BOUCLES ET TABLEAUX

BOUCLES

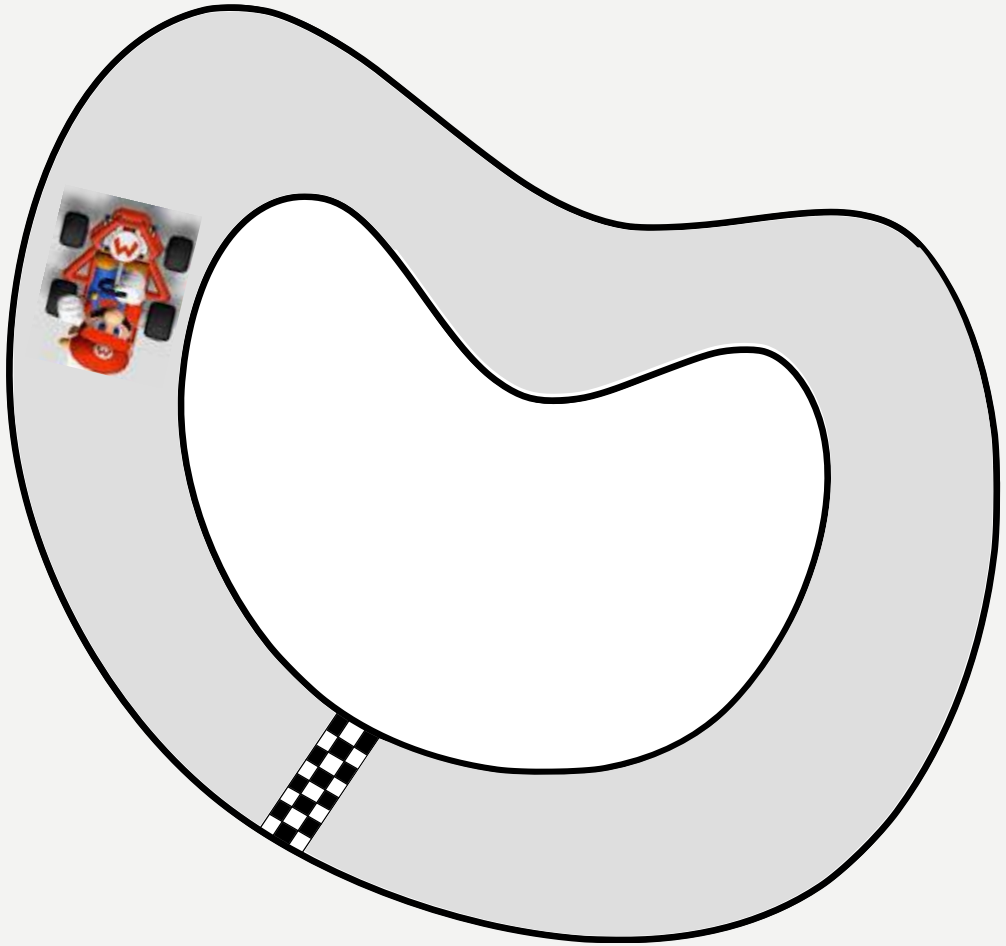
Quezaco

On sait maintenant ce qu'est une condition. Une boucle n'est ni plus ni moins une répétition des instructions contenues dans la condition tant que cette dernière n'est pas fausse.
On peut énoncer une boucle de la façon suivante:
« Tant que tu n'as pas la tête qui tourne, fais un tour sur toi-même »
Ici la condition est « Est-ce que ta tête ne tourne pas? »
L'instruction est « Fais un tour sur toi-même » et elle est répétée tant que la condition n'est pas fausse, c'est-à-dire quand la réponse à « Est-ce que ta tête ne tourne pas? » est **non**, soit « ta tête tourne ».

BOUCLES

Autre exemple

Tant que la voiture n'a pas fait 10 tours, alors elle ne s'arrête pas.



BOUCLES

GDScript : while

Dans Godot, pour réaliser une boucle, il existe deux façons:

- while
- for

Le mot clé « while » fonctionne exactement de la même façon que le mot clé « if », c'est-à-dire:

while [Conditions] :
[Instructions]

Le bloc [Instructions] sera exécuté jusqu'à ce que le bloc [Conditions] soit faux.

GODOT

```
4  func _ready():
5      var numero_tour = 0
6      var nombre_de_tours = 10
7      while numero_tour < nombre_de_tours:
8          print(numero_tour)
9          numero_tour += 1
```

```
Output:

0
1
2
3
4
5
6
7
8
9
```

BOUCLES

Examples

```
6  func _ready():
7      var a = 50.0
8      var b = 10.0
9      while a != -50.0 || a + b == 10.0:
10         a -= 1
```

```
6  func _process():
7      self.translate(Vector2(0.0,1.0))
```

```
4  func _ready():
5      var deplacement = Vector2(1,0)
6      var position = Vector2(0,0)
7      while position.x < 50.0 :
8          position += deplacement
```

```
6  func _ready():
7      var a = 10.0
8      var b = 1.0
9      while divide(a, b) != 1.0:
10         b += 1
```

```
4  func _ready():
5      while true:
6          print("hello")
```

```
6  func _ready():
7      var i = 0
8      while i < 10:
9          while i > 8:
10             if i == 5.0:
11                 print(i)
12             i -= 1
13         if i == 9.0:
14             print(i)
15         i += 1
```

BOUCLES

Exercices

- Créer une variable « a » avec comme valeur 0
- **Tant que** « a » est inférieure à 10 **alors** incrémenter a de 2

- Créer une variable « a » avec comme valeur 10
- **Tant que** « a » est supérieure à 10 **alors** décrémenter a

- Créer une variable « pos » de type Vector2 avec une valeur initiale Vector2(0,0)
- **Tant que** la variable « pos » n'est pas égale à Vector2(100,200) **alors** ajouter Vector2(1,2) à « pos »
- Afficher « Arrivé » en sortie de boucle

- Créer une variable « pos » de type Vector2 avec une valeur initiale Vector2(0,0)
- **Boucle 1 : Tant que** la composante x de « pos » n'est pas égale 100 **alors**
  - **Boucle2** : **Tant que** composante y de « pos » n'est pas égale 100 **alors** ajouter à la composante y de de « pos » la valeur 1.
  - **Dans Boucle 1:** Ajouter à la composante x de de « pos » la valeur 1.
  - **Dans Boucle 1** : Afficher le vecteur « pos »

TABLEAUX

Quezaco

Un tableau est une suite d'éléments rangée dans une seule variable. Par exemple un tableau d'entiers est un ensemble d'entiers.
Un tableau a donc une taille qui correspond aux nombre d'éléments à l'intérieur.

Ce tableau contient 5 entiers →

1	5	3	8	2

Ce tableau contient 3 Vector2 →

Vector2(0,10)	Vector2(4,2)	Vector2(-7,8)

Ce tableau contient 3 Node2D →

Node2D@1	Node2D@2	Node2D@3

TABLEAUX

GDScript

Pour créer un tableau en GDScript, il faut déclarer une variable comme d'habitude, mais cette fois-ci la valeur qui est donné à la variable est un type tableau. Une valeur de type tableau s'écrit de la façon suivante:
[valeur1, valeur2, valeur3, …]

GODOT

```
8   ⇥   var nombres = [1,5,3,8,2]
9   ⇥   var vectors = [Vector2(0,10), Vector2(4,2), Vector2(-7,8)]
10  ⇥   var nodes = [self, get_parent(), get_child(0)]
11  ⇥   var values = [1, 5, Vector2(4,2), get_parent()]
```

TABLEAUX

Accès à un élément

Un tableau est indexé de 0 à sa taille -1.
Le premier élément du tableau est donc à la position 0.
Le dernier élément du tableau est donc à la position [taille du tableau] – 1
L'élément n°3 du tableau alors à la position 2 (0, 1, 2)
Pour accéder à un élément du tableau:

nom_variable[i]

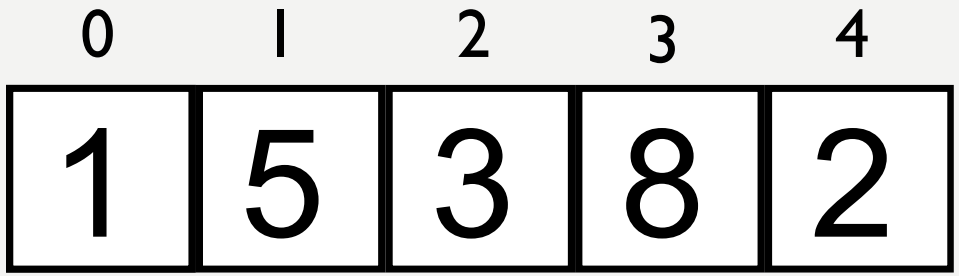
Avec i un nombre entier

GODOT

```
6  func _ready():
7
8      var nombres = [1,5,3,8,2]
9      print(nombres[0] , " " , nombres[1] , " " , nombres[4])
```

Output:

```
1 5 2
```



TABLEAUX

Définir la valeur d'un élément

De la même façon qu'on accède à un élément du tableau, on peut y définir sa valeur:

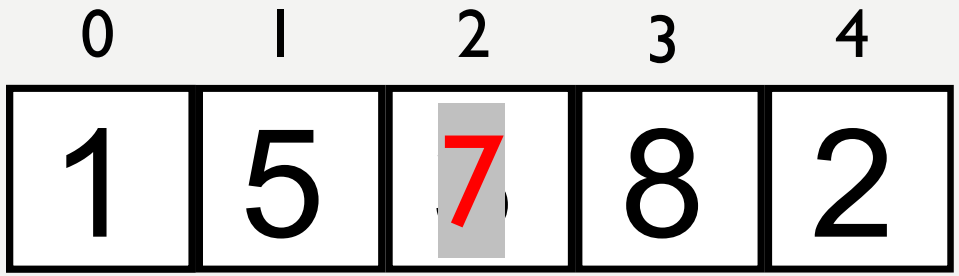
nom_variable[i] = valeur

GODOT

```
6  func _ready():
7
8      var nombres = [1,5,3,8,2]
9      print(nombres[2])
10
11     nombres[2] = 7
12     print(nombres[2])
```

Output:

```
3
7
```



TABLEAUX

D’autres opérations

- On peut ajouter ou enlever d’autres éléments d’un tableau
- On peut concaténer deux tableaux
- On peut trier des tableaux
- On peut mélanger des tableaux
- …

TABLEAUX

Examples

```
6  func _ready():
7      var nombres = [1,5,3,7,6,8,2]
8      print(nombres[5])
```

```
6  func _ready():
7      var nombres = [1,5,3,7,6,8,2]
8
9      print(nombres[-1])
10
11     print(nombres[7])
```

```
6  func _ready():
7      var nombres = [1,5,3,7,6,8,2]
8
9      if nombres[1] == 6:
10         nombres[4] = 3
11     elif nombres[6] == 2:
12         nombres[2] += 1
13     else:
14         nombres[4] = 4
```

```
6  func _ready():
7      var vectors = [Vector2(5,8), Vector2(3,7), Vector2(1,2), Vector2(0,0)]
8      var i = 0
9      var taille_tableau = vectors.size()
10     while i < taille_tableau:
11         print(vectors[i])
12         i+=1
```

TABLEAUX

Exercices

- Déclarer une variable « tab » et lui affecter un tableau de 3 entiers : 4, 3, 2
- Afficher le deuxième élément de ce tableau

- Déclarer une variable « tab » et lui affecter un tableau contenant un Vector2(0,1), un entier 4 et self
- Afficher chacun de ces éléments dans une boucle

- Créer une fonction « add_value_to_array » qui prend en entrée un tableau et un entier
- Cette fonction doit afficher chaque élément du tableau ajouté à la valeur
- Dans la fonction _ready, déclarer une variable « tab » et lui affecter un tableau de 3 entiers : 4, 3, 2
- Appeler la fonction « add_value_to_array » en lui envoyant « tab » et 4

- Déclarer une variable « positions » et lui affecter un tableau de Vector2 : Vector2(0,1), Vector2(0,8), Vector2(3, 4)
- Déclarer une variable « deplacements » et lui affecter un tableau de Vector2 : Vector2(10, 9), Vector2(10, 2), Vector2(7, 6)
- Ajouter un à un les Vector2 de « deplacements » aux Vector2 de « positions » (utiliser une boucle)
- Afficher « positions »

BOUCLES

GDScript : for

La boucle for est utilisée spécialement lorsque l'on veut parcourir un tableau.
Par exemple, on peut parcourir un tableau d'entiers directement en faisant:

***for** valeur **in** tableau:*
print(valeur)

« valeur » est une variable utilisée seulement dans la boucle qui va, à chaque tour de boucle, prendre la valeur d'un élément du tableau en partant de l'élément 0.
« tableau » est une variable contenant un tableau.
Le mot clé « in » permet une lecture intuitive de la boucle: **Pour chaque** « valeur » **dans** « tableau » **alors**:

GODOT

```
5  func _ready():
6      var tours = [0,1,2,3,4,5,6,7,8,9]
7      for numero_tour in tours:
8          print("tour n°", numero_tour)
```

```
Output:
tour n°0
tour n°1
tour n°2
tour n°3
tour n°4
tour n°5
tour n°6
tour n°7
tour n°8
tour n°9
```

A noter que la boucle for a son équivalent avec la boucle while

```
10     var i = 0
11     while i < tours.size():
12         print("tour n°", tours[i])
```

BOUCLES

GDScript : range

La boucle for est utilisée souvent lorsque l'on veut exécuter un certains nombre de fois un bloc d'instructions.

Déplaces-toi 10 fois à droite

On peut utiliser un boucle *while* avec la condition *i < 10* avec *i = 0* au départ ou alors utiliser la boucle *for* avec une fonction très utile appelée « *range* ».

range(initial, final)

C'est une fonction qui prend en entrée le nombre de départ, le nombre d'arrivée et qui produit en sortie un tableau de tous les nombres entre la borne de départ incluse et la borne d'arrivée exclue.

GODOT

```
5 func _ready():
6     for i in range(0,10):
7         print(i)
```

```
Output:

0
1
2
3
4
5
6
7
8
9
```

A noter que range est une fonction qui fournit un tableau de valeurs à partir des paramètres en entrée.

A noter également que si l'on donne un seul paramètre à range, la valeur de départ sera 0 par défaut.

BOUCLES

Examples

```
5 func _ready():
6     for a in range(5, 20):
7         print(a)
```

```
5 func _ready():
6     var tab = [Vector2(5,3), Vector2(3,1), Vector2(0,0)]
7     for a in tab:
8         print(a)
```

```
5 func _ready():
6     for i in range(0,10):
7         for j in range(0,10):
8             print(i,j)
```

```
5 func _ready():
6     for i in range(0,10):
7         if i == 4:
8             i = 7
9         print(i)
```

```
5 func _ready():
6     var tab = [5, 0, 2, 3]
7     var maximum = 0
8     for i in tab:
9         if maximum < i:
10            maximum = i
11    print(maximum)
```

BOUCLES

Exercices

- Créer un tableau de 3 Vector2 : Vector2(0,0), Vector2(10,6), Vector2(6,-5)
- Parcourir le tableau et affiché chacun des éléments

- Ecrire l'équivalent avec une boucle while du code suivant:

```
for i in range(0,4):
    print(i)
```

- Créer une fonction « minimum_value » qui prend en entrée un tableau
- Retourner la valeur minimale du tableau
- Dans la fonction _ready(), créer un tableau de 5 entier : 1, 9, 6, 5, 3
- Appeler la fonction « minimum_value » avec ce tableau
- Afficher le résultat