

KOTLIN SPACE INVADERS

Lógica e Implementación

Deconstrucción técnica de un motor de juego en consola



A technical line drawing of a vintage-style arcade cabinet. The cabinet is shown from a three-quarter perspective, highlighting its front control panel with a joystick and four buttons, and its side panels. Various dimensions are labeled in millimeters: height (1253), width (942), depth (409), and base width (491). A vertical dimension of 900 is also indicated. On the right side of the drawing, a sequence of binary digits (01001011, 01101011, 01001010, etc.) is overlaid, suggesting a digital or data-related theme.

El Desafío de Ingeniería

A 5x8 grid of white downward-pointing chevrons on a dark gray background. Each chevron is a V shape pointing downwards. The grid is composed of five rows and eight columns.

A

¿Cómo convertimos reglas abstractas en un **sistema interactivo**? Este proyecto reduce un videojuego a sus componentes atómicos utilizando Kotlin.

```

// Project: Alien Invasion Console Game
package com.gaeem.alien

import java.util.Scanner

// Main game loop structure
fun main() {
    var gameState = GameState()
    var reciever = ConstantReceiver()
    var inputHandler = InputHandler()

    // The infinite game loop
    while (gameState.isRunning) {
        // Update game state based on input and logic
        inputHandler.processInput(gameState)
        gameState.update()

        // Render the current state to console
        reciever.render(gameState)
        // Control frame rate (approximated)
        Thread.sleep(200)
    }
}

// Game structure for aliens
class GameState {
    var isRunning: Boolean = true
    var alienList: List<Alien> = generateAliens()
    var ship: Ship = Ship()

    // Random lists for game objects
    fun generateAliens(): MutableList<Alien> {
        // Implementation of alien generation...
        return mutableListOf(Alien())
    }

    fun update() {
        // Logic to move aliens, check collisions, etc.
    }
}

```

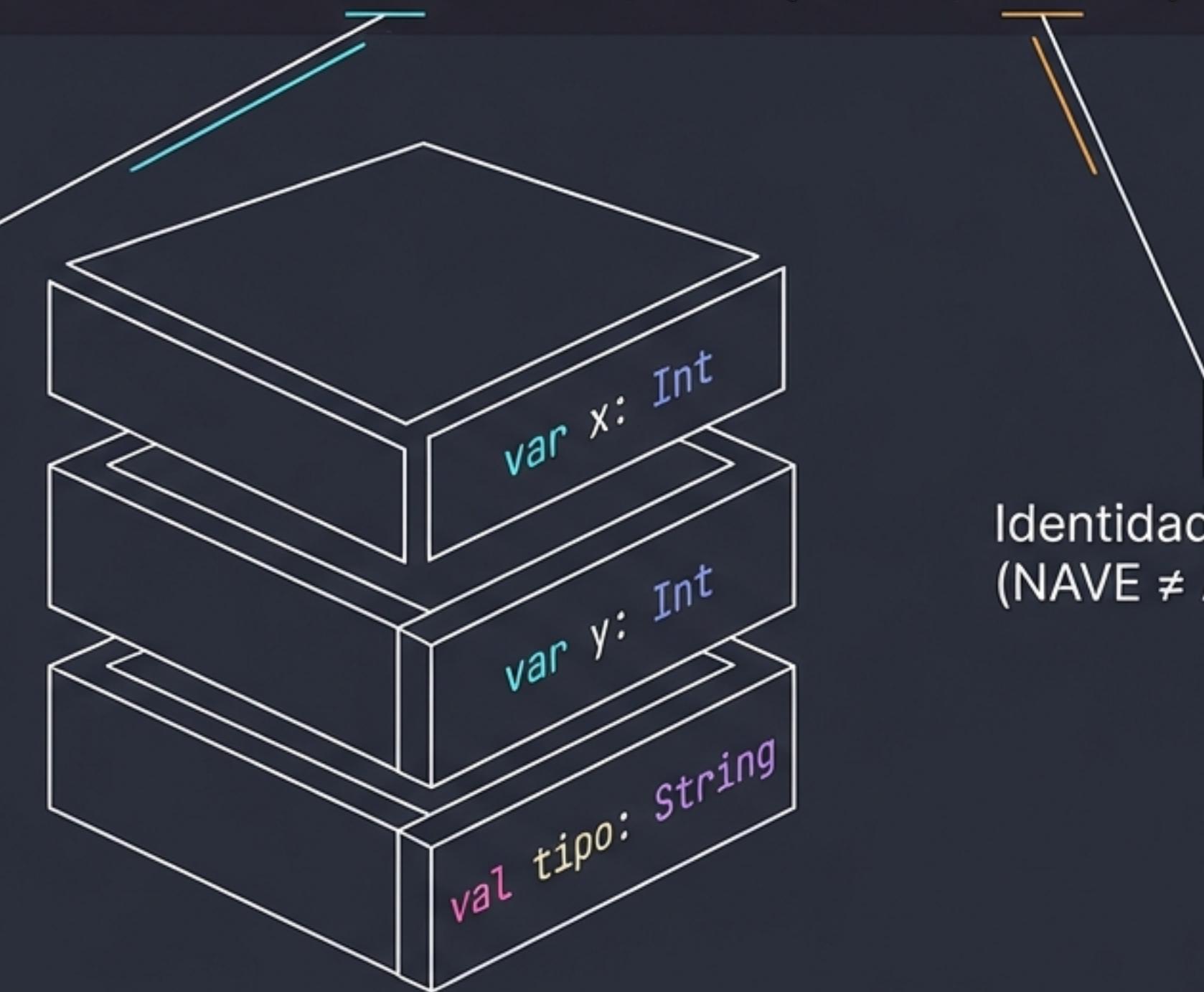
- **Arquitectura:** Programación Orientada a Objetos (POO)
 - ◊ **Estructura de Datos:** Listas mutables dinámicas
 - **Ciclo de Vida:** El bucle infinito (Game Loop)

El Plano Maestro: Clase Entidad

Antes de dibujar nada,
definimos qué es un objeto.
La clase **Entidad** es el molde
para todo lo que existe en
nuestro universo.

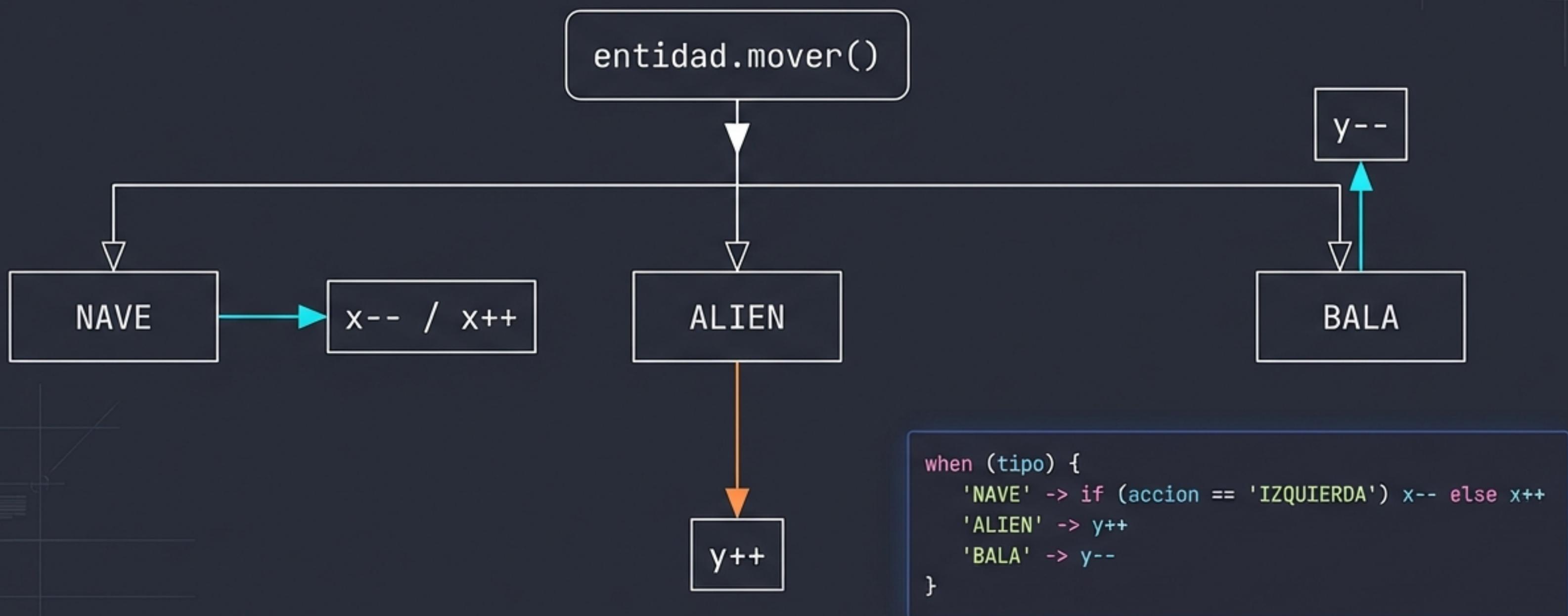
Posición mutable
(cambia frame a frame)

```
class Entidad(var x: Int, var y: Int, val tipo: String)
```



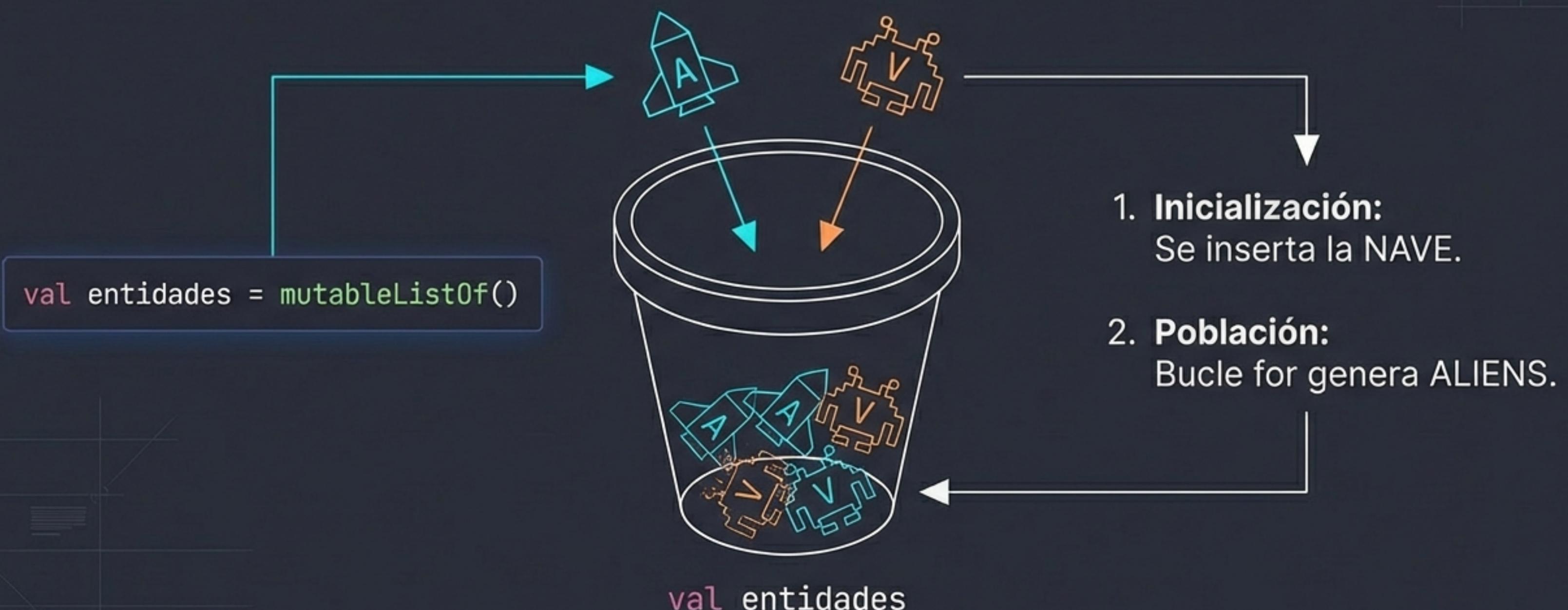
Comportamiento Polimórfico

La función mover() dicta la reacción a los estímulos. Usamos 'when' como un selector inteligente.

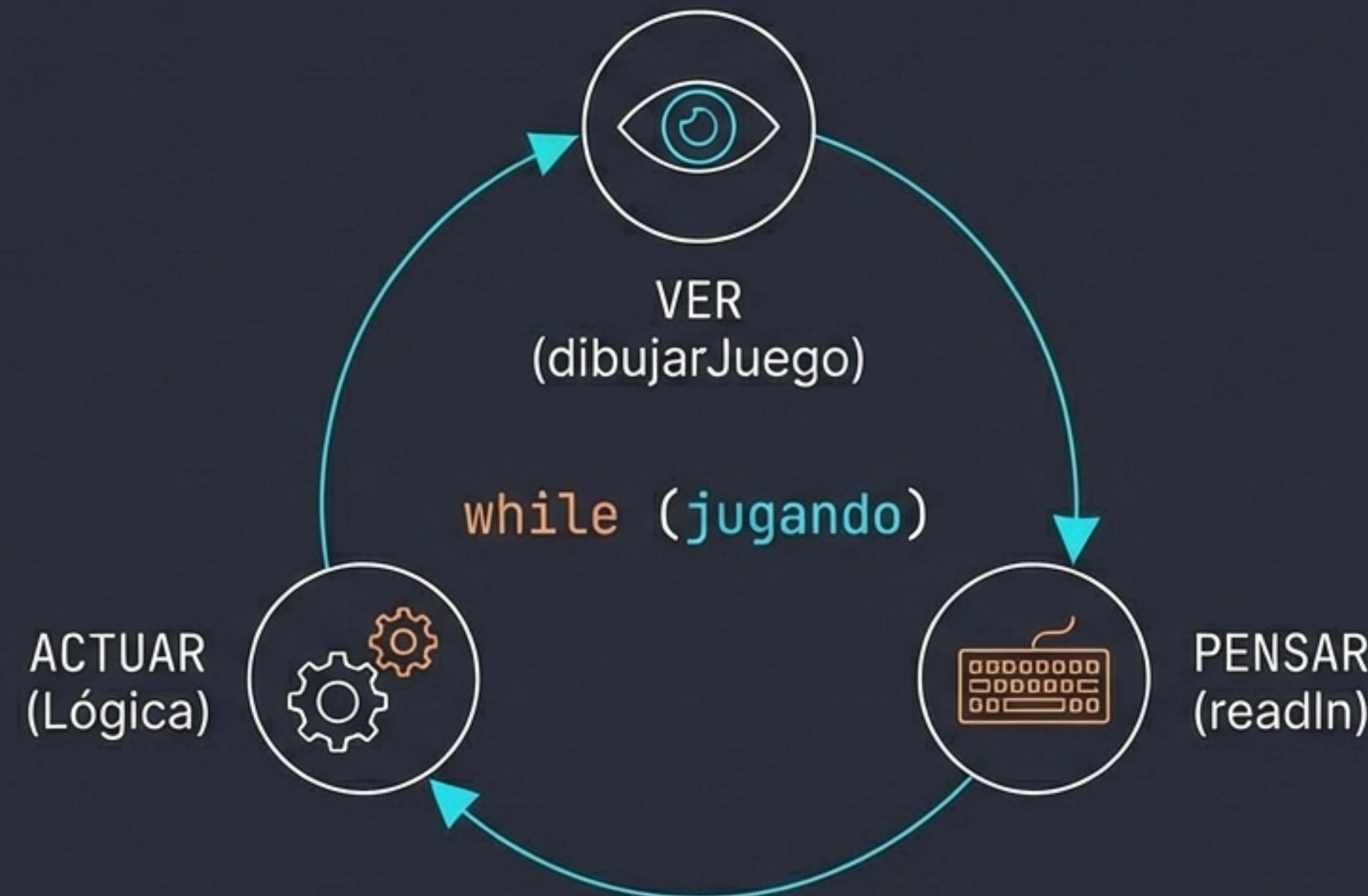


El Contenedor del Universo

El juego vive en una 'mutableListOf'. Un hangar dinámico que se expande y contrae.



El Motor: The Game Loop



Un videojuego es una ilusión de movimiento. El sistema repite estas tres fases críticas infinitamente.

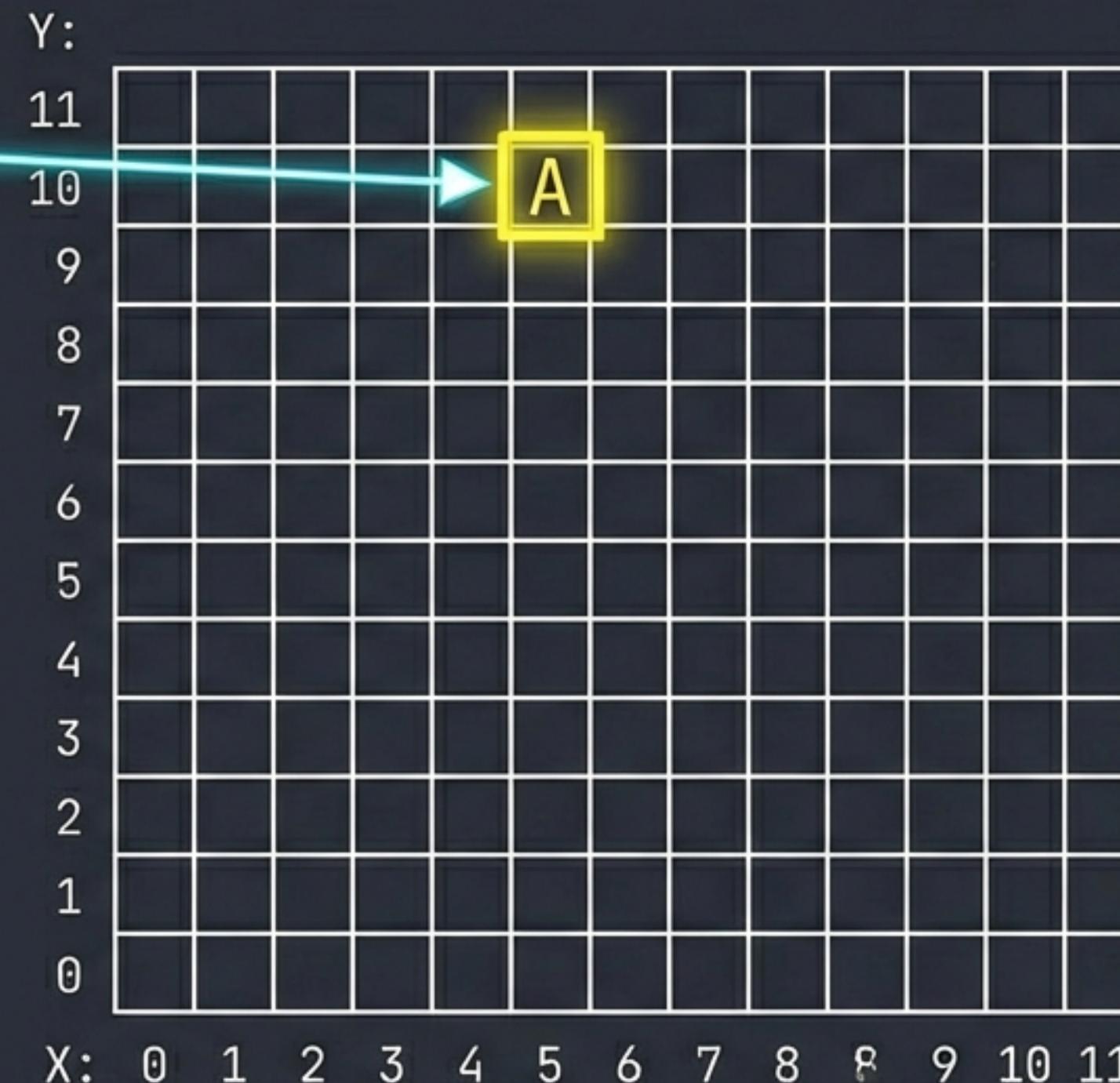
Visualización: De Datos a Caracteres

Bucle Anidado:
Recorre filas (Y) y
columnas (X).

Entidad(x=5,
y=10,
'NAVE')

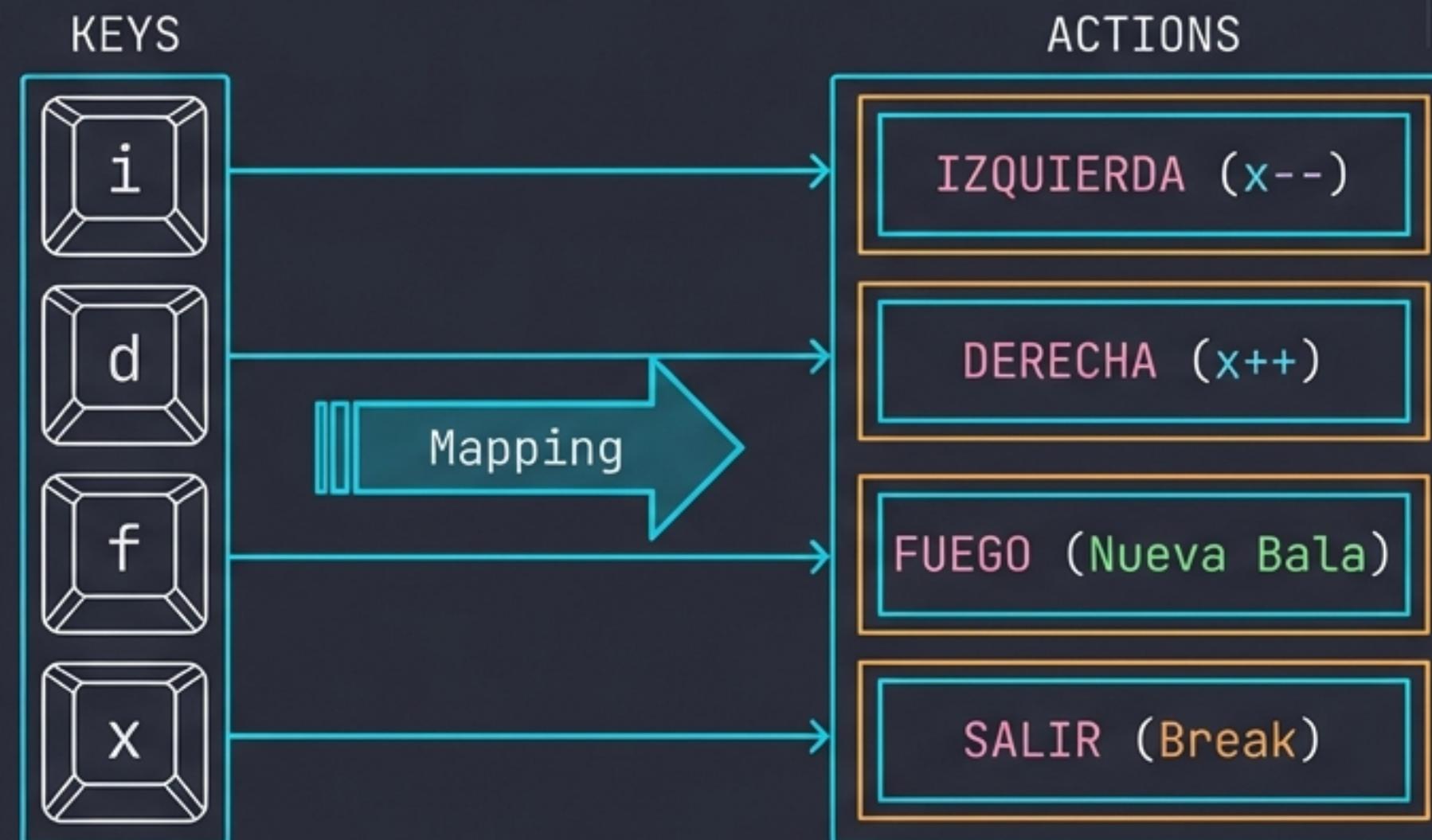
Búsqueda:
`lista.find`
`{ it.x == x && it.y == y }`

Mapeo de Caracteres	
NAVE	-> A
ALIEN	-> V
BALA	->



Interfaz de Entrada

`readln()` detiene el tiempo para esperar órdenes. Traducimos teclas físicas a intenciones semánticas.



```
val entrada = readln()
```

Acción y Reacción

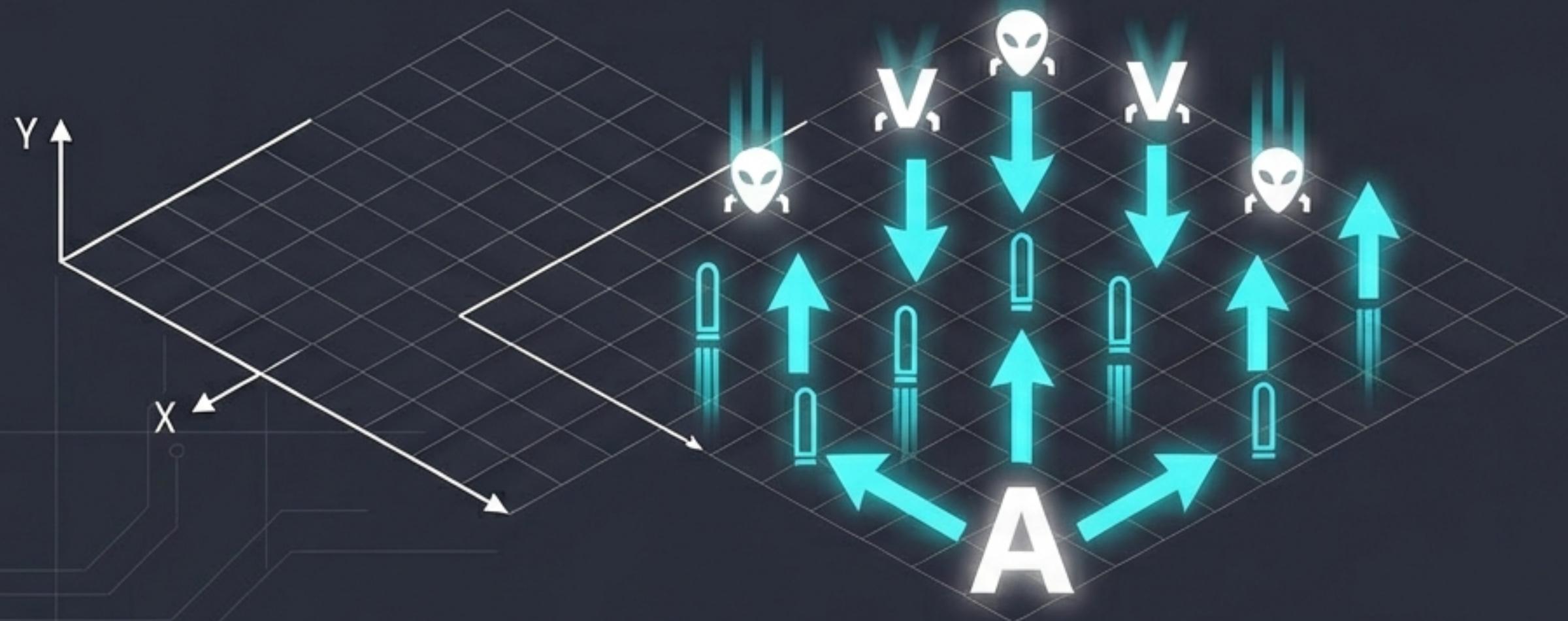
Cuando el usuario elige FUEGO, modificamos la estructura del universo.



```
if (accion == 'FUEGO') {  
    entidades.add(  
        Entidad(  
            nave.x,  
            nave.y,  
            'BALA'  
    )  
}
```

Actualización de Estado

Una vez procesado el input, el universo avanza un 'tick'.



```
for (entidad in entidades) {  
    entidad.mover(accion)  
}
```

Gestión de Memoria y Limpieza



¡Cuidado! No puedes borrar elementos de una lista mientras la lees.



Algoritmo de Colisión

¿Hubo impacto? Comparamos coordenadas.



Impacto Directo

```
entidad.y == otra.y
```

Cruce en Aire

```
entidad.y == otra.y + 1
```

Resultado: Ambos se añaden a paraBorrar.

Condiciones de Victoria y Derrota



GAME OVER

Alien alcanza suelo ($y \geq \text{alto}$).

return



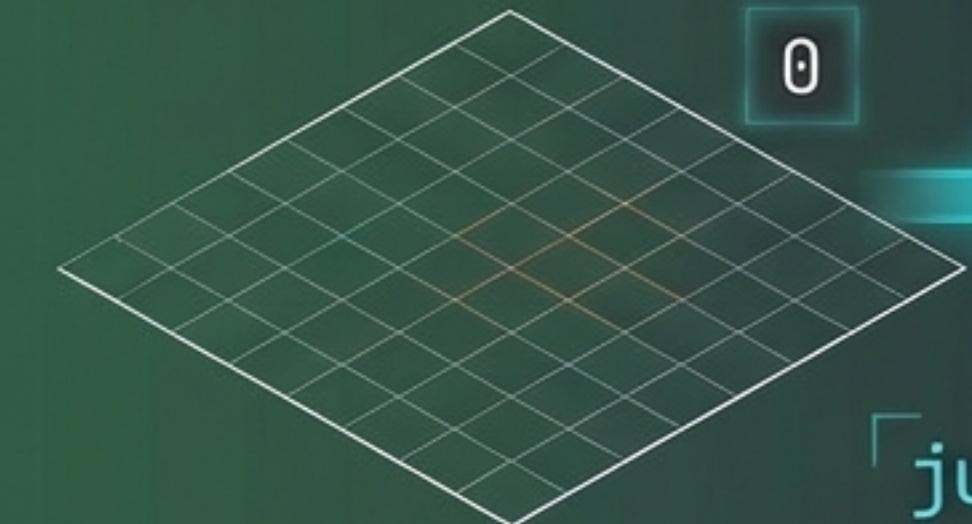
return



VICTORIA

Contador de Aliens == 0.

jugando = false



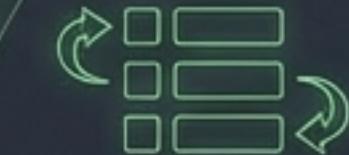
jugando = false

Herramientas Clave de Kotlin



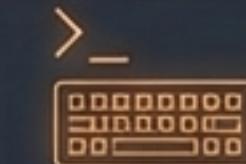
data class

(Estructura de Datos)



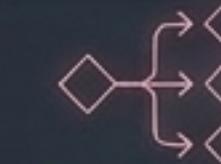
mutableListOf

(Colecciones Dinámicas)



readln()

(Entrada Síncrona)



when

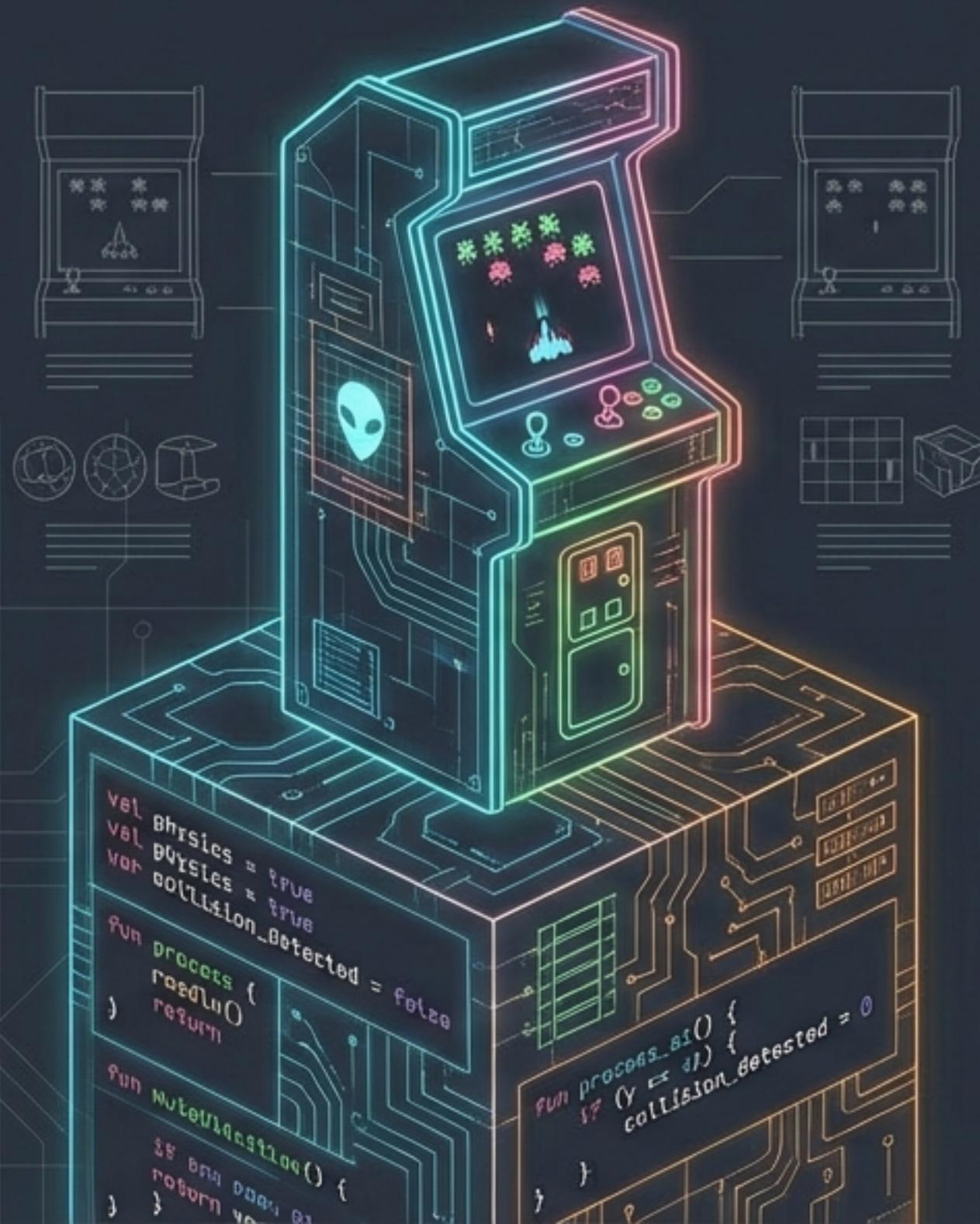
(Control de Flujo)



**Lambda
Filters**

(.find, .count)

La Máquina Viva



Con menos de 150 líneas de código, hemos construido un sistema con física, colisiones e inteligencia artificial básica.

**El código es el engranaje;
la diversión es el resultado.**