PROJET DE FIN D'ANNÉE – SPÉCIALITÉ NSI

Recréation d'un système de combat Pokémon en 1 contre 1

Nom : Gaël Hallopeau

Classe : Terminale Générale – TG6

Année scolaire : 2023–2024

Introduction

Dans le cadre du projet de fin d'année en spécialité NSI, j'ai réalisé un jeu vidéo en Python reproduisant un système de combat Pokémon en 1 contre 1, en tour par tour.

L'objectif était de concevoir une application complète combinant programmation orientée objet, gestion d'une base de données relationnelle et création d'une interface graphique interactive.

Table des matières

1. Objectif du projet

Le but principal était de recréer un combat fidèle à l'univers Pokémon de la première génération (151 Pokémon), incluant la sélection d'un Pokémon, l'affichage des statistiques, le choix d'attaques, un système de dégâts calculé dynamiquement et la gestion des points de vie jusqu'au KO d'un des deux combattants.

2. Technologies utilisées

- Python pour la logique du jeu
- Pygame pour l'interface graphique et la gestion des événements
- SQLite pour la base de données relationnelle
- Krita pour la création des interfaces graphiques

3. Structure de la base de données

La base de données contient plusieurs tables relationnelles :

- Table Pokémon : id, nom, pv, attaque, défense, attaque spéciale, défense spéciale, vitesse
- Table Attaque : id, nom, puissance, type
- Table Pokemon_Attaque : relation entre un Pokémon et ses 4 attaques
- Table Faiblesse : gestion des multiplicateurs selon les types

Table : Pokemon

Filtrer dans n'importe

	id	nom	pv	attaque	defense	attaque_speciale	defense_speciale	vitesse
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Bulbizarre	45	49	49	65	65	45
2	2	Herbizarre	60	62	63	80	80	60
3	3	Florizarre	80	82	83	100	100	80
4	4	Salamèche	39	52	43	60	50	65
5	5	Reptincel	58	64	58	80	65	80
6	6	Dracaufeu	78	84	78	109	85	100
7	7	Carapuce	44	48	65	50	64	43
8	8	Carabaffe	59	63	80	65	80	58
9	9	Tortank	79	83	100	85	105	78
10	10	Chenipan	45	30	35	20	20	45
11	11	Chrysacier	50	20	55	25	25	30
12	12	Papilusion	60	45	50	90	80	70
13	13	Aspicot	40	35	30	20	20	50
14	14	Coconfort	45	25	50	25	25	35
15	15	Dardargnan	65	90	40	45	80	75
16	16	Roucool	40	45	40	35	35	56
17	17	Roucoups	63	60	55	50	50	71
18	18	Roucarnage	83	80	75	70	70	101
19	19	Rattata	30	56	35	25	35	72
20	20	Rattatac	55	81	60	50	70	97

4. Conception orientée objet

Une classe Pokemon a été créée pour récupérer dynamiquement les données depuis la base SQL et gérer le système de combat.

Les principales méthodes permettent :
- d'obtenir les attaques
- de retirer des points de vie
- de vérifier si un Pokémon est encore en vie
- de gérer le calcul des dégâts

```
class Pokemon:
    def __init__(self, db_connection, id): #Class Pokémon qui permet d'utiliser plus facilement les données de la base
        self.id = id

        cursor = db_connection.cursor()


        cursor.execute('SELECT nom, pv, attaque, defense, attaque_speciale, defense_speciale, vitesse FROM Pokemon WHERE id = ?',(self.id,))

        result = cursor.fetchone()

        if result:
            self.nom, self.pv, self.sattaque, self.defense, self.attaque_speciale, self.defense_speciale, self.vitesse = result
            self.faiblesses = []
        else:
            raise ValueError("Le Pokémon n'existe pas")
```

5. Système de combat

Le combat fonctionne en tour par tour :

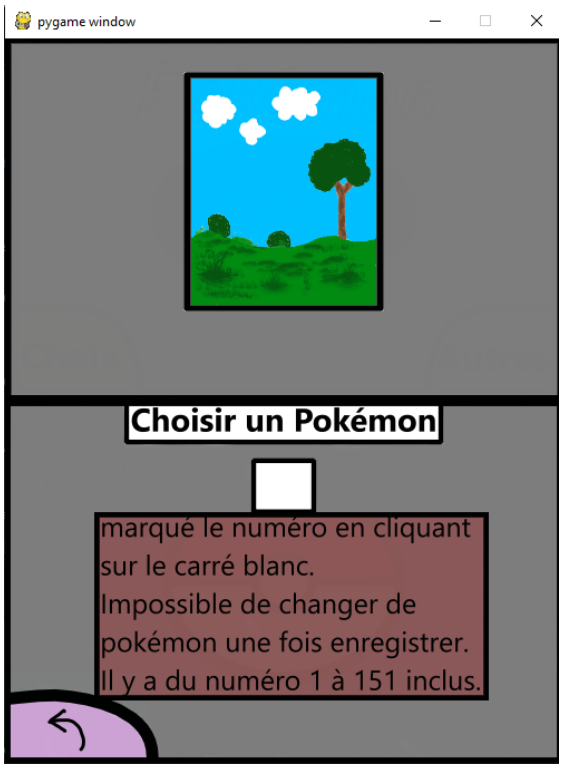
1. Le joueur choisit une attaque.
2. Les dégâts sont calculés selon une formule utilisant les statistiques.
3. Les PV sont mis à jour.
4. L'adversaire attaque aléatoirement.
5. Le combat se termine lorsqu'un Pokémon atteint 0 PV.

```
def combat(self, attaque_choisie_id, adversaire): # à chaque clique sur un des boutons attaque la fon
    cursor = db_connection.cursor()
    cursor.execute('SELECT puissance FROM Attaque WHERE id = ?', (attaque_choisie_id,))
    result = cursor.fetchone()

    if result:
        puissance_attaque = result[0]
        if puissance_attaque == None:
            degats = 0
            self.enlever_pv(degats)
            print(f"{self.nom} a infligé {degats} points de dégâts à {adversaire.nom}")
            adversaire.afficher_pv()
        else:
            degats = (((((50*0.4+2)*self.sattaque*puissance_attaque) / adversaire.defense)/50) + 2 )
            adversaire.enlever_pv(degats)
            print(f"{self.nom} a infligé {degats} points de dégâts à {adversaire.nom}")
            adversaire.afficher_pv()
    else:
        # Gérer le cas où l'attaque choisie n'existe pas dans la base de données
        print("L'attaque choisie n'existe pas.")

    # Vérifier si le Pokémon est KO après l'attaque
    if adversaire.pv <= 0:

        print(f"{adversaire.nom} est KO.")
```

6. Interface graphique

Les interfaces ont été conçues avec Krita puis intégrées dans Pygame.

Plusieurs écrans ont été réalisés :
- Menu principal
- Sélection du Pokémon
- Arène de combat
- Écran de fin



Les boutons sont interactifs grâce à la détection des clics souris.

```
#bouton start
blue = (0,255,255)
hauteur_bouton_start = 245
longueur_bouton_start = 150
rect_bouton_start = pygame.draw.rect(fenetre,blue,[125,85,hauteur_bouton_start,longueur_bouton_start])
```

7. Difficultés rencontrées

Les principales difficultés ont été :
- La prise en main de Pygame
- La gestion des événements multiples
- L'organisation des requêtes SQL
- L'intégration des 151 images

Ces difficultés ont permis de développer ma rigueur et mes compétences techniques.

8. Améliorations possibles

Plusieurs améliorations pourraient être envisagées :
- Ajout d'animations d'attaque
- IA plus avancée
- Barre de vie graphique
- Sélection aléatoire de l'adversaire
- Mode multijoueur

9. Conclusion

Ce projet m'a permis de mobiliser l'ensemble des compétences vues en NSI : programmation orientée objet, gestion de bases de données, création d'interfaces graphiques et logique algorithmique.

Il représente une réalisation complète et structurée illustrant ma capacité à concevoir un projet informatique de bout en bout.