

pokemon.zip sur site (pokedex.csv+dossier photos)

Pokémon Go

Christophe Viroulaud

Première NSI

Pokémon Go

Christophe Viroulaud

Première NSI

Problématique

Informations
disponibles

Interface graphique

Bibliothèque tkinter

Ajouter un composant

Construire par bloc

Remplissage des labels

Gestion des images

Interaction entre
les composants

Liste de choix : Combobox

Choisir un Pokémon : un
événement



FIGURE – Illustration du jeu Pokémon GO

Problématique

Le jeu pour smartphone *Pokemon Go* reprend l'univers du manga éponyme. Il utilise la réalité augmentée pour donner une expérience utilisateur nouvelle.



FIGURE – Illustration du jeu Pokémon GO

Devant le succès du jeu des communautés se créent et tentent d'établir des stratégies pour optimiser leurs résultats.

On se propose de construire un programme pour aider les joueurs dans leurs quêtes.

Devant le succès du jeu des communautés se créent et tentent d'établir des stratégies pour optimiser leurs résultats.

On se propose de construire un programme pour aider les joueurs dans leurs quêtes.

Problématique

Informations
disponibles

Interface graphique

Bibliothèque tkinter

Ajouter un composant

Construire par bloc

Remplissage des labels

Gestion des images

Interaction entre
les composants

Liste de choix : Combobox

Choisir un Pokémon : un événement

Quand on joue à Pokémon Go on trouve des Pokémon sur notre chemin, mais également des œufs. Il faut parcourir une certaine distance pour faire éclore un œuf. Enfin, il est possible de faire évoluer un Pokémon à l'aide de friandises. Un fichier de données (`pokedex.csv`) recense l'ensemble des Pokémon utilisables dans le jeu.

Informations disponibles

Quand on joue à *Pokémon Go* on trouve des Pokémon sur notre chemin, mais également des œufs. Il faut parcourir une certaine distance pour faire éclore un œuf. Enfin, il est possible de faire évoluer un Pokémon à l'aide de friandises. Un fichier de données (`pokedex.csv`) recense l'ensemble des Pokémon utilisables dans le jeu.

- ▶ num : Number of the Pokémon in the official Pokédex
- ▶ name : Pokémon name
- ▶ img : URL to an image of this Pokémon
- ▶ type : Pokémon type
- ▶ height : Pokémon height (m)
- ▶ weight : Pokémon weight (kg)
- ▶ candy : type of candy used to evolve Pokémon or given when transfered
- ▶ candy_count : amount of candies required to evolve
- ▶ egg : Number of kilometers to travel to hatch the egg
- ▶ weakness : Types of Pokémon this Pokémon is weak to
- ▶ next_evolution : Number of evolution of Pokémon

Attributs du fichier

- ▶ num : Number of the Pokémon in the official Pokédex
- ▶ name : Pokémon name
- ▶ img : URL to an image of this Pokémon
- ▶ type : Pokémon type
- ▶ height : Pokémon height (m)
- ▶ weight : Pokémon weight (kg)
- ▶ candy : type of candy used to evolve Pokémon or given when transfered
- ▶ candy_count : amount of candies required to evolve
- ▶ egg : Number of kilometers to travel to hatch the egg
- ▶ weakness : Types of Pokémon this Pokémon is weak to
- ▶ next_evolution : Number of evolution of Pokémon

Activité 1 :

1. Ouvrir le fichier `pokedex.csv` pour observer les données fournies.
2. Créer un programme `pokemon.py`.
3. Importer les données du `pokedex` dans un tableau de dictionnaires `pokedex`. Typier correctement les données de poids et de taille.

Activité 1 :

1. Ouvrir le fichier `pokedex.csv` pour observer les données fournies.
2. Créer un programme `pokemon.py`.
3. Importer les données du `pokedex` dans un tableau de dictionnaires `pokedex`. Typier correctement les données de poids et de taille.

```

1 fichier = open("pokedex.csv")
2 data = csv.DictReader(fichier)
3 pokedex = []
4 for pokemon in data:
5     dico = {}
6     for cle, val in pokemon.items():
7         # validation des données
8         if cle == "height" or cle == "weight":
9             val = float(val)
10
11     dico[cle] = val
12     # ajout du pokemon dans le tableau
13     pokedex.append(dico)
14 fichier.close()

```

Code 1 – Importation des données

Correction

```

1 fichier = open("pokedex.csv")
2 data = csv.DictReader(fichier)
3 pokedex = []
4 for pokemon in data:
5     dico = {}
6     for cle, val in pokemon.items():
7         # validation des données
8         if cle == "height" or cle == "weight":
9             val = float(val)
10
11     dico[cle] = val
12     # ajout du pokemon dans le tableau
13     pokedex.append(dico)
14 fichier.close()

```

Code 1 – Importation des données

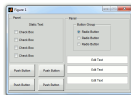


FIGURE – Exemple d'interface graphique

Interface graphique

L'interface permet à l'utilisateur d'interagir avec les données à l'aide de boutons, boîtes de dialogue...

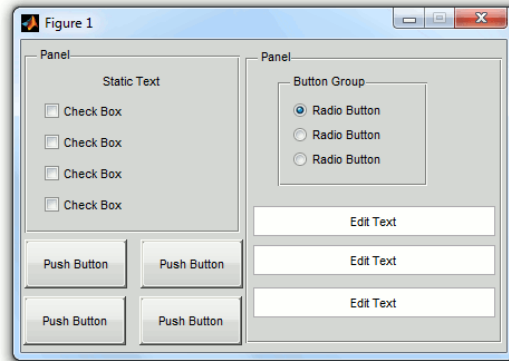


FIGURE – Exemple d'interface graphique


```

1 import tkinter
2 from tkinter import ttk
3
4 #création de la fenêtre
5 fenetre = tkinter.Tk()
6 fenetre.title("Pokemon Go")
7
8 # la construction des composants se placera ici
9
10 # dernière ligne du programme: met à jour les variables
11 fenetre.mainloop()

```

Code 2 – Créer une fenêtre d'interface

Bibliothèque tkinter

La bibliothèque tkinter fournit des *composants* (*widgets*) pour construire une interface graphique simple.

```

1 import tkinter
2 from tkinter import ttk
3
4 #création de la fenêtre
5 fenetre = tkinter.Tk()
6 fenetre.title("Pokemon Go")
7
8 # la construction des composants se placera ici
9
10 # dernière ligne du programme: met à jour les variables
11 fenetre.mainloop()

```

Code 2 – Créer une fenêtre d'interface

- » Créer le composant.
- » Placer le composant dans l'interface.
- » Remplir le composant.

Ajouter un composant

L'ajout d'un composant se déroule en trois étapes :

- Créer le composant.
- Placer le composant dans l'interface.
- Remplir le composant.

```
1 # création
2 etiquette = tkinter.Label(fenetre)
3 # placement
4 etiquette.pack()
5 # remplissage
6 etiquette["text"] = "Bonjour"
```

Code 3 – Placer un *label*
dans la fenêtre

Exemple

```
1 # création
2 etiquette = tkinter.Label(fenetre)
3 # placement
4 etiquette.pack()
5 # remplissage
6 etiquette["text"] = "Bonjour"
```

Code 3 – Placer un *label*
dans la fenêtre

Activité 2 : Construire une interface avec trois labels
nom, poids, taille.

**Activité 2 : Construire une interface avec trois labels
nom, poids, taille.**

```

1 label_nom = tkinter.Label(fenetre)
2 label_nom.pack()
3 label_nom["text"] = "Bulbasaur"
4
5 label_poids = tkinter.Label(fenetre)
6 label_poids.pack()
7 label_poids["text"] = "6.9kg"
8
9 label_taille = tkinter.Label(fenetre)
10 label_taille.pack()
11 label_taille["text"] = "0.71m"

```

Code 4 – Création d'une carte Pokémon

Correction

```

1 label_nom = tkinter.Label(fenetre)
2 label_nom.pack()
3 label_nom["text"] = "Bulbasaur"
4
5 label_poids = tkinter.Label(fenetre)
6 label_poids.pack()
7 label_poids["text"] = "6.9kg"
8
9 label_taille = tkinter.Label(fenetre)
10 label_taille.pack()
11 label_taille["text"] = "0.71m"

```

Code 4 – Création d'une carte Pokémon



Bulbasaur
6.9kg
0.71m

FIGURE – Interface obtenue

Correction

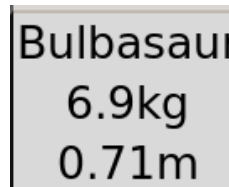


FIGURE – Interface obtenue

```
1 etiquette.grid(column=0, row=0)
```

Code 5 – Place le composant `etiquette`
aux coordonnées (0,0)**Commentaire**On ne peut pas utiliser deux géométries de placement différents (`pack`, `grid`) dans un même bloc.

Placement plus précis

La méthode `pack` place les éléments les uns en dessous des autres. Il existe la méthode `grid` qui utilise un système de coordonnées.

```
1 etiquette.grid(column=0, row=0)
```

Code 5 – Place le composant `etiquette`
aux coordonnées (0,0)

Commentaire

On ne peut pas utiliser deux géométries de placement différents (**`pack`**, **`grid`**) dans un même bloc.

Activité 3 : Présenter les trois labels sous la forme d'une grille.

nom
poids taille

Activité 3 : Présenter les trois labels sous la forme d'une grille.

nom
poids taille


```

1 label_nom = tkinter.Label(fenetre)
2 label_nom.grid(column=0, row=0)
3 label_nom["text"] = "Bulbasaur"
4
5 label_poids = tkinter.Label(fenetre)
6 label_poids.grid(column=0, row=1)
7 label_poids["text"] = "6.9kg"
8
9 label_taille = tkinter.Label(fenetre)
10 label_taille.grid(column=1, row=1)
11 label_taille["text"] = "0.71m"

```

Code 6

Correction

```

1 label_nom = tkinter.Label(fenetre)
2 label_nom.grid(column=0, row=0)
3 label_nom["text"] = "Bulbasaur"
4
5 label_poids = tkinter.Label(fenetre)
6 label_poids.grid(column=0, row=1)
7 label_poids["text"] = "6.9kg"
8
9 label_taille = tkinter.Label(fenetre)
10 label_taille.grid(column=1, row=1)
11 label_taille["text"] = "0.71m"

```

Code 6

Construire par bloc

Pour organiser les composants de manière plus ordonnée, il est préférable de ne pas tous les plaquer directement dans la fenêtre *tkinter* principale. On crée alors des blocs (**Frame**) pour découper notre interface.

```

1 # création d'un bloc
2 bloc_carte = tkinter.Frame(fenetre)
3
4 # on plaque les composants dans le bloc
5 label_nom = tkinter.Label(bloc_carte)
6 label_nom.grid(column=1, row=0)
7
8 label_poids = tkinter.Label(bloc_carte)
9 label_poids.grid(column=0, row=1)
10
11 label_taille = tkinter.Label(bloc_carte)
12 label_taille.grid(column=1, row=1)
13
14 # On plaque le bloc dans la fenêtre
15 bloc_carte.pack()

```

Code 7 – Création d'un bloc

```

1 # création d'un bloc
2 bloc_carte = tkinter.Frame(fenetre)
3
4 # on plaque les composants dans le bloc
5 label_nom = tkinter.Label(bloc_carte)
6 label_nom.grid(column=1, row=0)
7
8 label_poids = tkinter.Label(bloc_carte)
9 label_poids.grid(column=0, row=1)
10
11 label_taille = tkinter.Label(bloc_carte)
12 label_taille.grid(column=1, row=1)
13
14 # On plaque le bloc dans la fenêtre
15 bloc_carte.pack()

```

Code 7 – Création d'un bloc

Remarque

Dans le code 7, la `Frame bloc_carte` a utilisé une géométrie `grid` alors que la fenêtre principale c'est `pack` qui est utilisé. Il n'y a pas d'incompatibilités car il s'agit de deux blocs différents.

Remarque

Dans le code 7, la `Frame bloc_carte` a utilisé une géométrie `grid` alors que la fenêtre principale c'est `pack` qui est utilisé. Il n'y a pas d'incompatibilités car il s'agit de deux blocs différents.

Le tableau `pokedex` est utilisé ici comme une variable globale → pour faciliter l'usage du callback plus tard.

Pour remplir le texte des composants on peut choisir de créer une fonction.

Activité 4 : Écrire la fonction `remplir_carte(num_pok: int) → None` qui remplit les trois labels créés en fonction du numéro de Pokémon choisi dans le Pokédex.

Remplissage des labels

Pour remplir le texte des composants on peut choisir de créer une fonction.

Activité 4 : Écrire la fonction

`remplir_carte(num_pok: int) → None` qui remplit les trois labels créés en fonction du numéro de Pokémon choisi dans le Pokédex.

```

1 def remplir_carte(num_pok: int) -> None:
2     # choix du pokemon dans le pokedex
3     pok = pokedex[num_pok]
4
5     label_nom["text"] = pok["name"]
6     label_poids["text"] = str(pok["weight"])+"kg"
7     label_taille["text"] = str(pok["height"])+"m"

```

Code 8 – Fonction de remplissage

Commentaire

La fonction utilise ici des variables du programme principal. Avec nos connaissances actuelles, c'est la manière la plus simple de procéder.

Correction

```

1 def remplir_carte(num_pok: int) -> None:
2     # choix du pokemon dans le pokedex
3     pok = pokedex[num_pok]
4
5     label_nom["text"] = pok["name"]
6     label_poids["text"] = str(pok["weight"])+"kg"
7     label_taille["text"] = str(pok["height"])+"m"

```

Code 8 – Fonction de remplissage

Commentaire

La fonction utilise ici des variables du programme principal. Avec nos connaissances actuelles, c'est la manière la plus simple de procéder.

```
1 remplir_carte(12)
```

Code 9 – Remplissage des labels

Correction

Dans le programme principal, l'appel 9 après la création des labels, affiche les informations du Pokémon 12.

```
1 remplir_carte(12)
```

Code 9 – Remplissage des labels

L'affichage d'une image demande un peu plus de travail. Il faut créer un objet `PhotoImage` qui sera ensuite affiché dans un composant `label_photo`. Commençons par créer le composant.

```
1 label_photo = tkinter.Label(bloc_carte)
2 label_photo.grid(column=1, row=0)
```

Code 10 – `label_photo` est placé à droite de `label_nom`

Gestion des images

L'affichage d'une image demande un peu plus de travail. Il faut créer un objet `PhotoImage` qui sera ensuite affiché dans un composant `label_photo`. Commençons par créer le composant.

```
1 label_photo = tkinter.Label(bloc_carte)
2 label_photo.grid(column=1, row=0)
```

Code 10 – `label_photo` est placé à droite de `label_nom`

Pokémon Go

- Interface graphique
- Gestion des images

Il faut ensuite modifier la fonction `remplir_carte` pour afficher la photo.

```
1 def remplir_carte(num_pok: int) -> None:
2     global photo # garde une référence de l'image
3
4     pok = pokedex[num_pok]
5
6     # affichage de l'image
7     photo = tkinter.PhotoImage(file=pok["img"])
8     label_photo["image"] = photo
9
10    label_nom["text"] = pok["name"]
11    label_poids["text"] = str(pok["weight"])+"kg"
12    label_taille["text"] = str(pok["height"])+"m"
```

Commentaire

La ligne 2 évite que le *garbage collector* efface l'image à la sortie de la fonction. Cette notion est hors programme.

Il faut ensuite modifier la fonction `remplir_carte` pour afficher la photo.

```
1 def remplir_carte(num_pok: int) -> None:
2     global photo # garde une référence de l'image
3
4     pok = pokedex[num_pok]
5
6     # affichage de l'image
7     photo = tkinter.PhotoImage(file=pok["img"])
8     label_photo["image"] = photo
9
10    label_nom["text"] = pok["name"]
11    label_poids["text"] = str(pok["weight"])+"kg"
12    label_taille["text"] = str(pok["height"])+"m"
```

Commentaire

La ligne 2 évite que le *garbage collector* efface l'image à la sortie de la fonction. Cette notion est hors programme.

Notre programme n'est pour l'instant que peut utile : il faut changer *en dur* (c'est à dire dans le programme) le numéro du Pokémon à afficher. La prochaine étape consistera à créer une liste de choix dans l'interface graphique pour changer dynamiquement la carte à afficher.

Interaction entre les composants

Notre programme n'est pour l'instant que peut utile : il faut changer *en dur* (c'est à dire dans le programme) le numéro du Pokémon à afficher. La prochaine étape consistera à créer une liste de choix dans l'interface graphique pour changer dynamiquement la carte à afficher.

Pour afficher les noms des Pokémons dans une liste de choix, il faut d'abord construire un tableau contenant tous ces noms.

Activité 5 : Construire par compréhension le tableau `noms_affiches` des noms de tous les Pokémons du Pokédex.

Combobox

Pour afficher les noms des Pokémons dans une liste de choix, il faut d'abord construire un tableau contenant tous ces noms.

Activité 5 : Construire par compréhension le tableau `noms_affiches` des noms de tous les Pokémons du Pokédex.

```
noms_affiches = [pokemon["name"] for pokemon in pokedex]
```

Code 11 – Tableau des noms

Correction

```
noms_affiches = [pokemon["name"] for pokemon in pokedex]
```

Code 11 – Tableau des noms

```

1 # création du composant
2 combo_pok = ttk.Combobox(fenetre, values=
  noms_affiches)
3 # valeur par défaut
4 combo_pok.current(0)
5 # placement
6 combo_pok.pack()

```

Code 12 – Création du composant

Combobox

Le composant **Combobox** crée une liste de choix.

```

1 # création du composant
2 combo_pok = ttk.Combobox(fenetre, values=
  noms_affiches)
3 # valeur par défaut
4 combo_pok.current(0)
5 # placement
6 combo_pok.pack()

```

Code 12 – Création du composant

Notre interface *tkinter* attend en permanence une action de l'utilisateur. Quand ce dernier change le Pokémon sélectionné dans la **Combobox**, un événement se produit. L'interface écoute les événements.

```
1 combo_pok.bind("<<ComboboxSelected>>", callback_combo)
```

Code 13 – Écouteur sur la **Combobox**

Il faut maintenant qu'elle sache comment réagir.

Choisir un Pokémon : un événement

Notre interface *tkinter* attend en permanence une action de l'utilisateur. Quand ce dernier change le Pokémon sélectionné dans la **Combobox**, un *événement* se produit. L'interface *écoute* les événements.

```
1 combo_pok.bind("<<ComboboxSelected>>", callback_combo)
```

Code 13 – Écouteur sur la **Combobox**

Il faut maintenant qu'elle sache comment réagir.

Quand l'utilisateur change la sélection dans la liste de choix, la fonction `callback_combo` est appelée. C'est une **fonction de rappel (*callback*)**. Un unique paramètre (noté ici `event`) est passé automatiquement à la fonction. Ce paramètre contient des informations sur le choix effectué.

```
1 def callback_combo(event):
2     # num contient l'indice de la ligne sélectionnée
3     # dans la liste
4     num = event.widget.current()
```

Code 14 – fonction de callback

Fonction de rappel (*callback*)

Quand l'utilisateur change la sélection dans la liste de choix, la fonction `callback_combo` est appelée. C'est une **fonction de rappel (*callback*)**. Un unique paramètre (noté ici `event`) est passé automatiquement à la fonction. Ce paramètre contient des informations sur le choix effectué.

```
1 def callback_combo(event):
2     # num contient l'indice de la ligne sélectionnée
3     # dans la liste
4     num = event.widget.current()
```

Code 14 – fonction de callback

Nous pouvons alors utiliser cette information pour mettre à jour la carte Pokémon.

```
1 def callback_combo(event):
2     """
3     fonction de rappel quand on change de pokemon
4     dans la combobox
5     """
6     num = event.widget.current()
7     # mise à jour de la carte
8     remplir_carte(num)
```

Nous pouvons alors utiliser cette information pour mettre à jour la carte Pokémon.

```
1 def callback_combo(event):
2     """
3     fonction de rappel quand on change de pokemon
4     dans la combobox
5     """
6     num = event.widget.current()
7     # mise à jour de la carte
8     remplir_carte(num)
```


Code complet

Le code complet est récupérable à l'adresse suivante :

[Code complet](#)