POLYTECH MARSEILLE
Aix*Marseille Université

PEIP2

Ingénierie Informatique

TP n°1

Premiers pas dans Candy Crush

Étape 1 – Installation de Candy Crush

En salle de TP, connectez-vous en utilisant l'image Windows :

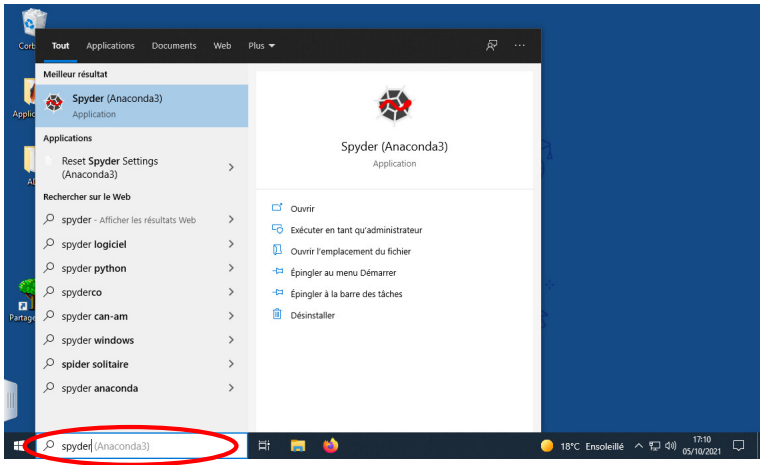
ETOILE-POLYTECH (Windows)

C'est en effet l'image qui contient tous les logiciels dont vous aurez besoin en TP.

Dans les énoncés, les URLs sont affichées en bleu et sont cliquables. Cliquez sur celle ci-dessous afin de télécharger le script d'installation de Candy Crush et sauvegardez le sur **votre bureau**.

https://pageperso.lis-lab.fr/christophe.gonzales/teaching/ing-info/ressources/ing-info-installer.py

Ensuite, comme le montre l'ellipse rouge de la figure ci-dessus, sous Windows 10, tout en bas de l'écran, dans la barre des tâches, vous pouvez rechercher une application. Tapez « spyder » afin de rechercher Spyder. Exécutez l'application Spyder (Anaconda3) proposée en haut de la fenêtre.



Dans le menu Fichier/File en haut à gauche de la fenêtre Spyder, sélectionnez l'item Ouvrir/Open et ouvrez le script `ing-info-installer.py` que vous venez de télécharger puis cliquez sur l'icône en forme de triangle vert afin de l'exécuter. La première fois que vous exécutez un fichier sous Spyder, une fenêtre de configuration apparaît afin de vous permettre de paramétrer la manière dont vous souhaitez réaliser l'exécution. Cliquez sur le bouton Run tout en bas de cette fenêtre (les paramètres par défaut

n'ont pas besoin d'être modifiés). Sous windows, les fenêtres graphiques s'ouvrent derrière la fenêtre de Spyder. Si celle de `ing-info-installer` est cachée par Spyder, regardez dans la barre des tâches en bas de votre écran (à droite de l'ellipse rouge sur la figure ci-dessus). Vous devriez voir l'icône de l'installer juste à droite de celle de Spyder. Cliquez dessus pour visualiser la fenêtre.

`ing-info-installer` va vous demander de sélectionner le répertoire dans lequel vous souhaitez installer l'archive de Candy Crush avec laquelle vous allez travailler en TP. Vous pouvez choisir n'importe quel répertoire, mais je vous suggère de sélectionner **votre bureau**. `ing-info-installer` va probablement télécharger et installer un package appelé `wget`. Le cas échéant, si vous voyez un échec d'installation de ce package, fermez Spyder et réexécutez-le. Cela mettra à jour certains chemins nécessaires pour que `ing-info-installer` puisse utiliser `wget`. Réexécutez `ing-info-installer`. Cela devrait télécharger l'archive et l'installer dans le répertoire que vous avez sélectionné. Ce dernier contiendra alors un nouveau répertoire `ing-info`, contenant lui-même tous les fichiers dont vous avez besoin, notamment `candy.py` qui sera l'unique fichier que vous éditerez durant les TP.

Ouvrez-le maintenant dans Spyder.

Étape 2 – Fonction getTypeBonbon

Parcourez le code de `candy.py` jusqu'à arriver à partie correspondant au TP n°1 et plus précisément à la fonction `getTypeBonbon`. Celle-ci prend en paramètre un entier et elle renvoie son chiffre des unités. Par exemple, pour un nombre entier égal à 24, elle renverra l'entier 4. Réécrivez la fonction `getTypeBonbon` et testez qu'elle produit bien le bon résultat. Par réécrire la fonction, on entend que vous remplaciez l'instruction « `return cdp.getTypeBonbon(bonbon)` » par vos propres instructions. Pour tester que votre fonction est correcte, vous pouvez la sélectionner à la souris dans Spyder et appuyer sur la touche F9. Vous la verrez alors apparaître sur la console Python en bas à droite. Vous pourrez l'exécuter dans cette console, par exemple en lui passant en paramètres 5 puis 24. Dans le 1er cas, elle doit renvoyer 5, et dans le second la valeur 4.

Utilité de la fonction :

Dans la modélisation de Candy Crush, un bonbon est représenté par un entier dont le chiffre des dizaines représente son *supertype* (bonbon normal, strié horizontalement, strié verticalement) et le chiffre des unités représente son *type* (bonbon ovale orange, carré vert, jaune en forme de poire, *etc.*). Cf. la vidéo sur les principes de programmation du jeu.
Le nombre passé en paramètre de la fonction est la représentation d'un bonbon. La fonction renvoie alors le type du bonbon (bonbon ovale orange, carré vert, jaune en forme de poire, *etc.*) encodé sous la forme d'un entier. Cette fonction est utile, par exemple, dans l'écriture des fonctions qui calculent le nombre de bonbons contigus de même type sur une ligne ou une colonne. Cela s'avère utile pour calculer les match3, qui impliquent au moins trois bonbons de même type (couleur/forme), peu importe qu'ils soient striés ou non.

Étape 3 – Fonction getSupertypeBonbon

La fonction `getSupertypeBonbon` prend en paramètre un entier compris entre 0 et 99 et elle renvoie son chiffre des dizaines. Par exemple, pour un nombre entier égal à 24, elle renverra l'entier 2. Réécrivez cette fonction et testez la.

Utilité de la fonction :

Le nombre passé en paramètre de la fonction est la représentation d'un bonbon. La fonction renvoie alors le supertype du bonbon (bonbon normal, strié horizontalement, strié verticalement) encodé sous la forme d'un entier. Cette fonction sera utile, notamment, dans les fonctions d'affichage. En effet, les images des bonbons à afficher sont stockées dans un tableau 2D dont la première dimension est le supertype et l'autre le type du bonbon.

Étape 4 – Fonction getBonbon

La fonction `getBonbon` prend en paramètres deux entiers. Elle renvoie le nombre entier dont le chiffre des dizaines correspond au 1er paramètre et le chiffre des unités au 2ème paramètre. Par exemple, si le 1er paramètre est égal à l'entier 2 et le 2ème à l'entier 4, la fonction renverra l'entier 24.

Réécrivez cette fonction et testez la, par exemple en lui passant en paramètres (0,5) puis (2,4). Dans le 1er cas, elle doit renvoyer 5, et dans le second la valeur 24.

Utilité de la fonction :

La fonction `getBonbon` est notamment appelée par la fonction `initialiseTableauJeu` du TP n°2 qui, comme son nom l'indique, initialise le tableau de jeu. Dans ce cas, le 1er paramètre passé à la fonction `getBonbon` représente le supertype d'un bonbon et le second son type. L'entier renvoyé est le bonbon correspondant.

Information :

À partir de maintenant, les paramètres des fonctions correspondant à des lignes ou des colonnes du tableau « modèle » de jeu seront notées `lig` ou `col`, et les coordonnées en pixels (pour les affichages) seront notées `x` ou `y`.

Étape 5 – Fonction getXPixel

La fonction `getXPixel` prend en paramètre le numéro de colonne `col` d'un bonbon dans le tableau « modèle » et renvoie le numéro `x_pixel` du pixel le plus à gauche de ce bonbon dans la fenêtre d'affichage. Ce numéro est défini par l'équation suivante :

$$\texttt{x\_pixel} = \text{X\_TABLEAU\_JEU} + \texttt{col} \times \text{TAILLE\_BONBON}. \quad (1)$$

Réécrivez la fonction `getXPixel`. Pour tester vos fonctions, à partir de maintenant, le plus simple est de sauvegarder votre fichier (ctrl-S) et d'exécuter le jeu (soit en cliquant sur le triangle vert dans la barre de menu, soit en tapant sur la touche F5). Si vous voyez le jeu s'afficher comme avant, c'est que votre fonction est correcte. Si vous observez des affichages ultra bizarroïdes, c'est qu'elle est incorrecte. Vous pouvez également réaliser l'opération décrite dans le dernier transparent de la vidéo « Principes de programmation du jeu » afin de vérifier que votre fonction est correcte.

Pour bien vérifier votre fonction, ne vous contentez pas de démarrer le jeu. Regardez également ce qui se passe si vous essayez d'intervertir des bonbons ne faisant pas partie de match3 et se trouvant sur

la ligne la plus haute ou la plus basse de l'espace de jeu ainsi que sur les colonnes les plus à gauche et les plus à droite.

Compléments d'informations :

On rappelle que, dans le « modèle » du jeu (cf. la vidéo sur les principes de programmation du jeu), l'espace de jeu est représenté par un tableau `tab_jeu` de `NB_COLONNES_JEU` colonnes et `NB_LIGNES_JEU` lignes (si vous n'avez pas modifié ces « constantes », celles-ci sont respectivement égales à 9 et 11 : il y a donc dans votre jeu 9 colonnes et 11 lignes de bonbons). Les colonnes sont la 1ère dimension du tableau et les lignes la 2ème. Autrement dit, `tab_jeu[col,lig]` correspond au bonbon sur la colonne `col` et ligne `lig`. Tout le jeu est réalisé avec ce tableau « modèle », excepté les affichages dans la fenêtre de jeu où l'on a besoin de déterminer en pixels sur la fenêtre de jeu l'endroit précis où l'on affiche les bonbons.
Pour en revenir à la correspondance entre `col` et `x_pixel`, comme le montre la figure 1, le bonbon hexagonal en haut à gauche est situé dans le tableau « modèle » en colonne 0 et le numéro de son pixel le plus à gauche est X_TABLEAU_JEU. La poire jaune située à sa droite est en colonne 1 et le numéro de son pixel le plus à gauche est X_TABLEAU_JEU + TAILLE_BONBON.

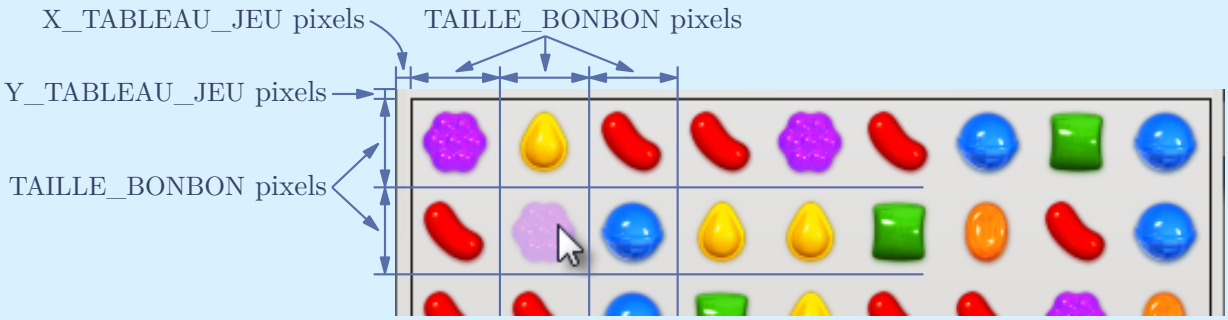


FIGURE 1 – Les tailles en pixels de l'espace de jeu.

Utilité de la fonction :

La fonction `getXPixel` est utilisée, notamment, par les fonctions d'affichage du jeu : pour afficher un bonbon donné, on récupère ses coordonnées `col` et `lig` dans le tableau « modèle » et on calcule les coordonnées (`x_pixel`,`y_pixel`) où il faut l'afficher dans la fenêtre de jeu.

Étape 6 – Fonction getYPixel

La fonction `getYPixel` prend en paramètre un numéro de ligne `lig` du tableau « modèle » et renvoie le numéro `y_pixel` du pixel dans la fenêtre de jeu le plus haut du bonbon correspondant. Cela correspond à l'équation suivante :

$$\texttt{y\_pixel} = \text{Y\_TABLEAU\_JEU} + \texttt{lig} \times \text{TAILLE\_BONBON}. \quad (2)$$

Réécrivez la fonction `getYPixel` et testez la.

Compléments d'informations :

Comme le montre la figure 1, le bonbon hexagonal en haut à gauche est situé dans le tableau « modèle » en ligne 0 et le numéro de son pixel le plus haut est donc Y_TABLEAU_JEU. Le tic-tac rouge situé en dessous est en ligne 1 et le numéro de son pixel le plus haut est

Y_TABLEAU_JEU + TAILLE_BONBON (Rappelez-vous que le pixel de coordonnées (0,0) correspond au coin supérieur gauche de la fenêtre).

Étape 7 – Fonction getColonne

C'est l'inverse de la fonction `getXPixel` : étant donné le numéro `x_pixel` du pixel le plus à gauche d'un bonbon, elle renvoie la colonne `col` à laquelle appartient le bonbon dans le tableau « modèle ». En utilisant l'équation (1), déduisez la formule permettant d'obtenir `col` en fonction de `x_pixel`. Utilisez cette formule pour réécrire la fonction `getColonne`. Testez votre fonction de la même manière que ce que vous avez fait pour les fonctions `getXPixel` et `getYPixel`.

Aide sur l'écriture de la fonction :

Attention : n'oubliez pas que ce que vous devez retourner est un entier (en informatique, un entier est différent d'un nombre réel !).

Utilité de la fonction :

Cette fonction est utile quand l'utilisateur clique sur un bonbon. Dans ce cas, le programme doit déterminer à quel bonbon cela correspond dans le tableau « modèle ». Or, l'environnement graphique ne transmet que la position en pixels (`x_pixel`,`y_pixel`) de la souris au moment du clic. Il faut donc transformer cette information en une colonne (`col`) et une ligne (`lig`) dans l'espace de jeu (le tableau « modèle »).

Étape 8 – Fonction getLigne

C'est l'inverse de la fonction `getYPixel` : étant donné le numéro `y_pixel` du pixel le plus haut d'un bonbon, elle renvoie la ligne `lig` à laquelle appartient le bonbon dans le tableau « modèle ». En utilisant l'équation (2), déduisez la formule permettant d'obtenir `lig` en fonction de `y_pixel`. Utilisez cette formule pour réécrire la fonction `getLigne`. Testez la.