# 🧾 Teacher Reference: Controls, Settings & Scoring

## BotBattles Quick Lookup Guide

This document is a **reference**, not a tutorial.

Use it to:

- Confirm what a control does
- Adjust match settings
- Interpret results
- Answer “is this expected?” questions quickly

It assumes you are already familiar with the basic BotBattles workflow.

## Interface Overview

The BotBattles interface is divided into four functional areas:

1. **Bot Loading & Management**
2. **Match Controls**
3. **Arena View**
4. **Results & Logs**

## Bot Loading & Management

### Load Bots / Select Bot Folders

- Opens a folder picker
- You can load bots:
    - either separately one at a time OR…
    - all at once, if they are in a parent folder like:

```
MatchBots/
    ├─ UberBot/
    └─ LazyBot/
    └─ SniperBot/
```

- Each separate bot folder must contain:
    - `bot.json` (must be a .json, name can be different otherwise)
    - `behavior.js` (MUST be called behavior.js)

### Bot Load Report

Displays:

- Successfully loaded bots
- Bots with pattern errors
- Bots with runtime or syntax errors
- Whether a previous valid version was used

**Notes**

- Bots load independently
- A single failure does not block others
- Reloading bots replaces existing ones

# Match Controls

## Start Battle

- Begins the simulation
- Locks bot state for the duration of the match
- Bots run autonomously until completion

## Pause/Resume

- Pauses sim execution
- Allows for tick-by-tick stepping

## Step

- Runs one tick of execution only

## Reset Match

- Re-randomizes starting positions
- Resets health, scores, timer, etc.

## Sim Speed

- Drag to speed up or slow down match execution

## Normalize Scores

- Default is off
- “Strict” setting will divide scores by bot config points used
    - Good for “efficiency” considerations
- “Gentle” will enforce only minor penalty on higher build budgets - less than strict

## Build Budget Selector

- Change to allow higher budgets if desired

## Match End On…

- “Timer” will use 120s and highest scores will win
- “Last bot standing” uses no timer and stops when only one bot remains

## Arena Behavior (Expected)

During a match, it is normal to observe:

- Bots colliding with walls
- Bots missing shots frequently
- Bots idling briefly between actions
- Bots failing or crashing individually

## Wall Behavior (Build Setting)

Bots define wall behavior in `bot.json`.

### Stop (0 pts)

- Bot halts briefly on collision
- Predictable, stable
- May appear as “freezing” to students

### Bounce (10 pts)

- Bot reflects off walls
- High momentum
- Less predictable trajectories

### Slide (20 pts)

- Bot moves along walls
- Maintains forward motion
- Requires spatial awareness

Wall behavior has a noticeable impact on survivability and control.

## Scoring: Conceptual Overview

You do **not** need to explain the full scoring formula.

Bots are generally rewarded for:

- Remaining active
- Engaging opponents
- Successfully landing hits
- Surviving longer than opponents

Bots are generally penalized (indirectly) for:

- Inactivity
- Random firing
- Early elimination
- Failing to engage

Scoring favors **consistent, purposeful behavior** over single lucky events.

Encourage interpretation across **multiple runs**, not single outcomes.

## Logs & Error Messages

### Load-time errors

- Related to `bot.json` schema or syntax
- Prevent the bot from entering the match

### Runtime errors

- Occur during `tick(api)` execution
- Affect only the bot that caused them
- Do not halt the simulation

Runtime errors are part of normal iteration and should be a sign to follow up with bot file review/corrections.

## Recommended Defaults (When Unsure)

- Default build budget (100)
- Multiple match runs (3–5)
- No mid-match interruptions
- Design changes happen between rounds, not between individual matches

These defaults maximize clarity and minimize classroom friction.

Have fun :)