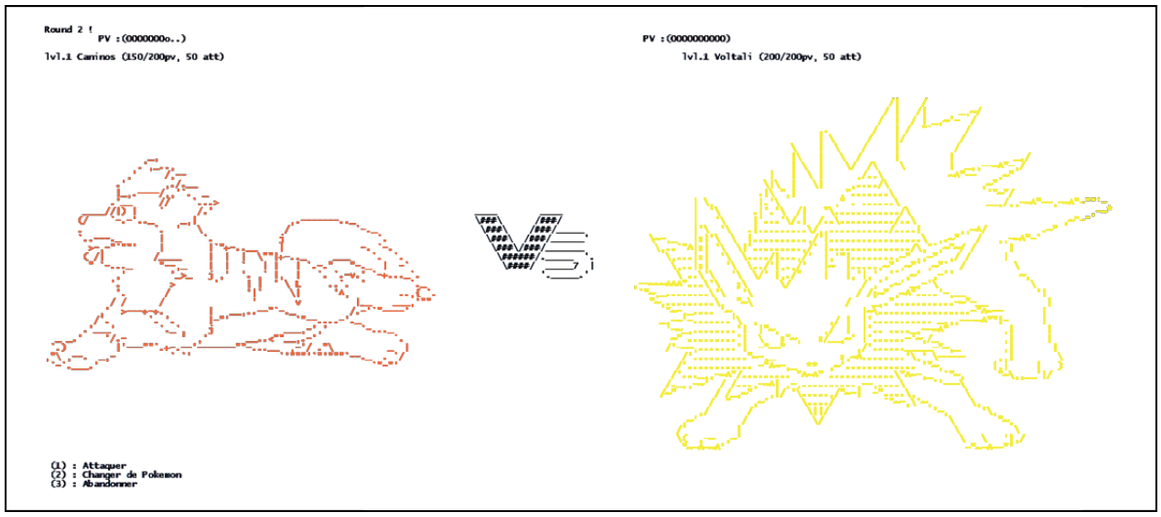
Projet Pokemon

2I002

NINI Yang
PINTO NUNES André



Quelques notes sur le projet

Dans ce projet, on cherche à créer un jeu qui ressemble au jeu Pokemon. Ce jeu simulera un monde de deux dimensions dans lequel le joueur pourra se deplacer. Dans ce monde, il y aura des Dresseurs contre lesquels le Joueur pourra se battre et des Infirmeries qui lui permettront de soigner ses pokemons.

1 Classes

Ce jeu est constitué par 20 classes au total. Parmi elles on trouve la classe Combat qui est la classe qui contient la méthode main.

1.1 La Classe Monde

- Un monde possède un et un seul joueur
- Ce joueur ne commencera jamais dans une case déjà occupée par un Dresseur
- Un monde possède entre un et vingt Dresseurs
- Un monde possède entre une et cinq Infirmeries
- La classe Monde a une méthode choix_OS() qui demande à l'utilisateur de choisir un OS (Linux ou Windows). Cette méthode doit être appelée dans le main() avant le début du jeu. Si ce n'est pas le cas, on choisira Linux par défaut.
- La classe Monde a une methode clearScreenLinux(), une methode clearScreenWindows() et une methode clearScreen(). Au début d'un jeu, on demande à l'utilisateur quel est son OS. La methode clearScreen() se chargera d'appeller clearScreenLinux() ou clearScreenWindows(), selon l'OS choisi.
- La classe Monde a une méthode choix_clavier() qui demande à l'utilisateur de choisir un clavier (AZERTY ou QWERTY). Cette méthode doit être appelée dans le main() avant le début du jeu. Si ce n'est pas le cas, on choisira AZERTY par défaut.
- La classe Monde a une methode bouger_AZERTY(), une methode bouger_QWERTY() et une methode bouger(). Au début d'un jeu, on demande à l'utilisateur quel est son clavier. La methode bouger() se chargera d'appeller bouger_AZERTY() ou bouger_QWERTY(), selon le clavier choisi.
- La classe Monde a une méthode lire() qui est utilisée pour déplacer le joueur (utilisée par la fonction bouger()). Cette méthode lit l'entrée du joueur et la retourne si elle est valide. Pour savoir si une entree est valide, cette méthode verifie quel clavier a été choisi. Pour un QWERTY on accepte WASD, pour un AZERTY on accepte ZQSD.
- La classe Monde a une méthode jouer() qui gère le jeu. Un jeu se termine quand tous les Dresseurs ont été vaincus où quand le nombre de défaites est égal à la moitié du nombre de Dresseurs

1.2 La Classe Dresseur

- Seulement quelques Guetters ont été ajoutés au code de base dans cette classe

1.3 La Classe Joueur

La méthode combat() de la classe joueur a été codée de telle sorte que l'on peut 'naviguer' sur trois menus avant d'attaquer.
On commence par afficher les pokemons (en utilisant leurs toString()) côte à côte.
On initialise la variable "attaque_choisie" à -1 (ce qui est invalide).
On propose à l'utilisateur un premier menu (en appelant la fonction lireAction()). Dans ce menu, on peut choisir une action parmi 3: attaquer, changer de pokemon ou abandonner. Lorsque l'utilisateur fais un choix, on change la valeur de "action_choisie".
Quel que soit le choix fait par l'utilisateur, "attaque_choisie" reste invalide et on rentre dans une boucle while. Tant que la variable "attaque_choisie" reste invalide, on ne sort pas de cette boucle.

À chaque tour de boucle, on vérifie la valeur de "action_choisie":

- si action_choisie = 0, on lance le menu lireAction() (où on choisit une action);
- si action_choisie = 1, on lance le menu lireAttaqueClavier() (où on choisit une attaque);
- si action_choisie = 2, on lance le menu choisir_pokemon() (où on change de pokemon);

Si l'utilisateur décide d'attaquer, alors "action_choisie = 1". Ainsi il est renvoié dans un deuxième menu qui lui permettra de choisir une attaque parmi celles disponibles. Il aura aussi la possibilité de revenir en arrière en choisissant l'attaque 0. S'il choisit une attaque valable, on sort de la boucle while et on lance le combat avec l'attaque choisie; s'il choisit l'attaque 0 (revenir en arrière), on ne change pas la valeur de "attaque_choisie" (elle reste non valide et on reste dans la boucle while) et "action_choisie" prends la valeur 0 ce qui lancera le menu lireAction().

Si l'utilisateur décide de changer de pokemon, alors "action_choisie = 2". Ainsi il est renvoié vers un autre menu qui lui permettra de choisir un pokemon parmi ceux disponibles. Il aura aussi la possibilité de revenir en arrière en choisissant l'option 0. S'il choisit un pokemon valable, on choisit l'action attaquer et on paralyse le pokemon (on n'a pas le droit d'attaquer quand on vient de changer de pokemon); s'il choisit l'option 0, alors "action_choisie" prends la valeur 0 et "attaque_choisie" reste non valide. Ainsi, on reste dans la boucle while et au prochain tour de boucle on lance le menu lireAction().

Si l'utilisateur décide d'abandonner, on sort du combat (avec un break).

Après chaque combat, si un des pokemons du joueur est KO, on l'oblige à changer de pokemon.

1.4 La Classe Pokemon et ses subclasses

Chaque pokémon hérite d'un type (class Herbe, class Feu, ...). Chaque type hérite de class Pokemon. Or un pokémon peut avoir 2 types. Puisque l'héritage multiple est impossible en Java, nous avons dû trouver une alternative. Nous avons décidé de créer des pokémons "schizophrénes". Un pokémon type Terre + Feu est un pokémon qui hérite de Terre et qui a pour atribut une instance de Feu. C'est comme si ce pokémon avait un autre pokémon qui vit dans sa tête, une double personnalité. Ainsi, lorsque ce pokémon utilise une attaque type Feu, c'est sa deuxième personnalité qui attaquera, sinon, c'est lui qui attaque. C'est pourquoi dans chaque pokémon schizophrène, nous avons dû faire un override de attaque().

Nous avons aussi changé les deux fonctions attaque() de pokémon pour la raison suivante:
Si un pokémon est fort contre un autre pokémon (à cause de leurs types), un multiplicateur s'aplique lors de chaque attaque. Nous avons trouvé plus logique de ne faire attention qu'à la nature de l'attaque. En effet, si un pokémon Èlectricité + Feu attaque un pokémon Eau, il peut utiliser 3 types d'attaque:

- Une attaque Feu qui serait peu éfficace;
- Une attaque Électricité qui serait très éfficace;
- L'attaque défaut qui n'a pas de type qui serait simplement éfficace;

Cependant, si un pokémon qui a deux types subit une attaque, on prends en compte ses deux natures. En effet, si un pokémon Èlectricité + Feu attaque un pokémon Eau + Herbe, il peut utiliser:

- Une attaque Feu qui serait à la fois peu éfficace et très éfficace (donc simplement efficace);
- Une attaque Électricité qui serait très éfficace;
- L'attaque défaut qui n'a pas de type qui serait simplement éfficace;

Nous avons aussi ajouté une méthode health_bar() qui crée une barre de vie pour les pokémons.