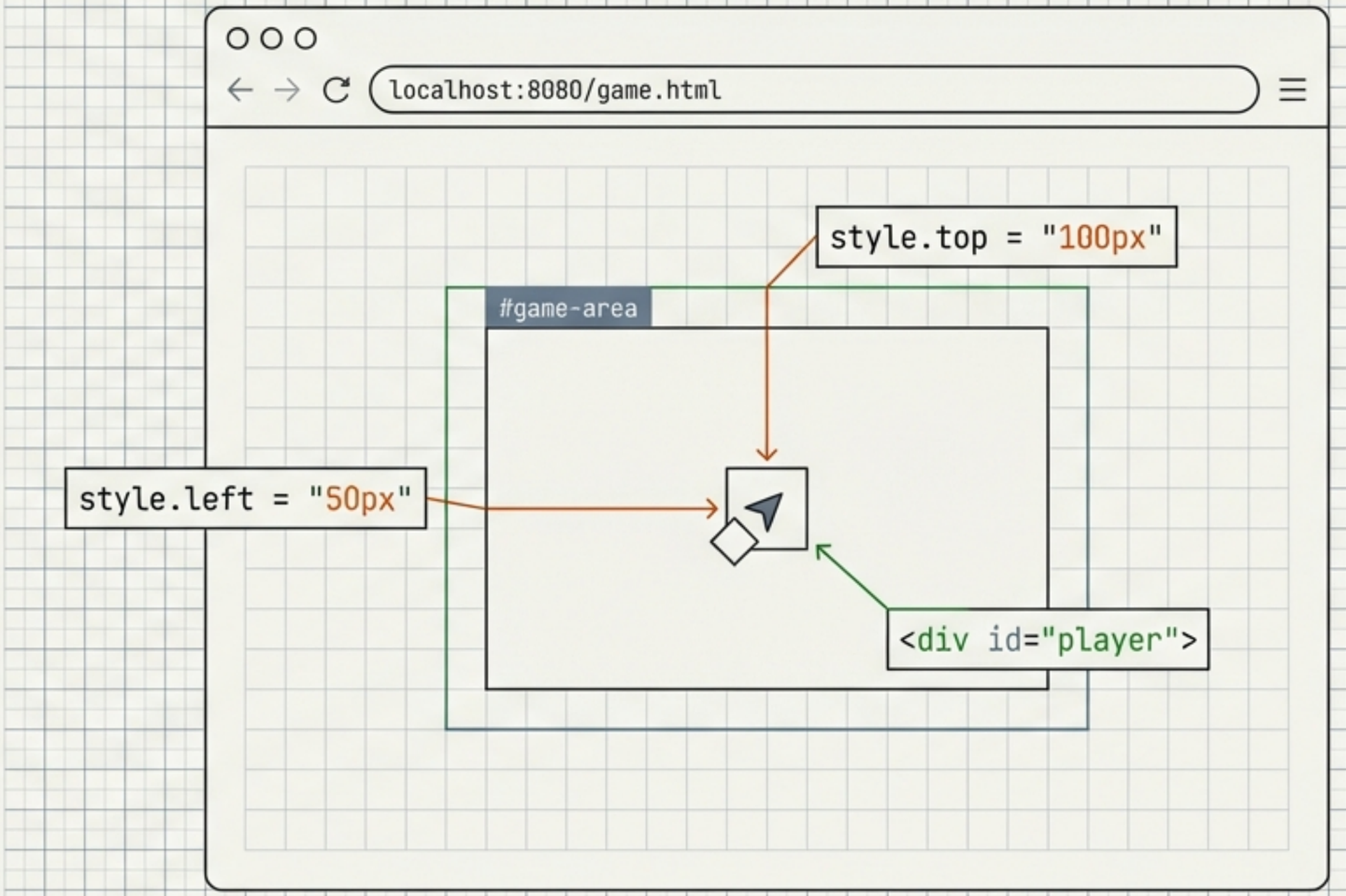
The Browser Is the Engine

A Technical Manifesto for Explicit State and DOM-Based Architecture.

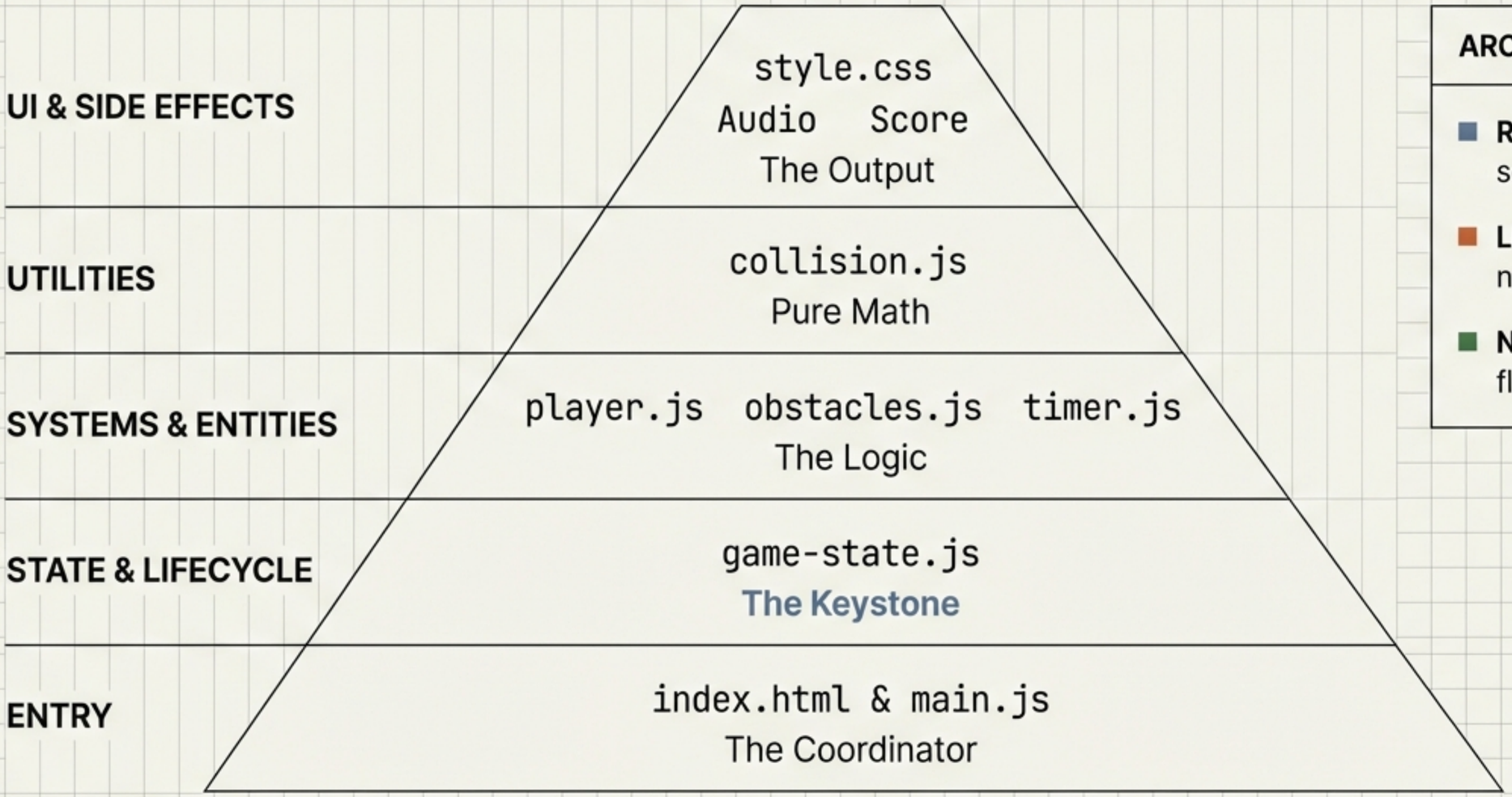
This architecture creates a game with no hidden runtime, no framework, and no “black box” engine. Everything happens because the code explicitly decides it should run in that moment.

- SPECIFICATIONS
- FRAMEWORK: NONE (VANILLA JS)
- PHYSICS: AABB GEOMETRY
- RENDERING: DOM MANIPULATION
- STATE: EXPLICIT STATE MACHINE



Complexity Through Clarity

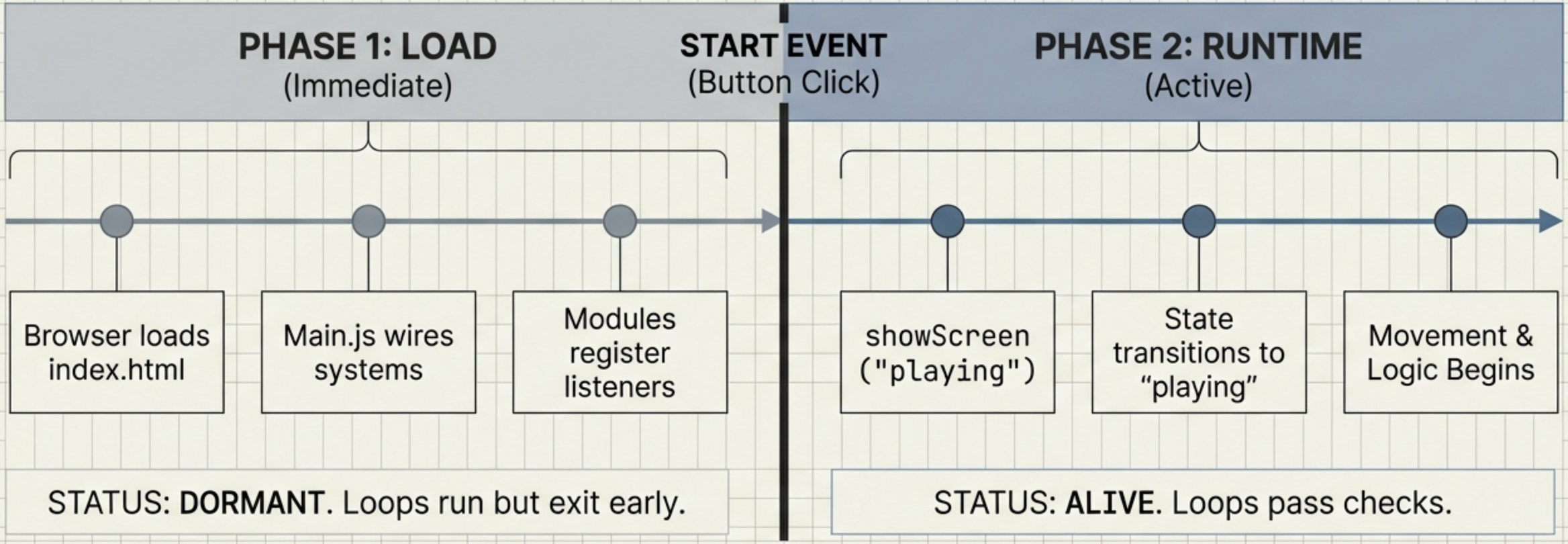
The Architectural Stack



ARCHITECTURAL RULES

- **ROOTS (Main):** Coordinate setup; import systems.
- **LEAVES (Collision):** Import nothing; pure utility.
- **NO CYCLES:** Dependency flow is strictly top-down.

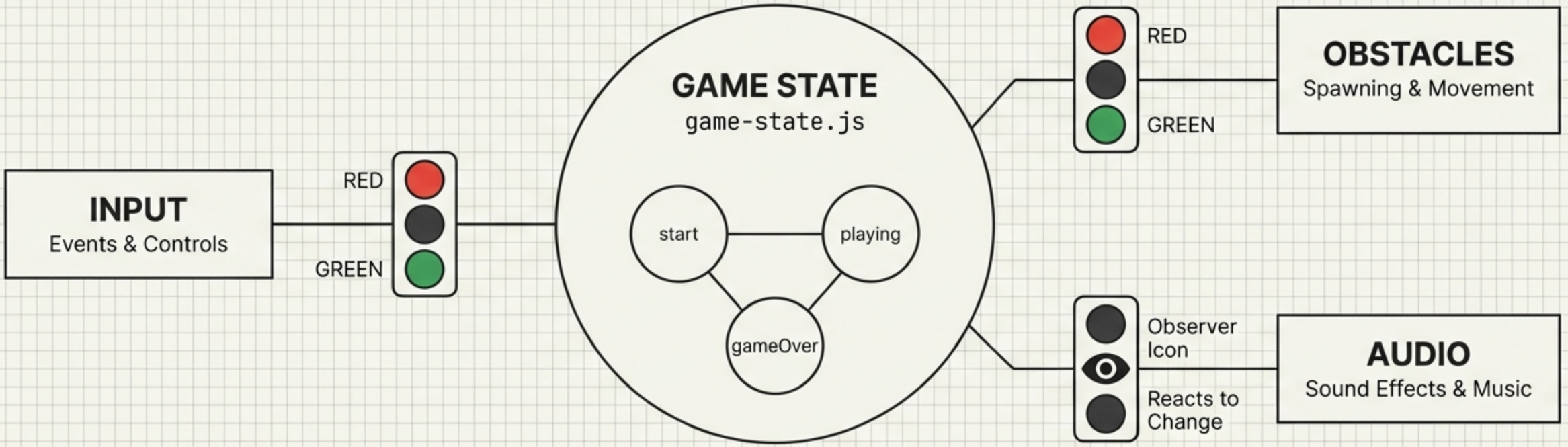
The Boot Flow: Armed and Waiting



Initialization registers the ***potential*** for behavior. **State** permits the ***execution*** of behavior.

State Gates Everything

The Architectural Stack




```
// Inside every loop
if (getGameState() !== "playing") {
  return; // The Gate is Closed
}
```

The Three Timing Models

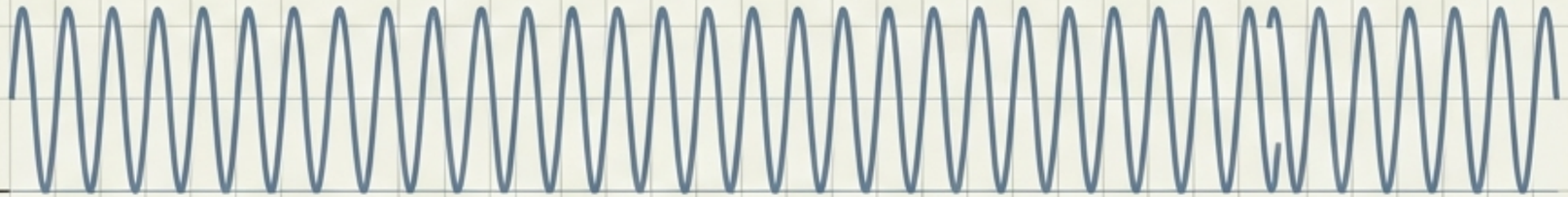
Separation by Purpose

RUNNING IN PARALLEL

PLAYER INPUT
(Zero Latency)

Fires only on keypress. No polling.

OBSTACLES LOOP
(50ms / 20Hz)



Smooth motion. Independent of frame rate.

TIMER LOOP
(1000ms / 1Hz)

High accuracy. Drift prevention.

The DOM Is The Reality

The Architectural Stack

Visual Reality

#game-area

X,Y

A,B

No Virtual Physics World.
State = Style.

Code Reality

```
<div id="game-area">
  <div id="player"
    style="left: 50px; top: 200px;">
  </div>
  <div id="box"
    style="left: 300px; top: 200px;">
  </div>
</div>
```

The Entity Model

Roles and Lifecycles

Static Entities (Persist)

Player

Controls own position. Observes bounds. Cannot change global state.

Box

Passive. Moves only when pushed by Player.

Goal

Static. Owns the 'Win' trigger.

Dynamic Entities (Created/Destroyed)

Cones

Static obstacles. Created/cleared by obstacles.js.

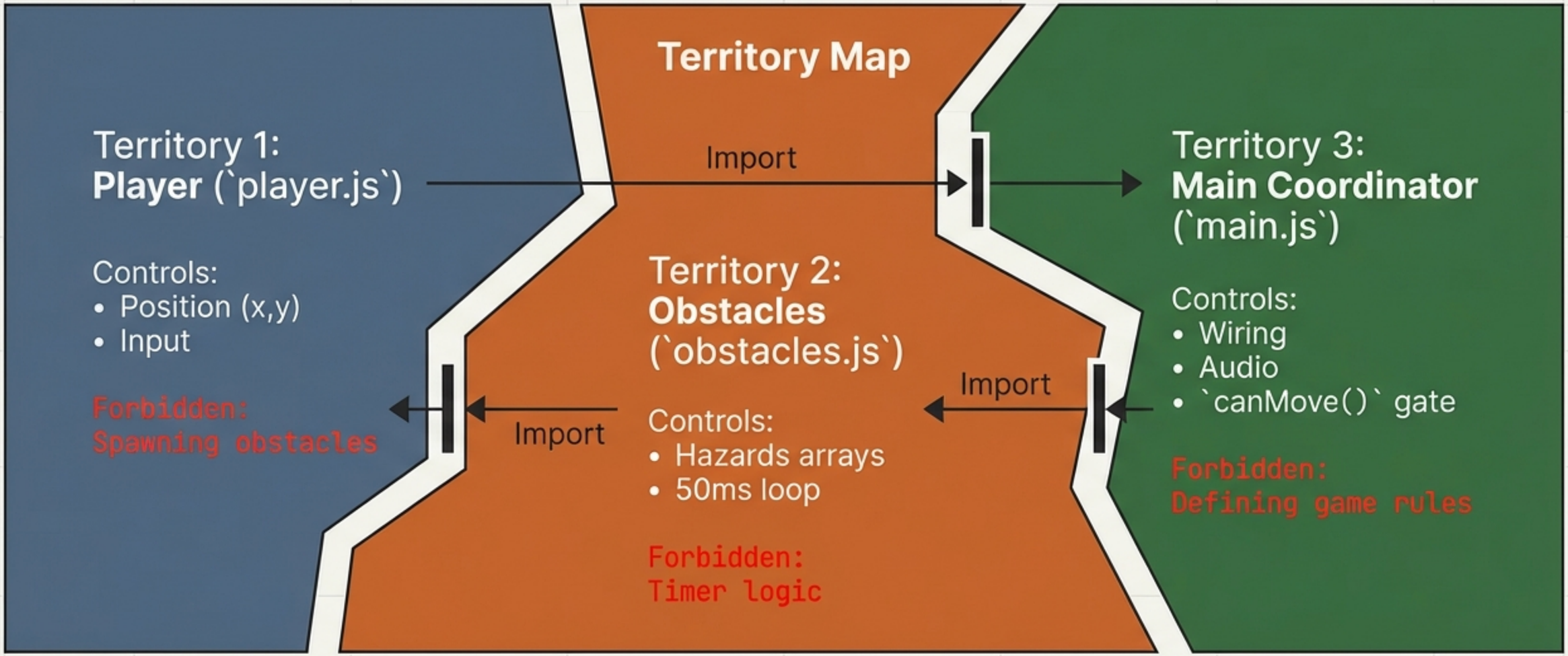
Persons

Moving hazards. Lifecycle managed by 50ms loop.

Rat

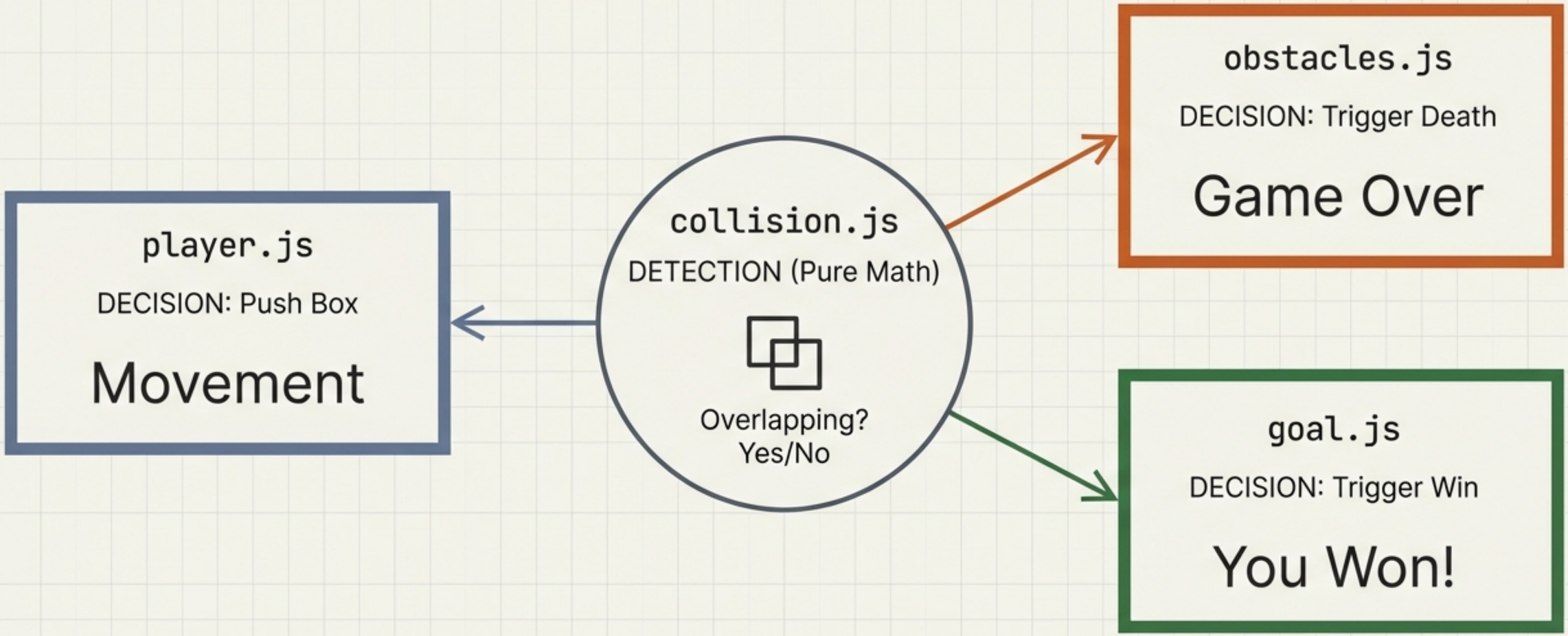
Temporary hazard. 5-second self-destruct timer.

Responsibility Boundaries



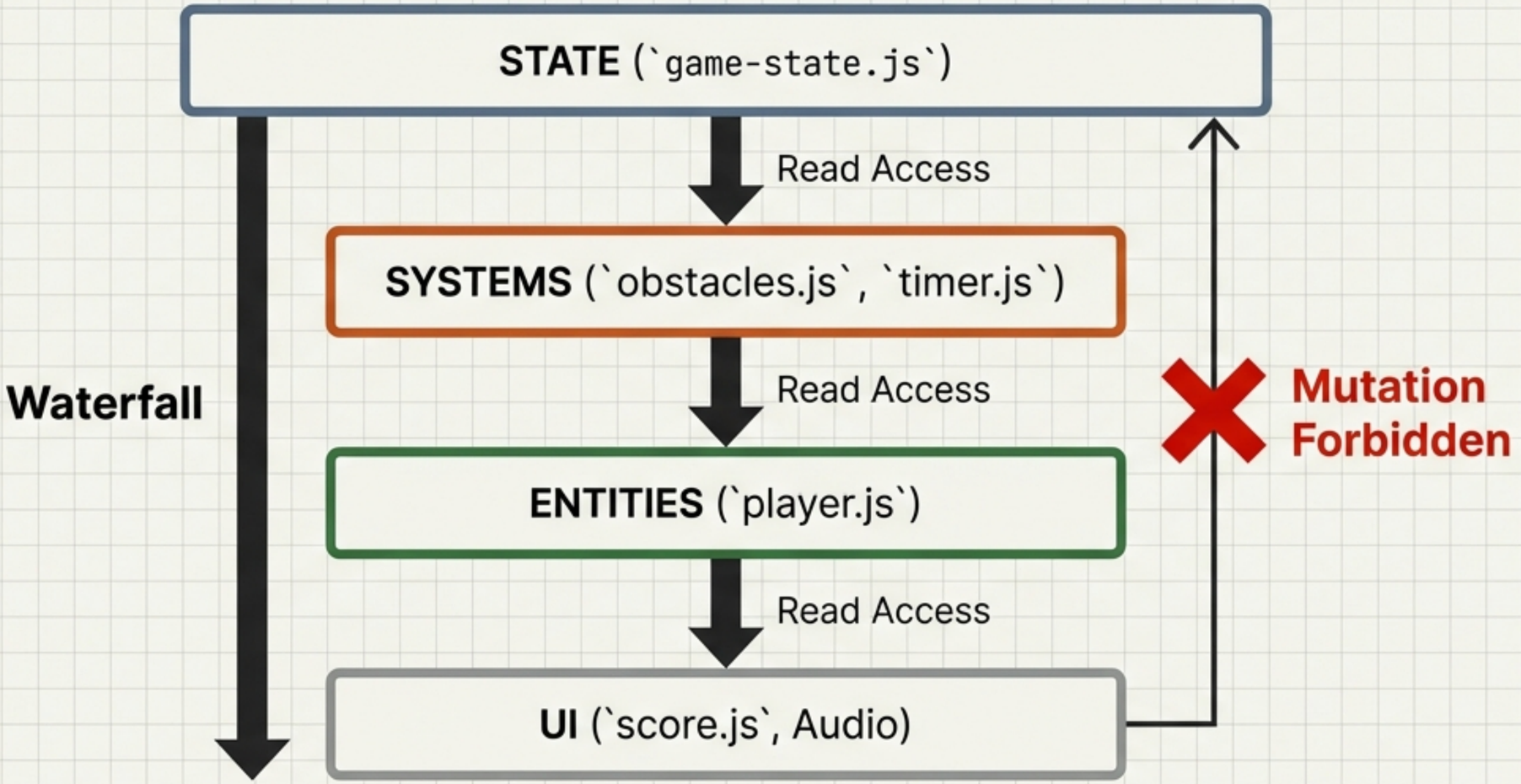
"Files are boundaries, not buckets. Stay in your lane."

Collision: Detection vs. Decision



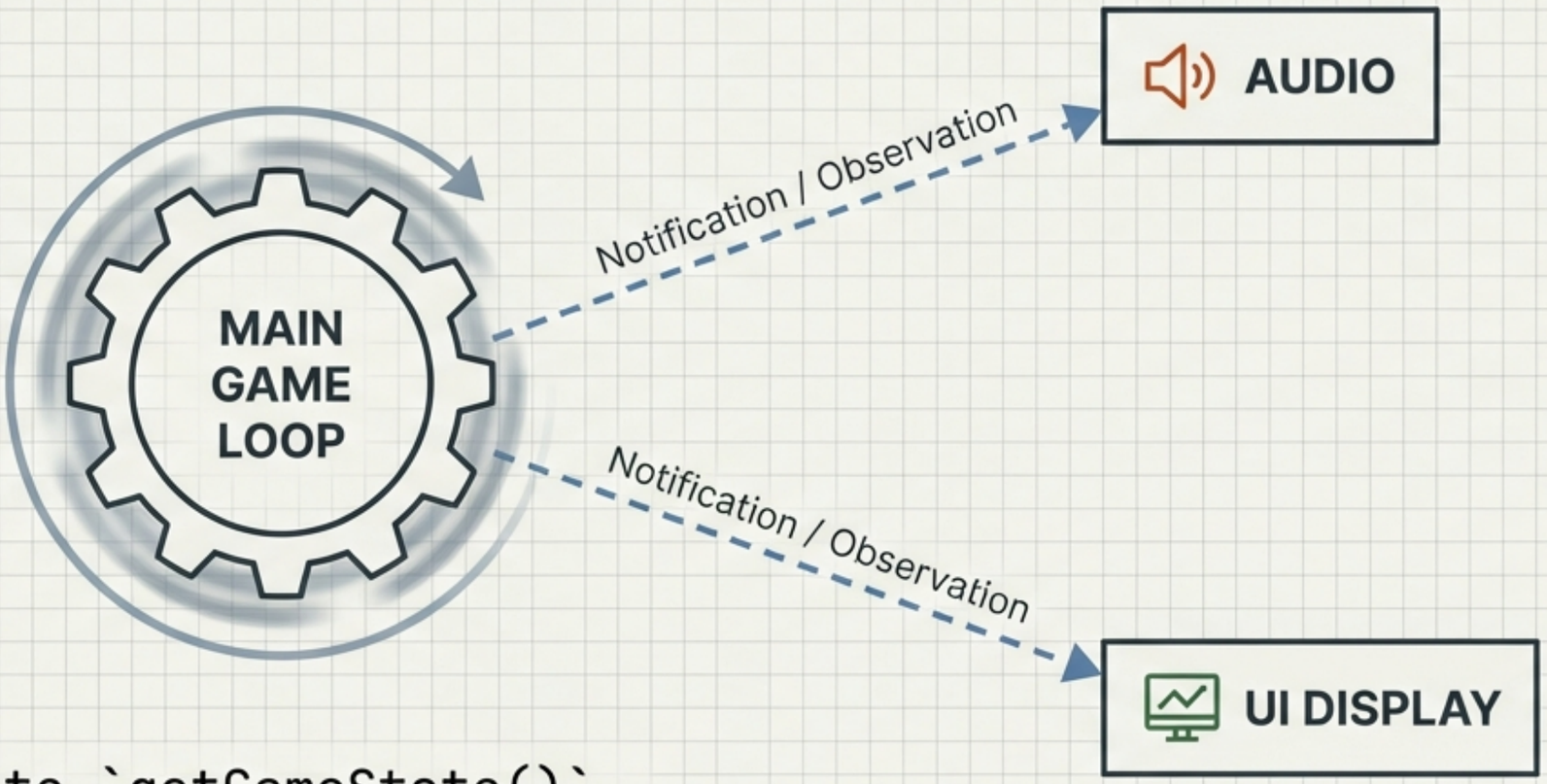
Math detects. Rules decide. The utility knows nothing about the game.

Unidirectional Data Flow



Data flows **down**. It never climbs back up. Only the owner mutates.

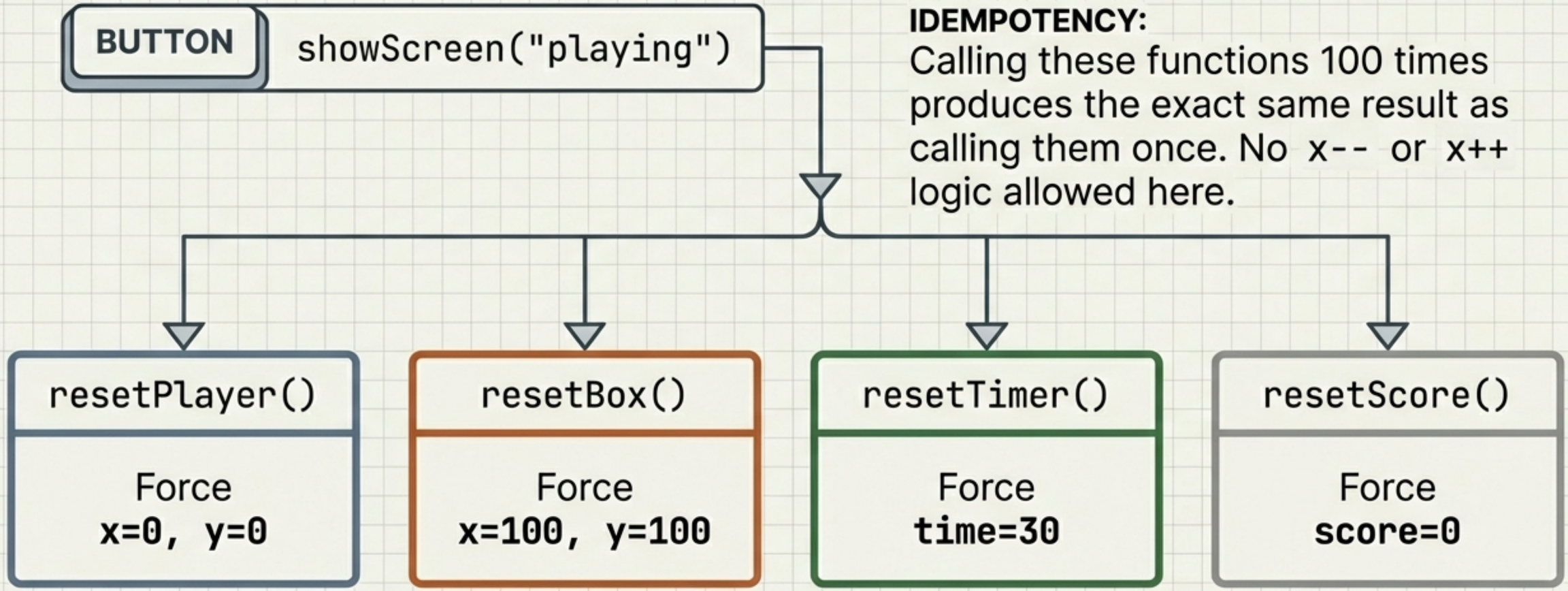
Side Effects Are Observers



Audio listens to `getGameState()`.
Audio never changes state.
Mute button affects speakers, not logic.

Reset Logic & Idempotency

Overwrite, Don’t Calculate



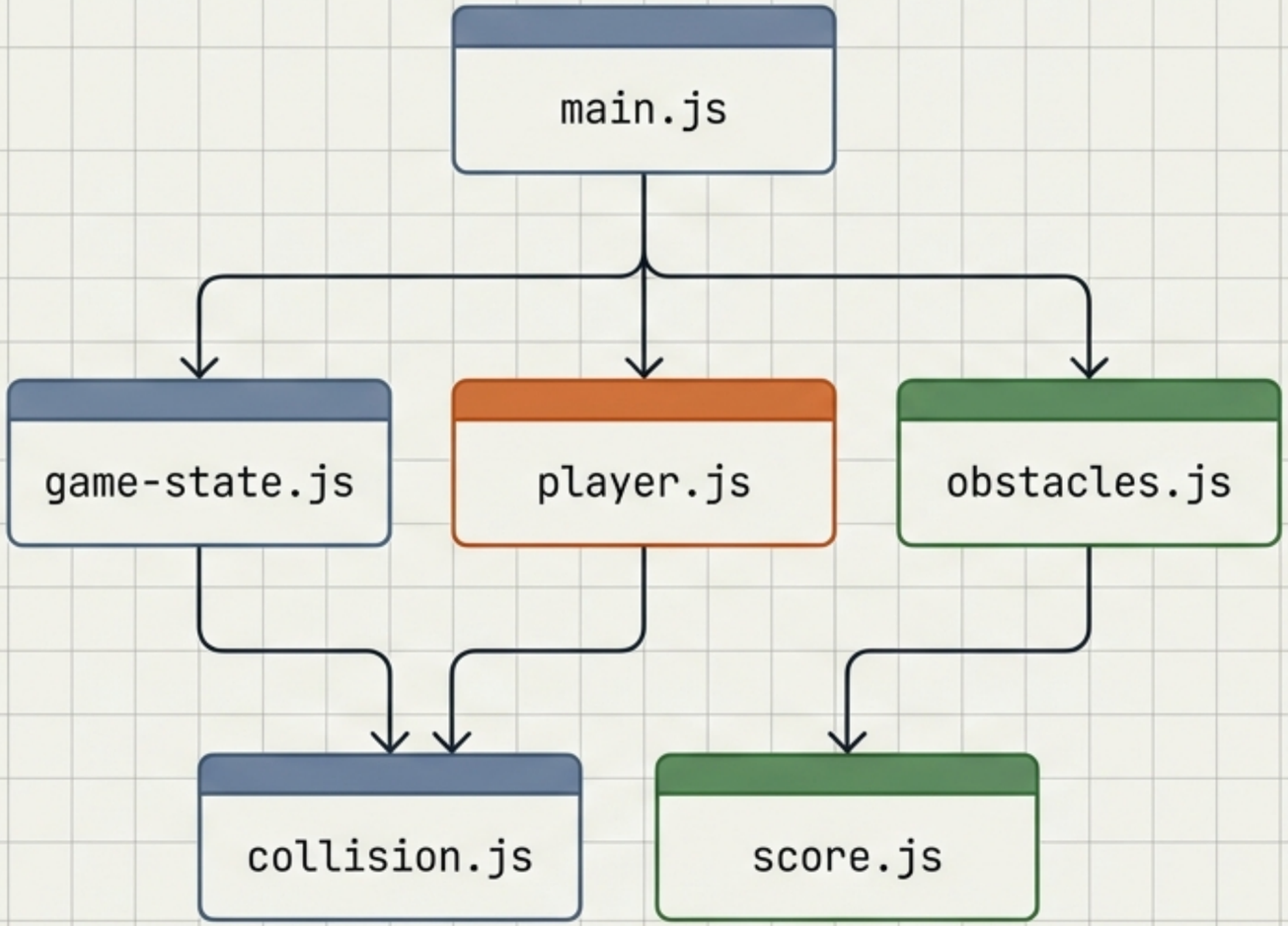
IDEMPOTENCY:
Calling these functions 100 times produces the exact same result as calling them once. No `x--` or `x++` logic allowed here.

The Dependency Graph

Level 1 - Root

Level 2 - Coordinators

Level 3 - Leaves



KEY PRINCIPLES:

- No Circles. No Knots.
- Roots import Leaves.
- Leaves import Nothing.
- UI imports Nothing.

Architecture Guardrails

What Breaks The Machine

Entity Suicide

Entities deciding their own death.

FIX
`Obstacles.js must decide outcomes.`

God Objects

Putting game rules in collision.js.

FIX
`Keep utilities pure math.`

Loop Merging

Putting Timer logic in the Obstacle loop.

FIX
`Keep timing models separate.`

UI Drivers

UI manipulating variables directly.

FIX
`UI only signals State Intent.`

The Mental Model

Summary of Principles

STATE GATES EVERYTHING

Behavior runs only when the Traffic Light is Green.

SEPARATION OF TIME

Events, Smoothness, and Accuracy live in separate loops.

DETECTION VS DECISION

Math detects collisions; Systems decide consequences.

ONE-WAY FLOW

Data flows down. Only owners mutate.

SIMPLICITY IS STRENGTH