VIDEOJUEGO CALDERO ORIGINAL

MARIO JIMÉNEZ MARSET

ÍNDICE

1. ENUNCIADO – OBJETIVOS

En esta práctica se pedía compilar y ejecutar el juego del caldero a partir del código fuente facilitado. Hecho esto y viendo que funciona, se pedía sustituir los comentarios del programa por los propios (demostrando así que se comprende el videojuego).

Para todo esto se necesitaba importar cinco jars externos, además de las clases y demás documentación.

2. DESARROLLO – PROCEDIMIENTOS

Se muestra el código ejecutable con comentarios propios:

Código Clase DesktopDrop:

```
package juego;
import com.badlogic.gdx.backends.lwjgl.LwjglApplication;
import com.badlogic.gdx.backends.lwjgl.LwjglApplicationConfiguration;

//clase principal que, al ser ejecutada, muestra el juego disponible para ser utilizado
public class DesktopDrop {
	public static void main(String[] args) {
		//se crea un objeto el cual configura el juego
		LwjglApplicationConfiguration configuracion = new
LwjglApplicationConfiguration();
		//se llama a la clase System para fijar la propiedad que permite utilizar
el software OpenGL (líos de licencias)
		System.setProperty("org.lwjgl.opengl.Display.allowSoftwareOpenGL",
"true");
		//se configura el título y las dimensiones de la pantalla
		configuracion.title = "Drop";
		configuracion.width = 1024;
		configuracion.height = 768;
		//se instancia una nueva configuración, llamando a otra clase
		new LwjglApplication(new Drop(), configuracion);
	}
}
```

Código Drop:

```
package juego;
import com.badlogic.gdx.Game;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;

public class Drop extends Game {
	//se crean las variables necesarias a utilizar en la clase
	OrthographicCamera camara;
	SpriteBatch spriteBatch;
	BitmapFont fuente;
	int gotasRecogidas;
	//se crea un método el cual se invoca al momento de ejecutar la aplicación
	@Override
	public void create() {
```

```
			//se inicializan las variables, fijando la pantalla, la fuente de las letras y el spriteBatch
			spriteBatch = new SpriteBatch();
			fuente = new BitmapFont();
			setScreen(new MainMenuScreen(this));
		}
		//método sobrescrito que se invoca cuada vez que toca renderizar, actualizándose la lógica del juego
		@Override
		public void render() {
			super.render();
		}
		//método sobrescrito que coloca el spriteBatch y la fuente de las letras
		@Override
		public void dispose() {
			spriteBatch.dispose();
			fuente.dispose();
		}
}
```

Código GameOverScreen:

```
package juego;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input.Keys;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.GL10;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.scenes.scene2d.Stage;

//esta clase presenta cómo se ve la pantalla al finalizar el usuario la partida
public class GameOverScreen implements Screen {
		//se crean las variables a utilizar en la clase
		//se crea un objeto d ela clase Drop
		final Drop juego;
		Stage menu;
		OrthographicCamera camara;
		//en el constructor se parametriza el objeto Drop, además de inicializar la cámara
		public GameOverScreen(Drop juego) {
			this.juego = juego;
			camara = new OrthographicCamera();
			camara.setToOrtho(false, 1024, 768);
		}
		//se sobrescribe el método de renderizado
		@Override
		public void render(float delta) {
			//se establecen los colores exactos del juego
			Gdx.gl.glClearColor(0, 0.3f, 0.6f, 1);
			Gdx.gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
			//se actualiza la cámara, además de fijar cuál será su proyección
			camara.update();
			juego.spriteBatch.setProjectionMatrix(camara.combined);
			//se muestra el menú de inicio
			juego.spriteBatch.begin();
			//se establecen los mensajes al terminar el juego y, si se quiere salir o volver a jugar, la tecla la cual clicar
			juego.fuente.draw(juego.spriteBatch, "Fin del juego!!!!", 100, 150);
```

```
                juego.fuente.draw(juego.spriteBatch, "Tu puntuación: " +
juego.gotasRecogidas, 100, 130);
                juego.fuente.draw(juego.spriteBatch, "Si quieres jugar otra partida pulsa
la tecla 'N'", 100, 110);
                juego.fuente.draw(juego.spriteBatch, "Pulsa 'ESCAPE' para SALIR",
100, 90);
                juego.spriteBatch.end();
                //si el usuario toca la pantalla, se inicia la partida
                if (Gdx.input.isKeyPressed(Keys.N)) {
                        //al empezar una nueva partida, no debe de haber gotas
recogidas
                        juego.gotasRecogidas = 0;
                        //se fija la pantalla llamando a otra clase
                        juego.setScreen(new GameScreen(juego));
                }
                //al pulsar la tecla ESCAPE, se permite salir del juego
                else if (Gdx.input.isKeyPressed(Keys.ESCAPE)) {
                        dispose();
                        System.exit(0);
                }
        }
        //diferentes métodos auto-implementados debido a la interfaz Screen
        @Override
        public void resize(int width, int height) {
        }
        @Override
        public void show() {
        }
        @Override
        public void hide() {
        }
        @Override
        public void pause() {
        }
        @Override
        public void resume() {
        }
        @Override
        public void dispose() {
                juego.dispose();
        }
}
```

Código Clase GameScreen:

```
package juego;
import java.util.Iterator;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.InputProcessor;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.Input.Keys;
import com.badlogic.gdx.audio.Music;
import com.badlogic.gdx.audio.Sound;
import com.badlogic.gdx.graphics.GL10;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.math.MathUtils;
import com.badlogic.gdx.math.Rectangle;
```

```
import com.badlogic.gdx.math.Vector3;
import com.badlogic.gdx.utils.Array;
import com.badlogic.gdx.utils.TimeUtils;
import com.badlogic.gdx.utils.Timer;
import com.badlogic.gdx.utils.Timer.Task;

//pantalla donde el usuario juega la partida
public class GameScreen implements Screen, InputProcessor {
	//se crean las variables necesarias en la clase
	//se crea una constante de la clase Drop
	final Drop juego;
	//se implementan las texturas e imágenes cargas en el proyecto para dar vida a los elementos del juego
	Texture spriteGota;
	Texture spriteCubo;
	Texture spriteRoca;
	Sound sonidoGota;
	Music musicaLluvia;
	Sound sonidoRoca;
	//se representan los elementos del juego como rectángulos; se utilizan para comprobar las colisiones entre los mismos
	Rectangle cubo;
	Array<Rectangle> gotas;
	Array<Rectangle> rocas;
	//controlan el ritmo al que van cayendo las gotas y las rocas
	long momentoUltimaGota;
	long momentoUltimaRoca;
	float tiempoJuego;
	//indica si el juego está en pausa
	boolean pausa = false;
	//cámara del juego
	OrthographicCamera camara;
	//constructor que parametriza el objeto de la clase Drop
	public GameScreen(Drop juego) {
		this.juego = juego;
		//duración exacta de la partida
		tiempoJuego = 50;
		//carga las imágenes del juego, cuyos elementos se encuentran cargados en el proyecto
		spriteGota = new Texture(Gdx.files.internal("droplet.png"));
		spriteCubo = new Texture(Gdx.files.internal("bucket.png"));
		spriteRoca = new Texture(Gdx.files.internal("rock.png"));
		//carga los sonidos del juego
		sonidoGota = Gdx.audio.newSound(Gdx.files.internal("waterdrop.wav"));
		musicaLluvia = Gdx.audio.newMusic(Gdx.files.internal("undertreeinrain.mp3"));
		sonidoRoca = Gdx.audio.newSound(Gdx.files.internal("rock.mp3"));
		//inicia la música de fondo del juego (en bucle hasta terminar)
		musicaLluvia.setLooping(true);
		//se establecen los pixeles y dimensiones del cubo en el juego
		cubo = new Rectangle();
		cubo.x = 1024 / 2 - 64 / 2;
		cubo.y = 20;
		cubo.width = 64;
		cubo.height = 64;
		//genera la lluvia de gotas
		gotas = new Array<Rectangle>();
```

```
                generarLluvia();
                //se lanza la primera roca
                rocas = new Array<Rectangle>();
                lanzarRoca();
                //se crea la cámara y se define la zona de visión del juego (toda la
pantalla)
                camara = new OrthographicCamera();
                camara.setToOrtho(false, 1024, 768);
                Gdx.input.setInputProcessor(this);
        }
        //método sobrescrito que tiene el objetivo de renderizar
        @Override
        public void render(float delta) {
                //se pinta el fondo de pantalla con un color exacto
                Gdx.gl.glClearColor(0, 0, 0.2f, 1);
                //se limpia la pantalla
                Gdx.gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
                //se actualiza la cámara
                camara.update();
                //se comprueba la entrada del usuario. Además, se actualiza la posición
de los elementos del juego y se dibujan en la pantalla
                if (!pausa) {
                        comprobarInput();
                        actualizar();
                }
                //se redibuja la pantalla (obligatorio)
                dibujar();
        }
        //método que comprueba la entrada del usuario (teclado o pantalla)
        private void comprobarInput() {
                //se establecen las operaciones necesarias para hacer que el cubo se
mueva pulsando en la pantalla
                if (Gdx.input.isTouched()) {
                        Vector3 posicion = new Vector3();
                        posicion.set(Gdx.input.getX(), Gdx.input.getY(), 0);
                        //se transforman las coordenadas de la posición al sistema de
coordenadas de la cámara
                        cubo.x = posicion.x - 64 /2;
                }
                //se mueve el cubo pulsando LEFT y RIGHT
                if (Gdx.input.isKeyPressed(Keys.LEFT))
                        cubo.x -= 200 * Gdx.graphics.getDeltaTime();
                if (Gdx.input.isKeyPressed(Keys.RIGHT))
                        cubo.x += 200 * Gdx.graphics.getDeltaTime();
        }
        //se actualiza la posición de todos los elementos del juego
        private void actualizar() {
                //se comprueba que el cubo no se salga de los límites de la pantalla
                if (cubo.x < 0)
                        cubo.x = 0;
                if (cubo.x > 1024 - 64)
                        cubo.x = 1024 - 64;
                //se generan nuevas gotas dependiendo del tiempo que ha pasado
desde la última
                if (TimeUtils.nanoTime() - momentoUltimaGota > 100000000)
                        generarLluvia();
                //se generan nuevas rocas
                if (TimeUtils.nanoTime() - momentoUltimaRoca > 1000000000)
```

```
                    lanzarRoca();
                //se actualizan las posiciones de las gotas (si llega al suelo, se elimina;
si llega al cubo, suena y se elimina)
                Iterator<Rectangle> iter = gotas.iterator();
                while (iter.hasNext()) {
                    Rectangle gota = iter.next();
                    gota.y -= 200 * Gdx.graphics.getDeltaTime();
                    if (gota.y + 64 < 0)
                        iter.remove();
                    if (gota.overlaps(cubo)) {
                        sonidoGota.play();
                        iter.remove();
                        juego.gotasRecogidas++;
                    }
                }
                //se actualizan las posiciones de las rocas (si llega al suelo, se elimina;
si llega al cubo, lo rompe y se termina la partida)
                Iterator<Rectangle> iterRoca = rocas.iterator();
                while (iterRoca.hasNext()) {
                    Rectangle roca = iterRoca.next();
                    roca.y -= 200 * Gdx.graphics.getDeltaTime();
                    if (roca.y + 64 < 0)
                        iterRoca.remove();
                    if (roca.overlaps(cubo)) {
                        sonidoRoca.play();
                        pausa = true;
                        Timer.schedule(new Task(){
                           @Override
                           public void run() {
                              dispose();
                                    juego.setScreen(new
GameOverScreen(juego));
                           }
                        }, 2);
                    }
                }
                //se actualiza el tiempo de juego
                tiempoJuego -= Gdx.graphics.getDeltaTime();
                if (tiempoJuego < 0) {
                    dispose();
                    juego.setScreen(new GameOverScreen(juego));
                }
            }
            //se dibujan los elementos del juego en pantalla
            private void dibujar() {
                //pinta la imágenes del juego en la pantalla
                juego.spriteBatch.begin();
                juego.spriteBatch.draw(spriteCubo, cubo.x, cubo.y);
                for (Rectangle gota : gotas)
                    juego.spriteBatch.draw(spriteGota, gota.x, gota.y);
                for (Rectangle roca : rocas)
                    juego.spriteBatch.draw(spriteRoca, roca.x, roca.y);
                juego.fuente.draw(juego.spriteBatch, "Puntos: " +
juego.gotasRecogidas, 1024 - 100, 768 - 50);
                juego.fuente.draw(juego.spriteBatch, "Tiempo: " + (int) (tiempoJuego),
1024 - 100, 768 - 80);
                juego.spriteBatch.end();
            }
```

```
		//se genera una gota de lluvia en una posición aleatoria de la pantalla y anota el
momento de generarse
		private void generarLluvia() {
			Rectangle gota = new Rectangle();
			gota.x = MathUtils.random(0, 1024 - 64);
			gota.y = 768;
			gota.width = 64;
			gota.height = 64;
			gotas.add(gota);
			momentoUltimaGota = TimeUtils.nanoTime();
		}
		//genera una roca y la deja caer
		private void lanzarRoca() {
			Rectangle roca = new Rectangle();
			roca.x = MathUtils.random(0, 1024 - 64);
			roca.y = 768;
			roca.width = 64;
			roca.height = 64;
			rocas.add(roca);
			momentoUltimaRoca = TimeUtils.nanoTime();
		}
		//método sobrescrito que se invoca cuando esta pantalla es la que se está
mostrando
		@Override
		public void show() {
			musicaLluvia.play();
		}
		//método que se invoca cuando esta pantalla deja de ser la principal
		@Override
		public void hide() {
			musicaLluvia.stop();
		}
		@Override
		public void dispose() {
			//se liberan los recursos utilizados
			spriteGota.dispose();
			spriteCubo.dispose();
			spriteRoca.dispose();
			sonidoGota.dispose();
			musicaLluvia.dispose();
			sonidoRoca.dispose();
			gotas.clear();
			rocas.clear();
		}
		//métodos sobrescritos implementados por la interfaz Screen
		@Override
		public void resize(int width, int height) {
		}
		@Override
		public void pause() {
			pausa = true;
		}
		@Override
		public void resume() {
			pausa = false;
		}
		@Override
		public boolean keyDown(int keycode) {
```

```
                return false;
        }
        @Override
        public boolean keyUp(int keycode) {
                //se pone el juego en pausa
                if (keycode == Keys.P)
                        pausa = !pausa;
                return false;
        }
        @Override
        public boolean keyTyped(char character) {
                return false;
        }
        @Override
        public boolean touchDown(int screenX, int screenY, int pointer, int button) {
                return false;
        }
        @Override
        public boolean touchUp(int screenX, int screenY, int pointer, int button) {
                return false;
        }
        @Override
        public boolean touchDragged(int screenX, int screenY, int pointer) {
                return false;
        }
        @Override
        public boolean mouseMoved(int screenX, int screenY) {
                return false;
        }
        @Override
        public boolean scrolled(int amount) {
                return false;
        }
}
```

Código MainMenuScreen:

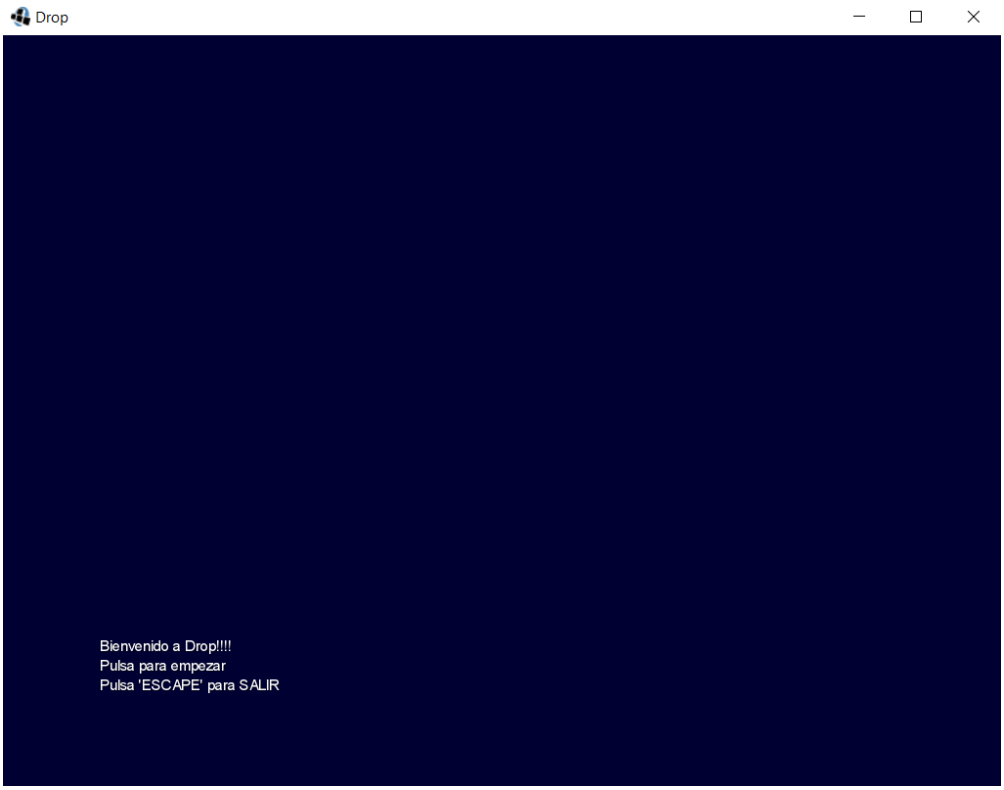
```
package juego;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input.Keys;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.GL10;
import com.badlogic.gdx.graphics.OrthographicCamera;

//pantalla de incio que muestra el menú del juego
public class MainMenuScreen implements Screen {
        //se crean las variables a utilizar en la clase
        //se crea un objeto de la clase Drop
        final Drop juego;
        OrthographicCamera camara;
        //se parametriza el objeto
        public MainMenuScreen(Drop juego) {
                this.juego = juego;
                camara = new OrthographicCamera();
                camara.setToOrtho(false, 1024, 768);
        }
        //método que tiene el objetivo de renderizar
        @Override
```

```
		public void render(float delta) {
			//se establecen los colores exactos del menú
			Gdx.gl.glClearColor(0, 0, 0.2f, 1);
			Gdx.gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
			//se actualiza la cámara
			camara.update();
			juego.spriteBatch.setProjectionMatrix(camara.combined);
			//se muestra el menú de inicio
			juego.spriteBatch.begin();
			juego.fuente.draw(juego.spriteBatch, "Bienvenido a Drop!!!!", 100, 150);
			juego.fuente.draw(juego.spriteBatch, "Pulsa para empezar", 100, 130);
			juego.fuente.draw(juego.spriteBatch, "Pulsa 'ESCAPE' para SALIR",
100, 110);
			juego.spriteBatch.end();
			//si el usuario toca la pantalla, se inicia la partida
			if (Gdx.input.isTouched()) {
				juego.setScreen(new GameScreen(juego));
				dispose();
			}
			if (Gdx.input.isKeyPressed(Keys.ESCAPE)) {
				dispose();
				System.exit(0);
			}
		}
		//métodos implementados por la interfaz Screen
		@Override
		public void resize(int width, int height) {
		}
		@Override
		public void show() {
		}
		@Override
		public void hide() {
		}
		@Override
		public void pause() {
		}
		@Override
		public void resume() {
		}
		@Override
		public void dispose() {
		}
}
```

CAPTURAS VIDEOJUEGO EN EJECUCIÓN:

Menú principal, donde se clica para empezar y se pulsa ESC para salir/cerrar el programa:



Juego en ejecución, con contador de puntos y tiempo:

Juego al terminar la partida, ya que el cubo ha tocado una roca:

Pantalla final del juego, donde se presenta un menú el cual elegir si volver a jugar o salir: