Documentación Juego Retro LBreakOut2

Simón Sloan García Villa Alejandro Tirado Ramirez

October 28, 2025

Contents

0.1 Introducción 1
0.2 Sobre el juego 1
0.3 Sobre el diseño general escogido 1
0.3.1 Square.jack 2
0.3.2 Ball.jack 2

0.1 Introducción

En esta oportunidad estamos trabajando un juego retro clásico llamado *LBreakOut2*, mediante el lenguaje de programación *Jack* del programa educativo *Nand2Tetris*, usando por supuesto el compilador correspondiente *JackCompiler* para pasarlo a *Aritmética de Stack* y luego ejecutarlo en *VMSimulator*. En las próximas secciones se ahondará más sobre cada uno de los aspectos particulares tanto del juego, como su solución y por supuesto los aprendizajes adquiridos.

0.2 Sobre el juego

Como se dijo en la introducción, en esta oportunidad nos propusimos la creación del clásico juego retro *LBreakOut2* donde tenemos una plataforma que se mueve, de izquierda a derecha (o viceversa) por medio de las teclas ←, →. Además tenemos cierto número de obstáculos, en el caso particular de nosotros, 4 filas y 8 columnas de rectángulos, que tenemos que destruir sin que la pelota se escape por la parte inferior de la pantalla. Por otra parte, tenemos un sistema de vidas, donde solo se pierde si se te escapa la pelota 3 veces (cada vez que pierdas una vida, aparecerá un contador con una vida menos) y se gana si destruyes todos los obstáculos.

0.3 Sobre el diseño general escogido

En términos generales tenemos *6 archivos*, estos son:

- `Ball.jack` - Control del movimiento de la pelota
- `Gameboard.jack` - Gestión del tablero de juego

- `Main.jack` - Punto de entrada principal
- `Obstacle.jack` - Manejo de obstáculos/bloques
- `Square.jack` - Elementos gráficos básicos
- `SquareGame.jack` - Lógica principal del juego

A continuación se explicará de manera un poco más precisa, cada uno de los documentos.

0.3.1 Square.jack

Nuestro juego, al igual que la gran mayoría de los juegos retro usa la figura geométrica *Cuadrado* como base. En esta línea de ideas, este archivo es bastante importante en nuestro juego, porque los otros documentos usarán sus funciones para ejecutar ciertas acciones como mostrar las paredes, los obstáculos e incluso la propia barra. Además en este archivo están las acciones fundamentales de movimiento de la barra con funciones como *moveLeft* o *moveRight*.

Consultar el archivo Square.jack para más detalle.

0.3.2 Ball.jack

Este archivo es bastante corto pero tiene todo el movimiento de la pelota (incluyendo la lógica necesaria para que rebote exactamente en un ángulo de 45 grados).

Movimiento básico

Para lograr el movimiento, tenemos el siguiente método principal:

```
1     method void move() {
2       do erase();
3       let x = x + dx;
4       let y = y + dy;
5       do draw();
6       return;
7     }
```

Listing 1: Método move() de la pelota

Aquí está la magia, y es que como podemos ver, tenemos un movimiento tanto en x como en y, donde **y** y **x** son las posiciones iniciales en **y** y en **x**, y donde **dy** y **dx** son la posición deseada (después del movimiento).

Física del movimiento a 45°

Para garantizar el movimiento a 45 grados, inicializamos `dx` y `dy` con valores absolutos iguales (generalmente 1 o -1). Esto asegura que:

$$\text{velocidad} = \sqrt{dx^2 + dy^2} = \sqrt{1^2 + 1^2} = \sqrt{2}$$

Y el ángulo se mantiene en:

$$\theta = \arctan\left(\frac{dy}{dx}\right) = \arctan(1) = 45^\circ$$

Sistema de rebotes

Los métodos de rebote mantienen este ángulo cambiando solo una componente a la vez:

```
    // Rebote vertical - invierte direcci n Y
    method void bounceY() {
      let dy = -dy;
      return;
    }

    // Rebote horizontal - invierte direcci n X
    method void bounceX() {
      let dx = -dx;
      return;
    }
```

Listing 2: Métodos de rebote

Este diseño garantiza que la pelota siempre se mueva en diagonales de 45 grados, simplificando la física del juego mientras se mantiene un comportamiento predecible y jugable.