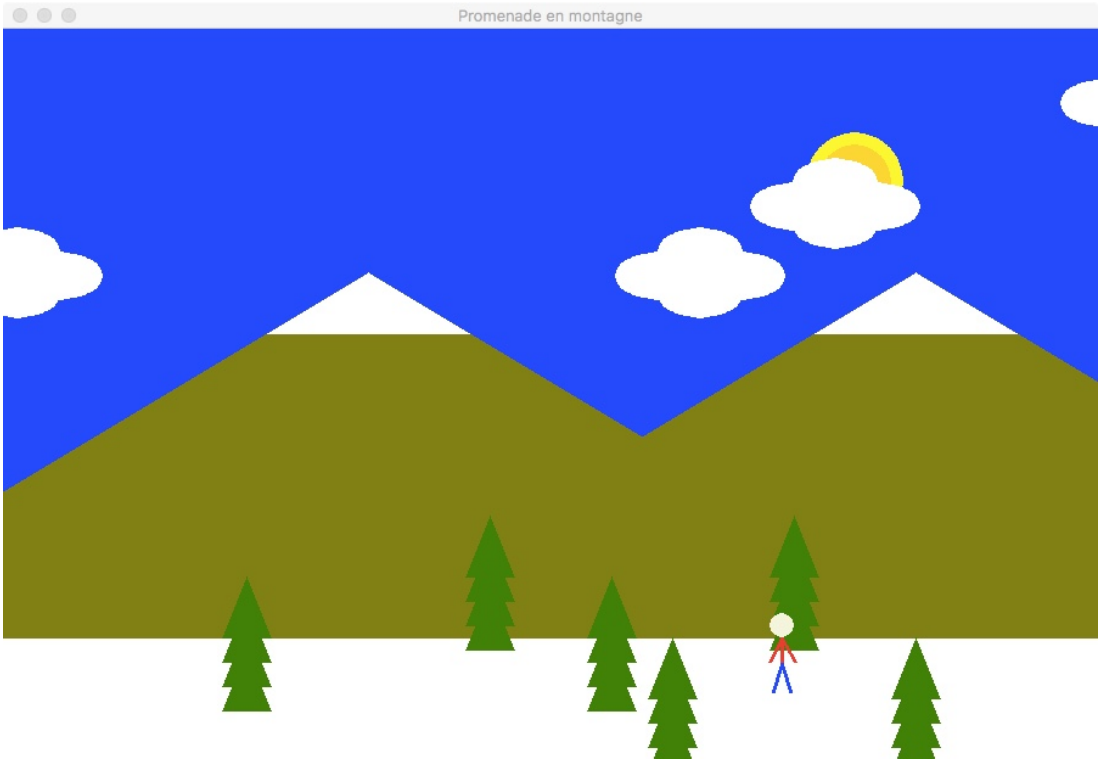
LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1

Fiche de TP numéro 11 - Animation d'une scène avec Pygame

Aujourd'hui, vous allez créer une animation en Python avec Pygame. L'objectif est de pratiquer l'utilisation des modules et des fonctions graphiques. Voici une photo de l'animation que vous allez créer, vous pouvez également voir le film de cette animation sur Moodle.



Vous devez réaliser une animation qui ressemble à peu près celle ci, mais vous n'êtes pas obligé de la faire au pixel près ! Si vos nuages sont un peu plus grands, si votre soleil n'est pas exactement au même endroit, ou bien si les montagnes n'ont pas tout à fait la même couleur, ce n'est pas grave ! L'important est de programmer une animation qui ressemble à celle-ci en respectant les quelques restrictions imposées.

Pour réaliser cette animation, vous devez utiliser les fonctions `pygame.draw.rect`, `pygame.draw.circle`, `pygame.draw.polygon`, etc. de Pygame. Vous pouvez consulter la documentation de la bibliothèque Pygame (en anglais) :

`http://www.pygame.org/docs/ref/draw.html`

Par ailleurs, n'oubliez pas de faire la documentation de vos fonctions.

1 Le module principal

Pour pouvoir tester votre animation au fur et à mesure de sa création, vous devez commencer par le module principal du programme. La première chose à faire est donc de créer un dossier appelé `tp10` et à l'intérieur de ce dossier, vous devez créer le fichier `__main__.py`. Ce fichier constitue le module principal du programme.

1 La fonction principale

Dans le module principal, écrivez la fonction `main` qui ne reçoit pas d'argument et n'a pas de valeur de retour. Cette fonction va être simplifiée par rapport au tp précédent. On la construit comme la fonction `main` du cours pour le programme de la neige qui tombe. Dans ce module, seul le module `anim` est importé :

```
import anim

def main() :
    att_anim = anim.init()
    anim.boucle(att_anim)
    anim.quitte()

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

2 Premières fonctions du module `anim`

Créez le fichier `anim.py` dans votre répertoire `tp10`. Dans ce module, vous devez :
— importer `pygame`
— écrire les fonctions `init()`, `boucle()` et `quitte()` détaillées ci-dessous.

2.1 La fonction `init()`

La fonction `init()` doit :

1. initialiser Pygame.
2. ouvrir la fenêtre de l'application, lui attribuer le titre "Promenade en montagne" et récupérer la surface de dessin dans une variable.
3. initialiser l'horloge.
4. créer un dictionnaire `scene` qui va contenir quatre éléments pour le moment :
   — associée à la clé `"surface"`, la surface de dessin
   — associée à la clé `"horloge"`, l'horloge
   — associé à la clé `"dim_fen"`, un couple qui contient les dimensions de la fenêtre de l'application
   — associé à la clé `"continuer"`, un booléen intialisé à `True`
5. retourner ce dictionnaire.

N'oubliez pas de spécifier votre fonction.

2.2 La fonction `boucle()`

La fonction `boucle()` prend en paramètre un dictionnaire du type de celui construit par `init()`. Elle ne retourne rien. Cette fonction doit faire :

— Tant que le programme doit continuer : :

1. Traiter l'événement `pygame.QUIT`
2. Mettre à jour la fenêtre de l'application.
3. Ajuster la vitesse de la boucle.

Piste : utilisez les informations contenues dans le dictionnaire.

2.3 La fonction `quitte()`

La fonction `quitte()` doit juste quitter pygame.

3 Appel à la fonction principale

Ajoutez l'appel à votre fonction principale dans le module principal.

Testez votre programme maintenant et vérifiez que la fenêtre de l'application s'ouvre. À ce stade, elle est encore vide.

2 Module couleur

Vous devez créer un module `couleurs.py` dans le dossier `tp10`. Définissez toutes les couleurs utilisées dans le scénario dans ce module, sachant que le dessin que vous allez créer doit ressembler à celui ci-dessus. Ne perdez pas trop de temps pour trouver les couleurs exactes. Vous n'êtes pas obligés d'utiliser exactement les mêmes couleurs dans votre figure.

Pour trouver le code RVB des couleurs dont vous aurez besoin dans votre animation, consultez la liste de couleurs sur wikipedia :

`http://fr.wikipedia.org/wiki/Liste_de_noms_de_couleur`

3 Module `anim`

1 Dessin

Dans la fonction `boucle()` du module `anim`, ajoutez un appel à la fonction `dessine` qui reçoit en paramètre le dictionnaire en paramètre de la fonction `boucle()`. Cet appel doit se trouver entre le traitement des événements et la mise à jour de la fenêtre de l'application.

Écrivez la fonction `dessine`. Pour le moment, elle prend en paramètre un dictionnaire et ne fait rien. Sa seule instruction est l'instruction `pass`. La fonction `dessine` va être en charge du dessin de toute la scène : le ciel, le sol, les montagnes, le soleil, mais aussi les nuages et le personnage. Au fur et à mesure que vous allez créer les autres fonctions de dessin (`dessine_ciel`, `dessine_montagne`, etc...), vous allez ajouter leur appel dans la fonction `dessine` et tester votre programme.

Vous pouvez à nouveau tester votre programme. Il ne se passe pas plus de choses, mais il ne doit pas y avoir d'erreur.

2 Le ciel

Nous allons maintenant commencer à dessiner. Écrivez la fonction `dessine_ciel` qui reçoit en paramètre `surface` et les dimensions de la surface (clé `'dim_fen'`). La fonction doit dessiner le ciel du scénario (sans les nuages). Le ciel occupe tout l'espace.

Ajoutez l'appel à cette fonction dans la fonction `dessine`, et supprimez l'instruction `pass`. Testez votre programme.

3 Le soleil

Écrivez une fonction `dessine_soleil` qui reçoit `surface` argument. La fonction doit dessiner le soleil.

Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme.

4 Les montagnes

Écrivez une fonction `dessine_montagnes` qui reçoit `surface` en paramètre.. La fonction doit dessiner les montagnes.

Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme.

5 Le sol

Écrivez une fonction `dessine_sol` qui reçoit en paramètre `surface` et les dimensions de la surface (clé `'dim_fen'`). La fonction doit dessiner le sol. Le sol occupe un quart de l'espace.

Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme.

6 Les arbres

Écrivez une fonction `dessine_arbre` qui reçoit un tuple de deux entiers `(x, y)` et `surface` en arguments. La fonction doit dessiner un seul arbre dans la position passée en argument.

Piste : les arbres sont dessinés en utilisant plusieurs triangles. Pour pouvoir dessiner un triangle à la position `(x, y)` vous devez utiliser la technique du *décalage*. Par exemple, au lieu de faire :
`pygame.draw.polygon(surface, VERT, ((0, 50), (20, 0), (40, 50)))`
Vous pouvez faire :
`pygame.draw.polygon(surface, VERT, ((x+0, y+50), (x+20, y+0), (x+40, y+50)))`.
Et pour dessiner 4 triangles, vous pouvez faire un `for i in range(4)` et utiliser la variable `i` pour calculer la position de chaque triangle.

Ajoutez plusieurs appels à cette fonction dans la fonction `dessine`. Variez les positions des arbres pour créer l'impression d'un petit bois. Testez votre programme.

7 Mise à jour

Il s'agit de la fonction qui sert à mettre à jour le scénario, c'est-à-dire à changer l'image dessinée à chaque passage dans la boucle principale. Elle fera appel à d'autres fonctions pour changer les positions des nuages et du personnage. Pour l'instant, cette fonction contiendra uniquement l'instruction : `pass`. En d'autres mots, pour l'instant, elle ne fait rien.

Ajoutez l'appel à cette fonction dans la boucle du programme principal. L'instruction `anim.met_a_jour()` doit être placée juste avant l'appel à dessine.

4 Module nuages

Créez un module `nuages.py` dans le dossier `tp10`. Comme le module `anim`, le module `nuages` doit contenir les fonctions `init`, `dessine` et `update`.

La programmation des nuages est très semblable à celle de la neige vue en cours. Mais au lieu de bouger du haut vers le bas, les nuages bougent de la gauche vers la droite de l'écran.

1 Initialisation

La fonction `init()`, dans le module `nuages`, reçoit un entier `nombre` en argument et retourne un dictionnaire contenant les attributs des nuages. La fonction doit donc créer un dictionnaire `att` contenant les attributs des nuages listés ci-dessous :

- `'vit'` : la vitesse de déplacement d'un nuage (tous les nuages se déplacent à la même vitesse). Il s'agit du nombre de pixels de décalage entre deux images.
- `'lim'` : les coordonnées `(x, y)` limites pour un nuage à l'écran (après lesquelles les nuages disparaissent de l'écran).

— `'objs'` : une liste contenant `nombre` tuples de deux entiers `(x, y)`, chacun correspondant à la position d'un nuage à l'écran.

Pour créer la liste des nuages (`objs`), utilisez une boucle et initialisez leurs positions aléatoirement à l'écran. Retournez le dictionnaire avec les attributs.

Importez le module nuages dans le module `anim` :

```
import nuages
```

Ajoutez l'appel à la fonction `init` que vous venez de créer dans la fonction `init` du module `anim` de façon à ajouter les attributs de nuages aux attributs du scénario. Par exemple, ajoutez : `scene['nuages'] = nuages.init(4)`
pour ajouter 4 nuages au scénario.

2 Dessin

Dans le module `nuages`, écrivez la fonction `dessine` qui reçoit les attributs des nuages `att` et `surface` en arguments. La fonction doit dessiner tous les nuages à leur positions respectives sur l'écran.

Piste : Les nuages sont dessinés avec des ellipses.

Ajoutez l'appel à la fonction `nuages.dessine` dans la fonction `dessine` du module `anim` pour dessiner les nuages.

Piste : Si les nuages sont dessinés avant les montagnes ces dernières apparaissent devant les nuages et vice-versa.

Testez votre programme (les nuages s'affichent, mais ne se déplacent pas encore).

3 Mise à jour

Toujours dans le module `nuages`, écrivez la fonction `met_a_jour` qui reçoit les attributs des nuages `att` en argument. La fonction doit faire bouger les nuages. C'est-à-dire que pour chaque nuage de la liste des nuages (`att['objs']`), vous devez changer sa position en ajoutant sa vitesse `att['vit']`. Si la position dépasse la limite (`att['lim']`), alors recréez un nuage aléatoirement à l'autre bout de l'écran.

Piste : Jetez un oeil sur le programme de la neige.

Ajoutez l'instruction :

```
nuages.met_a_jour(att['nuages'])
```

dans la fonction `met_a_jour` du module `anim` pour mettre à jour les nuages.

Testez votre programme. Vos nuages doivent bouger maintenant.

5 Le personnage

Créez le module `perso.py` dans le dossier `tp10`. Encore une fois, nous allons créer un module avec les fonctions `init`, `dessine` et `met_a_jour`.

1 Initialisation

Écrivez une fonction `init` qui reçoit un tuple de deux entiers `(x, y)` en argument. La fonction doit initialiser le dictionnaire `att` des attributs du personnage et ensuite le retourner. Le dictionnaire doit contenir les champs suivants :

— `'pos'` : un tuple de deux entiers contenant la position du personnage. Ceci est égal à la position passée en argument.

— `'lim'` : un entier contenant la position limite du personnage à l'écran. Il s'agit de l'endroit où il arrêtera de marcher vers la gauche et se mettra à marcher vers la droite. Ceci est égal à la largeur de l'écran moins la largeur du dessin de l'homme-bâton.

— `'vit'` : un entier contenant la vitesse de déplacement du personnage.
— `'img'` : un entier entre 0 et 2. Ceci représente l'image actuelle du personnage. Si cet attribut est égal à 0, vous allez dessiner l'homme-bâton avec les jambes ouvertes, s'il est égal à 1, dessiner avec les jambes demi-ouvertes, s'il est égal à 2 dessiner avec les jambes fermées.

Ajoutez les instructions nécessaires pour ajouter un personnage au scénario.

2 Dessin

Écrivez la fonction `dessine` qui reçoit une surface et **un tuple** de deux entiers `(x, y)` en argument. La fonction doit dessiner l'homme-bâton qui correspond à l'image actuel du personnage (attribut `att['img']`).

Piste : Vous pouvez créer une fonction différente pour chaque position des jambes de l'homme-bâton. Vous appellerez ici la fonction qui correspond selon la valeur de l'attribut `img`.

Ajoutez les instructions nécessaires pour dessiner le personnage en même temps que les autres éléments du scénario.

Piste : Comme pour les nuages, si vous dessinez le personnage avant un arbre il apparaîtra derrière cet arbre et vice-versa.

Testez votre programme maintenant. Il doit dessiner le personnage, mais celui-ci ne bouge pas encore.

3 Mise à jour

Toujours dans le module `perso`, écrivez une fonction `met_a_jour` qui reçoit un dictionnaire d'attributs d'un personnage comme argument. La fonction doit réaliser les trois opérations suivantes :

1. Mettre à jour l'image du personnage. C'est à dire que l'attribut `img` doit être incrémenté, ou bien revenir à 0 dans le cas où il a dépassé sa valeur maximale 2.
2. Mettre à jour la position du personnage. C'est à dire que l'attribut `pos` doit être incrémenté de la valeur de l'attribut `vit`.
3. Rebondir sur le bord de l'écran si nécessaire. C'est à dire, vérifier si le personnage a dépassé la limite de l'écran (c.à.d. : vérifier la valeur de l'attribut `pos` par rapport à `lim`). Auquel cas, la vitesse doit être multipliée par -1.

Ajoutez les instructions nécessaires pour mettre à jour le personnage en même temps que les autres éléments du scénario.

Testez votre programme maintenant. C'est fini !