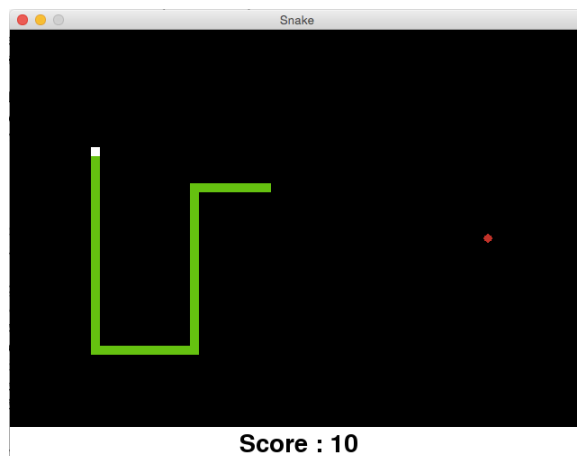


Travaux pratiques

9 Snake

Nous allons réaliser le Jeu du serpent (aussi appelé Snake) en Python avec Pygame. Notre objectif est de pratiquer tous les notions vues en cours jusqu'à présent, en particulier les fonctions et les listes. Ce TP nous montrera également comment programmer un jeu vidéo avec animation.



Quelques pistes:

- Pendant la programmation, rappelez-vous de laisser des lignes blanches entre les groupes d'instructions pour augmenter la lisibilité du programme.
- Utilisez des commentaires afin d'expliquer ce que fait chaque partie de votre programme. Ceci vous aidera à réutiliser des morceaux de votre programme dans d'autres programmes.
- Testez régulièrement votre programme et corrigez les éventuelles erreurs. Si vous laissez les tests seulement pour la fin, les erreurs seront plus difficiles à trouver.
- Si vous êtes incertain sur la valeur d'une variable durant l'exécution du programme, vous pouvez l'afficher avec une instruction `print`. Ceci apparaîtra dans la console de Python et vous aidera à trouver des éventuelles erreurs.

1 Introduction

Dans le jeu du serpent le joueur doit guider le serpent afin de manger les fruits (petits ronds rouges) qui apparaissent à l'écran. Cependant, si le serpent heurte le mur (le bord de l'écran) ou s'il mord sa propre queue, il meurt! Cherchez sur Google "snake game" pour voir des nombreuses exemples de ce jeu très amusant.

2 Avant de commencer à jouer

1. Tout d'abord, nous allons définir quelques constantes. **Attention:** il s'agit ici des **constantes** définies au niveau zéro (c.a.d.: des constantes globales du programme). Ceci est toléré seulement parce qu'il s'agit des constantes. Rappelez-vous, **les variables globales sont à éviter!!**

Déclarer les constantes de vos couleurs préférés, par exemple:

```
BLANC = ( 255, 255, 255)
NOIR   = (   0,   0,   0)
VERT   = (   0, 192,   0)
ROUGE  = ( 192,   0,   0)
...
```

Ensuite, déclarer deux constantes qui serviront à paramétrer le jeu:

```
L_CARRE = 10    # Largeur d'un carre du serpent.
VITESSE = 10    # Vitesse de déplacement du serpent (doit etre egale a L_CARRE).
```

2. Définir la fonction principale du programme:

```
def main():
    ''' Fonction principale du programme. '''
```

3. Dans la fonction `main`:

- initialiser Pygame;
- ouvrir la fenêtre de l'application;
- créer la boucle principale du jeu (n'oubliez pas l'horloge);
- traiter l'événement `pygame.QUIT`;
- finaliser Pygame.

4. Écrire l'instruction qui exécute la fonction principale quand le programme est exécutée:

```
if __name__ == '__main__':
    main()
```

5. Créer la fonction `ouverture`, qu'affiche l'écran d'ouverture du jeu. L'ouverture doit contenir:

- le nom du jeu ("Snake") en grandes lettres;
- le nom de l'auteur du jeu (vous!);
- les instructions pour jouer (mais ne soyez pas trop long);
- la version et la date de création du jeu.

L'ouverture doit rester quelques instants à l'écran. Pour cela, vous pouvez utiliser la fonction `pygame.time.wait(<millisec)`.

6. Appeler la fonction `ouverture` dans la fonction `main`.

7. Tester votre programme. Vérifier que l'ouverture s'affiche correctement et que l'utilisateur a assez de temps pour lire toutes les informations affichées.

3 Création du niveau

Maintenant, nous allons créer l'écran du premier niveau du jeu. En effet, notre jeu n'aura qu'un seul niveau ou phase de jeu, mais vous pouvez en créer d'autres si vous voulez! Le niveau du jeu contient le score et l'aire de jeu. Dans notre cas, l'aire de jeu n'est rien d'autre qu'un rectangle noir où le serpent se déplace, mais dans un niveau de jeu plus avancé, ceci contiendrait aussi des obstacles, des ennemis, des pièges, etc.

1. Créer une fonction `niveau_init` qui prend comme argument la surface de la fenêtre de l'application. Cette fonction initialise le niveau du jeu. La fonction doit créer un dictionnaire `niveau` qui contiendra toutes les informations concernant le niveau du jeu. La fonction doit retourner le niveau créé. Exemple:

```
def niveau_init(surface):
    ''' Init niveau. '''
    niveau = {}
    niveau['surf'] = surface
    niveau['score'] = 0
    ...
    return niveau
```

Les informations concernant le niveau du jeu:

- `niveau['surf']`: la surface de la fenêtre de l'application.
 - `niveau['score']`: le score initial du niveau.
 - `niveau['max_score']`: le score que le joueur doit attendre pour passer de niveau.
 - `niveau['rect_score']`: une tuple de quatre nombres représentant le rectangle où le score est affiché sur l'écran;
 - `niveau['rect_jeu']`: une tuple de quatre nombres représentant l'aire du jeu. C'est à dire, le rectangle noir où le serpent se déplace.
 - d'autres informations qui vous jugerez importantes, si vous voulez.
2. Créer une fonction `niveau_dessine` qui prend comme argument une variable `niveau` et qui dessine le rectangle `niveau['rect_score']` et le score `niveau['score']` dans ce rectangle. Cette fonction doit également dessiner le rectangle `niveau['rect_jeu']` qui représente l'aire où le serpent se déplace durant le jeu.
 3. La fonction `niveau_init` doit être appelée dans la fonction `main` pour initialiser le niveau. Ceci doit être fait après l'écran d'ouverture du jeu.

```
|niveau = niveau_init(surface)
```

Maintenant, la variable `niveau` contient les informations du niveau du jeu (`score`, `max_score`, etc.). La fonction `niveau_dessine` peut être appelée pour dessiner le niveau créé:

```
|niveau_dessine(niveau)
```

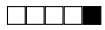
Ceci doit être fait dans la boucle principale du programme. (N'oubliez pas de mettre à jour l'écran.)

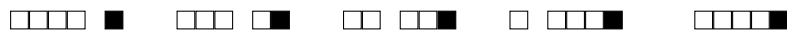
4. Testez votre programme. Vérifier que le score s'affiche correctement.

4 Création du serpent

Maintenant, nous allons créer le serpent. Dans ce jeu, le serpent se déplace tout le temps. On ne peut pas l'arrêter. Il se déplace soit horizontalement, soit verticalement (jamais en diagonale). Tout ce que le joueur peut faire est de changer la direction de déplacement du serpent.

Pour pouvoir simuler le déplacement du serpent, nous allons le représenter sous forme de plusieurs petits carrés. Le premier carré du serpent représente sa tête, ce qui donne un indice sur la direction de son déplacement actuel.

Par exemple, voici un serpent de longueur 5 avec la tête à droite.  Pour continuer le déplacement du serpent vers la droite, nous devons placer sa tête un peu plus à droite. Ensuite, le deuxième carré prend la place qui occupait la tête, ensuite le troisième prend la place du premier et ainsi de suite jusqu'au dernier carré:



Après le déplacement de tous les carrés du corps du serpent, nous les redessignons sur l'écran. Ceci donne l'impression que le serpent s'est déplacé un peu vers la droite.

Maintenant, suppose que le joueur a appuyé sur le clavier pour changer le déplacement du serpent vers le haut. Dans ce cas, la tête du serpent doit donc être placée un peu plus vers le haut. Ensuite, le deuxième carré prend sa place, le troisième prend la place du deuxième et ainsi de suite jusqu'au dernier carré du corps du serpent:



Après le déplacement de tous les carrés du corps du serpent, nous redessignons tous les carrés du corps du serpent sur l'écran et ceci donne l'impression que le serpent a tourné vers le haut.

Si le serpent continue à se déplacer vers le haut, le processus se répète:



Encore une fois, nous redessignons tous les carrés du serpent et nous avons l'impression qu'il continue à se déplacer vers le haut.

Ce processus donnera l'impression d'un serpent que ce déplace sur l'écran. Donc, le serpent n'est rien d'autre qu'une liste de petits carrés de largeur `L_CARRE`. Chaque carré de la liste est lui même une liste de deux nombres: le premier nombre représente la position horizontale du carré (coordonnée x) et le second représente la position verticale du carré (coordonnée y) sur l'écran.

1. Créer une fonction `snake_init` qui prend comme argument un niveau, la position initiale du serpent, sa longueur et sa vitesse d'agrandissement. Cette fonction initialise le serpent. De manière similaire à `niveau_init`, elle doit créer un dictionnaire `snake` qui contiendra toutes les informations concernant le serpent. À la fin, la fonction doit retourner le serpent créé. Les informations concernant le serpent sont:
 - `snake['niv']`: le niveau passé comme argument.
 - `snake['vit_agr']`: la vitesse d'agrandissement du serpent qui a été passé en argument.
 - `snake['depl_x']`: la vitesse de déplacement horizontal du serpent, initialement égal à `VITESSE`. Ceci veut dire que, initialement, le serpent se déplace vers la droite.
 - `snake['depl_y']`: la vitesse de déplacement vertical du serpent, initialement égal à 0. C'est à dire, initialement le serpent ne se déplace pas verticalement.
 - `snake['corps']`: la liste de carrés qui représente la tête et le corps du serpent. Ceci est une liste de liste de deux nombres représentant la position de chaque carrés du serpent à l'écran. Le premier carré (la tête) doit se trouver à la position passée comme argument. Le second doit se trouver exactement à sa gauche et ainsi de suite. Le nombre de carrés dans la liste doit correspondre à la longueur du serpent passée comme argument.
 - `snake['larg']`: la largeur du serpent qui est égale à la largeur de chaque carré, qui est égale à `L_CARRE`.
2. Créer une fonction `snake_dessine` qui prend comme argument un serpent et le dessine à l'écran. Il suffit donc de dessiner chaque carré dans la liste `snake['corps']` dans sa position. Notez que pour pouvoir dessiner les carrés vous devez accéder à la surface de la fenêtre de l'application. Il suffit donc d'utiliser `snake['niv']['surf']`.
3. Maintenant vous pouvez appeler la fonction `snake_init` dans la fonction `main` pour créer un serpent. Ensuite, dans la boucle principale du jeu, ajoutez l'appel à la fonction `snake_dessine`. Testez votre programme et vérifiez que le serpent s'affiche correctement à l'écran.
4. Pour l'instant, le serpent ne se déplace pas. Pourtant, dans le jeu, il doit se déplacer sans arrêt. Nous allons donc créer la fonction `snake_update_pos` qui prend comme argument un serpent et actualise sa position. Pour actualiser la position du serpent, la fonction doit donc vérifier les valeurs de `snake['depl_x']` et `snake['depl_y']`. Selon ces valeurs, la tête du serpent doit donc être déplacée vers la bonne direction. La tête se déplace toujours de `VITESSE` pixels. Ensuite, tous les autres carrés du corps du serpent doivent la suivre, comme dans l'explication donnée auparavant.
5. Maintenant vous pouvez appeler la fonction `snake_update_pos` dans la boucle principale du jeu, juste avant l'appel à la fonction `snake_dessine`. Testez votre programme et vérifiez que le serpent se déplace comme prévu vers la droite.
6. Nous ne pouvons pas encore changer la direction du serpent. C'est pour cela que maintenant, nous allons rajouter le traitement des touches du clavier. Dans la boucle principale du jeu, dans la partie de traitement des événements, ajouter une instruction qui vérifie si une touche du clavier a été appuyée:

```
| if event.type == pygame.KEYDOWN:
```

Si oui, testez quelle touche a été appuyée. Pour cela, testez si la variable `event.key` est égale à `pygame.K_LEFT`, `pygame.K_RIGHT`, `pygame.K_UP` ou `pygame.K_DOWN` et faite le traitement correspondant. C'est à dire, pour chacune des directions possibles, changez correctement les valeurs de `snake['depl_x']` et `snake['depl_y']`. Rappelez vous que la vitesse de déplacement du serpent est `VITESSE`. Elle doit être négative si le serpent se déplace vers la gauche ou vers le haut.

7. Testez votre programme et vérifiez que les touches du clavier changent la direction de déplacement du serpent correctement.
8. Pour l'instant, rien ne se passe si le serpent sort de l'aire du jeu. Nous allons donc créer la fonction `snake_heurte_mur` qui prend un snake comme argument et retourne `True` si sa tête a sortie de l'aire du jeu. C'est à dire, il faut vérifier si le carré correspondant à la tête du serpent a dépassé le bord du rectangle `snake['niv']['rect_jeu']`.
9. Dans la boucle principale du jeu, ajouter une instruction pour vérifier si le serpent a heurté le mur. Si c'est le cas, le jeu doit s'arrêter avec un message "Vous êtes mort !".
10. Dans ce jeu, le serpent meurt aussi s'il mort sa propre queue. Pour vérifier cela, créer une fonction `snake_mort_queue` qui vérifie si la tête du serpent essaye de passer sur l'un des carrés de son corps.
11. Dans la boucle principale du jeu, ajouter une instruction pour vérifier si le serpent mort sa propre queue. Si oui, le jeu doit s'arrêter avec un message "Vous êtes mort!".
12. Testez votre programme.

5 Manger les fruits

Le serpent de notre jeu est affamé. Il doit manger tous les fruits pour pouvoir s'agrandir.

1. Créer une fonction `fruit_init` qui prend comme argument un niveau. Cette fonction initialise le fruit. De manière similaire à `niveau_init` et `snake_init`, elle doit créer un dictionnaire `fruit` qui contiendra toutes les informations concernant le fruit. À la fin, la fonction doit retourner le fruit créé. Les informations concernant le fruit sont:
 - `fruit['niv']`: le niveau passé en argument.
 - `fruit['pos']`: la position du fruit sur l'écran. Vous devez générer un couple de coordonnées (x , y) de manière aléatoire. Attention: ne générez pas le fruit en dehors de l'aire du jeu. Important: le programme sera beaucoup plus facile si les fruits sont générés sur des coordonnées qui sont multiples de `L_CARRE`. Utilisez donc la fonction `random.randrange` de la manière suivante:


```
random.randrange(debut, fin, step)
```

 qui retourne un nombre aléatoire entre `debut` et `fin` qui est multiple de `step`.
 - `fruit['larg']`: la largeur (ou taille) du fruit, qui doit être égale à `L_CARRE`.
2. Créer une fonction `fruit_dessine` qui prend comme argument un fruit et le dessine sur l'écran.
3. Vous pouvez maintenant appeler les fonctions `fruit_init` et `fruit_dessine` dans les bons endroits de la fonction `main` pour dessiner le premier fruit.
4. Créer la fonction `snake_mange` qui prend comme argument un snake et un fruit et retourne si la tête du serpent est sur le fruit, auquel cas le serpent mange le fruit.
5. Créer la fonction `snake_grandit` qui prend un snake comme argument et ajoute `snake['grand']` nouveaux petits carrés au corps du snake.
6. Dans la boucle principale du jeu, ajouter une instruction pour vérifier si le serpent a mangé le fruit. Si oui, vous devez ajouter 1 à `niveau['score']`, agrandir le serpent et créer un nouveau fruit.
7. Testez votre programme et vérifiez qu'un nouveau fruit est créé, que le score change et que le serpent s'agrandit quand sa tête passe sur le fruit.
8. Dans la boucle principale du jeu, vérifiez si le score atteint son maximum, auquel cas le jeu termine avec le message "Vous avez gagné!".