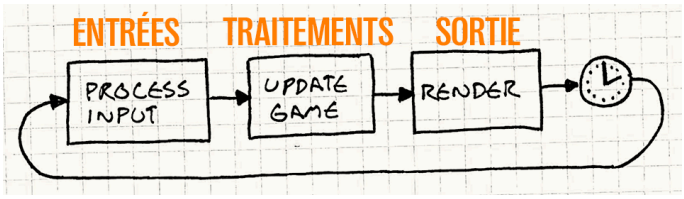
Introduction à la boucle de jeu (game loop)

La boucle de jeu, ou **game loop** en anglais, est l'un des concepts les plus fondamentaux dans le développement de jeux vidéo. Que vous utilisiez un moteur de jeu avancé ou que vous programmiez un jeu en 2D avec une bibliothèque comme PyGame, la boucle de jeu est essentielle pour réagir aux actions du joueur, gérer la logique du jeu et mettre à jour les éléments graphiques.

Qu'est-ce qu'une boucle de jeu ?

Une boucle de jeu est une structure répétitive qui fait tourner le jeu en continu jusqu'à ce que le joueur quitte. Elle permet de :

1. **Lire les entrées du joueur** (clavier, souris...).
2. **Mettre à jour l'état du jeu** (déplacements, collisions...).
3. **Rafraîchir l'affichage** pour montrer les changements à l'écran.



Structure d'une boucle de jeu

1. **Initialisation** (création de la fenêtre, chargement des ressources...)
2. **Boucle principale** :
   - Gérer les entrées
   - Mettre à jour le jeu
   - Redessiner l'écran
3. **Sortie propre** (fermeture du jeu)

Exemple simple en PyGame

Ton programme ci-dessous fonctionnera si ton écran est noir avec une petite boîte carrée affichée.

```python
import pygame
import sys

# Initialisation
pygame.init()

# Création de la fenêtre de jeu
screen = pygame.display.set_mode((800, 600))
clock = pygame.time.Clock()

# position initiale du joueur
player_x = 100
player_y = 100

```

```python
player_color = (255, 0, 255)  # Rose

# Boucle de jeu
running = True
while running:

    # Permet de quitter le programme proprement
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Mise à jour de l'écran
    # Dessiner le joueur (rectangle)
    pygame.draw.rect(screen, player_color, (player_x, player_y, 50, 50))
    pygame.display.flip()

    # Limiter la vitesse d'exécution (60 FPS)
    clock.tick(60)

pygame.quit()
sys.exit()
```

🧱 `pygame.draw.rect()`

La fonction `pygame.draw.rect()` permet de **dessiner un rectangle** sur une surface (par exemple l'écran de jeu). C'est très utile pour représenter un joueur, un mur, un bouton ou tout autre élément graphique simple.

📝 Syntaxe

```python
pygame.draw.rect(surface, couleur, rectangle)
```

📌 Paramètres

Paramètre	Type	Description
`surface`	`pygame.Surface`	L'endroit où le rectangle doit être dessiné (souvent l'écran principal, par exemple `screen`).
`couleur`	`tuple`	La couleur du rectangle, sous forme `(R, G, B)`, avec des valeurs entre 0 et 255.
`rectangle`	`tuple` ou `pygame.Rect`	La position et la taille du rectangle : `(x, y, largeur, hauteur)`. `x, y` = coin supérieur gauche. `largeur`, `hauteur` = taille en pixels.

✅ Exemple

```python
pygame.draw.rect(screen, player_color, (player_x, player_y, 50, 50))
```

- **`screen`** : l'écran sur lequel on dessine.
- **`player_color`** : une couleur, par exemple `(255, 0, 0)` pour rouge.
- **`(player_x, player_y, 50, 50)`** : on dessine un rectangle de 50×50 pixels à la position `(player_x, player_y)`.

🖼️ Résultat

Cela dessine un carré rouge (ou de la couleur choisie) à l'écran, à la position indiquée, de taille 50x50 pixels. Il est visible **à la prochaine mise à jour de l'affichage** avec `pygame.display.flip()`.

💡 Bonnes pratiques

- **Garde la boucle claire** : sépare les entrées, la logique et l'affichage (**entrée**, **traitements**, **sorties**).
- **Utilise un framerate constant** (clock.tick(60)) pour éviter les variations de vitesse.
- **Ne surcharge pas la boucle** : place les calculs lourds dans des fonctions à part.

Exercices à essayer

- Change les coordonnées (`player_x` et `player_y`) du joueur pour changer sa position à l'écran.
    - Identifie l'impact de ces changement sur ton personnage
- Affiche plusieurs rectangles avec des tailles et des couleurs différentes
- Grâce à une boucle `for` affiche 10 rectangles à des positions aléatoires (au hasard)
- Pour les ninjas: affiche 10 rectangles avec des positions, tailles et couleurs aléatoires

Cette boucle de jeu est la base pour créer un jeu interactif. Ajoute des éléments et expérimente !