Gérer les entrées au clavier

Dans un jeu vidéo, gérer les entrées clavier est essentiel pour contrôler les déplacements du personnage.

Gestion des entrées clavier dans PyGame

La notion de **vitesse** est également importante : elle détermine combien de pixels le personnage se déplacera à chaque action. Une vitesse plus élevée entraînera des déplacements plus rapides et plus grands à l'écran, tandis qu'une vitesse plus faible permettra des mouvements plus précis et contrôlés.

Déplacement avec les flèches

Utilise les constantes de PyGame pour détecter les touches fléchées :

```
keys = pygame.key.get_pressed() # Récupère la liste de toutes les touches qui sont appuyées pa

if keys[pygame.K_LEFT]: # gauche
    player_x -= vitesse
if keys[pygame.K_RIGHT]: # droite
    player_x += vitesse
if keys[pygame.K_UP]: # haut
    player_y -= vitesse
if keys[pygame.K_DOWN]: # bas
    player_y += vitesse
```

- `K_LEFT` : gauche
- `K_RIGHT` : droite
- `K_UP` : haut
- `K_DOWN` : bas

Dans la boucle de jeu

Gérer les entrées utilisateurs est généralement la première chose que tu fais dans ta boucle de jeu. Voici la boucle de jeu originale des exercices précédents adaptées avec la gestion des touches:

```
import pygame
import sys

# Initialisation
pygame.init()

# Création de la fenêtre de jeu
screen = pygame.display.set_mode((800, 600))
player = pygame.image.load('player.png')

clock = pygame.time.Clock()

# position initiale du joueur
```

```
player_x = 100
player_y = 100
vitesse = 5

# Boucle de jeu
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    keys = pygame.key.get_pressed()

    if keys[pygame.K_LEFT] or keys[pygame.K_q]:
        player_x -= vitesse
    if keys[pygame.K_RIGHT] or keys[pygame.K_d]:
        player_x += vitesse
    if keys[pygame.K_UP] or keys[pygame.K_z]:
        player_y -= vitesse
    if keys[pygame.K_DOWN] or keys[pygame.K_s]:
        player_y += vitesse

    # Mise à jour de l'écran
    screen.blit(player, (player_x, player_y))
    pygame.display.flip()

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

pygame.quit()
sys.exit()

```

Pourquoi je peins mon écran ?

Il manque un petit quelque chose dans la gestion de l'affichage de notre jeu. A chaque fois qu'on dessine l'image à une nouvelle position, Pygame se contente de dessiner l'image une nouvelle fois sur l'écran. Et en fonction de la vitesse de déplacement, celle ci vient se dessiner au dessus des autres images dessinnées précédemment. Ce qui donne un petit peu l'effet pinceau.

Pour corriger ce problème: à chaque fois, avant d'afficher ton personnage, il faut entièrement recolorier/effacer le fond de l'écran. Dans un premier temps, on se contente de le colorier en noir et on apprendra plus tard comment changer la couleur et afficher des images de fond.

```
ecran.fill((0, 0, 0))  # Effacer l'écran en noir
```

A ajouter juste avant la ligne `screen.blit(player, (player_x, player_y))`.

Touches gamer habituelles

Pour une expérience de jeu plus complète, on utilise souvent les touches suivantes :

```python
if keys[pygame.K_z]:  # Monter ou avancer
    player_y -= vitesse
if keys[pygame.K_q]:  # Gauche
    player_x -= vitesse
if keys[pygame.K_s]:  # Bas
    player_y += vitesse
if keys[pygame.K_d]:  # Droite
    player_x += vitesse
```

- **ZQSD** : déplacements haut, gauche, bas, droite

Bonnes pratiques

- Regroupe la gestion des entrées dans une seule section de la boucle de jeu.
- Ajuste la variable `vitesse` pour contrôler la rapidité des mouvements.
- Sépare clairement le traitement des entrées du reste du code pour une meilleure lisibilité.

Avec ces bases, tu peux efficacement contrôler ton personnage dans PyGame !