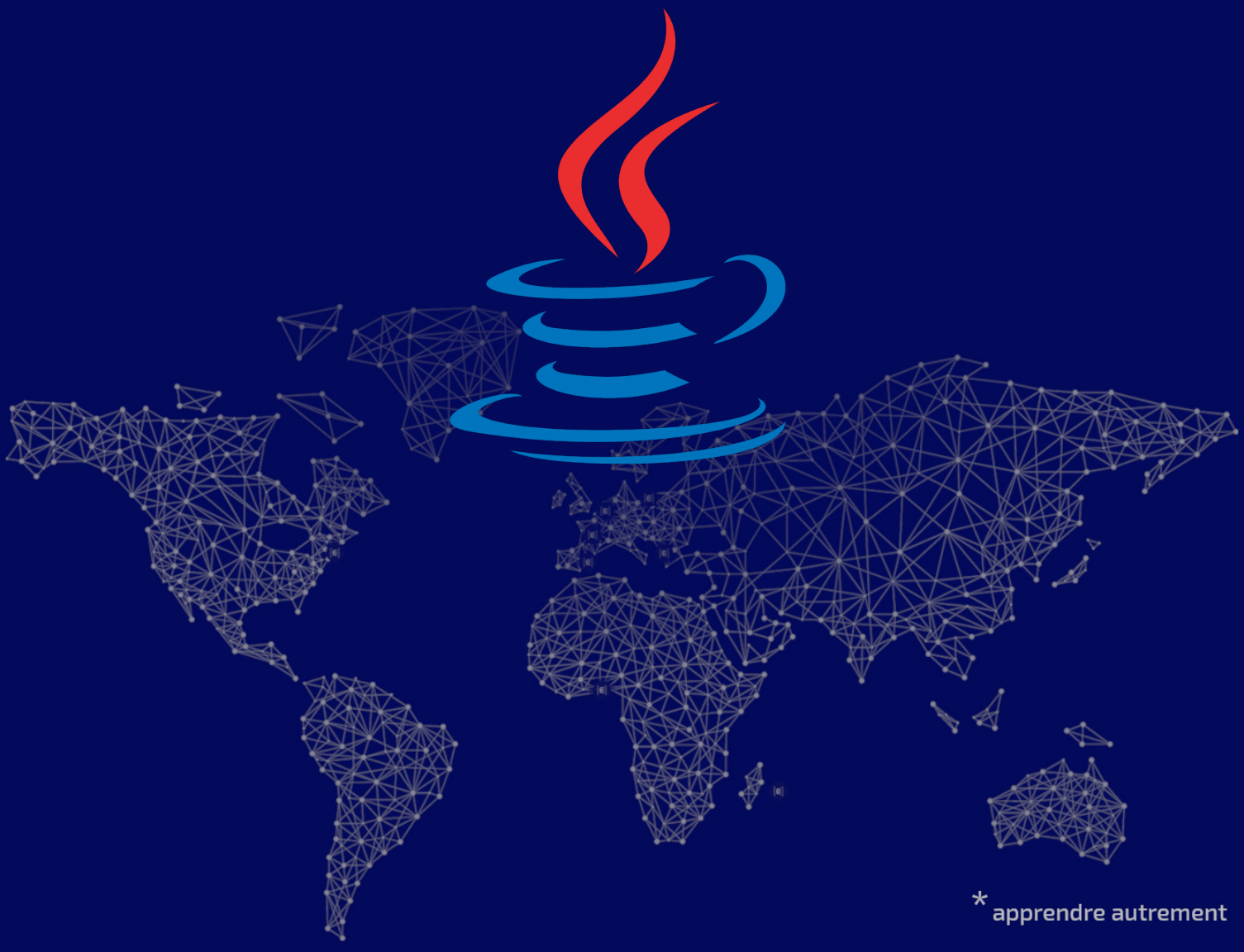
{EPITECH}
LEARN DIFFERENT*

2DGAME

LET'S MAKE A GAME IN JAVA



* apprendre autrement

2DGAME

Your goal is to **put OOP principles into practice** by making a game in Java. Starting from this bootstrap *and* OOP paradigm, move your game in (almost) any direction you want. **Let your imagination speak, be creative !**.

Here are a (non-exhaustive) couple of possibilities:

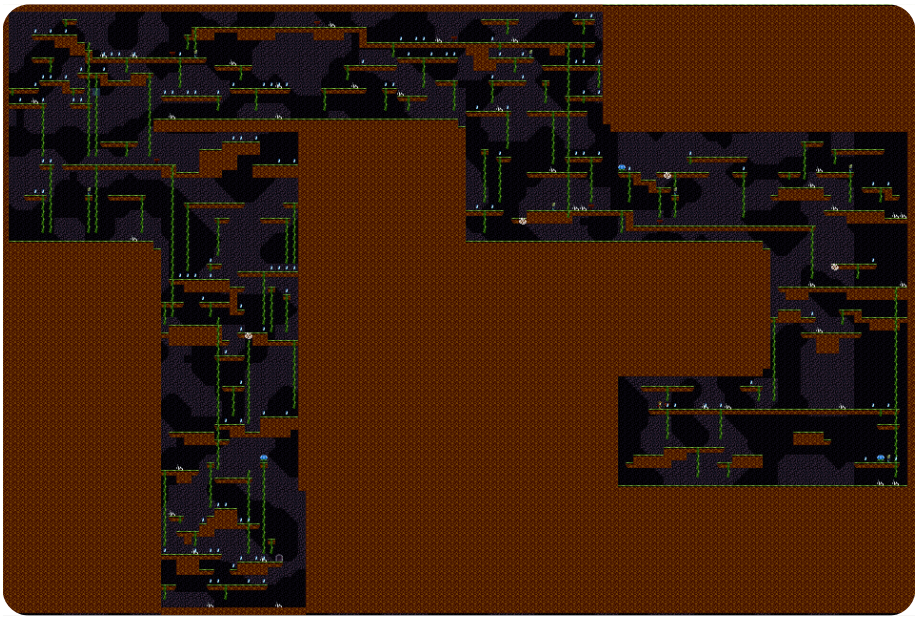
Beware! completing the bootstrap is mandatory.

Level Generation

The bootstrap took place into a fixed 2D-grid, the simplest vision of a two-dimensional world.

- ✓ Why not put obstacles on the grid (walls, doors, ladders…), make a maze, a platformer… ?
- ✓ Why not imagine more complex, generated levels ?

There are many ways to generate a world automatically.
Choose (and justify) the one that suits your needs.



Interaction with items and other characters

The bootstrap was limited to the simplest interactions : kill NPCs, consume or store items.

- ✓ Why fight and only fight ? Why not dialogue ? Or at the very least, why not avoid ?
- ✓ Why not equip the items, combine them, use them on the environment ?
- ✓ Why not use dialogue choices and/or items (or combination of items) to solve puzzles ?

Adventure games, point'n'clicks, require puzzle solving and dialogue choices.

Many role-playing games include quests, dialogues and complex inventory management.

Many infiltration games require avoiding opponents rather than killing them.

Character behavior and pathfinding

The bootstrap included static monsters, waiting to harm or be harmed.

- ✓ Why couldn't they move, towards the player or any specific direction, avoiding obstacles ?

There are a lot of pathfinding algorithms that could help.

Real (?) time

The bootstrap required a turn-by-turn gameplay, where time didn't count.

- ✓ Why not include time ? Even real-time ?

When turn lasts only a few milliseconds, it starts looking like real-time (hello Warcraft 2 !).

Input and character control

The bootstrap required to move the character with the keyboard.

✓ Why not pick the mouse to make the character orient itself or point the direction to go ?

Character control with a mouse is at the core of many strategy and adventure games.

Think about mouse and key events.

Party management

The bootstrap made the player move a single character. No more.

✓ Why just one ? Why not control several characters ?
✓ Why not choose the suitable character to recruit (or fire) ?

Many role-playing games provide a complex party management, including affinities between party members, hiring and firing, loyalty quests, etc…

The basis of strategy games and wargames is the management of an army, or a squad, or a small town where people of different kinds (peasant, infantry…) are recruited.

Passive management

Bis repetita: The bootstrap made the player move a single character. No less.

✓ Why control anybody anyway ? Why not train the character before or during the game?

Many sports management games do not allow direct control over the party. Instead, they require the player to train properly the character, or the team, before the game.

Training a computer-controlled character means artificial intelligence, machine learning, and a whole set of exciting, disciplines. Many of them taught at Epitech !

Combat system

The bootstrap required the simplest, kindergarten-like, form of fighting : hit or be hit.

✓ Why not a more complex combat system, taking into accounts stats, skills, spells, … ?

The very core of most role playing games is their combat system. The algorithm use to resolve a fight can also be used to evaluate the success of a dialogue choice.

Many role-playing games use the D20 system that includes characteristics (strength, dexterity…), skills, spells, and several kinds of dices (4-sides, 6-sides, 20-sides…).

Et cetera

To make it short, you must create a 2-dimension game in Java *while respecting OOP principles*. From here, you can do whatever you want. And the more you'll dedicate yourself to it, the better your project will be evaluated.

Think of your project as a POC, where you choose either to **focus entirely on a single aspect** or **pick two or more aspects and mix them** to validate the whole gameplay.

We just provided a couple of ideas. But you might have other ideas that you want to push further. Feel free to try… on condition that your pedago agrees.

But wait, what about the graphics ?

You'll need a GUI to display the game and character informations. The choice is yours.

JavaFX can help. Or at least AWT and Swing. Or a Javascript-based frontend language.

Always prioritize the gameplay over the graphics. A beautiful but empty shell is not a game.

Oh, wait further ! There's the review !

A proof or concept means you have something (a playable demo) to prove to somebody (us).

The final goal of Epitech projects is to have something to show to future recruiters.

v 4.0.1

* apprendre autrement