Labyrinthe v0.2 - Personnage

Dans cet article, nous allons faire progresser notre projet de labyrinthe en ajoutant un personnage que le joueur pourra déplacer avec les touches du clavier. Nous allons d'abord charger une image fixe pour le personnage, la dessiner sur l'écran, puis ajouter la gestion des déplacements en fonction des entrées clavier.

Dans cette version, nous allons apprendre à:

1. Charger une image représentant le personnage.
2. Dessiner ce personnage sur l'écran à une position initiale.
3. Gérer les déplacements du personnage avec les touches fléchées.

Étape 1 : Charger l'image du personnage

Tout d'abord, il faut trouver ou créer une image pour représenter votre personnage. Par exemple, vous pouvez utiliser une image au format PNG d'un personnage de jeu vidéo. Assurez-vous que l'image est de taille similaire aux tuiles de votre labyrinthe (par exemple, 32x32 pixels).

Utilisez la fonction `pygame.image.load()` pour charger l'image du personnage, puis stockez-la dans une variable pour l'utiliser dans le jeu. Voici comment charger une image de personnage à partir d'un fichier :

```
# Charger l'image du personnage
image_personnage = pygame.image.load('personnage.png')
```

Veillez à ce que le fichier `personnage.png` soit dans le même dossier que votre script Python ou fournissez le chemin correct vers l'image.

Étape 2 : Dessiner le personnage sur l'écran

Maintenant que l'image du personnage est chargée, nous devons la dessiner sur l'écran. Tout comme nous avons dessiné les tuiles du labyrinthe (les murs et le sol), nous allons utiliser la fonction `blit()` de PyGame pour afficher l'image du personnage à une position donnée.

Pour cela, nous devons choisir une position de départ pour le personnage. Par exemple, nous pouvons commencer au centre du labyrinthe, ou à une position spécifique comme `(1, 1)` (première colonne, première ligne).

```
# Position initiale du personnage (en pixels)
x_personnage = 32  # 2ème colonne (1 * taille de tuile)
y_personnage = 32  # 2ème ligne (1 * taille de tuile)
```

Pour dessiner l'image du personnage sur l'écran à cette position :

```
# Dessiner le personnage à sa position actuelle
fenetre.blit(image_personnage, (x_personnage, y_personnage))
```

Étape 3 : Gérer les déplacements du personnage

Pour permettre au joueur de déplacer le personnage à l'aide des touches du clavier, nous devons :

1. **Capturer les événements clavier** (lorsque l'utilisateur appuie sur une touche).
2. **Modifier la position du personnage** en conséquence (se déplacer vers la gauche, la droite, le haut ou le bas).

Le déplacement du personnage se fait en utilisant les **touches fléchées** du clavier : **gauche**, **droite**, **haut** et **bas** (ou les touches `zqsd`). Chaque fois que l'utilisateur appuie sur une touche, le personnage se déplace d'une case dans la direction correspondante. En termes de **pixels**, cela signifie que la position du personnage sur l'écran est modifiée par exemple de **32 pixels** (qui correspond à la taille d'une tuile) à chaque pression de touche.

Le nombre de pixels utilisés pour le déplacement de votre personnag représente en fait sa **vitesse de déplacement**. Au plus cette valeur est grande, au plus votre personnage **se déplace rapidement à l'écran**. A savoir que ses déplacements deviennent aussi **moins fluides**.

Généralement, **la vitesse est stockée dans une variable**, ce qui permet au jeu de la faire... varier (votre personnage poeut marcher, courir, 'manger un piment'...).

Avec `x` (position horizontale) et `y` (position verticale), les déplacements se font comme suit:

- Vers la **gauche**: `x = x - vitesse`
- Vers la **droite**: `x = x + vitesse`
- Vers le **haut** `y = y - vitesse`
- Vers le **bas**: `y = y + viitesse`

Utiliser une vitesse égale à la taille de tes tuiles permet de synchroniser les mouvements du personnage avec la grille du labyrinthe pour un déplacement précis et aligné.

Dans PyGame, la fonction `pygame.key.get_pressed()` permet de détecter quelles touches sont pressées. Nous allons l'utiliser pour détecter les flèches du clavier (`K_LEFT`, `K_RIGHT`, `K_UP`, `K_DOWN`).

À chaque itération de la boucle de jeu, nous allons mettre à jour la position du personnage en fonction des touches pressées.

Code pour les déplacements avec les touches fléchées :

```
# Vitesse du personnage (combien de pixels il se déplace à chaque pression)
vitesse = 32  # Correspond à une case de la grille --> MODIFIEZ pour obtenir la bonne vitesse

# Boucle de jeu principale
continuer = True
while continuer:
    # Gestion des événements
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            continuer = False

    # Récupérer les touches pressées
    touches = pygame.key.get_pressed()

    # Mettre à jour la position du personnage en fonction des touches
    if touches[pygame.K_LEFT]:
        x_personnage -= vitesse  # Déplacer vers la gauche
    if touches[pygame.K_RIGHT]:
        x_personnage += vitesse  # Déplacer vers la droite
    if touches[pygame.K_UP]:
        y_personnage -= vitesse  # Déplacer vers le haut
    if touches[pygame.K_DOWN]:
        y_personnage += vitesse  # Déplacer vers le bas

    # Remplir l'écran avec les murs et le sol
    # ....

```

```
        # Dessiner le personnage à sa nouvelle position
        fenetre.blit(image_personnage, (x_personnage, y_personnage))

        # Mettre à jour l'affichage
        pygame.display.flip()

        # Limiter le framerate
        pygame.time.Clock().tick(60)

    pygame.quit()  # Quitter PyGame proprement
```

Explication du code :

1. **Récupération des touches pressées** :

   ```
   touches = pygame.key.get_pressed()
   ```

   Cette fonction renvoie un tableau où chaque index correspond à une touche. Si la touche est enfoncée, l'index correspondant vaut `True`.

2. **Déplacement du personnage** :

   - Si la touche fléchée gauche (`K_LEFT`) est pressée, on soustrait `vitesse` à `x_personnage` pour le déplacer vers la gauche.
   - Si la touche fléchée droite (`K_RIGHT`) est pressée, on ajoute `vitesse` à `x_personnage` pour le déplacer vers la droite, et ainsi de suite pour les directions haut et bas.

3. **Rafraîchissement de l'écran** : Après chaque déplacement, le labyrinthe et le personnage sont redessinés dans leur nouvelle position. La fonction `pygame.display.flip()` met à jour l'affichage.

4. **Limitation de la vitesse** : Pour éviter que le jeu ne tourne trop vite, nous limitons le nombre d'images par seconde avec `pygame.time.Clock().tick(60)` (ici, 60 FPS).

Améliorations possibles

1. **Collision avec les murs** : Pour l'instant, le personnage peut passer à travers les murs. Vous pouvez ajouter une gestion des collisions en vérifiant si la case vers laquelle le personnage veut se déplacer est un mur avant de modifier ses coordonnées.

2. **Animations du personnage** : Si vous voulez aller plus loin, vous pouvez ajouter une animation au personnage lorsqu'il se déplace en changeant son sprite à chaque pas.

3. **Limites de la fenêtre** : Assurez-vous que le personnage ne puisse pas sortir des limites du labyrinthe en ajoutant des vérifications sur `x_personnage` et `y_personnage`.

4. **Orientation**: Quand votre personnage se déplace dans différentes directions, il sera nécessaire de modifier son orientation pour que le rendu soit plus réaliste.

Conclusion

Dans cet article, nous avons ajouté un personnage à notre jeu PyGame en le dessinant à partir d'une image et en gérant ses déplacements au clavier. C'est une base essentielle pour rendre le jeu interactif et amusant pour les joueurs. Vous pouvez maintenant enrichir le jeu en améliorant la gestion des collisions, en ajoutant des niveaux ou même des ennemis pour rendre le jeu plus complexe !

Améliorations

- Modifie ton code pour qu'il supporte aussi les touches Z Q S D pour les déplacements.
- Modifie ton code pour faire courir ton personnage quand la touche SHIFT est appuyée.