Rapport de développement

L’équipe

Participants

Le projet a été réalisé par :

- Ewen Bouquet,
- Alexandre Da Silva Maia.

Nous sommes tous les deux en deuxième année de DUT Informatique à l’UPEM et nous sommes alternants (TD1, TP Alpha).

Tâches réalisés

Les tâches ont été réparties de la manière suivante :

- Ewen s’est occupé du modèle (package model) et du controller (package controller),
- Alexandre s’est occupé de la vue (package view).

Difficultés rencontrées

Le manque de temps

La principale difficulté du projet a été la gestion du temps. En effet, nous avons eu moins d’un mois pour réaliser le jeu. De plus, comme nous sommes alternants, nous n’avons pas eu de vacances et donc il était difficile de trouver du temps à consacrer au projet. Cela a été d’autant plus compliqué que la semaine de “rentrée scolaire” est pourvue de nombreux partiels.

Se former sur les librairies graphiques

Une autre difficulté importante a été de se former à la bibliothèque graphique Swing. En effet, les méthodes de conception utilisées sont spécifiques et complexes d’utilisation. Une partie importante du projet a donc été consacrée à la lecture de la javadoc associée et autres tutoriels trouvés sur internet.

Faire communiquer le contrôleur et la vue

Enfin, étant donné que nous nous sommes répartis les tâches de sorte à pouvoir travailler de son côté sans se gêner, il a été difficile de faire communiquer le contrôleur et la vue. Nous avons donc dû nous concerter afin de trouver la meilleure solution.

L'architecture du projet

Ce qui a été implanté

En se basant sur le sujet, nous avons réalisé :

- Les types,
- Le pokédex national,
- Les capacités,
- Les pokémons,
- Le chargement,
- La sauvegarde,
- Le mode original des combats,
- La ligue par défaut,
- La ligue “made in DUT2”,
- La gestion des types,
- Le multijoueur,
- L’acquisition des capacités (sans prise en charge du niveau),
- Le soin de l’équipe (choix fait de soigner que les pokémons vivants).

La structure du projet

Le projet est basé sur le MVC (modèle / vue / contrôleur). Il est organisé de la manière suivante :

- model
    - *Loader.java (Interface de chargement)*
    - *Model.java (Classe contenant les constantes et méthodes utilitaires)*
    - capacities
        - *Capacity.java (Classes des capacités d'attaque des pokémons)*
        - *CapacityLoader.java (Classe qui charge les capacités)*
        - *CapacityType.java (Enumération des types de capacités des pokémons)*
    - fights
        - *ComputerFight.java (Classe de combat contre l'ordinateur)*
        - *Fight.java (Classe de combat multijoueurs)*
        - *FightTurn.java (Enumération des actions possibles d'un combat)*
    - league
        - *League.java (Classe de league pokémon)*
    - pokedex
        - *Pokedex.java (Classe de conteneur de modèles de pokémons)*
    - pokemon
        - *DamagesCalculator.java (Classe qui calcule les dégâts d'une attaque)*
        - *Pokemon.java (Classe de pokémons parés au combat)*
        - *PokemonModel.java (Classe de modèle de pokémon)*
    - trainer
        - *Trainer.java (Classe d'équipe de pokémons)*
        - *UserTrainer (Classe d'équipe de pokémons de l'utilisateur)*
    - type
        - *Type.java (Enumération des types de pokémon)*
        - *TypeLoader.java (Classe de chargement du type de pokémons)*
        - *TypesCell.java (Classe utilitaire qui contient des types de pokémons)*
- view
    - *GameWindow.java (Classe qui définit la fenêtre principale de jeu)*
    - *ImageLoader.java (Classe qui permet de charger les images)*
    - *ViewManager.java (Classe qui gère l'affichage graphique)*
    - panels
        - *ComputerBattlePanel.java (Classe d'affichage d'un combat)*
        - *LeagueBattlePanel.java (Classe d'affichage d'un combat de league)*
        - *LeaguePanel.java (Classe d'affichage d'une league)*
        - *MenuPanel.java (Classe d'affichage du menu)*
        - *MultiplayerBattlePanel.java (Classe d'affichage d'un combat multijoueur)*
        - *MyTeamPanel.java (Classe d'affichage de l'équipe)*
        - *PokedexPanel.java (Classe d'affichage du pokédex)*
        - *PokemonAttacksPanel.java (Classe d'affichage des capacités)*
        - *PokemonPanel.java (Classe d'affichage d'un pokémon)*
    - util

        - *JUtilities.java (Classe contenant des méthodes utiles pour la manipulation de l'affichage)*
- controller
    - *Controller.java (Classe qui contient les méthodes de contrôle du jeu)*

Des fichiers annexes ont aussi été utilisés :

- assets/grid_types.csv (Types des pokémons et constantes d'attaque)
- assets/moves.csv (Capacités des pokémons)
- assets/pokedex.csv (Pokédex et pokémons associés)
- assets/pictures/*.png (Images des pokémons)
- assets/other/*.png (Autres images associées)
- config/serializable.ser (Fichier de sauvegarde de l'équipe de pokémons de l'utilisateur).

Pour plus d'informations, vous pouvez vous référez à la javadoc, rédigée entièrement en anglais pour chaque composants et méthodes du projet.

Ce qu'il reste à implanter

Nous avons réalisé l'ensemble des points obligatoires à implémenter. Nous avons même programmé certaines caractéristiques du sujet spécifiées dans le sujet qui étaient optionnels pour les alternants.

Perspectives d'évolution du projet

En se basant sur le sujet, nous n'avons pas réalisé les points optionnels suivants (à partir du niveau VIII) :

- Les objets,
- L'état du pokémon,
- Level Up,
- L'évolution du pokémon,
- Tenez moi,
- Le mode duo et trio.

Pour le développeur

Ajout d'un mode de combat

Pour ajouter un mode de combat, il faut créer une nouvelle classe qui hérite de Fight.java. On peut alors gérer les caractéristiques de combats avec des @Override sur les méthodes précédemments écrites. Pour ajouter un mode de combat contre l'ordinateur, on peut d'ailleur directement de ComputerFight.java et utiliser la même méthode.

Ajout d'une league spéciale

Il est facile de spécifier les pokémons souhaités via les constructeurs préexistants de la classe League.java. Cependant, si une league plus complexe veut être réalisée, on utilisera l'héritage sur la classe League.java.

Gestion des caractéristiques des pokémons

La gestion de caractéristiques précises sur les pokémons s'effectue directement dans la classe Pokemon.Java. Pour l'intant, aucune classe n'hérite de Pokemon.java mais cela pourrait être interessant notamment dans le cas de la gestion d'effets et d'attaques spéciales (vol-vie, dégats sur le long terme, etc...).

Ajout d'un écran

Pour ajouter un écran, il faut créer une nouvelle classe héritant de JPanel (librairie Swing) et initialiser tous les composants comme souhaité. Il faut ensuite ajouter une instance de cet objet au niveau de la classe ViewManager.java et au niveau du "CardLayout" présent dans la classe GameWindow.Java. Finalement, il reste à implémenter la méthode d'affichage de la carte correspondante.

Modification des constantes de jeu

En fonction des constantes, il suffit juste de changer les valeurs dans les fichiers annexes (csv) ou bien dans la classe Model.java. Il n'y a pas de modifications à faire directement dans le code source.