Eva BEZIAU - Juliette VAUCHEZ

Bordeaux INP ENSC

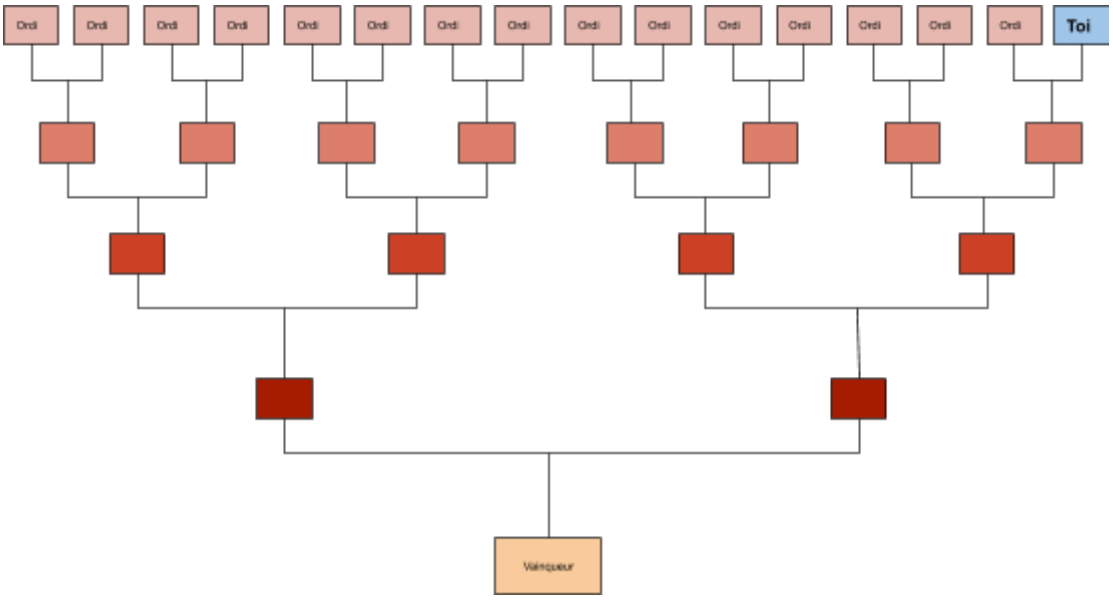
Programmation avancée - Projet 2019

Pokédojo

CONTEXTE

Dans ce projet, il nous est demandé de créer un tournoi du meilleur dresseur, dans lequel le joueur s'inscrit parmis les autres compétiteurs (qui sont des ordinateurs) et affronte ses adversaires avec son équipe de pokémons pour tenter de devenir le grand vainqueur !

Le tournoi s'organise initialement avec 16 joueurs, dont 15 sont simulés par l'ordinateur. Il faudra ainsi au joueur gagner 3 tours avant d'accéder à la finale qui opposera les deux joueurs restants.



Nous avons choisi d'intégrer une option qui permet au joueur de choisir sa difficulté de jeu entre trois niveaux, proposant ainsi 8 joueurs (3 tours de jeu), 16 joueurs (4 tours) et 32 joueurs (5 tours). Cela permet de moduler la longueur de jeu également.

Pour ce tournoi, chaque joueur dispose d'une équipe de 3 pokémons, provenant d'une liste de prédéfinie.

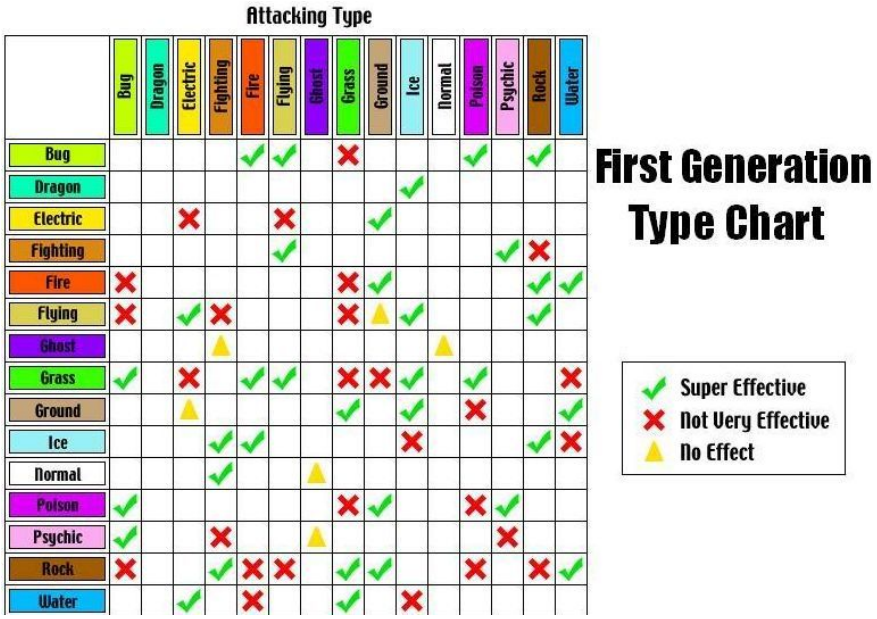
La BDD, interne, propose des pokémons différents, correspondant à 60 pokémons de la génération 1. Nous avons pensé à externaliser dans un second temps cette liste dans un fichier csv, ce qui permettrait de modifier les pokémons sans compiler le code.

FONCTIONNALITÉS DU JEU & CONCEPTION ADAPTÉE

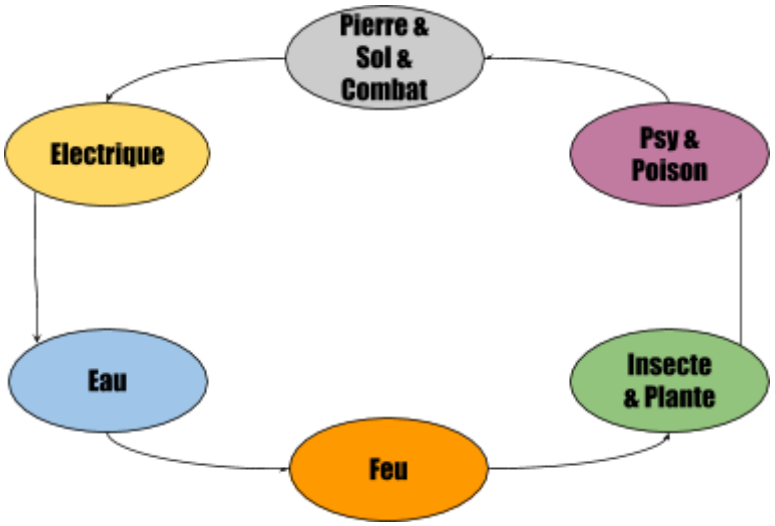
Ce jeu consiste en des combats de pokémons, et nous avons donc créé des classes en fonction, séparant Pokémon, Type, Dresseur et Tournoi. Nous avons également créé une différenciation entre les classes Dresseur et Joueur. La classe Joueur hérite de la classe Dresseur, afin de faciliter les comportements différents au cours du jeu.

Nous avons dans un premier temps créé la classe Pokemon.cs, possédant les attributs nom, PV et PV provisoires, puissance d'attaque, son type et son nombre de victoires consécutives. Nous avons opté pour une version simplifiée du jeu d'origine, à savoir que nous n'avons conservé que le premier type de chaque pokémon de façon à ce que chacun n'en possède qu'un.

Nous avons choisi de ne pas déclarer de faiblesses dans cette classe, mais d'implanter dans la classe Type, une fonction qui détermine la faiblesse associée selon un schéma bien précis, que nous avons établi suite à une rapide étude de l'existant. Ci-dessous un tableau récapitulatif des types effectifs du jeu pokémon et de leurs effets sur les uns sur les autres.



A partir de ce modèle, nous avons créé une version simple indiquant quel type agit plus efficacement sur l'autre. On obtient un cycle qui va nous être utile dans le code. Ce schéma peut aussi être utile dans les stratégies mises en place, que ce soit côté joueur ou pour l'ordinateur puisque l'ordinateur pourrait choisir des pokémons plus efficaces contre ceux du dresseur adverse.



Le joueur se voit attribuer une équipe de trois pokémons attribués aléatoirement. Nous avions réfléchi à une fonctionnalité qui aurait permis au joueur réel de choisir dans un premier temps son équipe avant de répartir aléatoirement le reste des pokémons disponible entre les ordinateurs. Nous n'avons finalement pas retenu cette idée pour deux raisons. Premièrement, cela enlèverait un aspect du jeu selon lequel les joueurs doivent composer avec ce qu'ils ont. Un bon dresseur est censé l'être avec n'importe quel pokémon. Cela permet de rajouter du challenge aux joueur expérimentés et de rendre le jeu accessible aux novices qui ne connaîtraient pas toutes les capacités des pokémons. La deuxième raison est que nous voulions que plusieurs dresseurs puissent avoir les mêmes pokémons (un pokémon n'est pas exclusif à un dresseur). Nous avons également choisi, pour les mêmes raisons, de laisser possible le tirage de deux mêmes pokémons dans la même équipe.

Nous avons dû créer une fonction Clone() pour pouvoir assigner plusieurs fois le même pokémon à des dresseurs. En effet sans cela, le pokémon était "physiquement" assigné à plusieurs dresseurs, et ainsi les dégâts infligés sur le pokémon étaient en même temps infligés sur tous les dresseurs possédant le même pokémon. Cela entraînait des erreurs telles qu'au début d'un combat, parfois, un de nos pokémon était déjà KO (l'initialisation des équipes se faisant à la fin d'un tour).

En étudiant rapidement les capacités des pokémons, nous avons remarqué que certains (Rondoudou par exemple) étaient de force bien supérieure (PV très élevés avec attaque comme les autres) ; il était alors quasiment impossible de gagner face à eux excepté avec le même pokémon. De même, de manière générale les PV étaient trop bas pour les attaques et les pokémons étaient mis k.o en 1 ou 2 coups maximum. Pour empêcher que cela ne fausse notre jeu, nous avons procédé à quelques remaniements de notre liste de pokémons pour lisser un peu ces capacités et rendre le jeu plus équitable.

Rappelons brièvement les règles d'un tournoi :

★ Tout d'abord, un tournoi est composé de 3, 4 ou 5 tours. Durant chacun des tours, sont effectués des combats entre 2 dresseurs. Les gagnants d'un tour s'affrontent ensuite au tour d'après.

★ Durant un combat, chaque joueur choisit un pokémon de son équipe qui devient le pokémon actif. Cette sélection est aléatoire pour les joueurs simulés par l'ordinateur. Ce choix est réalisé au début de la fonction LancerCombat() de la classe Tournoi.

★ Les deux pokémons actifs s'affrontent au tour par tour. Celui qui attaque en premier est tiré au sort. Pour cela nous avons pour cela choisi de réaliser "mélange" aléatoire à l'instanciation de la classe Tournoi : le positionnement aléatoire se fait donc à la création d'un nouveau tournoi. Ainsi, il n'y a plus de tirage aléatoire des dresseurs dans le tournoi et on fait toujours commencer un combat entre les dresseurs D1 et D2 par une attaque de D1.
Le pokémon attaquant inflige au pokémon actif de son adversaire un nombre de marqueurs de dégâts égal à sa puissance d'attaque, doublée en fonction de la faiblesse de son adversaire. Pour cela nous avons créé, sous forme de propriété, un attribut PVProvisoire dont la valeur est diminuée au fil des attaques.
Le pokémon attaqué devient ensuite attaquant et la boucle recommence, jusqu'à ce qu'un des dresseurs soit k.o.

★ Lorsque le nombre de marqueurs de dégâts sur un pokémon devient supérieur ou égal à son nombre de PV (c'est-à-dire dans notre code quand PVProvisoire atteint 0) celui-ci est mis k.o. et le joueur doit sélectionner un nouveau pokémon actif parmi les pokémons restants de son équipe. Le match continue entre le pokémon actif survivant et le nouveau pokémon actif, et ainsi de suite jusqu'à la fin du combat.

★ Le premier joueur dont tous les pokémons sont mis k.o. perd le combat, et le vainqueur avance au tour suivant.

★ Un centre de soin est installé dans ce tournoi, ainsi les pokémons de tous les dresseurs sont soignés à la fin de tous les combats d'un tour. Cette fonction, appelée InitPokemon(), assigne aux PVProvisoire la valeur des PV.

Un Tournoi est donc composé de plusieurs tours, qui correspondent à des phases de combats, des duels qui se déroulent simultanément. Nous avons choisi de ne pas créer de classe Combat car nous ne l'avons pas jugée utile.

Une fonction LancerCombat() exécute le match, et l'organise en boucles. Tant qu'aucun dresseur n'est k.o. (ce qui est vérifié par la fonction DresseurKO()), on continue à jouer. Les joueurs choisissent leurs pokémons, puis commencent à attaquer à tour de rôle. Une condition au début de la fonction permet de faire changer de pokémon si celui qui est actif est k.o. Lorsqu'un dresseur a gagné, il passe au tour suivant.

Après avoir mis en place ces règles de base, nous avons prévu des fonctions supplémentaires pour rendre le jeu plus intéressant.

★ Après deux victoires consécutives, un pokémon évolue grâce à la fonction EvolutionPokemon(). Ses caractéristiques de vie et d'attaque s'améliorent en s'incrémentant de +10. De plus, afin d'avantager ce pokémon au cours du combat, PVProvisoire est également incrémenté de 10. Ce phénomène peut se réitérer, et ainsi gagner +10 en PV et en attaque à chauque fois. Son nom change alors et devient "Super*nompokémon*" puis "SuperSuper*nompokémon*". Cette évolution est permanente, c'est-à-dire qu'elle existe jusqu'à la fin du tournoi.

★ Un joueur (non ordinateur) a désormais la possibilité de faire battre en retraite son pokémon actif pour le remplacer par un autre au cours de l'affrontement. Il conserve par contre les dégâts infligés. Cela ne prend pas la place d'un tour d'attaque. La fonction associée est SwitchPokemon(). A noter que lorsqu'il choisit cette option, le nombre de victoire consécutives est remis à 0. Cette fonctionnalité n'est possible que si le joueur ne vient pas de changer de pokémon pour cause de k.o.

Il y a certaines règles avancées que nous aurions aimé mettre en place mais que nous n'avons pas pu réaliser compte-tenu du temps restant, car notre souhait était avant tout de fournir une réponse fonctionnelle au projet.

D'une part, nous avons réfléchi à des stratégies d'attaque particulières. Il y en aurait trois : l'attaque normale (celle présentée dans le code), l'attaque double (qui inflige le double des dégâts mais fatigue tellement le pokémon qu'il ne peut plus attaquer au tour suivant), et l'attaque spéciale (qui inflige un peu moins de dégâts mais des effets à long terme propres aux types, tels que le gel, la brûlure, l'endormissement etc). Cette capacité aurait pu permettre d'autres stratégies d'intelligence comme réaliser une attaque si elle lui permet de gagner, ou sinon jouer aléatoirement.

La stratégie d'intelligence de l'ordinateur qui consiste à mettre un pokémon de type plus fort que l'adversaire (en fonctionnant avec les relations de faiblesse entre types) a été envisagée également.

Nous avons produit une version papier de ces réflexions et les avons jugées trop longues à mettre en place pour le temps restant.

Au niveau de l'affichage, nous avons représenté graphiquement la barre de vie pour une meilleure visualisation. Nous avions également souhaité faire une réalisation graphique de l'avancée du tournoi (similaire au schéma que vous pouvez trouver au début de ce rapport).

Ces pistes pourraient être des améliorations à apporter au jeu si le projet devait être continué.

GESTION DE PROJET

Le rendu de projet est prévu pour le vendredi 24 mai 2019 avant 23h et nous nous sommes organisées en conséquence. Nous avons utilisé GitHub pour partager notre code et nous permettre de travailler en même temps sur le projet, étant de ce fait plus efficaces. Nous avions parfois les mêmes créneaux en TP et nous pouvions alors échanger

directement, mais nous travaillions aussi indépendamment et avons veillé à garder une bonne communication afin d'assurer la cohérence du code.

Tâche	22/04 au 28/04	29/04 au 5/05	6/05 au 12/05	13/05 au 19/05	20/05 au 24/05
Premières réflexions sur le sujet					
Elaboration de schémas récapitulatifs					
Formation Git et GitHub					
Définition des classes nécessaires					
Création de la classe Pokemon					
Création des autres classes					
Elaboration du système de combat					
Création du système de combat					
Phase de test du jeu					
Dernières modifications du code					
Représentation graphique de l'évolution du tournoi					
Rédaction du rapport					

Nous avons réalisé plusieurs tests au cours de la conception afin de vérifier la cohérence du code et son bon fonctionnement.

Nous n'avons pas conduit des tests unitaires sur chacune des classes car elles comportent beaucoup d'interactions et il a fallu attendre qu'elles soient toutes relativement avancées.

Cependant, toutes les classes ont finalement pu être testées à travers les tests des fonctionnalitées du jeu, qui utilisent les classes pour fonctionner. Le fonctionnement correct d'une fonction nous a permis de conclure que les classe étaient valides. Une fois les relations primaires codées, nous avons procédé à des tests réguliers puisque à chaque nouvelle fonctionnalité nous faisions tourner le programme afin de vérifier les effets escomptés. Nous avons souvent passé des parties de code en commentaire pour tester avec ou sans, etc.

A chaque partie lancée nous assurions le suivi avec attention. Nous avons testé le jeu toutes les deux indépendamment pour pouvoir rendre compte au mieux d'une part de son bon fonctionnement, ainsi que de l'organisation « graphique » que nous avons mise en place afin de rendre le jeu plus lisible et agréable.

C'est de cette façon que nous avons pu remarquer et corriger notamment le déséquilibre des points de vie et d'attaque, ainsi que des erreurs inhérentes à notre code.

CONCLUSION

Nous fournissons un programme abouti qui permet d'effectuer des combats de pokémons selon les règles de tournoi détaillées dans les instructions et dans ce rapport. Certaines règles avancées ont pu être mises en place et rendent le jeu plus complexe. La possibilité que nous avons pu mettre en place de moduler la longueur du tournoi selon les envies du joueur ajoute une plus-value à notre jeu.

Nous pouvons cependant souligner certaines limites, qui résident dans le temps nécessaire pour les mettre en place. Nous avions réfléchi à ces solutions sur papier et elles

pourraient être mises en place en quelques jours, qui nous manquent. Cela n'est pourtant pas dû à une mauvaise gestion du planning prévisionnel puisque nous sommes restées dans les temps le concernant.

Ce projet nous a permis d'une part de nous former à Git et à GitHub, que nous n'avions pas utilisé sur les précédents projets. Nous sommes désormais conscientes et convaincues du réel avantage que ces plateformes apportent et nous les utiliserons de nouveau par la suite.

Nous avons également pu, à travers ce projet, mettre en pratique les techniques de programmation orientée objet vues au cours du semestre et nous les approprier.