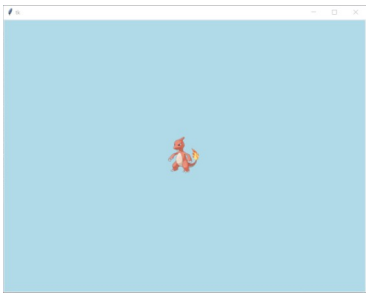
Projet n°1 : Battle of Pokemons

Le contexte

On veut créer une arène de combat virtuelle avec une multitude de Pokemons de plusieurs espèces. Pour cela, on utilisera la programmation orientée objet.

Le projet

1. Le premier Pokemon



Récupérer sur Pealrtrees le dossier du projet. Le fichier *ma_poke_battle.py* contient les instructions de base pour créer la zone de dessin et afficher un premier Pokemon (fichier *Reptincel.png*). Vous devriez voir cela :

Créer ensuite une classe `Pokemon` dans laquelle il n'y aura pour l'instant que le constructeur `__init__` : il déclarera les coordonnées `xLoc` et `yLoc` de l'image, et créera l'image elle-même.

```
class Pokemon:
    # constructeur
    def __init__(self, xLoc, yLoc, image, puissance):
        self.xLoc      = xLoc # entier
        self.yLoc      = yLoc # entier
        self.image     = image # image du Pokemon
        self.puissance = puissance # puissance du Pokemon
        self.ko        = False # état du Pokemon : KO ou pas KO
        self.img       = mon_canvas.create_image(self.xLoc, self.yLoc, image =
img_reptincel)
```

Créer enfin les accesseurs et mutateurs nécessaires. Tester la classe `Pokemon` en créant 2 ou 3 Pokemons et en les affichant.

2. L'arène

On veut créer une classe `Arene`, pour pouvoir instancier un objet arène, contenant des Pokemons, des éléments de décor, et gérer le tout. On définit les premières méthodes de cette classe comme ci-dessous (à compléter) :

```
class Arene:
    # listeDePokemons est de type list
    def __init__(self,nom):
        self.listeDePokemons=[]# Liste de Pokemons

    # pour ajouter un Pokemon de classe Pokemon
    def ajouter(self, pokemon):
        ...

    # pour supprimer un Pokemon KO
    def retirerPokemonKo(self):
        ...

    # pour avoir le nombre de Pokemons de l'arène
    def nbPokemons(self): #
        ...
```

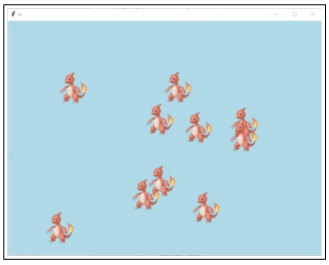
Tester cette implémentation en créant une arène contenant 2 Pokemons.

3. Plein de Reptincel

Au sein du constructeur de la classe `Arene`, ajouter les instructions suivantes au code afin de créer une dizaine de Pokemons (à compléter) :

```
for i in range(10):
    temp_xLoc      = random.randint(100, 700)
    temp_yLoc      = random.randint(50, 550)
    temp_puissance = random.randint(10, 300)
    ...
```

Vous devez voir quelque chose comme ça :

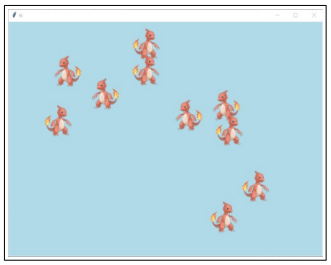


4. Dans les 2 directions

On veut que les Pokemons soient orientés vers la gauche ou vers la droite. Pour cela, à l'aide du fichier *Reptincel.png* et du logiciel Paint 3D, créer une image inversée et l'appeler *Reptincel_inv.png*.
Il faudra aussi créer une nouvelle variable image (`PhotoImage`) et faire un choix aléatoire parmi une liste des variables image à l'aide de l'instruction `random.choices()`.

Conseil : ajouter la variable image au constructeur de la classe `Pokemon`. Le choix aléatoire pourra ainsi se faire dans le constructeur de la classe `Arene`.

Vous devez voir quelque chose comme ça :



5. Des Pokemons différents

De nombreuses images de Pokemons sont disponibles à l'adresse suivante : https://www.pokepedia.fr/

Ainsi l'image vectorielle de Reptincel a été récupérée ici : https://www.pokepedia.fr/Reptincel

Conseil : prendre une petite image, située dans le tableau de la famille. Télécharger les images (format PNG) de trois ou quatre sortes de Pokemons et constituer une arène diversifiée.

6. Les Pokemons bougent !

Dans la classe `Pokemon`, ajouter les deux méthodes suivantes :

```
    def affiche(self):
        """Affiche le Pokemon selon ses nouvelles coordonnées
           Uniquement si celui-ci n'est pas KO """
        # move de l'image dans la zone de dessin
        ...
```

```
    def deplacement(self):
        """Mets à jour les coordonnées du Pokemon
           selon son vecteur (xVel,yVel)
           Uniquement si celui-ci n'est pas KO """
        # si le Pokemon peut sortir de l'arène, sa direction s'inverse
        ...
```

Pour la méthode `affiche`, il est conseillé d'utiliser la méthode `move` de la classe `Canvas`.

Enfin, pour animer le tout, utiliser le code suivant à la fin du programme, en adaptant le nom des variables :

```
def fonction_principale():
    for i in range(mon_arene.nbPokemons()):
        mon_arène.listeDePokemons[i].deplacement()
        mon_arène.listeDePokemons[i].affiche()
    mon_canvas.after(100, fonction_principale)

fonction_principale()
```

7. Let's fight !

On veut que les Pokemons puissent s'affronter au sein de l'arène, jusqu'à ce qu'il n'en reste qu'un. Les régles sont les suivantes :

- Quand deux Pokemons se rencontrent (coordonnées proches), on compare leur puissance.
- Le plus fort élimine le plus faible.
- Il absorbe la puissance du perdant.
- Et ainsi de suite.

Les ajouts à effectuer sont les suivants.

Dans la classe `Pokemon`	Dans la classe `Arene`
La méthode `setKO` doit effacer l'image du Pokemon éliminé, grâce à la méthode `delete` de la classe `Canvas`.	Créer une méthode `combat` qui va : • Tester les coordonnées de tous les Pokemons vivants ; • Comparer leur puissance s'ils sont proches ; • Modifier la puissance du Pokemon vainqueur et l'attribut `ko` du perdant.

Dans la fonction principale, avant de gérer le déplacement et l'affichage des Pokemons, il faut dorénavant :

- Lancer la méthode `combat` s'il reste au moins deux Pokemons ;
- Retirer le·s Pokemon·s KO.

8. C'est l'heure de libérer la créativité !

Maintenant que vous avez une architecture fonctionnelle, vous pouvez améliorer votre arène selon vos envies. Vous pouvez :

- Ajouter des éléments de décor (nouvelle classe) ;
- Afficher la puissance des Pokemons ou leur nombre de kills ;
- Ajouter une image de fond ;
- Ajouter une bande son ou des bruitages ;
- Etc.

Notes de projet