Java Racer

Versión 4 – versión final

Versión final

- Agregar:
- Carteles y árboles.
- Otros automóviles.
- Detección de colisiones.
- Inteligencia artificial (AI) rudimentaria.
- HUD (head-up display) con la velocidad, cronómetro de vuelta y de vuelta más rápida.

Estructura del código

- Clases nuevas: Car, Sprite.
- Clases que cambian: Segment,
- Las demás clases no cambian.

Sprites

- El archivo sprites.png contiene imágenes de todos los automóviles, árboles y carteles.

Sprites

- La clase Sprite contiene las coordenadas x, y, w, h de las imágenes individuales.

Clase Sprites

```
public class Sprites {
    public static final Rectangle
        PALM_TREE = new Rectangle(5, 5, 215, 540),
        BILLBOARD08 = new Rectangle(230, 5, 385, 265),
        // … etc
        PLAYER_STRAIGHT = new Rectangle(1085, 480, 80, 41),
        PLAYER_RIGHT = new Rectangle(995, 531, 80, 41);
        // … etc
}
```

Agregando sprites y automóviles

- Se agregan dos listas a cada segmento de la carretera:
- Una lista guarda los sprites (carteles y árboles) en el segmento.
- Otra lista guarda los automóviles en el segmento.
- La diferencia es que los automóviles se pueden mover de un segmento a otro.

Clase Segment

```
public class Segment {
    public int index;
    public PointProj p1, p2;
    public ColorModel color;
    public boolean looped;
    public float fog;
    public double curve;
    public List<Sprite> sprites;
    public List<Car> cars;
    public int clip;
```

Clase Car

- Cada automóvil tiene un offset horizontal aleatorio, velocidad aleatoria, posición z y un rectángulo que representa el sprite.

Clase Car

```
public class Car {
    public double offset, z, speed, percent;
    public Rectangle sprite;
    public Car (double offset, double z, Rectangle sprite, double speed) {
        this.offset = offset;
        this.z = z;
        this.sprite = sprite;
        this.speed = speed;
        this.percent = 0;
    }
}
```

Clase Sprite

- Cada sprite fijo tiene un rectángulo que representa el sprite y un offset horizontal.
- Un offset de -1 indica el lado izquierdo de la carretera.
- Un offset de +1 indica el lado derecho de la carretera.

Clase Sprite

```
public class Sprite {
    public Rectangle source;
    public double offset;
    public Sprite(Rectangle source, double offset)
    {
        this.source = source;
        this.offset = offset;
    }
}
```

Clase Game modificada

- Se declara una lista para guardar los automóviles.

```
private final List<Car> cars = new ArrayList<>();
```

- Se declara una constante para los automóviles y variables para el HUD.

```
private final int totalCars = 200;
private long currentLapTime = 0;
private long lastLapTime = -1;
private long fastestLapTime  = 999999;
```

Clase Game modificada

- En la clase Game se agregan los métodos addSprite, resetSprites y resetCars.
- addSprite agrega un sprite (cartel, árbol o columna) a un segmento.
- resetSprites agrega varios sprites a la carretera.
- resetCars agrega automóviles a la carretera.
- resetSprites y resetCars se invocan en resetRoad.

Método addSprite

```
private void addSprite(int n, Rectangle sprite, double offset)
{
    segments.get(n).sprites.add(new Sprite(sprite, offset));
}
```

Método resetSprites

- Algunos sprites se colocan de forma fija y otros de manera aleatoria.

```
private void resetSprites()
{
    addSprite(20,  Sprites.billboard07, -1);
    addSprite(40,  Sprites.billboard06, -1);
    addSprite(60,  Sprites.billboard08, -1);
    addSprite(80,  Sprites.billboard09, -1);
    addSprite(100, Sprites.billboard01, -1);
    // … etc
```

Método resetSprites

```
for (int n = 10 ; n < 200 ; n += 4 + Math.floor(n / 100)) {
    addSprite(n, Sprites.PALM_TREE, 0.5 + Math.random() * 0.5);
    addSprite(n, Sprites.PALM_TREE, 1 + Math.random() * 2);
}
for (int n = 250; n < 1000; n += 5) {
    addSprite(n, Sprites.COLUMN, 1.1);
    addSprite(n + Util.randomInt(0, 5), Sprites.TREE1, -1 - (Math.random() * 2));
    addSprite(n + Util.randomInt(0, 5), Sprites.TREE2, -1 - (Math.random() * 2));
}
```

Método resetCars

```
private void resetCars()
{
    cars.clear();
    Car car;
    Segment segment;
    double offset, z;
    Rectangle sprite;
```

Método resetCars

```
        for (int n = 0; n < totalCars; n++) {
            offset = Math.random() * Util.randomChoice(new double[]{-0.8, 0.8});
            z = Math.floor(Math.random() * segments.size()) * segmentLength;
            sprite = Util.randomChoice(Sprites.CARS);
            speed = maxSpeed / 4 + Math.random() * maxSpeed / (sprite == Sprites.SEMI ?
4 : 2);
            car = new Car(offset, z, sprite, speed);
            segment = findSegment(car.z);
            segment.cars.add(car);
            cars.add(car);
        }
    }
```

Automóviles

- Los automóviles están guardados en dos estructuras de datos: la lista global cars y la lista cars de cada segmento.
- Ventajas:
- Se puede iterar sobre todos los automóviles en el método update, moviéndolos de un segmento a otro si es necesario.
- Permite renderizar (método render) solo los automóviles en los segmentos visibles.