Objectif : Différencier et utiliser plusieurs paradigmes de programmation.

1 Problématique : Fortnite

Chaque combattant de Fortnite est extrêmement personnalisable. C'est une des caractéristiques qui fait le succès du jeu.
Il est ainsi possible de choisir les éléments : outfit, glider, contrail, back bling, emote, pickaxe, toy.

			
outfit	glider	emote	pickaxe

TABLE 1 – Exemples d'éléments

Il faut ajouter à cela les objets récupérés en cours de partie et gérer l'évolution de l'énergie et du bouclier du personnage.

> Quel concept de programmation mettre en place pour manipuler les combattants et les faire évoluer dans une partie ?

2 Un concept déjà utilisé : paradigme impératif

2.1 Stocker les informations

Différentes structure de données abordées en Première permettent de contenir les informations du combattant.

Activité 1 :

1. Proposer un type (construit) de données et construire la structure permettant de stocker les éléments qui composent un combattant.
2. Ajouter une structure permettant de stocker et gérer les objets du sac à dos.
3. Ajouter enfin deux variables exprimant l'énergie et le bouclier. Ces variables seront initialisées à *100*.

2.2 Modifier les informations

Lors d'une partie le joueur peut ramasser des objets pour les mettre dans son sac. Il peut également subir des dégâts lors d'une attaque.

Activité 2 :

1. Implémenter une fonction *subir_attaque* qui prendra un paramètre *degat* de type *integer* et qui modifiera les caractéristiques *shield* et *energy* du combattant.
2. Implémenter une fonction *ramasser_objet* qui ajoute un objet dans le sac s'il n'est pas plein (trois objets maximum). Cette fonction prendra un paramètre *objet* de type *string*.
3. Dans Fortnite chaque objet a également plusieurs propriétés. Construire un type construit de données qui stocke les caractéristiques des armes présentées.

Christophe Viroulaud

3 Une nouvelle approche : paradigme objet

3.1 Décrire une classe

3.1.1 Définition

Une *classe* définit une nouvelle structure de données. Elle regroupe plusieurs composantes de natures différentes. Dans notre cas c'est le combattant. Chaque langage implémentant la programmation orienté objet défini une syntaxe propre.

```
1 class Combattant:
```

La classe peut être vue comme le squelette générique du combattant.

3.1.2 Ajouter des attributs

La fonction *__init__* est appelée automatiquement quand nous créons un combattant. Chaque caractéristique est représentée par une variable propre à la classe : *un attribut*.

```
1 class Combattant:
2       """Crée un combattant avec ses caractéristiques propres"""
3       def __init__(self,tenue,parachute,trainee):
4             self.outfit=tenue
5             self.glider=parachute
6             self.contrail=trainee
```

3.2 Créer un objet

3.2.1 Créer un combattant

La classe est la structure (le squelette) d'un combattant. Pour l'instant aucun joueur n'a été construit. On crée une *instance* de la classe *Combattant* et on la stocke dans la variable *joueur1*.

```
1 joueur1=Combattant("cuddle-team-leader","hot-rod","flames")
```

Activité 3 :

1. Compléter la classe *Combattant* en ajoutant toutes les caractéristiques.
2. Créer un nouvel attribut permettant de stocker le contenu du sac à dos.

3.2.2 Manipuler les attributs

Nous pouvons consulter les valeurs d'un attribut mais aussi les modifier.

```
1 >>> joueur1.outfit
2 cuddle-team-leader
3 >>> joueur1.glider="pterodactyl"
```

3.3 Manipuler les données

La manipulation de l'objet passe de préférence par une interface constituée de fonctions définies dans la classe : *les méthodes*.

Christophe Viroulaud

```python
class Combattant:
	"""Crée un combattant avec ses caractéristiques propres"""
	def __init__(self,tenue,parachute,trainee):
		self.outfit=tenue
		self.glider=parachute
		self.contrail=trainee

	def get_outfit(self):
		return self.outfit

	def set_outfit(self,nouvelle):
		self.outfit=nouvelle
```

Activité 4 :

1. Ajouter les attributs *energy*, *shield* et *backpack*.
2. Adapter les fonctions *subir_attaque* et *ramasser_objet* pour en faire des méthodes de la classe *Combattant*.

3.4 Le monde est objet

Activité 5 :

1. Créer un objet *weapon* possédant les attributs présentés précédemment.
2. Ajouter une méthode *tirer* qui vide une balle du magasin. Cette méthode renverra le texte *Chargeur vide !* quand toutes les balles auront été tirées.
3. Implémenter une méthode *recharger* qui remplit le magasin.

Christophe Viroulaud