



INTRO:

Bienvenue dans PokéMon par Diego, Rémi et Maël ! Dans ce projet, notre objectif était de reproduire le jeu PokéMon que l'on connaît tous sur le langage de programmation Python. Rappelons le but du jeu avant de rentrer plus en détails dans les fonctionnalités et le code. Dans notre jeu, nous incarnons un dresseur qui a pour objectif de capturer tous les pokémons disponibles sur la map, en les combattants, à l'aide des pokémons qu'il possède déjà dans son pokédex. D'ailleurs, faites attention à où vous vous baladez, vous risqueriez de tomber sur les membres de la team Rocket !

Pour la réalisation de ce projet nous avons utilisé plusieurs librairies python, tel que PyQt5, numpy ou encore math. Nous avons aussi utilisé le modélisateur de fenêtres graphiques Qt Designer.

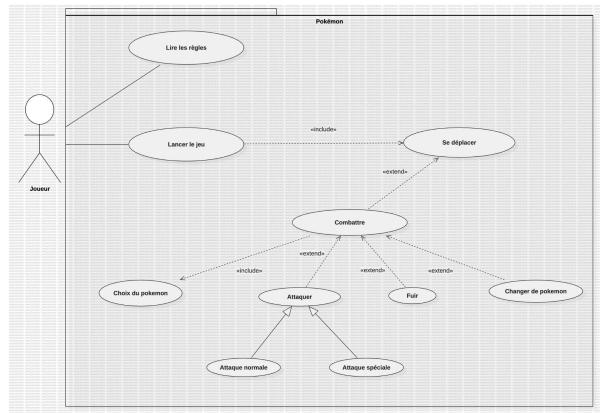
I - MODÉLISATION DU PROJET

A - Modélisation UML

1 - Diagramme de cas d'utilisation:

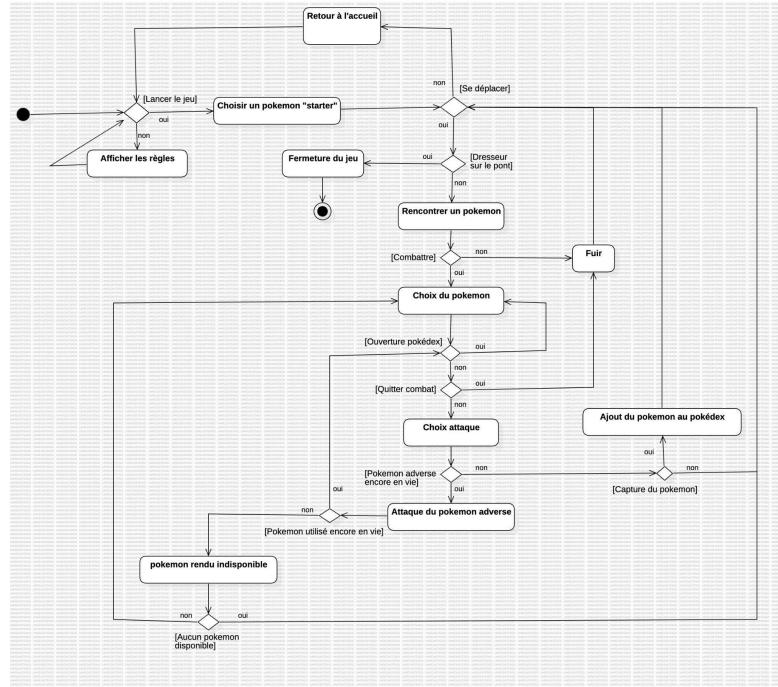
L'objectif de ce diagramme est d'identifier et clarifier les fonctionnalités que doit fournir un logiciel à un utilisateur.

Dans un premier temps, le joueur à le choix entre “lire les règles” ou “lancer le jeu”. “Lancer le jeu” inclut les déplacements (“se déplacer”) auxquels nous avons relié l'action de “combattre”, qui elle-même est associée aux options suivantes: “attaquer”, “fuir” et “changer de pokémon”, et qui inclut le “choix du pokémon”. Enfin, “attaque normale” et “attaque spéciale” sont une subdivision de “attaquer” spécifiant ainsi les types d’attaques qui en découlent.



2 - Diagramme d'activité:

Le diagramme d'activité quant à lui sert à décrire sous forme de flux ou d'enchaînement d'activités la logique du jeu.



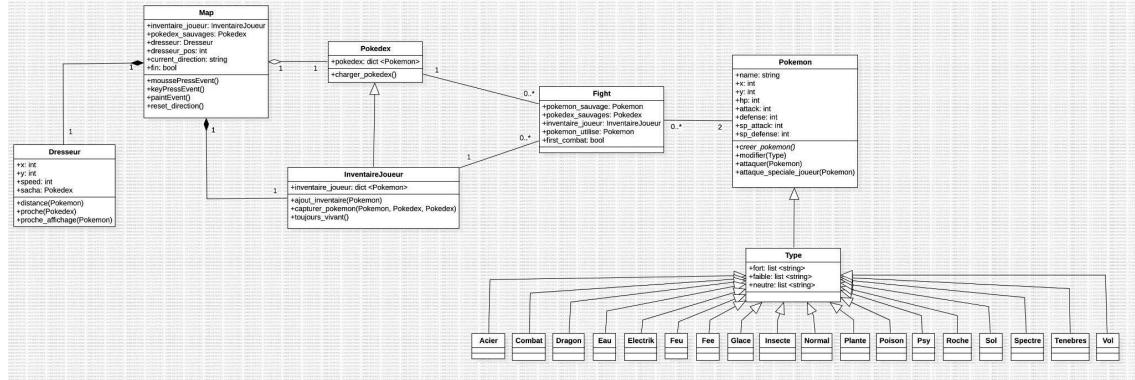
Les premières actions possibles par le joueur sont de lire les règles ou de commencer le jeu, menant au choix du pokémon starter. Le joueur a le choix entre Bulbizarre, Salamèche et Carapuce. Il peut ensuite se déplacer sur la carte jusqu'à ce qu'il se trouve sur la case d'un pokémon sauvage, dont les coordonnées sont déterminées de manière aléatoire à la création de la carte. Il peut choisir de combattre ou fuir. S'il choisit la première option, alors 2 conditions vont être vérifiées : le clic sur le pokédex ou sur le bouton de fuite. Si aucune n'est vérifiée, alors c'est le type d'attaque qui est attendu. Si le pokémon adverse est K.O., le joueur peut le capturer ou partir sans le capturer. Sinon, le pokémon adverse attaque à son tour, puis si le pokémon utilisé par le joueur est encore en vie, le combat continue. Sinon, on vérifie si aucun pokémon n'est disponible dans le pokédex. Si c'est le cas, le combat prend fin, sinon, le joueur doit choisir un autre pokémon.

La partie prend fin lorsque le dresseur est positionné au niveau du pont à gauche de la carte.

Il est possible en permanence de revenir à l'accueil, permettant de générer des emplacements de pokémons sauvages différents et de remettre à 0 le pokédex.

3 - Diagramme de classe:

Enfin le diagramme de classe est utilisé pour représenter l'architecture conceptuelle du jeu en décrivant les classes que le système utilise et les liens entre les classes.



Nous avons d'abord décidé de créer une classe Map regroupant l'ensemble des données permettant le déroulement du jeu à partir de la carte. C'est-à-dire :

- un attribut “pokedex_sauvage” de type Pokedex qui correspond à l'ensemble des pokémons sauvages présents sur la carte
- un attribut “inventaire_joueur” de type InventaireJoueur qui regroupe l'ensemble des pokémons détenus par le joueur
- un attribut “dresseur” de type Dresseur
- deux autres attributs : “current_direction” qui définit la direction actuelle du dresseur pour son affichage et “fin” qui affirme si le jeu est terminé ou non

La classe Dresseur stocke les informations essentielles liées au dresseur, telles que ses coordonnées, sa vitesse et enfin son pokédex. Les méthodes associées sont : “distance” qui donne la distance du dresseur à un pokémon. “proche” et “proche_affichage” qui traite de la proximité du dresseur pour l'affichage ou pour déclencher les combats.

La classe Pokedex qui correspond à un dictionnaire de pokémon, alors que la classe InventaireJoueur hérite de Pokedex à la différence qu'elle possède les méthodes “ajout_inventaire”, “capturer_pokemon” et “toujours_vivant”.

La classe Pokemon quant-à elle possède les attributs coordonnées, hp (niveau de vie), attack (points d'attaque), defense (points de défense), et enfin les points d'attaque et défense correspondant aux attaques spéciales. La classe Type hérite de Pokemon (les attributs fort, faible et neutre correspondent à des listes des types envers lesquels le type en question est faible, fort ou non) et toutes les classes des différents types héritent elles-mêmes de Type.

Enfin, la classe Fight utilise 2 Pokémons (le pokémon utilisé et le pokémon adverse), 1 Pokédex (celui contenant les pokémons sauvages), l'inventaire du joueur et un booléen first_combat qui différencie si le combat vient de commencer et le joueur doit attaquer ou s'il vient d'ouvrir le pokédex et que c'est l'ordinateur qui doit attaquer en premier.

B - MODÉLISATION ALGORITHMIQUE

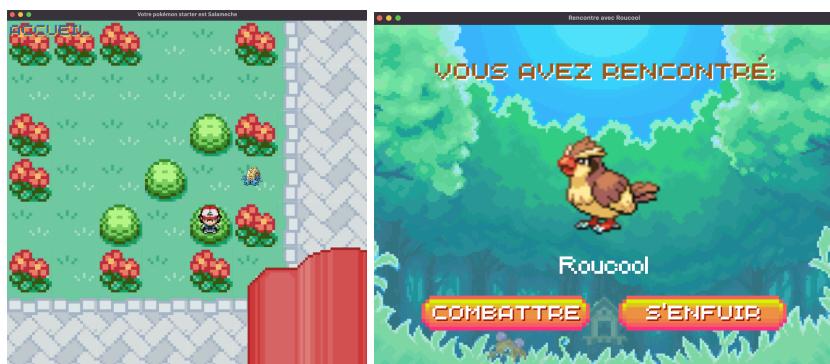
Notre code fonctionne avec différents fichiers. Le fichier principal Jeu.py fait appel à la plupart des fichiers pour jouer pleinement au jeu, c'est ici que la Map est générée et c'est ici que les déplacements visuels du dresseur sont gérés. Le fichier Poke.py permet de créer: la classe Pokemon (création d'un pokémon), la classe Type (création des différents types), une classe pour chaque type (feu, glace, plante,... ces classes héritent de Type), la classe Pokedex (création d'un dictionnaire contenant des pokémons) et la classe InventaireJoueur (création du pokédex du joueur, cette classe hérite de Pokedex). C'est à partir de ces classes que tout le jeu tourne. Le fichier dresseur.py permet de gérer ce qui est propre au dresseur: comment il se déplace et comment il rencontre un pokémon. La vitesse de déplacement ainsi que la distance de rencontre et d'affichage des pokémons sauvages sont gérés dans ce fichier et peuvent donc être modifiés avant de lancer le code. Le fichier Pokemon_coord.py permet de générer un fichier csv nommé "pokemons_a_capturer.csv" à partir du fichier "pokemon_first_gen.csv". Il crée un certain nombre de lignes (choisi dans le fichier Jeu.py en ligne 102) où chaque ligne correspond à un pokémon avec des coordonnées. Le taux de rareté de chaque pokémon (défini dans la colonne "Rare" du fichier "pokemon_first_gen.csv") augmente plus ou moins sa probabilité d'apparaître une ou plusieurs fois dans le fichier. Un Rattata peut apparaître plusieurs fois alors qu'un Artikodin a peu de chance d'apparaître. Ce fichier correspond à tous les pokémons présents sur la map. Enfin, tous les autres fichiers sont globalement des fichiers visuels. Le fichier Combat.py gère les combats entre un pokémon sauvage et le pokémon du dresseur. Il fait appel aux fonctions d'attaque définies dans le fichier Poke.py. Pour comprendre l'utilité de chaque fichier nous avons écrit un main pour que chaque fichier (notamment les fichiers visuels) puisse se lancer individuellement. En revanche, vous ne pourrez pas jouer entièrement au jeu à partir de ces mains, ils ne sont là que pour comprendre ce que fait le fichier exécuté.

II - MANUEL D'UTILISATEUR

Notre jeu commence avec une page d'accueil qui vous invite à choisir entre jouer ou lire les règles. Les règles vous précisent le but de notre jeu: capturer des Pokémons sauvages. Si vous choisissez de jouer vous allez ouvrir une nouvelle fenêtre vous proposant de choisir votre Pokémon starter, ce sera votre tout premier Pokémon et c'est avec lui que vous jouerez votre premier combat. En effet vous allez capturer des Pokémons en participant à des combats, si vous les remportez vous pourrez capturer le Pokémon sauvage.

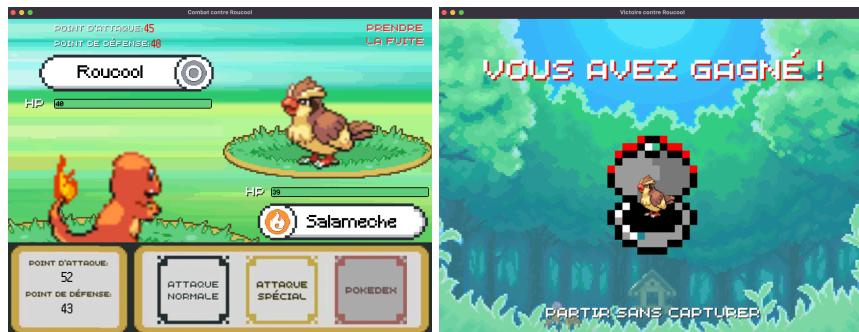


Après avoir choisi votre starter vous arrivez sur la Map, ici vous pouvez vous déplacer avec les flèches directionnelles de votre clavier, sur cette Map, des Pokémons sauvages sont cachés et apparaissent quand vous êtes proche d'eux. Ces Pokémons apparaissent de manière aléatoire sur la Map et possèdent un taux de rareté ce qui fait qu'un Rattata peut apparaître plusieurs fois alors qu'un Artikodin a peu de chance d'apparaître. Lorsque vous êtes "sur" un Pokémon (lorsque vous partagez la même position) vous aurez le choix entre le Combattre ou Fuir, car certains Pokémons peuvent être trop forts pour votre Pokémon.



Une fois le combat commencé vous avez plusieurs possibilités: utiliser l'attaque normale du Pokémon, utiliser son attaque spéciale ou changer de Pokémon. L'attaque normale du Pokémon repose sur ses statistiques d'attaque normale et les dégâts sont calculés de la manière suivante: partie entière de $((500 * \text{attaque normale} * \text{coef de type} / \text{défense normale}) / 50) + 2$. Avec attaque normale la statistique d'attaque normale de l'attaquant et défense normale la statistique de défense normale du défenseur. Coef de type correspond au multiplicateur de type, en effet les Pokémons possèdent des types et certains sont forts contre d'autres et faibles contre certains. Le détail des coefficients sera donné en annexe. Pour l'attaque spéciale, elle utilise la même formule mais avec les statistiques spéciales des Pokémons, cette attaque spéciale n'est disponible que tous les 2 tours et n'est pas forcément plus efficace car certains Pokémons ont des statistiques d'attaque spéciales plus faibles que l'attaque normale. Au dresseur de bien choisir et de bien connaître ses Pokémons. Pour changer de Pokémon il faut en posséder au moins 2 et le fait de changer vous utilisera un tour. Notre mode de combat fonctionne au tour par tour, on commence par attaquer ou changer de Pokémon ensuite le Pokémon sauvage contre-attaque. Le combat se finit quand le Pokémon adverse n'a plus de HP (point de vie) ou que tous vos Pokémons n'ont plus de HP. Une fois le combat fini vous pouvez choisir entre capturer le Pokémon ou

partir sans le capturer. Vous pouvez également fuir le combat même s'il est toujours en cours.



III - CONCLUSION ET REMARQUES:

Pour conclure nous voulons souligner que réaliser ce jeu a été un énorme plaisir pour tous les trois, le thème abordé a énormément aidé à nous motiver pour réaliser le meilleur jeu possible. Prendre en main PyQt et Qt Designer a été assez compliqué au début mais une fois la bibliothèque et le logiciel bien assimilé, tout déroulait. On a donc pris plaisir à rendre nos interfaces les plus plaisantes possibles. L'utilisation de GitHub a également été compliquée au début mais une fois bien comprise elle nous a grandement facilité la vie. Nous avons aussi pu nous rendre compte de l'utilité de la programmation orientée objet notamment pour la construction des pokémons. Enfin, si nous pouvions apporter des améliorations à notre jeu, améliorations que nous avons choisi de ne pas faire par choix lié au temps, nous implémenterions un système de sauvegarde pour pouvoir rejouer avec le même inventaire, nous ajouterions également un bouton qui permet d'accéder à notre inventaire à tout moment et ajouterions également un système d'évolution. Mis à part cela, nous sommes satisfaits de ce que nous avons réalisé dans le temps imparti.

ANNEXE:

	Défense																	
	Aacier	Combat	Dragon	Eau	Feu	Electrik	Fée	Glace	Insecte	Normal	Plante	Poison	Psy	Roche	Sol	Spectre	Ténèbres	Vol
Aacier	0,5	1	1	0,5	0,5	0,5	2	2	1	1	1	1	1	2	1	1	1	1
Combat	2	1	1	1	1	1	0,5	2	0,5	2	1	0,5	0,5	2	1	0	2	0,5
Dragon	0,5	1	2	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
Eau	1	1	0,5	0,5	2	1	1	1	1	1	0,5	1	1	2	2	1	1	1
Électrik	1	1	0,5	2	1	0,5	1	1	1	1	0,5	1	1	1	0	1	1	2
Feu	2	1	0,5	0,5	0,5	1	1	2	2	1	2	1	1	0,5	1	1	1	1
Fée	0,5	2	2	0,5	0,5	1	1	1	1	1	1	0,5	1	1	1	2	1	1
Glace	0,5	1	2	0,5	0,5	1	1	0,5	1	1	2	1	1	1	2	1	1	2
Insecte	0,5	0,5	1	1	0,5	1	0,5	1	1	1	2	0,5	2	1	1	0,5	2	0,5
Normal	0,5	1	1	1	1	1	1	1	1	1	1	0,5	1	2	2	1	1	1
Plante	0,5	1	0,5	2	0,5	1	1	1	0,5	1	0,5	0,5	1	2	2	1	1	0,5
Poison	0	1	1	1	1	1	2	1	1	1	2	0,5	1	0,5	0,5	0,5	1	1
Psy	0,5	2	1	1	1	1	1	1	1	1	1	2	0,5	1	1	1	0	1
Roche	0,5	0,5	1	1	2	1	1	2	1	1	1	1	1	0,5	1	1	1	2
Sol	2	1	1	1	2	2	1	1	0,5	1	0,5	2	1	2	1	1	1	0
Spectre	1	1	1	1	1	1	1	1	1	0	1	1	2	1	1	2	0,5	1
Ténèbres	1	0,5	1	1	1	1	0,5	1	1	1	1	1	2	1	1	2	0,5	1
Vol	0,5	2	1	1	1	0,5	1	1	2	1	2	1	1	0,5	1	1	1	1