

NANO DUNGEON

Deconstructing Roguelike Development
with JavaScript & Phaser

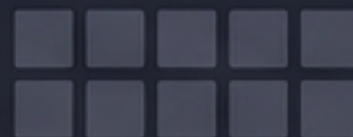
Based on 'Roguelike Development with JavaScript' by Andre Alves Garzia

 The Player

Hp: 13

Mp: 1

Ap: 1



 A Dangerous Monster

 A Dangerous Monster

 A Dangerous Monster

 A Dangerous Monster

 A Dangerous Monster

A Dangerous Monster was
killed.

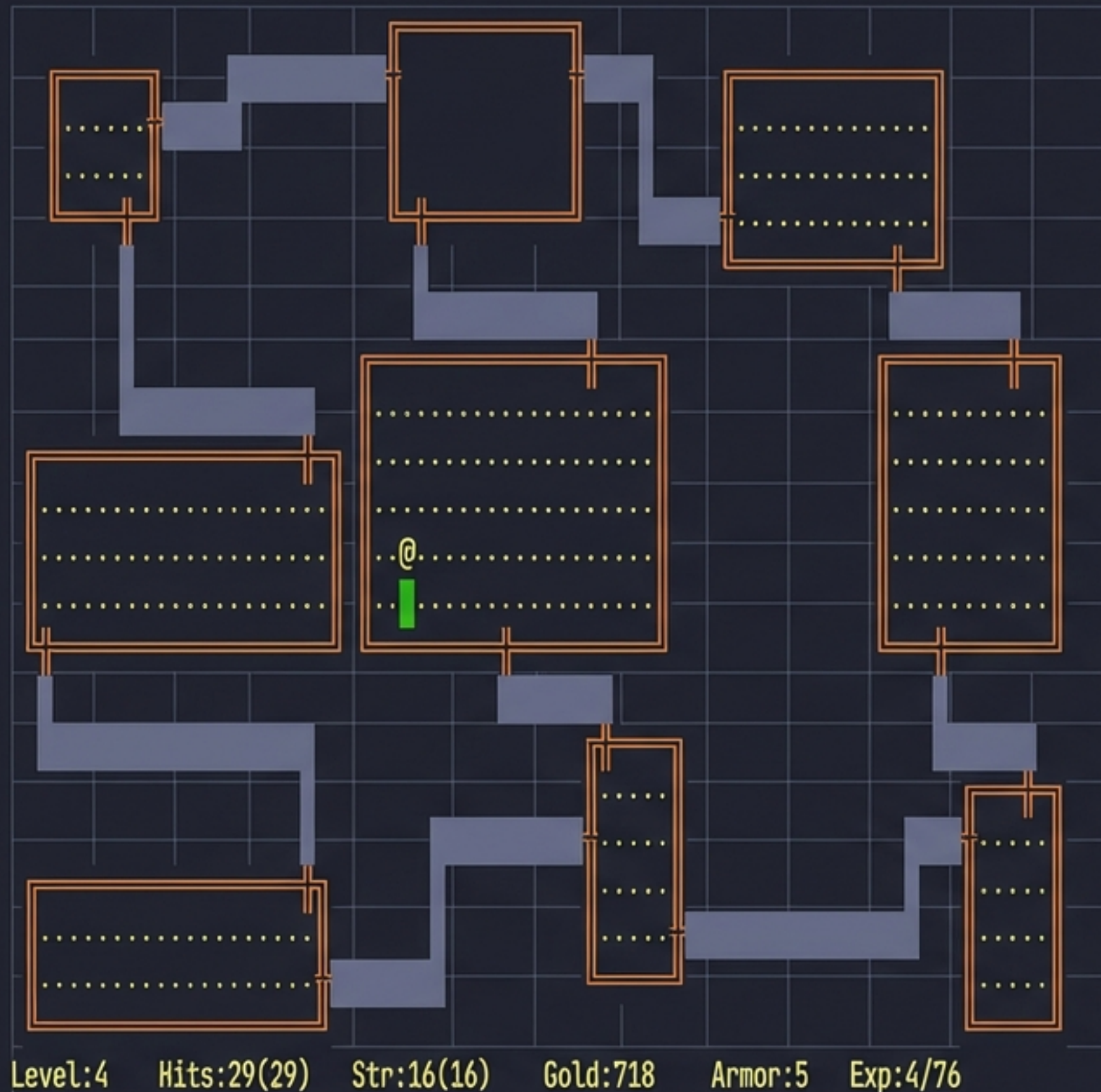
The Player does 1 damage to A
Dangerous Monster

A Dangerous Monster does 1
damage to The Player

A Dangerous Monster was
killed

The Player does 1 damage to A
Dangerous Monster

A Dangerous Monster does 1
damage to The Player



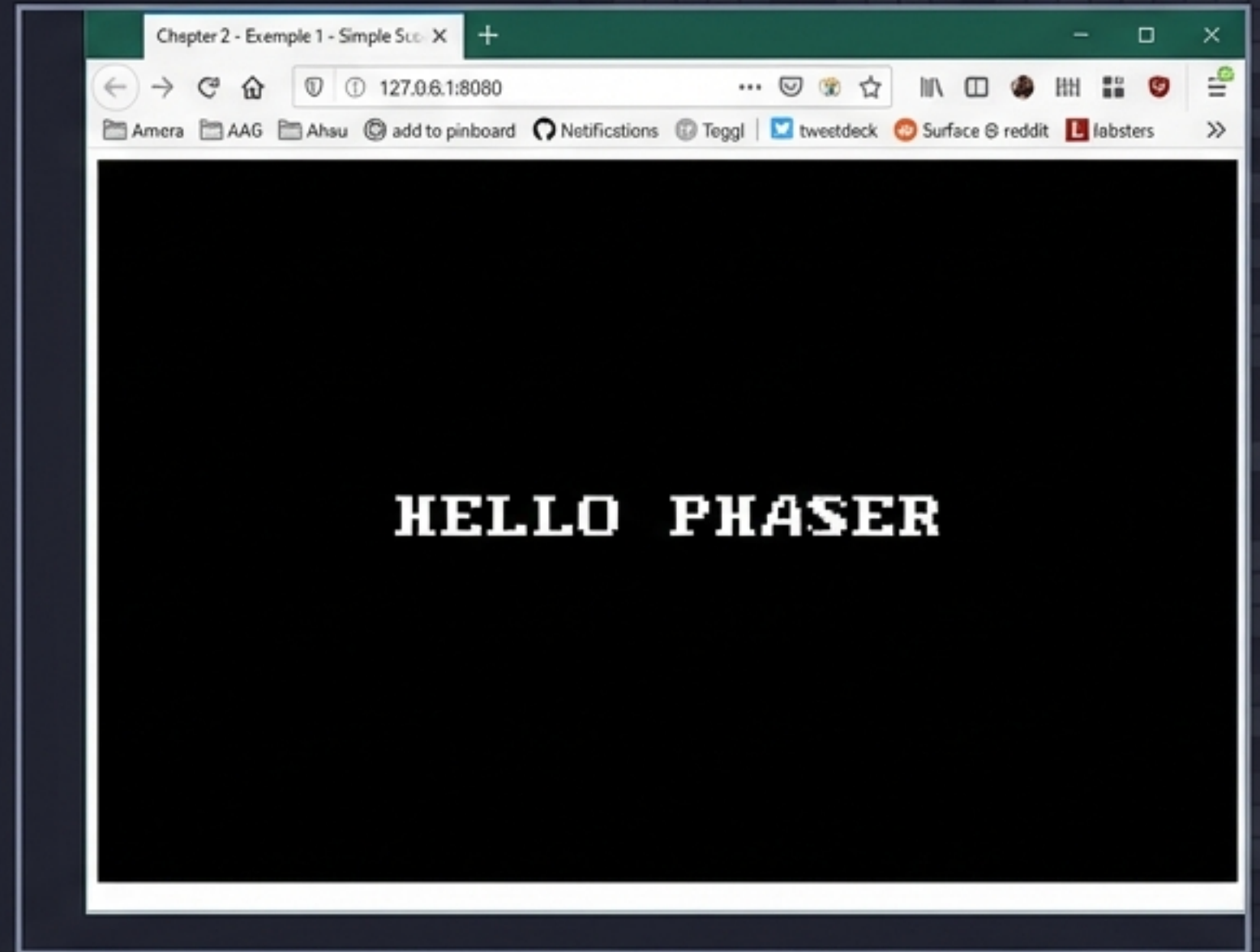
The Berlin Interpretation: 5 Key Pillars

- **Random Generation:** The world is new every run.
- **Permadeath:** No saving; death is final.
- **Turn-Based:** The world waits for you.
- **Grid-Based:** Everything exists on a discrete chessboard.
- **Resource Management:** Scarcity drives strategy.

The Tech Stack: Why Web?

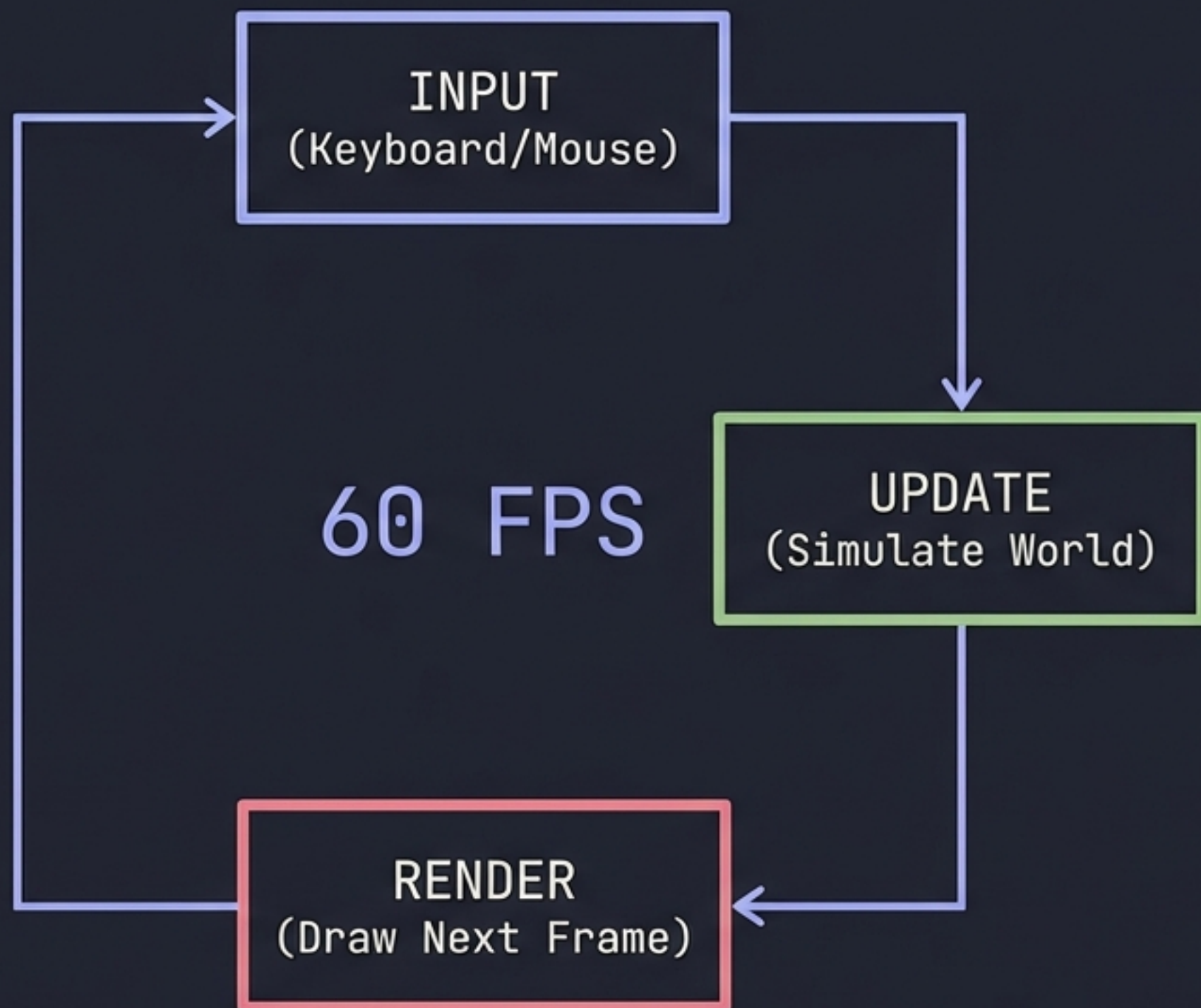
- ⇒ **Zero-Friction:** Runs in any browser.
- **Rapid Iteration:** Hot-reload with live-server.

“Always bet on JS.”



```
const config = {  
  type: Phaser.AUTO, // WebGL or Canvas  
  pixelArt: true,    // Crisp retro visuals  
  physics: { default: 'arcade' }  
};
```


The Heartbeat: The Game Loop



The Conflict:

Phaser runs at 60 frames per second (Real-time), but a Roguelike moves one step at a time (Turn-based).

We must enforce patience on the engine.

Anatomy of a Scene

PRELOAD

```
this.load.spritesheet(..., 'phaser/sv3.png" .png');
```

CREATE

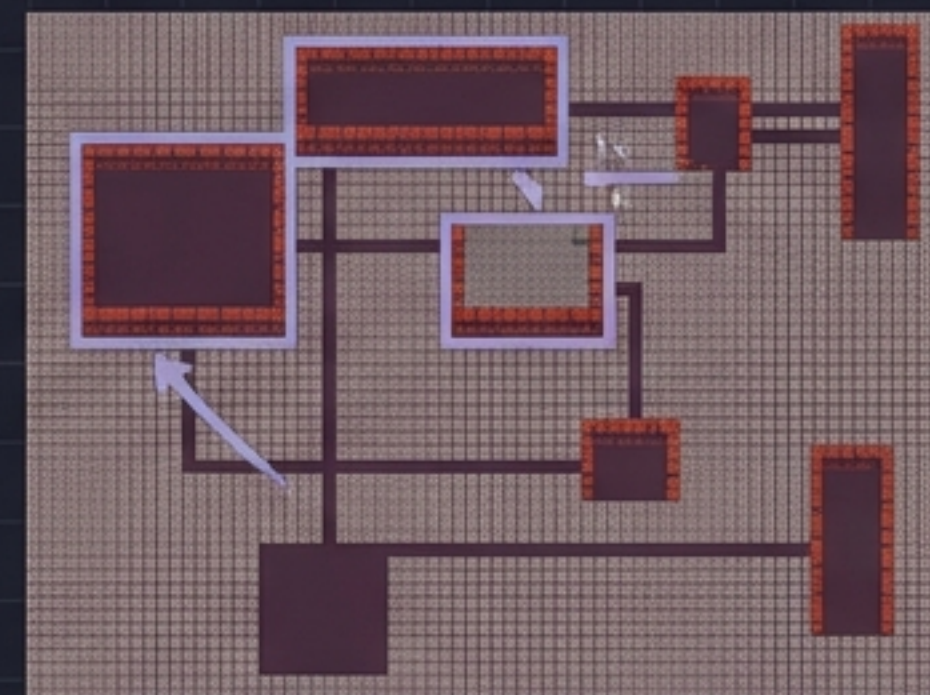
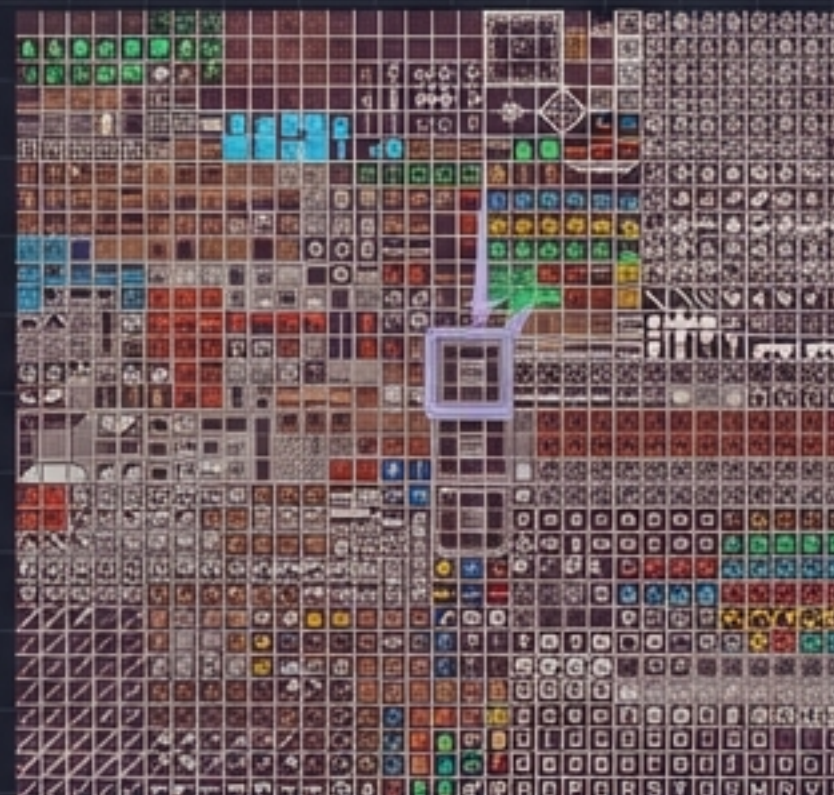
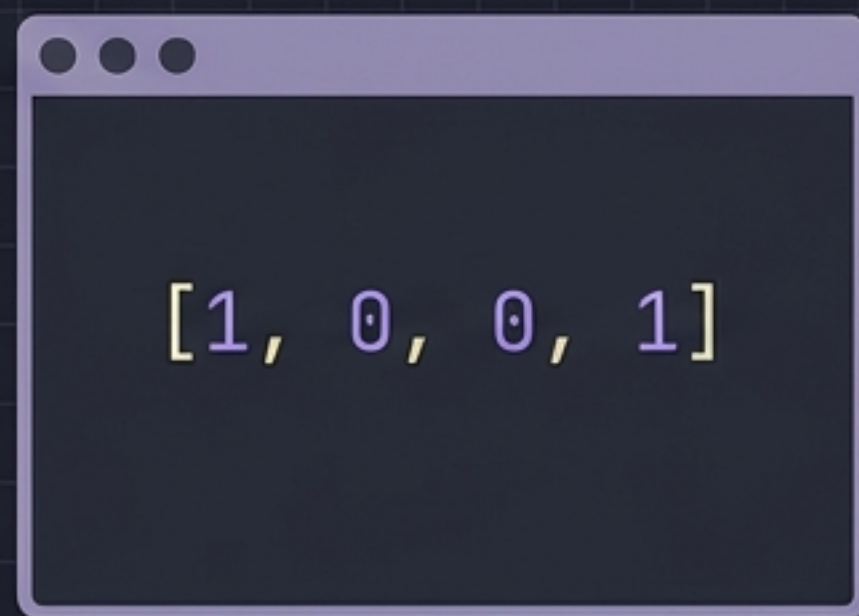
```
this.add.sprite(..., 2, color 60, );
```

UPDATE

```
if (input) {  
  move()  
}
```

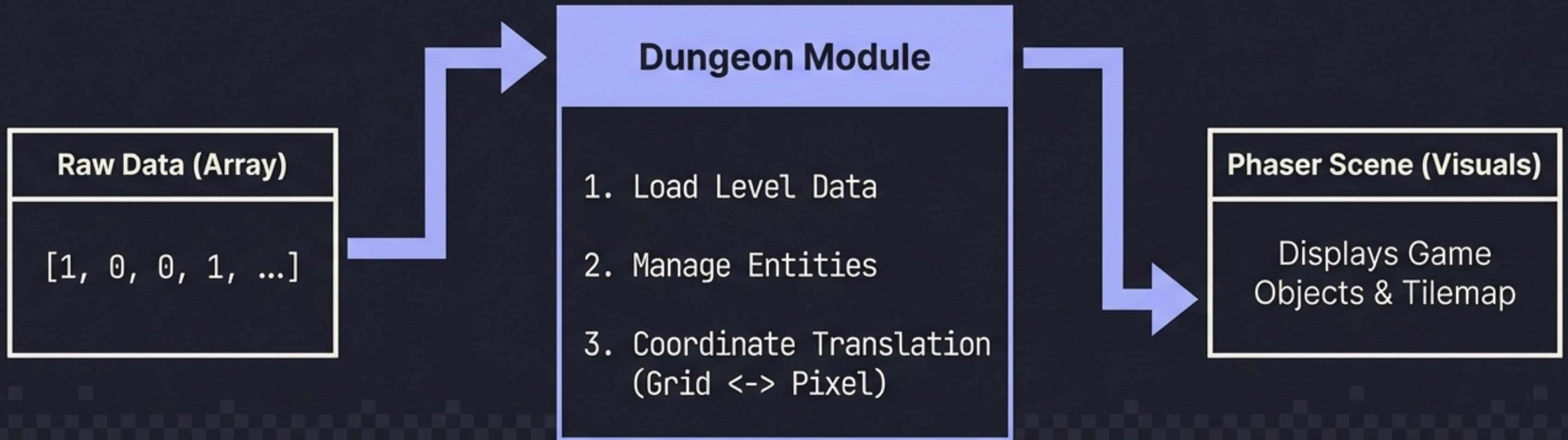
The Phaser
Scene Class

The Grid is the World

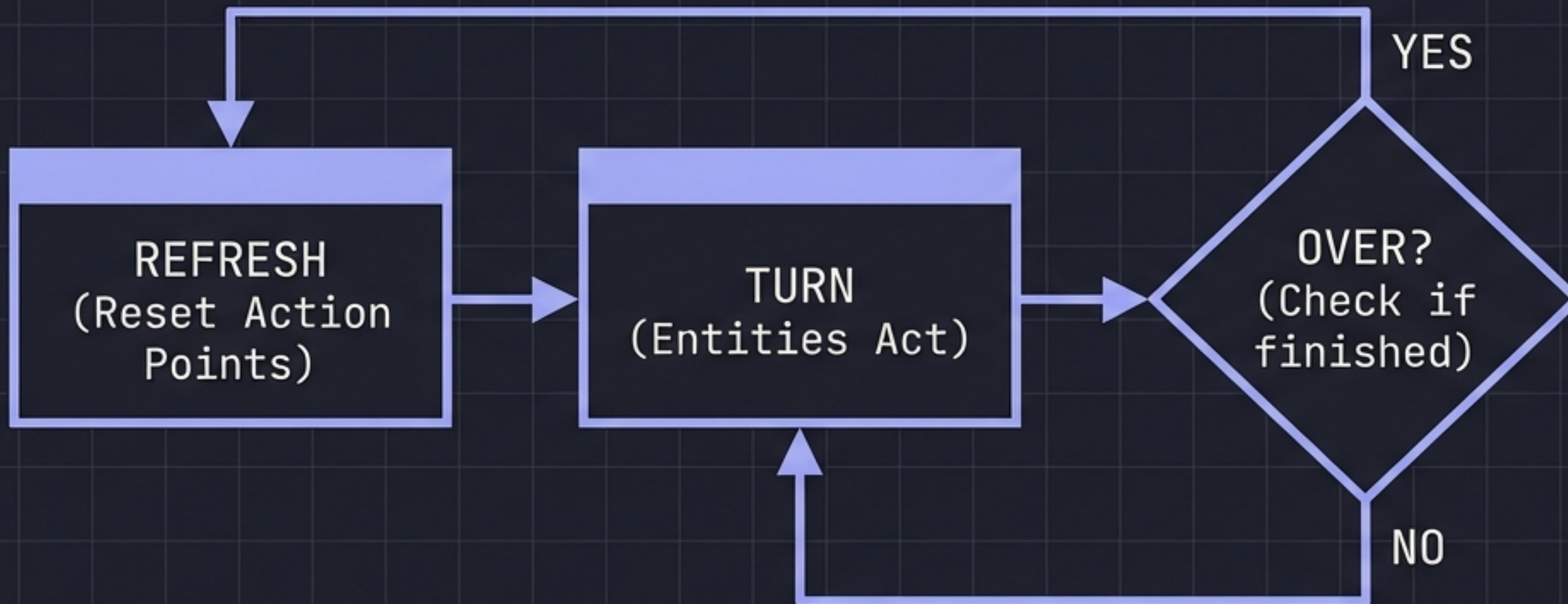


Data + Assets = Tilemap

The Architect: The Dungeon Manager



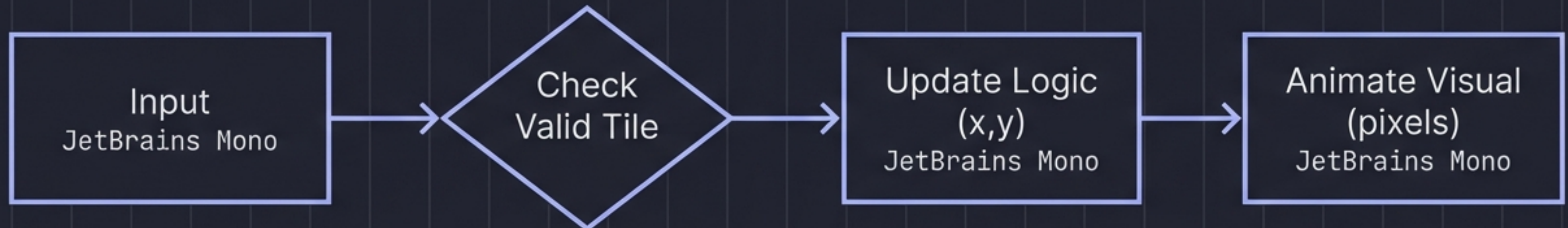
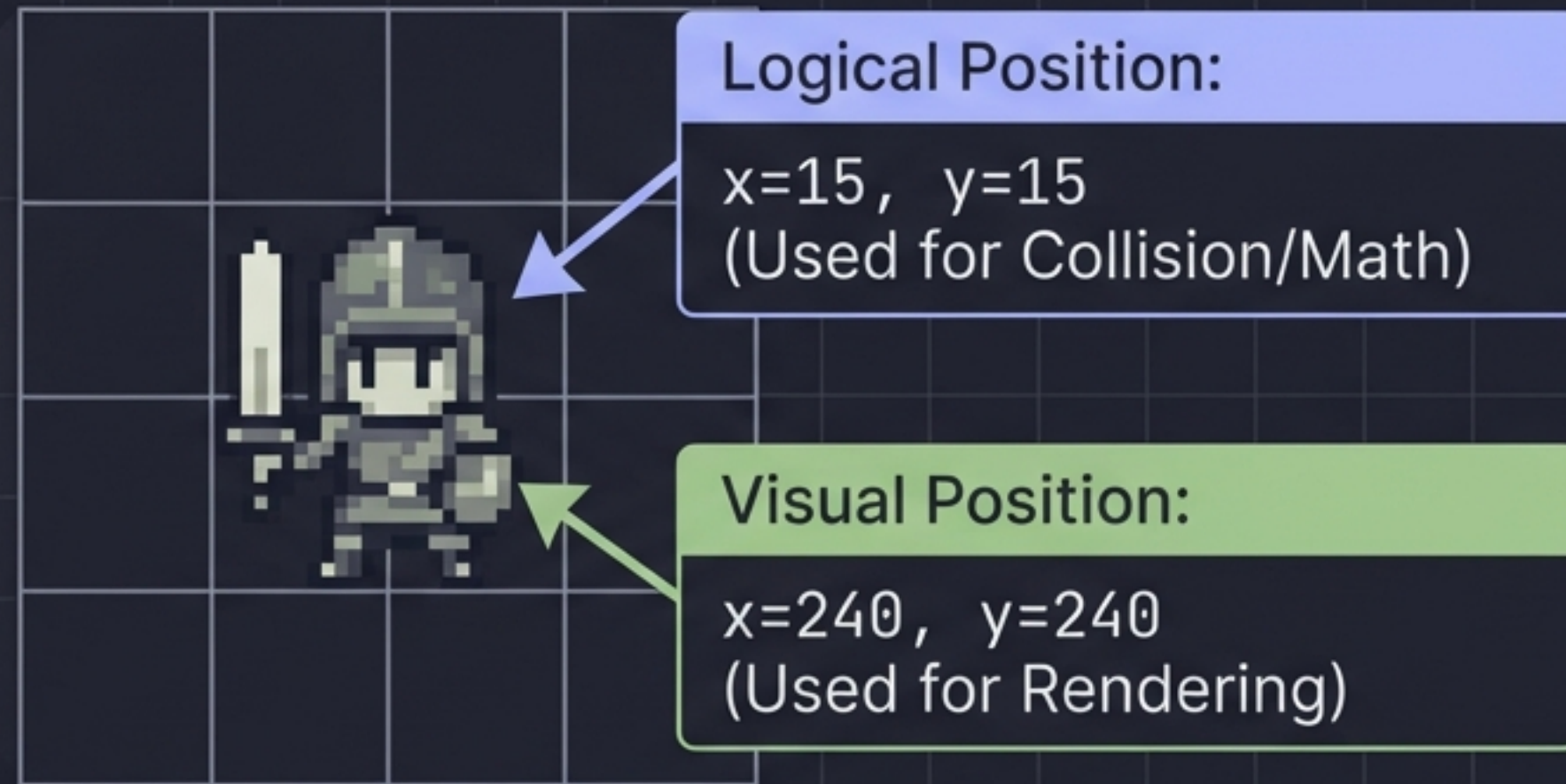
Controlling Time: The Turn Manager



The Debounce:

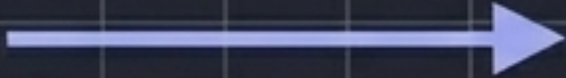
A **150ms** delay is added between **inputs** to prevent the player from accidentally sprinting across the map.

The Protagonist: Grid vs. Pixel



Smooth Operator: Tweens & Animation

Teleport vs. Glide



Instant Update (Jarring)

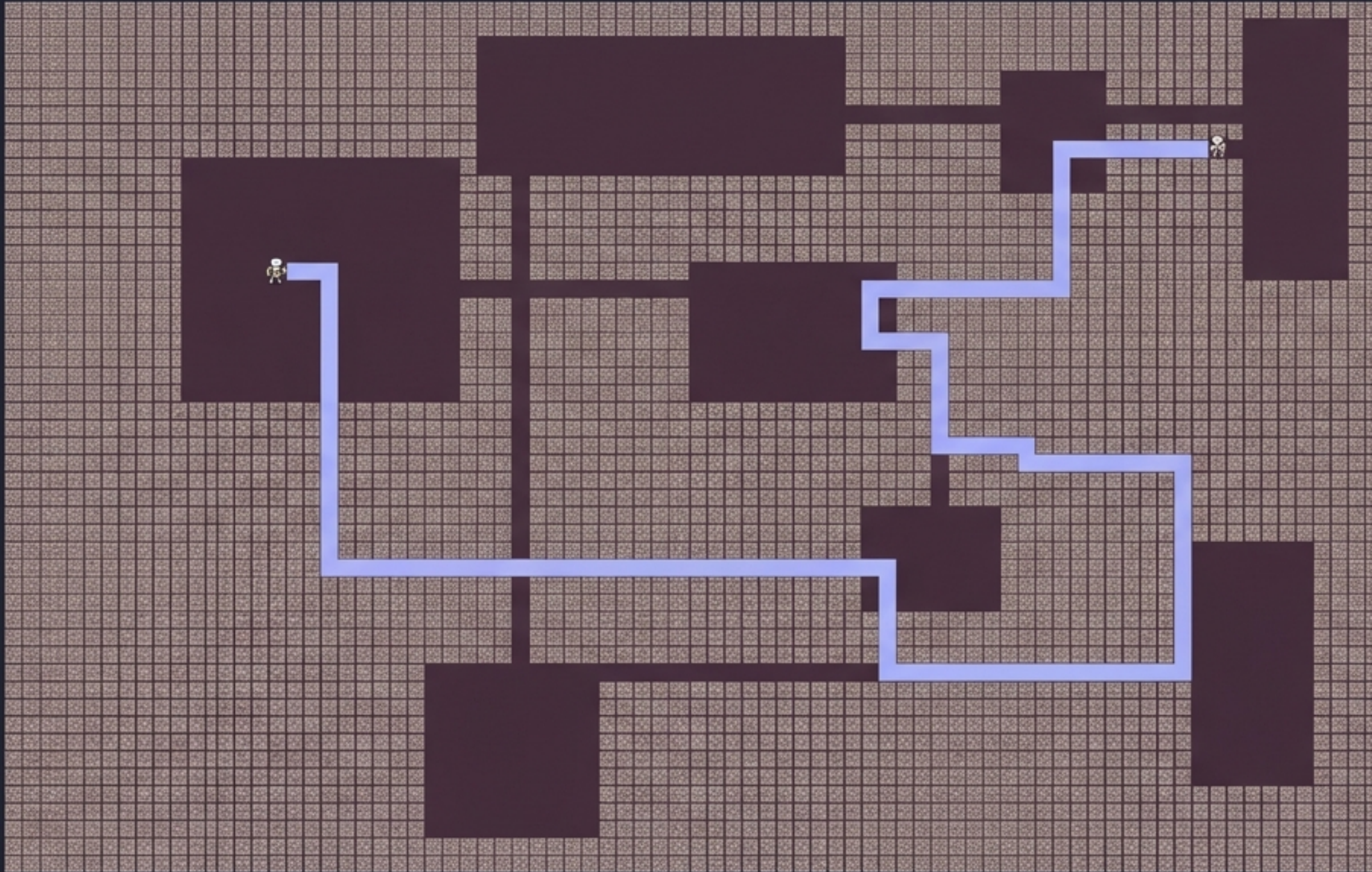


B

Tweened Movement (Fluid)

```
this.tweens.add({  
  targets: entity.sprite,  
  x: newX,  
  y: newY,  
  duration: 200, // 200ms glide  
  ease: "Power2"  
});
```


The Conflict: AI & Pathfinding



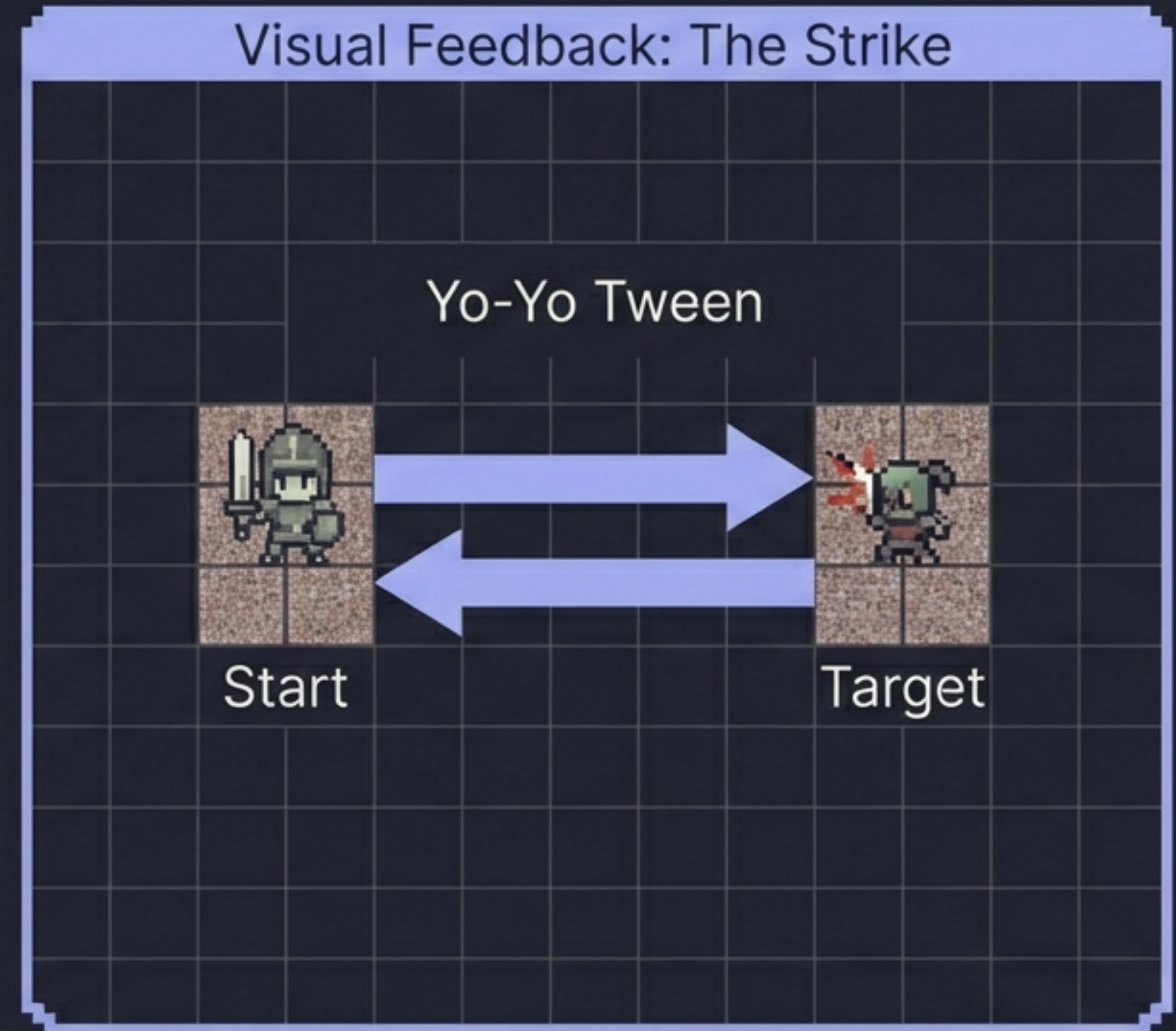
The Loop

1. Calculate path to Player (A* Algorithm).
2. If `Distance > 1`:
Move one step closer.
3. If `Distance = 1`:
ATTACK.

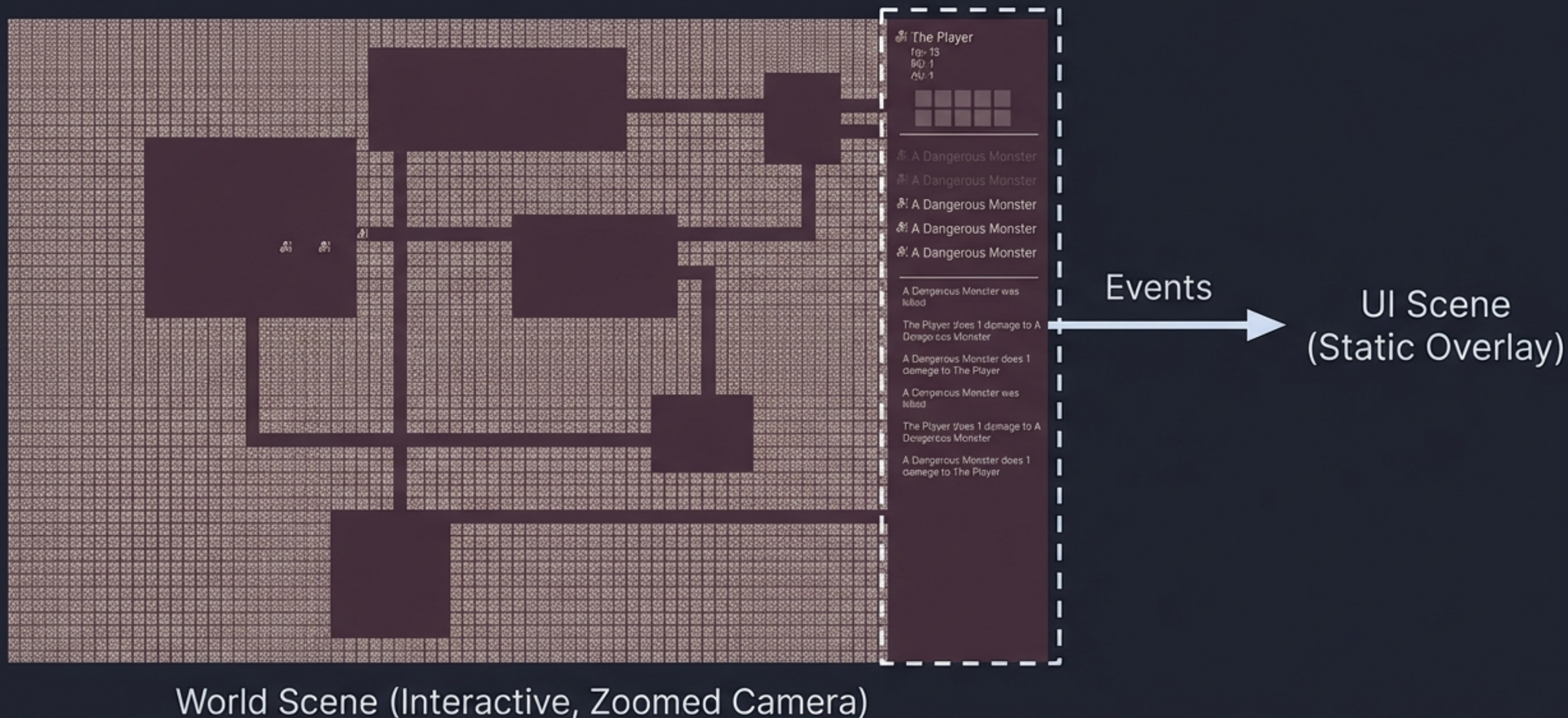
Combat & Permadeath

The Stakes

- ❖ Movement Points (**MP**):
To walk.
- ❖ Action Points (**AP**):
To strike.
- ❖ **Permadeath**: If $HP \leq 0$,
Reload Game.



The Overlay: Scene Stacking



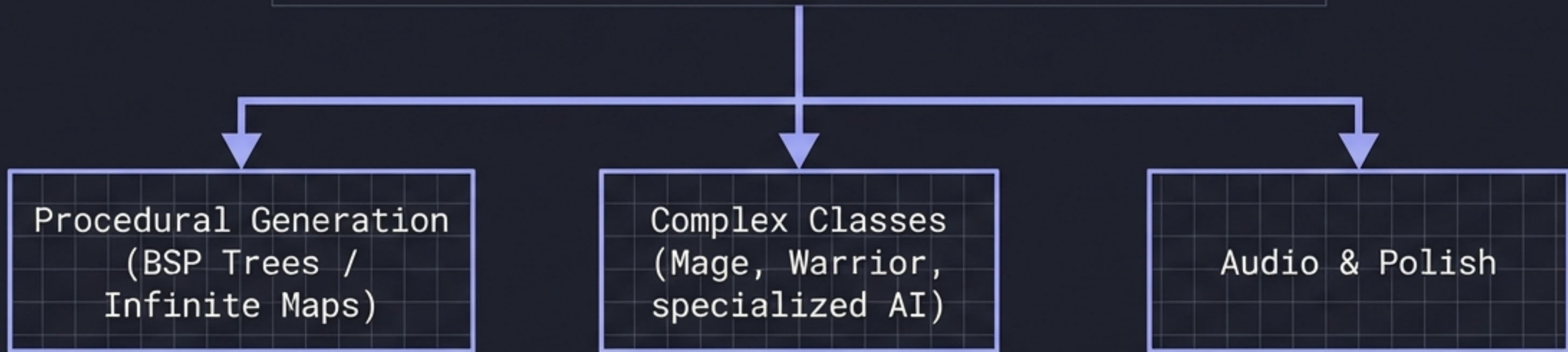
Loot & Inventory



- **Items are Entities:** They exist in the world logic.
- **Equip Logic:** Toggles 'active' flag.
- **Impact:** Active items modify the 'Attack()' calculation.

Beyond the Static Map

The Skeleton is Complete.



"Your journey starts when you change the code."