TD n° 2
Images, Sprites et Animation 2D

1 Des images pour votre jeu PyPong

Dans cette partie du TD, à l'aide des fichiers images `tennis_court.png` et `tennis_ball.png`, vous remplacerez respectivement les dessins du terrain et de la balle de tennis. Vous veillerez à corriger si nécessaire votre programme Pong réalisé lors du TD précédent afin que la balle rebondisse correctement par rapport au nouveau terrain et par rapport à la taille de la balle.

1.1 Faire bouger une image fixe

La première étape de ce petit programme est de réaliser une fonction qui permette de charger une image (nous utiliserons l'image tennis_court.png). Nous aurons en effet plusieurs images à charger dans ce programme, donc il est important de factoriser cette fonctionnalité. Pour cela, nous allons écrire une fonction dont la signature (type, nom, nombre et valeur par défaut des paramètres) devra être de la forme :

```
def load_image(filename, colorkey = None):
```

1.2 Faire tourner une image qui bouge

La deuxième étape de notre programme devra combiner le mouvement de la balle de tennis avec une rotation de celle-ci sur elle-même. Pour plus de réalisme, la rotation devra s'inverser lorsque la balle rebondie sur les raquettes. Pour réaliser cette fonctionnalité, vous pourrez avoir recours à la fonction `rotate` :

```
pygame.transform.rotate(tennis_ball, angle)
```

Dans le cas où l'on réalise cette opération sur une image rectangulaire, le calque subit alors un redimensionnement après la rotation qui impacte sur la position de l'objet. Ecrivez une fonction qui corrigera ce problème si vous constatez bien ce problème.

2 Sprite Sheet : Animation 2D

Nous allons maintenant remplacer les raquettes par des images de joueurs (un peu particulier, je vous l'accorde). Nous allons animer les images à l'aide de la technique de Sprite Sheet (table de dessins pour effectuer l'animation d'un personnage dans différents type de mouvements).

2.1 Lecture une planche de dessins

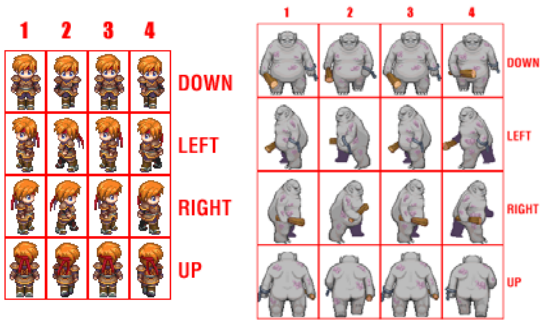
Pour réaliser ce nouveau programme, nous allons utiliser le canevas habituel. Créer une première classe SpriteSheet ayant l'interface suivante :

```
class SpriteSheet:
    def __init__(self, filename):
        …

    def split(self, width, height):
        …
```

Cette classe va vous permettre de découper les différentes images composant l'animation. Vous n'oublierez pas d'utiliser les fonctions déjà mises en place pour le chargement d'une image. Nous travaillerons sur les images `Perso.png` et `Monstre.png`. La fonction `__init__` est une fonction spécifique appelée le constructeur de classe qui est appelée quand on créé un objet de type `SpriteSheet`. Elle doit contenir les initialisations que vous souhaitez faire à la création.

TD n° 2
Images, Sprites et Animation 2D



Pour contrôler que le découpage s'opère bien, vous afficherez la planche des vignettes que vous avez identifiées.

2.2 Animation d'un personnage

Nous allons maintenant exploiter ces images pour réaliser l'animation d'un personnage qui se déplace (et ne pas uniquement faire bouger une image fixe à l'écran).

Comme nous souhaitons faire cette animation pour deux personnages (le monstre et notre petit héros), nous allons créer une classe (Character) pour gérer ce type d'objet :

```
class Character:
    def __init__(self, filename, width, height, side):
        …

    def set_default(self):
        …

    def goto(self, direction):
        …

    def start_animation(self):
        …

    def animation(self, speed):
        …

    def stop_animation(self):
        …

    def draw(self, screen, x, y):
        …
```

Modifier le programme afin de réaliser le déplacement du personnage ainsi que son animation lors de l'appui sur les touches de direction du clavier. Tout le travail simple de la classe va consister à calculer les bons indexes de ligne et de colonne pour faire l'animation.

La fonction `__init__` est comme pour la classe précédente le constructeur de la classe. Concernant les autres fonctions : `set_default` est là pour modifier les indexes de l'image aux personnage par défaut, `goto` va stocker la direction dans laquelle le personnage va se déplacer (ce qui nous permettra de savoir quelle ligne d'animation activer), les fonctions `start` et `stop` sont utilisées pour spécifier quand sera démarrée l'animation et quand elle sera stoppée (pour éviter tout problème), la fonction `animation` va stocker la vitesse du personnage (ce qui permettra de savoir quelle image afficher dans une ligne), et enfin, la fonction `draw` utilisera presque toutes les

variables de votre classe pour faire le `blit` de la bonne image de l'animation aux coordonnées `x`, `y` du `screen`. Si cette interface ne vous convient pas, vous pouvez redéfinir celle qui vous convient.

2.3 Animation de plusieurs personnages simultanément

Modifier votre programme pour réaliser la création de 1000 personnages positionnés aléatoirement à l'écran. Que pouvez-vous conclure sur les performances de la bibliothèque ainsi que sur la structuration de votre programme ?

3 Animation des joueurs dans votre programme Pong

Modifiez votre programme Pong pour remplacer les rectangles des raquettes par l'animation des personnages que nous utiliserons comme joueurs de tennis.

4 Pour aller plus loin

4.1 Collisions

Dans un jeu, nous avons besoin de tester si un objet entre en collision avec un autre objet (cas de la balle avec l'un de nos personnages). Modifier votre classe Character pour quelle utilise le classe Sprite de PyGame si ce n'est pas déjà le cas. Dans la partie gestion du jeu, vérifier si votre héro n'est pas en collision avec le monstre. Si c'est le cas, la partie est perdue.

4.2 Changer de point de vue pour votre jeu

Si maintenant, après avoir testé et jouer à votre jeu, vous pensez que le « game play » n'est pas des meilleurs, et que vous souhaitez tester le jeu sous en autre angle (changer la vue de côté par une vue de l'arrière du court), vous pouvez utiliser l'image tennis_court_3D.png mise à votre disposition.

Evaluez et si vous avez le temps, réalisez les modifications nécessaires à ce changement de point de vue sur votre programme Pong. Si nous avions été dans le cas d'une modélisation en 3D, il suffirait de changer l'emplacement de la caméra sans rien modifier au programme en lui-même. Il est donc important de bien prévoir l'angle de vue dans un programme en 2D car tous les dessins sont produits pour cet angle de vue et votre programmation est aussi souvent dépendante de celui-ci.