Below is a targeted guide to the project’s configuration system—how it’s structured, where it’s loaded, and exactly which simulation parameters you can tweak via the YAML file without touching any core Python code. At the end is a brief checklist to ensure we’ve covered every config-driven aspect of the game logic.

---

## 1. Project Layout: Where Configuration Lives

\* \*\*`config.yaml`\*\* sits at the project root alongside `\_\_main\_\_.py`.&#x20;

\* The loader is implemented in \*\*`core/config\_loader.py`\*\*, where `load\_config()` reads and validates that file.&#x20;

## 2. Loading Logic: `core/config\_loader.py`

\* \*\*Function\*\*:

```python

def load\_config(config\_path: str = DEFAULT\_CONFIG\_PATH) -> Dict[str, Any]:

... # safe\_load + type check + required-keys validation

```

\* \*\*Default Path\*\*:

```python

DEFAULT\_CONFIG\_PATH = "/home/ubuntu/coin\_clash/config.yaml"

```

You can override this by passing a different `config\_path` when calling `load\_config()`.

\* \*\*Validation\*\*: Throws an explicit error if any of these keys are missing:

````

min\_fee, default\_fee, max\_fee,

kill\_award\_rate\_min, kill\_award\_rate\_default, kill\_award\_rate\_max,

num\_players\_min, num\_players\_default, num\_players\_max,

chars\_per\_player\_min, chars\_per\_player\_max,

protocol\_cut, primary\_event\_weights, extra\_events, lethal\_modifiers

``` :contentReference[oaicite:4]{index=4}:contentReference[oaicite:5]{index=5}

````

## 3. Configurable Simulation Parameters (`config.yaml`)

You can modify \*\*all\*\* of the following without touching code:

### A. Entry Fees

````yaml

min\_fee: 0.5 # Minimum allowable entry fee per character

default\_fee: 1.0 # Fee used if no override

max\_fee: 5.0 # Maximum allowable entry fee

``` :contentReference[oaicite:6]{index=6}:contentReference[oaicite:7]{index=7}

### B. Kill-Award Rates

```yaml

kill\_award\_rate\_min: 0.0 # 0% of victim’s fee

kill\_award\_rate\_default: 0.1 # 10% of victim’s fee

kill\_award\_rate\_max: 0.5 # 50% of victim’s fee

````

These determine the slice of entry fees awarded per kill.&#x20;

### C. Simulated Players

````yaml

num\_players\_min: 3 # Minimum total participants

num\_players\_default: 10 # Used on startup unless overridden

num\_players\_max: 50 # Absolute cap

``` :contentReference[oaicite:10]{index=10}:contentReference[oaicite:11]{index=11}

### D. Characters per Player

```yaml

chars\_per\_player\_min: 1 # Each player must buy ≥ this number

chars\_per\_player\_max: 3 # Each player may buy ≤ this number

``` :contentReference[oaicite:12]{index=12}:contentReference[oaicite:13]{index=13}

### E. Protocol Cut Tiers

```yaml

protocol\_cut:

"1": 0.10 # 1 character ⇒ 10% taken

"2": 0.08 # 2 characters ⇒ 8% taken

"3": 0.06 # 3 characters ⇒ 6% taken

````

Affects how much of the entry pool is reserved versus prize-pool.&#x20;

### F. Primary Event Weights

```yaml

primary\_event\_weights:

direct\_kill: 50 # 50% chance

self: 15 # 15% chance

environmental: 20 # 20% chance

group: 5 # 5% chance

story: 10 # 10% chance (non-lethal)

```

Controls the distribution of the round’s single “primary” event.&#x20;

### G. Additional Events

````yaml

extra\_events:

non\_lethal\_story\_chance: 0.10 # 10% chance extra story

extra\_lethal\_base\_chance: 0.02 # 2% base chance extra kill

comeback\_base\_chance: 0.15 # 15% chance to resurrect (if dead pool non-empty)

``` :contentReference[oaicite:18]{index=18}:contentReference[oaicite:19]{index=19}

### H. Dynamic Lethal Modifiers

```yaml

lethal\_modifiers:

cap\_8\_plus: 0.10 # +10% extra kill chance when >8 alive

cap\_12\_plus: 0.20 # +20% extra kill chance when >12 alive

````

These stack on top of `extra\_lethal\_base\_chance` based on alive-pool thresholds.&#x20;

---

## 4. How to Tweak without Code Changes

1. \*\*Adjust the YAML values\*\* above to change match size, fees, reward rates, or event probabilities.

2. \*\*Add or remove scenarios\*\* by editing the JSON files in `/scenarios/`, but that’s outside `config.yaml`.

3. \*\*Override config path\*\* if you want multiple presets:

````python

from core.config\_loader import load\_config

config = load\_config("/path/to/your/custom\_config.yaml")

``` :contentReference[oaicite:22]{index=22}:contentReference[oaicite:23]{index=23}

````

4. \*\*Seed control\*\* for reproducible randomness lives in `MatchEngine`, not in config, but you can pass in a different `random\_seed` at instantiation.

---

## 5. Validation Checklist

Ensure each of these config-driven aspects is included in your custom YAML and recognized at runtime:

\* [ ] \*\*Entry Fees\*\* (`min\_fee`, `default\_fee`, `max\_fee`)

\* [ ] \*\*Kill-Award Rates\*\* (`kill\_award\_rate\_min/default/max`)

\* [ ] \*\*Player Counts\*\* (`num\_players\_min/default/max`)

\* [ ] \*\*Per-Player Character Limits\*\* (`chars\_per\_player\_min/max`)

\* [ ] \*\*Protocol Cuts\*\* (tiers “1”, “2”, “3”)

\* [ ] \*\*Primary Event Weights\*\* (all five categories)

\* [ ] \*\*Additional Event Chances\*\* (`non\_lethal\_story\_chance`, `extra\_lethal\_base\_chance`, `comeback\_base\_chance`)

\* [ ] \*\*Lethal Modifiers\*\* (`cap\_8\_plus`, `cap\_12\_plus`)

\* [ ] \*\*Correct Path\*\*: `config.yaml` at project root and loader pointing at it (or overridden)

With this guide, a new developer can immediately locate and adjust any high-level simulation setting without diving into core modules or altering engine logic—ensuring safe, structure-preserving tweaks for new objectives.