🧠 Student Guide: Building & Tuning Your Bot

From “It Does Something” to “It’s Doing What I Wanted”

If your bot loads, moves, and reacts, congratulations.
You’re past the hardest part.

This guide is about **making your bot fight/play better**.

You’ll learn how to:

- Tune your bot’s build
- Understand tradeoffs
- Predict how changes affect behavior
- Design *styles* of bots — not just random code

There is no “best build.”
There are only **better decisions for your idea**.

🏗️ Two Parts of Every Bot (Quick Reminder)

Every bot has:

1. **A build** → what it *can* do
2. **Behavior code** → how it *chooses* to act

This guide focuses mostly on the **build**, and how it shapes what your code can accomplish.

⚖️ Build Tiers: What They Really Mean

Your build uses **tiers**, not raw numbers.

Each tier:

- Improves one ability
- Costs build points
- Limits what other abilities you can afford

Think of build points like a budget. If you spend more in one place, you must spend less somewhere else.

That’s not a limitation, it’s the challenge of the game.

The “Worst” Build

```
{
  "name": "WorstBot",
  "color": [255, 255, 255],
  "build": {
    "maxSpeedTier": 1,
    "turnRateTier": 1,
    "sightRangeTier": 1,
    "sightFovTier": 1,
    "shotPowerTier": 1,
    "shotRangeTier": 1,
    "maxHealthTier": 1,
    "memoryTier": 0,
    "wallBehavior": "stop"
  }
}
```

No points were spent upgrading this bot. It will be a challenge to get the bot to perform well… but not impossible. Good behaviour can make a big difference.

The “Best” Build

```
{
  "name": "BestBot",
  "color": [100,100,100],
  "build": {
    "maxSpeedTier": 5,
    "turnRateTier": 5,
    "sightRangeTier": 4,
    "sightFovTier": 4,
    "shotPowerTier": 5,
    "shotRangeTier": 5,
    "maxHealthTier": 5,
    "memoryTier": 4,
    "wallBehavior": "slide"
  }
}
```

All tiers and options are MAXED out in this build. It would take 253 points to create this bot. Of course, you’d have a real advantage if you could make this, but you would still need to give it good decision-making power with the `behavior.js` file…

Build Configuration Costs Overview

Movement Stats	Tier values	Tier costs
Max speed (pix/tick)	12, 24, 36, 48, 60	0, 5, 12, 20, 30
Turn rate (deg/tick)	4, 7, 10, 13, 16	0, 4, 10, 18, 28

Sensor Stats	Tier Stats	Tier costs
Sight range (px)	120, 200, 280, 360	0, 6, 14, 24
Sight FOV (deg)	30, 60, 120, 180	0, 6, 14, 24

Weapon Stats	Tier Stats	Tier costs
Shot power (damage)	2, 4, 6, 8, 10	0, 6, 14, 24, 36
Shot range (pixels)	200, 260, 320, 380, 440	0, 4, 10, 18, 28

Health

Max health	Point costs
100, 140, 180, 220, 260	0, 8, 18, 30, 45

Memory

Memory slots	Point costs
0, 1, 2, 3, 4	0, 3, 7, 12, 18

Wall behavior	Point Costs
Stop	0
Bounce	10
Slide	20

🚶 Speed, 🔄 Turning, and 👀 Vision

These three stats shape *how your bot experiences the arena*.

🚶 Speed (`maxSpeedTier`) - five tiers available

- Higher speed = faster movement
- Great for chasing or escaping
- Harder to aim accurately

High speed bots

- Cover lots of ground
- Can sometimes overshoot targets
- May need careful turning logic

🔄 Turning (`turnRateTier`) - five tiers available

- Controls how quickly your bot can rotate
- Crucial for aiming and wall recovery

Lower turn rate

- Makes chasing bots frustrating
- Slow to line up shots

Higher turn rate

- Feels responsive
- Easier to line up shots

👀 Vision (`sightRangeTier`, `sightFovTier`) - four ties available (each)

- Determines *how early* your bot reacts
- Better scan()s allow quicker reaction, but cost more build points
    - Wide vision sees more of the arena
    - Far sight range can detect bots early

🔫 Firepower and ❤️ Health

These stats decide how long your bot survives — and how dangerous it is.

🔫 Shot Power (`shotPowerTier`) & Range (`shotRangeTier`)

- Power controls damage when a laser shot lands
- Range allows your laser to shoot farther
    - But again, both costs more build points

❤️ Health (`maxHealthTier`)

- Determines how much damage your bot can take

High health bots:

- Survive longer
- Forgive mistakes
- Often end up slower or less perceptive

Low health bots:

- Must avoid damage
- Require careful positioning

🧠 Memory: When (and When Not) to Use It

Memory lets your bot remember numbers between ticks.

This is powerful — and dangerous.

Good uses

- Tracking last seen enemy
- Counting turns
- Simple state (“searching” vs “attacking”)

Bad uses

- Overcomplicated plans
- Logic you can’t explain

Rule of thumb:
If you can’t describe your memory logic in one sentence, it’s probably too much.

🧱 Wall Behavior: Small Choice, Big Impact

Your wall behavior changes how your bot recovers from a wall impact.

`"stop"`

- Bot pauses briefly at walls
- Safe, predictable
- Can look like freezing

Best for:

- Beginners
- Saving point budget for investments elsewhere

`"bounce"`

- Bot reflects off walls
- Fast, chaotic
- Harder to predict

Best for:

- Aggressive bots
- High-speed builds
- Experimental strategies

`"slide"`

- Bot glides along walls
- Maintains momentum

Best for:

- Arena control
- Edge-following strategies

🎯 Build Budgets Can Change

Your teacher may change the total build budget.

This means:

- You might afford stronger builds
- Or be forced into harder tradeoffs

Important:
A higher budget does *not* guarantee better performance.

It just allows:

- More extreme designs
- More variety
- More interesting failures

📊 How Bots Score (Hints, Not Math)

You don’t need the exact formula, but you *should* know what matters.

Bots are rewarded for:

- Staying alive
- Hitting enemies
- Engaging in the fight
- Using time effectively

Bots are *not* rewarded for:

- Standing still forever
- Random firing
- Pure luck

A bot that **acts with purpose** usually scores better
than a bot that just survives.

🧪 How to Tune Like a Pro

1. Change **one build value**
2. Run several matches
3. Watch what changes
4. Ask *why*
5. Keep or undo the change

If you can explain:

“I changed X because I wanted Y”
…you’re tuning correctly.

🧠 Final Advice

A good bot is not:

- The fastest
- The strongest
- The most complicated

A good bot is:

- Understandable
- Intentional
- Consistent