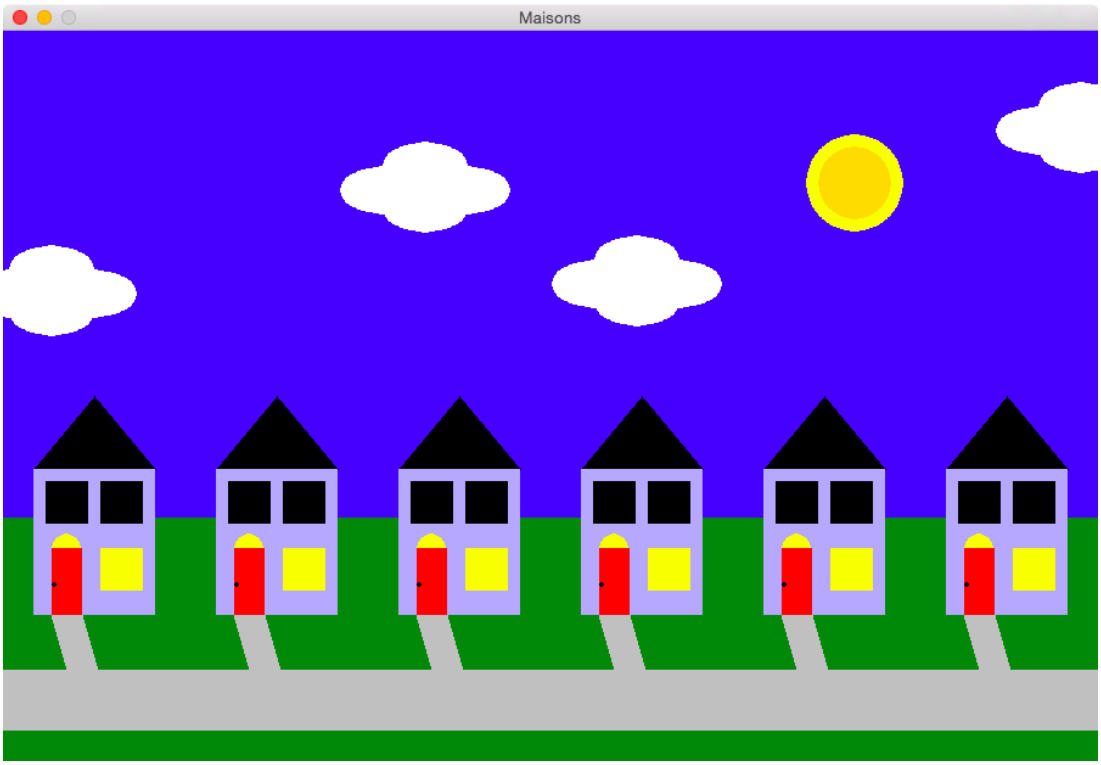
LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1

Fiche de TP numéro 9 - Des figures avec Pygame

Aujourd'hui, vous allez créer un dessin en Python avec Pygame. L'objectif est de pratiquer l'utilisation des modules et des fonctions graphiques. Voici une photo du dessin que vous allez créer.



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

Pour réaliser ce dessin, 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 programme

Le programme que vous allez créer est composé de trois fichiers :
— `__main__.py` (module principal)
— `couleur.py` (module couleur)
— `dessin.py` (module dessin)

2 Le module principal

Pour pouvoir tester votre dessin 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 `tp09`. À l'intérieur de ce dossier, créez le fichier `__main__.py`. Ce fichier constitue le module principal du programme.

2.1 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 doit :

1. Initialiser Pygame.
2. Ouvrir la fenêtre de l'application et lui attribuer un titre.
3. Initialiser l'horloge.
4. Tant que le programme n'est pas terminé :
   (a) Traiter l'événement `pygame.QUIT`
   (b) Effacer la surface de l'application.
   (c) Mettre à jour la surface de l'application.
   (d) Ajuster la vitesse de la boucle.
5. Terminer Pygame.

Pour écrire cette fonction, aidez-vous du cours (disponible sur Moodle).

2.2 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. Lorsque vous cliquez sur la croix, la fenêtre se ferme.

3 Module couleur

Vous devez créer un module `couleur.py` dans le dossier `tp09`. Définissez toutes les couleurs nécessaires pour réaliser le dessin dans ce module. Le dessin que vous allez créer doit ressembler à celui ci-dessus, néanmoins, 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, et vous pourrez affiner à la fin.

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

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

4 Module dessin

Vous devez créer un module `dessin.py` dans le dossier `tp09`. Importez le module `dessin` dans le module principal (`__main__.py`) :

```
import dessin
```

Ajoutez l'instruction :

```
dessin.dessine(surface)
```

dans la fonction principale `main` du module principal du programme, où `surface` correspond à la surface de l'application. Placez cette instruction dans la boucle principale juste avant la mise à jour de la surface de l'application.

4.1 Dessin

Dans le module `dessin`, écrivez la fonction `dessine` qui reçoit `surface` en argument. Cette fonction fera appel aux fonctions définies ci-dessous pour dessiner.

La fonction `dessine` est donc responsable de l'action *dessiner l'image*. Au fur et à mesure que vous allez créer les autres fonctions de dessin, (`dessine_ciel`, `dessine_maison`, etc.) vous devrez ajouter ces appels dans la fonction `dessine` et tester votre programme.

Dans un premier temps, votre fonction ne comporte qu'une seule instruction : `pass`. Cette instruction ne fait rien, elle va juste nous permettre de tester l'application sans que la fonction `dessine` soit réellement implémentée. Testez votre application maintenant : il ne doit rien se passer de plus (mais il ne doit pas y avoir non plus d'erreur !).

Tout ce qui suit se passe uniquement dans le module `dessin`.

4.2 Le ciel

Nous allons maintenant commencer à dessiner. Écrivez la fonction `dessine_ciel` qui reçoit `surface` en paramètre. La fonction doit dessiner le ciel de l'image (sans les nuages). Cela revient à dessiner un grand rectangle bleu. Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme.

4.3 Le soleil

Écrivez une fonction `dessine_soleil` qui reçoit `surface` en paramètre. La fonction doit dessiner le soleil (un disque jaune, ou, pour faire plus joli, deux disques jaunes, le plus petit et plus foncé sur le plus grand et plus clair). Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme. Que se passe-t-il si vous appelez la fonction `dessine_soleil` avant la fonction `dessine_ciel` ? après ?

4.4 Le sol

Écrivez une fonction `dessine_sol` qui reçoit `surface` en paramètre. La fonction doit dessiner le sol (vert) et le trottoir. Combien d'instructions sont-elles nécessaires ? On ne s'occupe pas ici de dessiner le petit trottoir devant la porte de chaque maison (car cela sera fait dans la fonction qui dessine la maison). Ajoutez l'appel à cette fonction dans la fonction `dessine` et testez votre programme.

4.5 Les maisons

Écrivez une fonction `dessine_maison` qui reçoit un couple de deux entiers `(x, y)` et `surface` en paramètres. La fonction doit dessiner une seule maison dans la position passée en argument, ainsi que le petit trottoir devant la porte. Les coordonnées `(x, y)` sont les coordonnées du coin en haut à gauche du rectangle dans lequel s'inscrit le dessin de la maison.

Écrivez maintenant une fonction `dessine_maisons` (pluriel !) qui reçoit un entier `nbre_maisons`, un entier `ecart`, un couple de deux entiers `(x, y)`, et `surface` en paramètres. Cette fonction appelle `dessine_maison` de manière à ce que `nbre_maisons` soit dessinées avec un écart entre deux maisons égal à `ecart`. Cela permettra de créer l'impression d'une rue résidentielle.

Ajoutez l'appel à fonction `dessine_maisons` dans la fonction `dessine`. Testez votre programme (et affinez pour avoir un nombre de maisons et un écart entre deux maisons qui vous paraisse joli). Vous pouvez commencer à ne dessiner qu'une seule maison, régler les dimensions des éléments de la maison, puis dessiner plusieurs maisons.

4.6 Les nuages

Écrivez la fonction `dessine_nuage` qui reçoit un couple `(x, y)` et `surface` en arguments. Cette fonction dessine un nuage, et les coordonnées `(x, y)` sont les coordonnées du coin en haut à gauche du rectangle dans lequel s'inscrit le dessin du nuage. **Piste :** Les nuages sont dessinés avec des ellipses.

Ajoutez trois ou quatre appels à la fonction `dessine_nuage` dans la fonction `dessine` pour dessiner les nuages. Vous choisirez les coordonnées de chaque nuage de manière à ce qu'ils soient répartis harmonieusement.

Testez votre programme, et affinez les derniers détails. C'est fini !