Warhammer 40'000

In the grim darkness of the far future
There is only WAR

Why Machine Learning ?

A logical journey

What is Warhammer 40'000 ?

1 – Turn-base Tabletop Game

- Played around a table
- Each player play alternatively

2 - Dark science-fantasy

- Set in the 41th millenium
- Every faction is evil…
- Or too stupid for that

3 – Next MCU

- The game / universe is more and more popular

How is it played ?

1 – Army preparation

- *Choose one of the 20+ factions*
- *Write down an army list within the agreed limit*
- *Collect and paint your models*

2 – Meet your friend

- *Reunite around a table*
- *Prepare the table setup*
- *Choose deployment and mission*

3 – Play !

- *Destroy your friend's army is not enough* (but still satisfying)
- *You must score more tactical objective points*

Game's Structure

Turn 1

=== Player 1 ===

--- Movement Phase---
Unit 1 moves
Unit 2 moves
↓
--- Shooting phase ---
Unit 1 shoots
Unit 2 shoots

=== Player 2 ===

--- Movement Phase---
Unit 3 moves
Unit 4 moves
↓
--- Shooting phase ---
Unit 3 shoots
Unit 4 shoots

Turn 2

=== Player 1 ===

--- Movement Phase---
Unit 2 moves
Unit 1 moves
↓
--- Shooting phase ---
Unit 2 shoots
Unit 1 shoots

=== Player 2 ===

--- Movement Phase---
Unit 4 moves
Unit 3 moves
↓
--- Shooting phase ---
Unit 4 shoots
Unit 3 shoots

...

Movement Phase

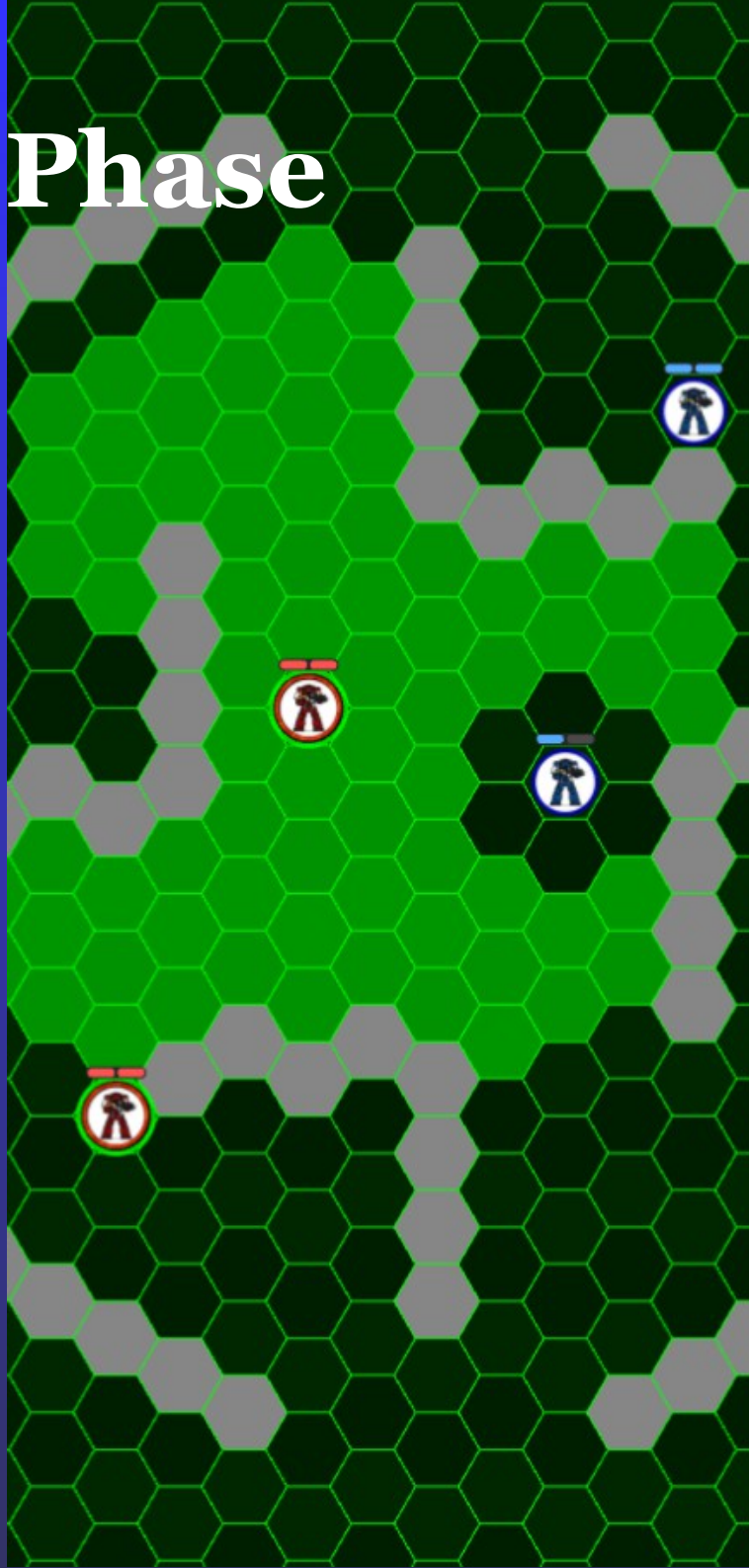
			RANGE WEAPON					
M	T	SV	RNG	A	BS	S	AP	D
6	4	3+	24"	2	3+	4	-1	1

Units can :

- Move up to **M** hexes

Units cannot :

- Move to or through a wall
- Move to another unit's hex
- Move to an hex adjacent to an enemy unit



Shooting Phase

			RANGE WEAPON					
M	T	SV	RNG	A	BS	S	AP	D
6	4	3+	24"	2	3+	4	−1	1

Units can shoot :

- Up to their **A** times
- At units in Line of Sight
- At units within their **RNG**

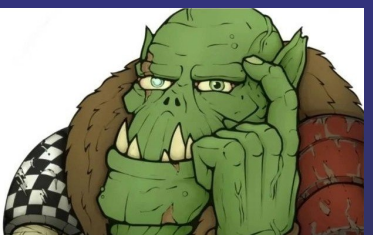
Shooting Sequence :

➡ Hit Roll : must roll >= **BS**
↳ Wound Roll : Weapon's **S** vs Target's **T**
↳ Armor Save (**SV**) – **AP**
↳ Inflict **D** damages

The wargamer's doom

To play a 2 players game ...

You need to be TWO !



You have a new friend !

No more need of an unfaithful human to play !

Welcome the Reinforcement Learning AI !!

A friend for life

Technical solution

Make it real

Frontend : React

Game Requierments :
- Board Management:
- Real-time UI Updates without full re-renders
- Event Handling: Click events

Performances :
- PIXI.js Integration:
    - React manages PIXI.js canvas
    - TypeScript Compatibility

Technical solution

Make it real

Backend : Python

Obvious Choice :
- Only viable option for serious deep reinforcement learning with DQN

Familiarity :
- I have made almost all the ML training on Python

Technical solution

The drama

Architecture conflict !

After 3 months of work, I had achieved :

- Complete PvP mode
- Fully functional agent training

Time to play versus the AI !?

I quickly realized a major flaw in my architecture :

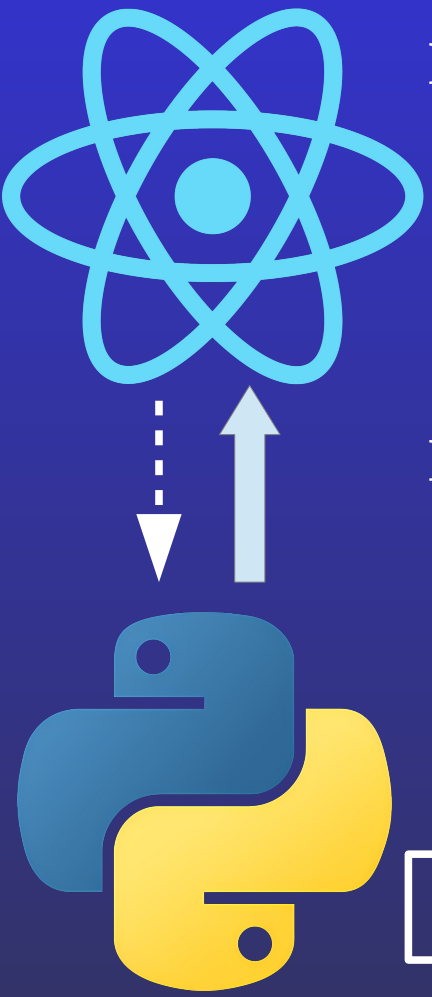
- The PvP was led by React (click reactive)
- The AI was led by Python, obviously

They could NOT work together !!!

Technical solution

Mind your own Business

Architecture Alignment



Backend drives :

- Sequential activation
- Phase transitions
- Game state authority
- AI decision making
- Action validation

Frontend displays :

- Visual representation
- User input
- Animation/UX

Clear Boundary :

Backend drives ALL game logic → Frontend focuses on the display

Preparation Work

Make the game follow ONE ruler

```
For each PLAYER unit
├── ELIGIBILITY CHECK (Pool Building Phase)
│   ├── unit.HP_CUR > 0?
│   │   └── NO → ❌ Dead unit (Skip, no log)
│   ├── unit.player === current_player?
│   │   └── NO → ❌ Wrong player (Skip, no log)
│   ├── units_fled.includes(unit.id)?
│   │   └── YES → ❌ Fled unit (Skip, no log)
│   ├── Adjacent to enemy unit within CC_RNG?
│   │   └── YES → ❌ In fight (Skip, no log)
│   ├── unit.RNG_NB > 0?
│   │   └── NO → ❌ No ranged weapon (Skip, no log)
│   ├── Has LOS to enemies within RNG_RNG?
│   │   └── NO → ❌ No valid targets (Skip, no log)
│   └── ALL conditions met → ✅ Add to shoot_activation_pool → Highlight the unit with a green circle around its icon
├── STEP : UNIT_ACTIVABLE_CHECK → is shoot_activation_pool NOT empty ?
│   ├── YES → Current player is an AI player ?
│   │   ├── YES → pick one unit in shoot_activation_pool
│   │   │   ├── Clear any unit remaining in valid_target_pool
│   │   │   ├── Clear TOTAL_ATTACK log
│   │   │   ├── SHOOT_LEFT = RNG_NB
│   │   │   └── While SHOOT_LEFT > 0
│   │   │       ├── Build valid_target_pool : All enemies within range AND in Line of Sight AND having HP_CUR > 0 → added to valid_target_pool
│   │   │       └── valid_target_pool NOT empty ?
│   │   │           ├── YES → SHOOTING PHASE ACTIONS AVAILABLE
│   │   │           │   ├── Display the shooting preview (all the hexes with LoS and RNG_RNG are red)
│   │   │           │   └── Display the HP bar blinking animation for every unit in valid_target_pool
│   │   │           │   ├── 🎯 VALID ACTIONS: [shoot, wait]
│   │   │           │   ├── ❌ INVALID ACTIONS: [move, charge, attack] → end_activation (ERROR, 0, PASS, SHOOTING)
│   │   │           │   └── AGENT ACTION SELECTION → Choose shoot?
│   │   │           │       ├── YES → ✅ VALID → Execute shoot
│   │   │           │       ├── Agent choose a target in valid_target_pool
│   │   │           │       │   ├── Execute attack_sequence(RNG)
│   │   │           │       │   ├── SHOOT_LEFT -= 1
│   │   │           │       │   ├── Concatenate Return to TOTAL_ACTION log
│   │   │           │       │   ├── selected_target dies → Remove from valid_target_pool, continue
│   │   │           │       │   ├── selected_target survives → Continue
│   │   │           │       │   └── GO TO STEP : PLAYER_ACTION_SELECTION
│   │   │           │       │   └── end_activation (ACTION, 1, SHOOTING, SHOOTING)
│   │   │           │       └── NO → Agent chooses: wait?
│   │   │           │           ├── YES → ✅ VALID → Execute wait action
│   │   │           │           │   └── end_activation (WAIT, 1, PASS, SHOOTING)
│   │   │           │           └── NO → Agent chooses invalid action (move/shoot/attack)?
│   │   │           │               └── ❌ INVALID ACTION ERROR → end_activation (ERROR, 0, PASS, SHOOTING)
│   │   │           └── NO → end_activation (PASS, 0, PASS, SHOOTING)
│   │   └── NO → Human player → STEP : UNIT_ACTIVATION → player activate one unit from shoot_activation_pool by left clicking on it
│   │       ├── Clear any unit remaining in valid_target_pool
│   │       ├── Clear TOTAL_ATTACK log
│   │       ├── SHOOT_LEFT = RNG_NB
│   │       ├── While SHOOT_LEFT > 0
│   │       │   ├── Build valid_target_pool : All enemies within range AND in Line of Sight AND having HP_CUR > 0 → added to valid_target_pool
│   │       │   └── valid_target_pool NOT empty ?
│   │       │       ├── YES → SHOOTING PHASE ACTIONS AVAILABLE
│   │       │       │   ├── STEP : PLAYER_ACTION_SELECTION
│   │       │       │   ├── Display the shooting preview (all the hexes with LoS and RNG_RNG are red)
│   │       │       │   └── Display the HP bar blinking animation for every unit in valid_target_pool
│   │       │       │       ├── Left click on a target in valid_target_pool
│   │       │       │       │   ├── Execute attack_sequence(RNG)
│   │       │       │       │   ├── SHOOT_LEFT -= 1
│   │       │       │       │   ├── Concatenate Return to TOTAL_ACTION log
│   │       │       │       │   ├── selected_target dies → Remove from valid_target_pool, continue
│   │       │       │       │   ├── selected_target survives → Continue
│   │       │       │       │   └── GO TO STEP : PLAYER_ACTION_SELECTION
│   │       │       │       ├── Left click on another unit in shoot_activation_pool ?
│   │       │       │       │   └── SHOOT_LEFT = RNG_NB ?
│   │       │       │       │       ├── YES → Postpone the shooting phase for this unit
│   │       │       │       │       │   └── GO TO STEP : UNIT_ACTIVABLE_CHECK
│   │       │       │       │       └── NO → The unit must end its activation when started
│   │       │       │       │           └── GO TO STEP : PLAYER_ACTION_SELECTION
│   │       │       │       ├── Left click on the active_unit → No effect
│   │       │       │       ├── Right click on the active_unit
│   │       │       │       │   └── SHOOT_LEFT = RNG_NB ?
│   │       │       │       │       ├── NO → end_activation (ACTION, 1, SHOOTING, SHOOTING)
│   │       │       │       │       │   └── GO TO STEP : UNIT_ACTIVABLE_CHECK
│   │       │       │       │       └── YES → end_activation (WAIT, 1, PASS, SHOOTING)
│   │       │       │       │           └── GO TO STEP : UNIT_ACTIVABLE_CHECK
│   │       │       │       └── Left OR Right click anywhere else on the board
│   │       │       │           └── GO TO STEP : PLAYER_ACTION_SELECTION
│   │       │       └── NO → SHOOT_LEFT = RNG_NB ?
│   │       │           ├── NO → shot the last target available in valid_target_pool → end_activation (ACTION, 1, SHOOTING, SHOOTING)
│   │       │           └── YES → no target available in valid_target_pool at activation → no shoot → end_activation (PASS, 1, PASS, SHOOTING)
│   │       └── End of shooting → end_activation (ACTION, 1, SHOOTING, SHOOTING)
│   └── No more activable units → pass
└── End of shooting phase → Advance to charge phase
```

This « algorythm » describes the way the shooting phase should be played

Back to the AI

What is Q learning ?

Overview

You are playing a new video game… like a kid!

⬇

Over time, you learn what works and what doesn't

This is the way Q-Learning works

How to help Bob to survive ?

Q learning to the rescue !

Bob's plan to survive :

1 - Bob explores his environment

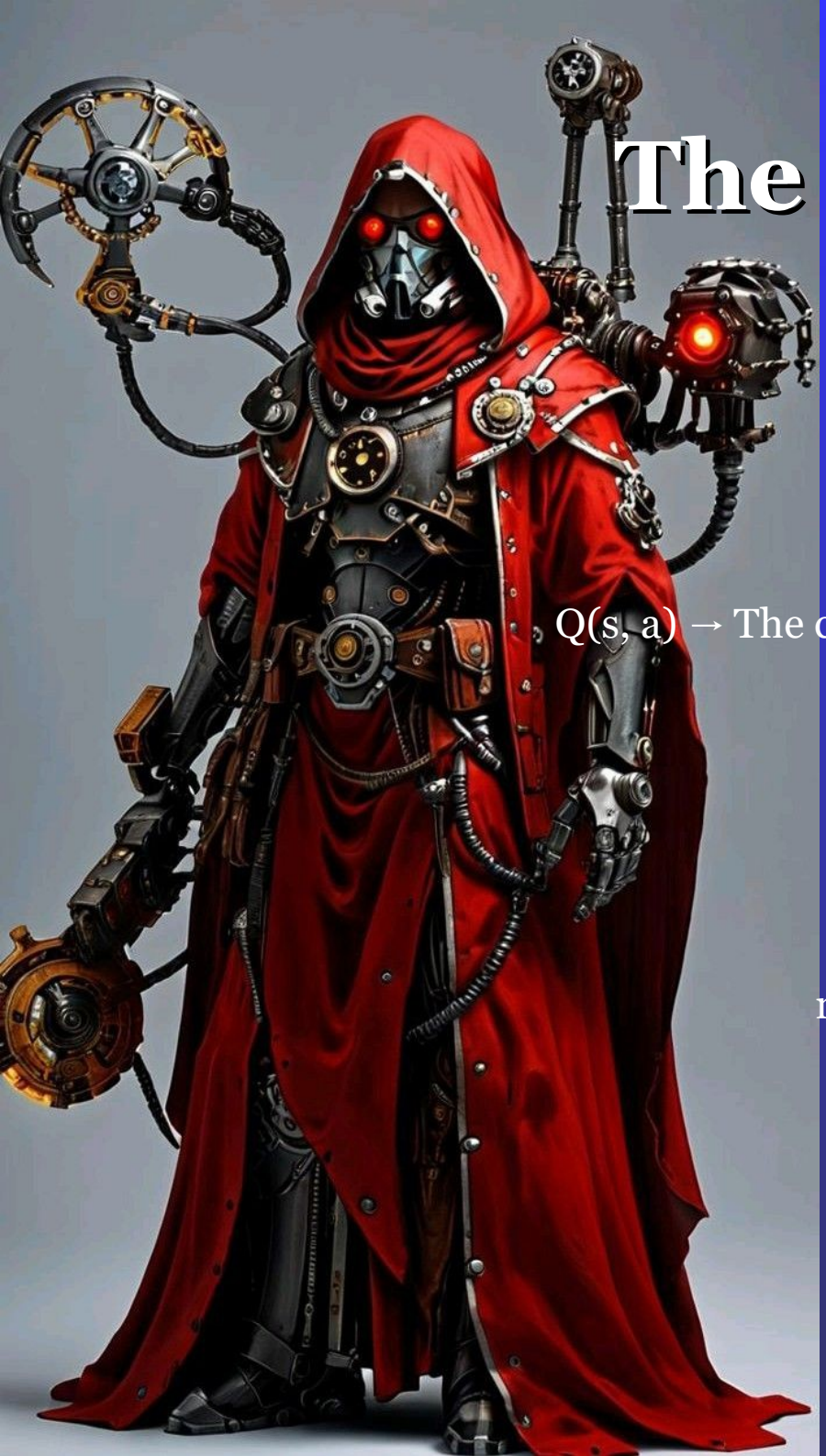
→ "What if I charge a Bloodthirster or rush an objective ?"

2 – Bob builds a strategy

→ "A Bloodthirster is too dangerous, go for the objective"

Over time, Bob remembers what works best and makes better choices !

(think Edge of tomorrow)



The Formula

$$Q(s,a)=Q(s,a)+\alpha[r+\gamma \max Q(s',a')-Q(s,a)]$$

Q(s, a) → The current guess for how good the action (a) in this state (s) is

α → How fast he updates what he knows

r → Reward

γ → How much we value current over future rewards

max Q(s', a') → The best future reward he can expect

This function will return how good is it to be in the state **s** and take an action **a** in this state

Learning parameters

Learning Rate (α) : How fast Bob adapts?

- Learns too fast → Forms an opinion too quickly

- Learns too slow → Keeps repeating the same mistakes

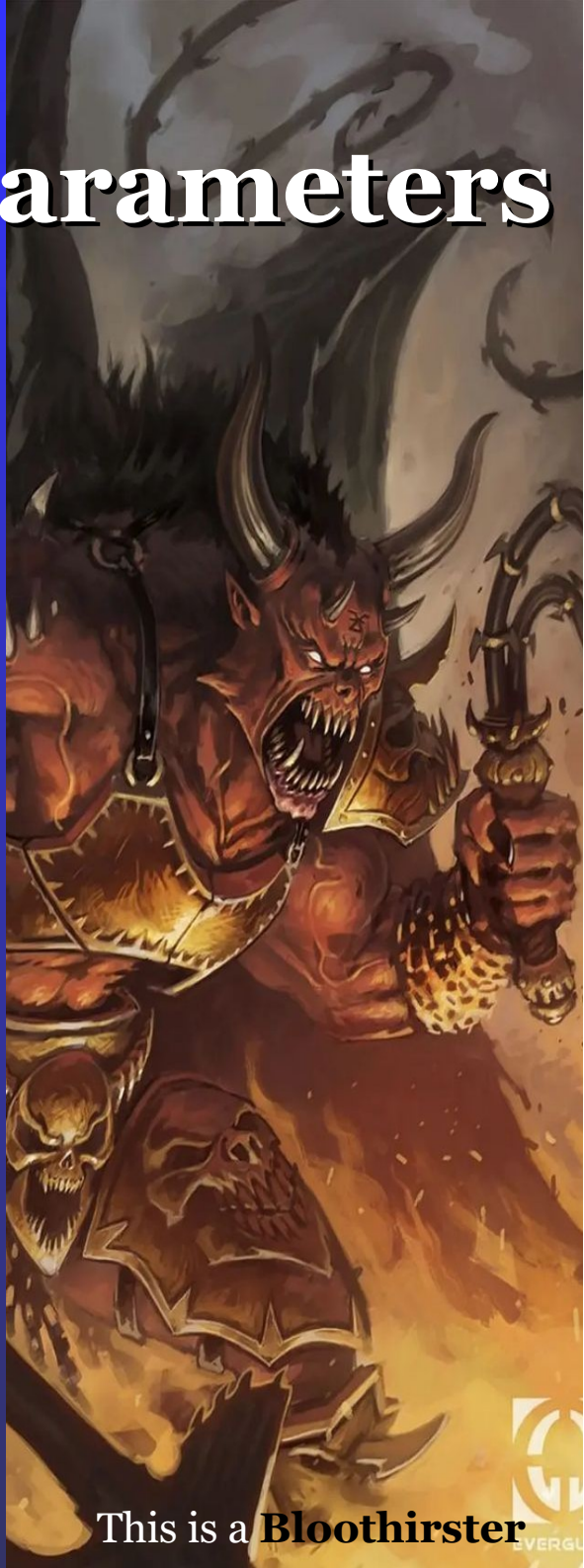
✓Smart Bob :
Remembers key lessons but still adapts

Future Discount (γ) : Thinking Long or Short-Term?

- Short-term → Misses big rewards

- Long-term → Waits too long & loses

✓Smart Bob :
Balances small wins & big rewards



This is a **Bloothirster**

The Q table

Think of the Q-table as Bob's personal cheat sheet

-1	S2 	-1
-1	-1	-1
-1	-10	+10

Q	a1	a2	a3	a4
S1	Q(S1,a1)	Q(S1, a2)	Q(S1, a3)	Q(S1, a4)
S2	Q(S2,a1)	Q(S?,a2)	Q(S2, a3)	Q(S2, a4)
S3	Q(S3,a1)	Q(S3,a2)	Q(S3, a3)	Q(S3, a4)
.	.	.	.	.

Goal :
Update and learn these Q-values such as the total reward is maximized

Apply it to Warhammer 40k

Observation

How Bob knows his environment ?

DQN Input

142 positions array :

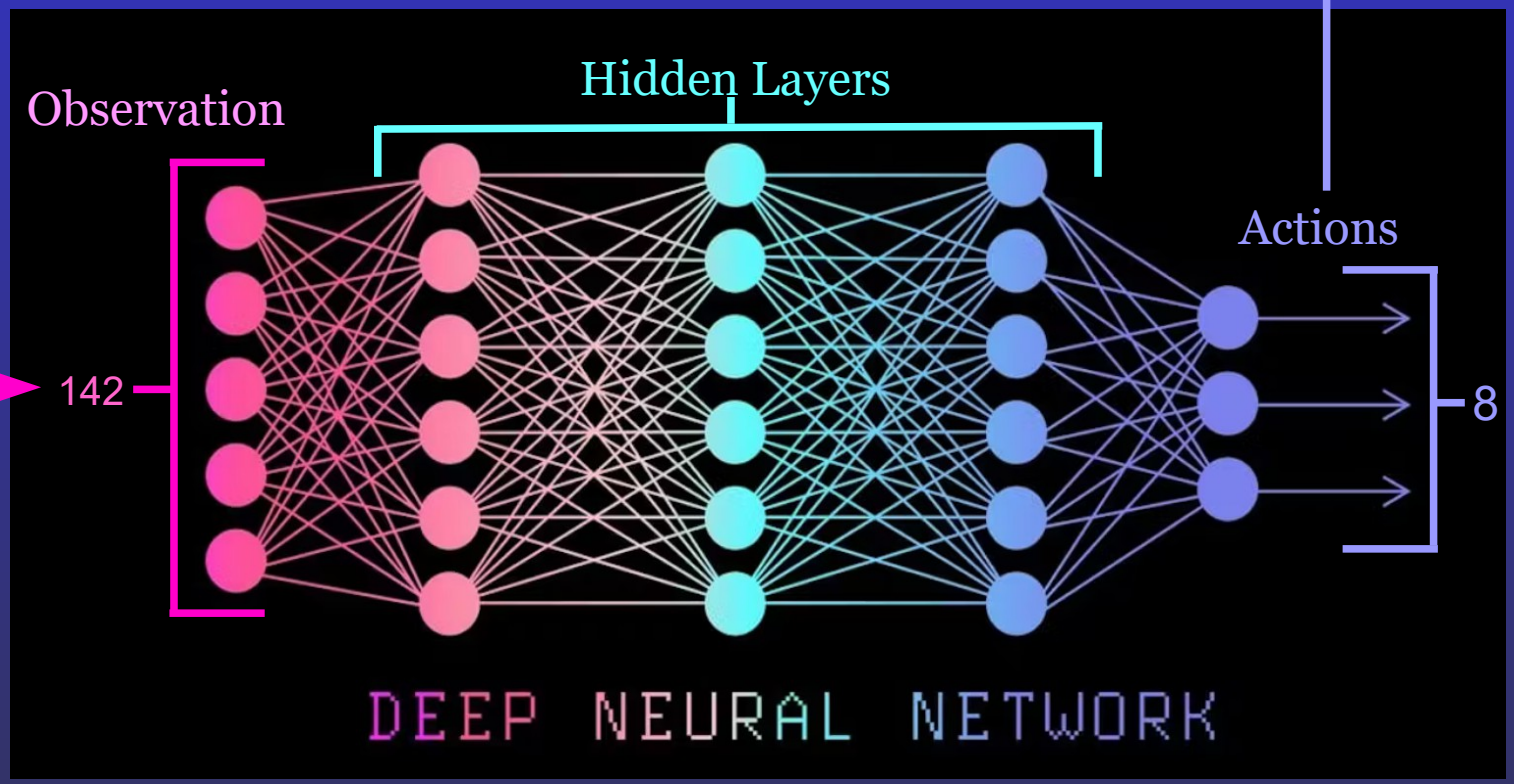
- General game status (Active player, turn, etc...)
- Informations about all the units on the board
- Situation on the board
- Board structure

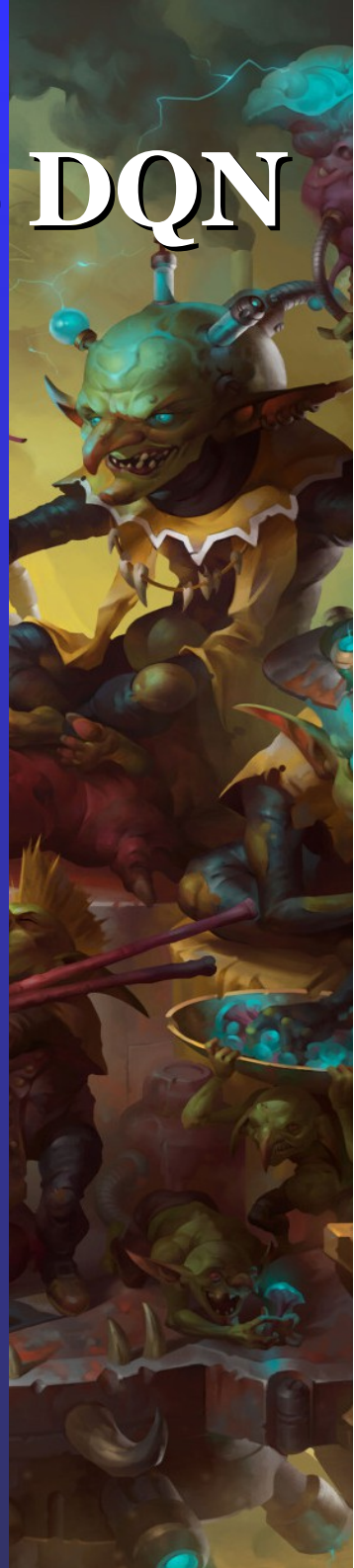
From Q-table to DQN

(Enter Bob's brain)

How does Bob handles his environment ?

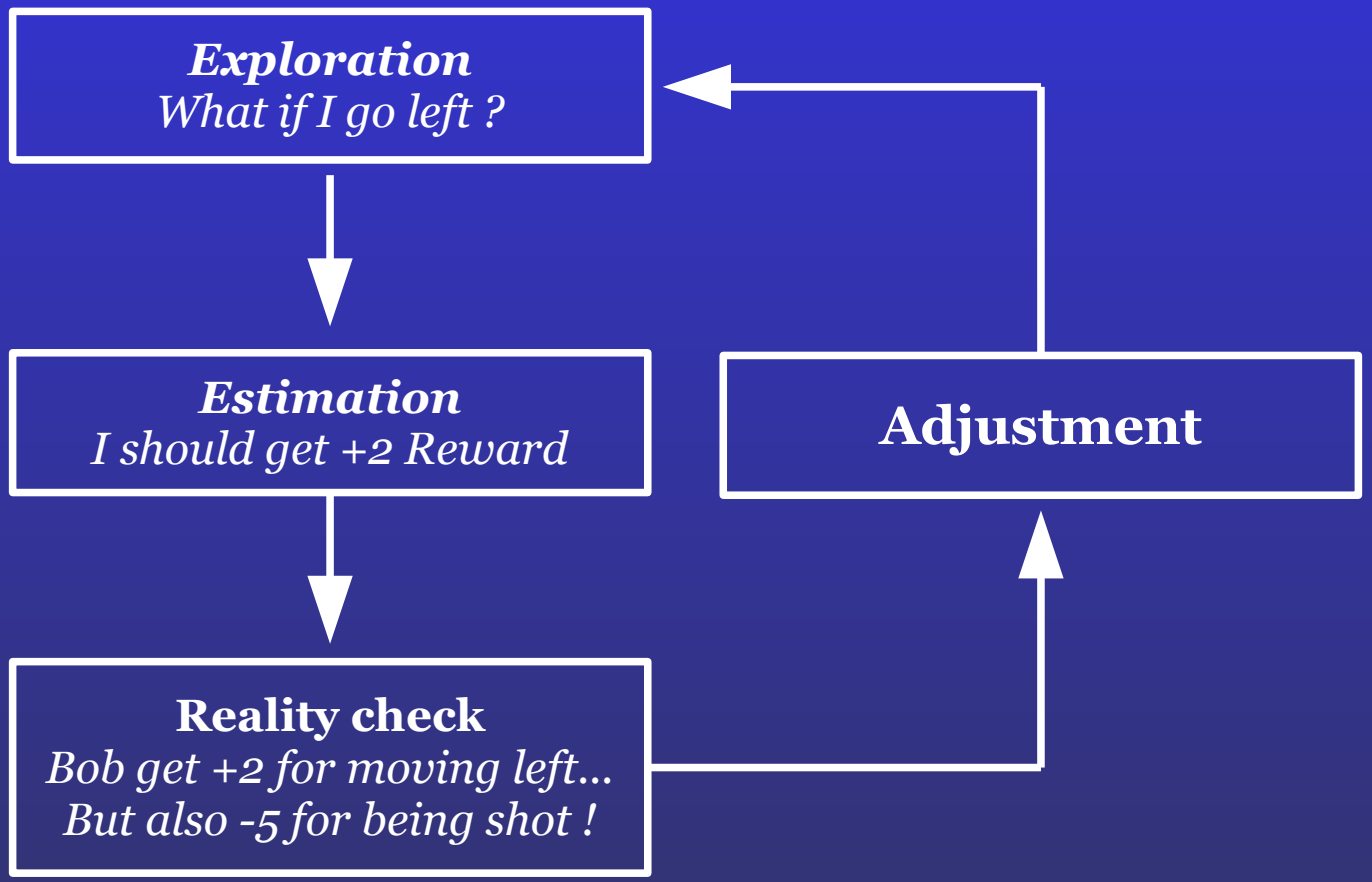
Q	a1	a2	a3	a4
S1	Q(S1,a1)	Q(S1, a2)	Q(S1, a3)	Q(S1, a4)
S2	Q(S2,a1)	Q(S2,a2)	Q(S2, a3)	Q(S2, a4)
S3	Q(S3,a1)	Q(S3,a2)	Q(S3, a3)	Q(S3, a4)
.	.	.	.	.





Tailor Strategy

How Bob learns to play ?



Rewards

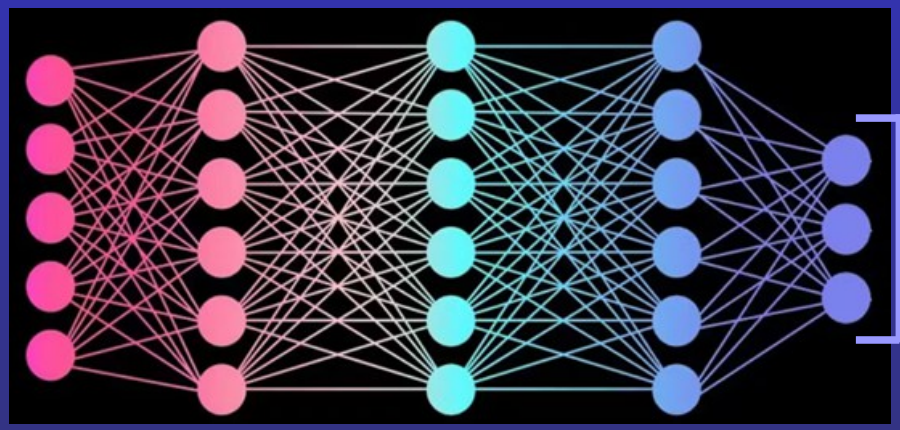
How Bob knows if he is doing good ?

```
"SpaceMarine_Infantry_Troop_RangedSwarm": {
  "base_actions": {
    "move_close": 0.4,
    "move_away": 0.5,
    "move_to_charge": 0.2,
    "move_to_los": 0.6,
    "ranged_attack": 0.5,
    "melee_attack": 0.1,
    "charge": 0.1,
    "charge_success": 0.1,
    "being_charged": -0.3,
    "wait": -0.5
  },
  "target_type_bonuses": {
    "vs_melee": 0.1,
    "vs_ranged": 0.0,
    "vs_swarm": 0.2,
    "vs_troop": 0.1,
    "vs_elite": 0.0,
    "vs_vehicle": -0.1,
    "vs_heavy": -0.2
  },
  "result_bonuses": {
    "kill_target": 0.2,
    "wound_target": 0.1,
    "no_overkill": 0.1,
    "target_lowest_hp": 0.1
```

Make a Choice

What will Bob decide to do ?

Action taken : Action with the highest Q-value

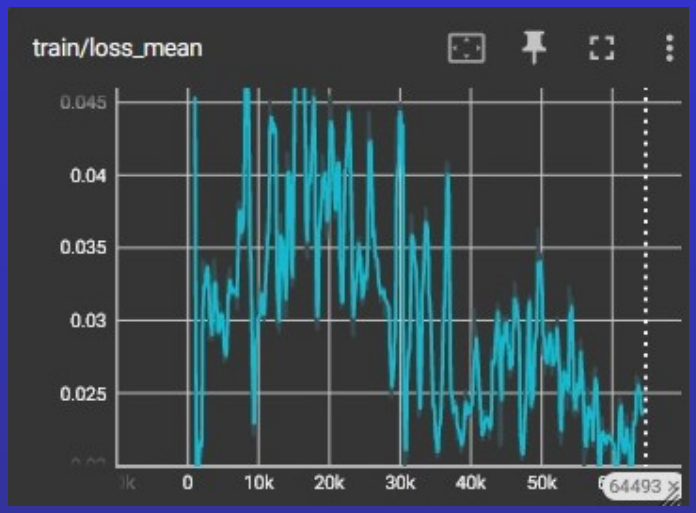


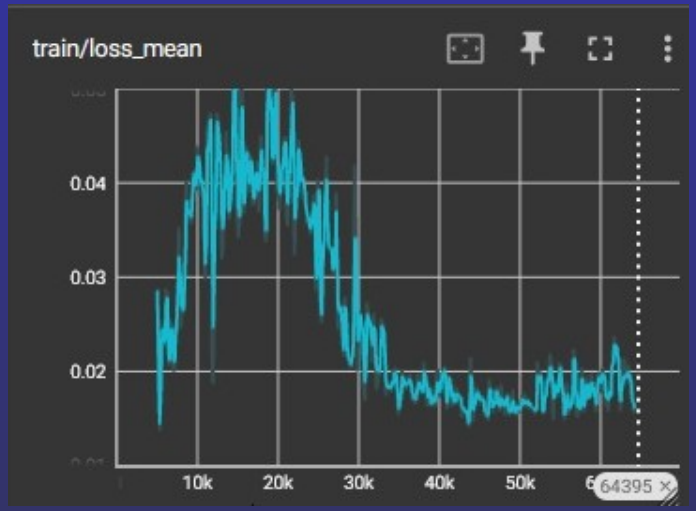
```
action_map = {
    0: "move_north",
    1: "move_south",
    2: "move_east",
    3: "move_west",
    4: "shoot",
    5: "charge",
    6: "attack",
    7: "wait"
}
```

Supervision

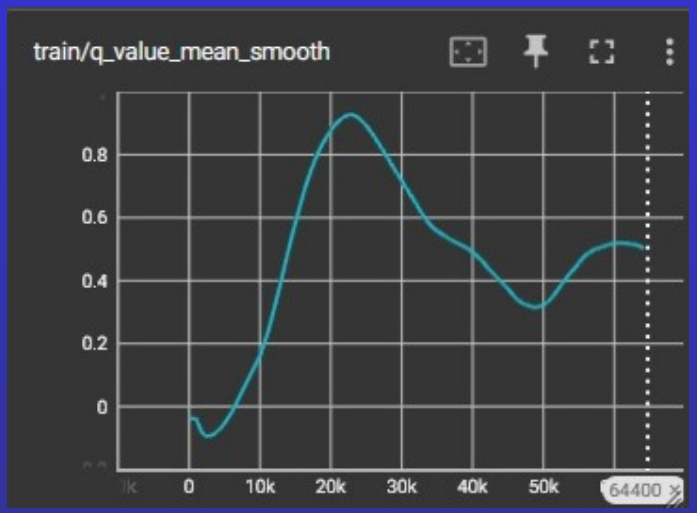
How to ensure Bob is well trained ?

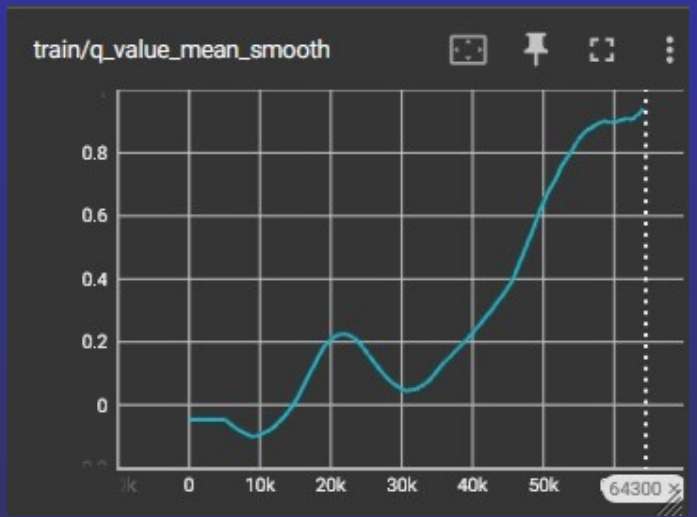
Loss





Q-Value






Let's Playyyy !!!

Future improvements

1 - Finish the game's core
- Add the charge and combat phases

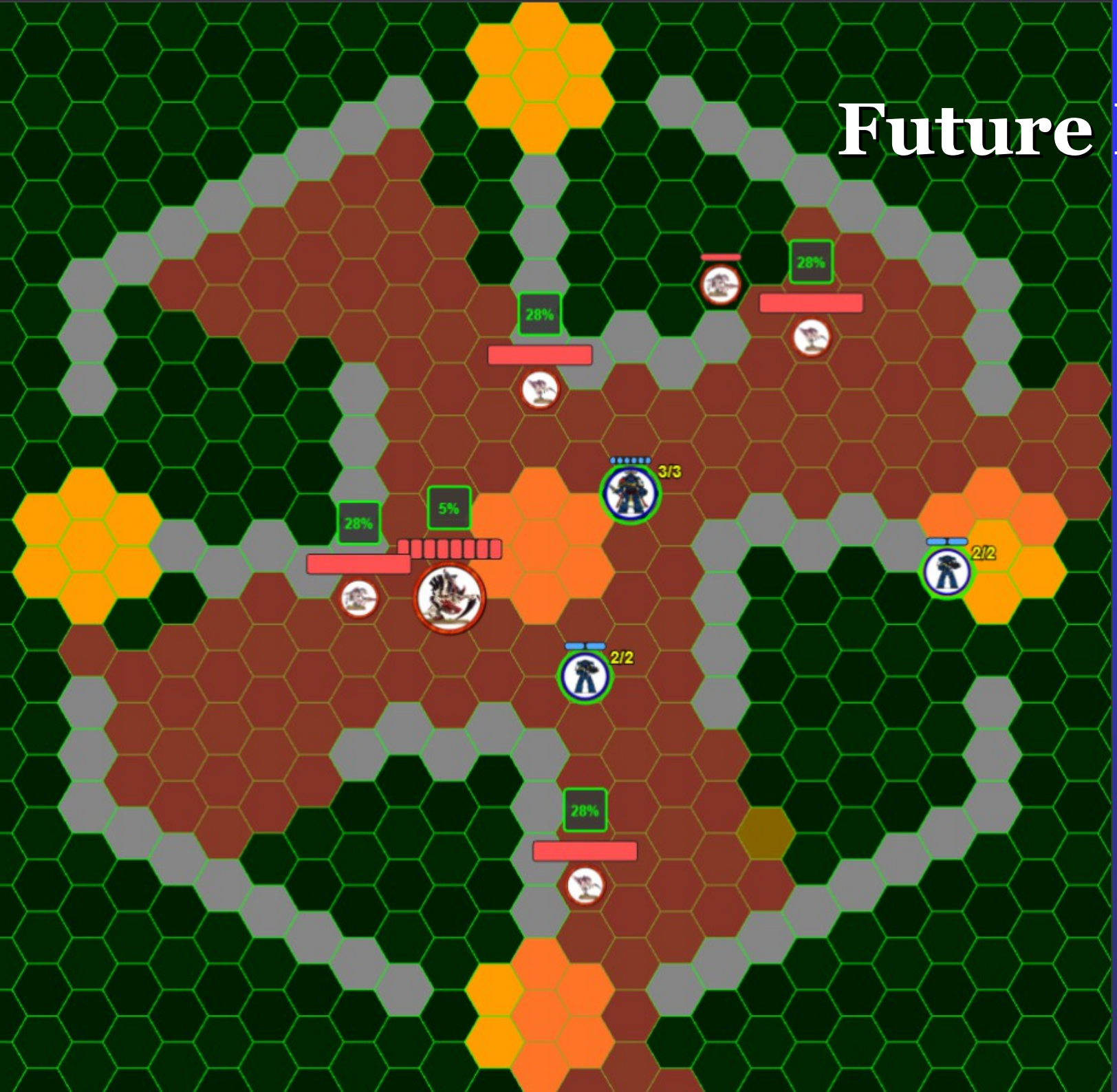
2 – Upgrade Observation
- Add tactical datas

3 – Optimize the agent training
- Make it worth playing against

4 - Add new units
- Assault Intercessors, Captain Gravis....

5 - Add new factions
- Tyranids, Orks, etc...

6 - Share the game !



Future Features

Complete the turn
- Add the last phases

Tactical Objectives

New Marins Unit
Captain Gravis

New Faction
Tyranids

Tyranids Unit
Carnifex
Hormagaunt
Termagant

My Future

Find a first job in AI

…

Anywhere in the galaxy !

(and keep learning)