
    public int kamehameListo() { //permite consultar cuanta energia tiene cargada el kamehame
        return energiaDeKamehameha;
    }
    public void cargado(Entorno entorno) {// dibuja el icono de kamehame listo para tirar
```

```
		entorno.dibujarImagen(cargado, 432, 20, 0,1);
	}
	public void confirmarImpacto() {//confirma colision de kameha con auto
		kamehamehaImpactado+=1;
	}
	public void agotarEnergia() {//una vez lanzado el kameja, resetea y comienza a cargar de nuevo
		energiaDeKamehameha=0;
	}
	public void aumentarSaltos() {// aumenta contador de saltos
		cantSaltos+=1;
	}
	public void aumentarEnergia() { // aumenta contador de energia pal kameha
		energiaDeKamehameha+=1;
	}
	public void aumentarTiempo() { // aumenta contador tiempo en el juego
		tiempo+=1;
	}
	public void aumentarPuntos() { // suma puntos
		puntos=tiempo/50+cantSaltos+kamehamehaImpactado*5;
	}
	public void resetearPuntajes() { // resetear stats
		cantSaltos=0;
		kamehamehaImpactado=0;
		tiempo=0;
		puntos=0;
	}
	//-------------------------si se Gana --------------------
	public void dibujarGano(Entorno entorno) { //dibuja el cartel de You Win
		entorno.dibujarImagen(win, entorno.ancho()/2, entorno.alto()/2-50,0.1);
		entorno.cambiarFont("Impact", 30, Color.BLUE);
		entorno.escribirTexto("'''Presione ENTER para ver los puntajes'",entorno.ancho()/2-250,
entorno.alto()/2+200);
	}
	//-------------------------dibujar Menu del game over --------------------
```

```
	public void gameOver(Entorno entorno) { //dibuja el cartel de game over
		entorno.dibujarImagen(gameOver, entorno.ancho()/2, entorno.alto()/2-50,0.1);
		entorno.cambiarFont("Impact", 30, Color.BLUE);
		entorno.escribirTexto("'''Presione ENTER para ver los puntajes'''",entorno.ancho()/2-250,
entorno.alto()/2+70);
	}
	//-------------------------dibujar Menu de la tabla final --------------------
	public void mostrarTablaDePuntaje() { // confirma la llamada a la tabla depuntajes
			estaTablaDePuntaje=true;
	}
	public void ocultarTablaDePuntaje() {
		estaTablaDePuntaje=false;
	}
public void dibujarTablaDePuntaje(Entorno entorno) {
	entorno.dibujarImagen(tablaPuntajeFinal, entorno.ancho()/2, entorno.alto()/2,0);
	entorno.cambiarFont("Impact", 18, Color.BLACK);
	entorno.escribirTexto(""+tiempo/50, entorno.ancho()/2-10, entorno.alto()/2-118);
	entorno.escribirTexto(""+cantSaltos, entorno.ancho()/2-6, entorno.alto()/2-30);
	entorno.escribirTexto(""+kamehamehaImpactado, entorno.ancho()/2-6, entorno.alto()/2+62);
	entorno.escribirTexto(""+puntos, entorno.ancho()/2-6, entorno.alto()/2+158);
	entorno.escribirTexto("'''Presione ENTER para volver a jugar'''",entorno.ancho()/2-125,
entorno.alto()-85);
}
}
```

Clase Kamehameha

```
package juego;

import java.awt.Image;
import entorno.Herramientas;
import entorno.Entorno;

public class Kamehameha {
  \\variables de instancia
```

```
  private double x;
  private double y;
  private double tamano;
  private Image bolaDeFuego;



  public Kamehameha(double x, double y) {
    this.x = x;
    this.y = y;
\\carga de imagen
this.bolaDeFuego=Herramientas.cargarImagen("bolaDeFuego.gif");
  }
\\verifica si el objeto kameha choca con algun vehículos del arreglo pasado como parámetro
  public int colicionConCoche(Vehiculo[] a) {
         for(int c=0;c<a.length;c++) {
               if(a[c].getX() - a[c].getAncho()/2 < this.x
                   && this.x < a[c].getX() + a[c].getAncho()/2
                   && this.y + this.tamano/2 > a[c].getY() - a[c].getAlto()/2
                   &&this.y - this.tamano/2 < a[c].getY() + a[c].getAlto()/2) {
                      return c;
               }
        }
         return -1;
  }
  \\dibuja la kameha en el entorno
  public void dibujar(Entorno entorno) {
            entorno.dibujarImagen(bolaDeFuego, x, y, 0,1);
  }
 \\desplaza la kameha hacia arriba
  public void desplazarArriba() {
        this.y=this.y-3;

  }
}
```

Clase Vehiculo

```
package juego;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;


public class Vehiculo {

\\variables de instancia

	private double alto;

	private double ancho;

	private double x;

	private double y;

    private Image autoDer;

    private Image autoIzq;

    \\getters

    public double getAlto() {

		return alto;

	}

	public double getAncho() {

		return ancho;

	}

	public double getX() {

		return x;

	}

	public double getY() {

		return y;

	}

	public void desplazarIzqVelocidad(double velocidad) { //aumenta velocidad de los autos que van
hacia la izquierda

		this.x-=velocidad;

	}

	public void desplazarDerVelocidad(double velocidad) { //aumenta la velocidad de los autos que van
hacia la derecha

		this.x+=velocidad;
```

```
	}
	public Vehiculo(double x, double y) {
		this.y = y;
		this.x = x;
		this.alto = 50;
		this.ancho = 100;

		double variableRandom=Math.random();
		//auto al azar mano izquierda
		if(variableRandom<0.3){//carga un coche al azar entre marron, azul, etc
			this.autoIzq= Herramientas.cargarImagen("autoamarilloIzq.gif");
		}
		else if(variableRandom>0.7){
			this.autoIzq=Herramientas.cargarImagen("autocelesteIzq.gif");
		}
		else{
			this.autoIzq=Herramientas.cargarImagen("autoverdeIzq.gif");
		}
		//auto al azar mano derecha
		if(variableRandom<0.3) {
			this.autoDer=Herramientas.cargarImagen("autonaranjaDer.gif");
		}
		else if(variableRandom>0.7){
			this.autoDer=Herramientas.cargarImagen("autocelesteDer.gif");
		}
		else{
			this.autoDer=Herramientas.cargarImagen("autorosaDer.gif");
		}
	}
	//autos que van a la derecha 50x100 pixel
	public void dibujarAutosDerecha(Entorno entorno) {
			entorno.dibujarImagen(autoDer, x, y, 0,1);
	}
//autos que van a la izquierda 50x100 pixel
```

```
	public void dibujarAutosIzquierda(Entorno entorno) {


		entorno.dibujarImagen(autoIzq, x, y, 0,1);

	}

	public void colicionConBordesDerecho(Entorno entorno,Vehiculo[] a) { //comportamiento con el borde derecho

		double min= 0;

		for(int c=0;c<a.length;c++) {

			if(a[c]==null&&c==a.length-1) {

				 break;

			}//se fija que no sea el ultimo en el arreglo para poder pasarle el valor del siguiente lugar

			if(a[c]==null) {

				c=c+1;

			}

			if(a[c].x<min){

				min=a[c].x;

			}

		}

			if (x > entorno.ancho()+50) { // borde derecho

				if(min<50) {

					x=min-150;

				}

				else{

					x=0;

				}

			}

	}

	public void colicionConBordesIzquierdo(Entorno entorno,Vehiculo[] a) { //comportamiento con el borde izquierdo

		double max=entorno.ancho();

		for(int c=0;c<a.length;c++) {

			if(a[c]==null&&c==a.length-1) {

				 break;

			}
```

```
                if(a[c]==null) {
                        c=c+1;
                }

                if(a[c].x>max){
                        max=a[c].x;
                }
                }
            if (x < -50) {
                if (max>entorno.ancho() -50){// borde derecho
                x =max+150;
            }
                else {
                        x=entorno.ancho();
                }
            }
        }
        public void colicionConBordeInferior(Entorno entorno,Vehiculo[] a) { //comportamiento con el borde
inferior
            if (y >= entorno.alto()+25) { //borde inferior
                y =25;
            }
        }
        public void bajar(int a) {// gravedad autos
            this.y+= a;
        }
}
```