(version 16/10/2021)

Notice d'utilisation de la bibliothèque graphique librapythk

La `libgrapythk` est une librairie "*maison*" qui va vous permettre de gérer des graphismes sans nécessiter de vous plonger dans le module `tkinter`. Elle est en fait une interface simple entre vous et ce module `tkinter` qui impose un certain apprentissage.

1 Ouverture de session graphique et objet graphique

La première méthode (c'est à dire fonction) à utiliser pour pouvoir lancer la `libgrapythk` est `ouvrirFenetre(x,y)`, où `x` indique la largeur en pixels de la fenêtre graphique, et où `y` indique sa hauteur. Exemple de lancement d'une session graphique :

```
g = ouvrirFenetre(800,600)
```

On a ici créé une fenêtre de 800 pixels de large sur 600 pixels de haut.
IMPORTANT : le point de coordonnées `(0,0)` se trouve toujours en haut à gauche de la fenêtre.

L'appel à `ouvrirFenetre` renvoie un `ObjetGraphique` (ici `g`). C'est une variable que vous devez conserver et utiliser tout au long de votre programme. Elle vous permettra de lancer les méthodes graphiques qui sont présentées ci-dessous.

2 Méthodes de `ObjetGraphique`

À partir du moment où vous disposez d'un `ObjetGraphique`, vous allez pouvoir créez et manipuler toutes sortes de figures géométriques, textes, images dans votre fenêtre graphique. Voici les méthodes permettant de créer ces figures.

Création de figures géométriques, images, textes

`dessinerRectangle(x, y, l, h, col)`

Crée un rectangle plein, dont le coin supérieur gauche se trouve en `(x,y)`, dont la largeur est `l`, la hauteur `h` et la couleur `col` (voir la section dédiée aux couleurs).

IMPORTANT : cette méthode, ainsi que toutes les méthodes qui créent des figures géométriques, renvoie un objet. Vous pouvez récupérer cet objet dans une variable, ce qui vous permettra ensuite de le modifier, le déplacer, le supprimer, ou bien ignorer cet objet si vous pensez ne plus en avoir besoin ultérieurement.

`dessinerLigne(x, y, x2, y2, col)`

Cette méthode dessine une ligne entre le point `(x,y)` et le point `(x2, y2)`, de couleur `col`.

`dessinerCercle(x, y, r, col)`

Dessine un cercle de centre `(x,y)` et de rayon `r`, de couleur `col`.

`dessinerDisque(x, y, r, col)`

Dessine un disque de centre `(x,y)` et de rayon `r`, de couleur `col`.

`changerPixel(x, y, col)`

Dessine un pixel de coordonnées `(x,y)` et de couleur `col`.

`afficherTexte(txt, x, y, col, sizefont)`

Écrit un texte `txt` en position `(x,y)`, de couleur `col` (blanc par défaut) et de taille `sizefont` (18 par défaut).

`afficherImage(x, y, filename)`

Affiche une image en position `(x,y)`, provenant du fichier `filename`. Le fichier doit être précisé **avec son chemin relatif au script python**. Cette méthode accepte les principaux formats bruts (`PNG, BMP`) ou compressés (`JPG, GIF`).

Méthodes de modification d'objets existants

Les méthodes qui suivent permettent de modifier un objet existant : changer ses caractéristiques (couleur, texte), le déplacer ou le supprimer. Pour ce faire, vous devez avoir conservé une référence à l'objet créé, au moment de sa création. Voir exemple ci-dessous.

`deplacer(obj, x, y)`

Permet de déplacer un objet `obj` vers la position `(x,y)`.
Exemple :

```
c = g.dessinerCercle(800,600,10,"pink") # c contient une rfrence au cercle cr
# plus tard dans le programme...
g.deplacer(c, 10, 10) # ...on peut dplacer le cercle en utilisant cette rfrence
```

`supprimer(obj)`

Supprime l'objet `obj`.

`changerCouleur(obj, col)`

Change la couleur de l'objet `obj` en `col`.

`changerTexte(obj, txt)`

Change le texte de l'objet `obj` (nécessairement un objet texte) en `txt`.

Gestion des événements

On appelle *événement* une interaction entre l'utilisateur et le programme : clic souris, appui touche clavier, déplacement souris.

`recupererTouche()`

Permet de récupérer la dernière touche pressée au clavier. un objet `obj` vers la position `(x,y)`. La variable renvoyée est une *string* qui contient la description de la touche : `"a", "b", "c",`... Elle peut également contenir le nom de certaines touches spéciales (curseurs, touche de fonction...) : `"Right", "Left", "Up", "Down",`...voir la section dédiée aux touches clavier. La variable contient `None` si aucune touche n'a été presséei depuis la dernière récupération de touche.

`recupererClic(obj)`

Permet de récupérer la position du dernier clic gauche de souris. La variable renvoyée est un *tuple* qui contient les coordonnées `(x,y)` du clic, ou `None` si aucun clic n'a eu lieu depuis la dernière récupération de position.

`recupererPosition()`

Permet de récupérer la dernière position de souris suite à un déplacement de souris. La variable renvoyée est un *tuple* qui contient les coordonnées `(x,y)` de la souris, ou `(0,0)` si aucun déplacement n'a eu lieu depuis le lancement du programme.

Autres fonctions

`actualiser()`

Cette méothode, placée après une méthode graphique, force le rafraîchissment. À utiliser avec modération au risque de ralentir votre programme.

IMPORTANT : Lorsqu'on crée ou qu'on modifie un objet graphique, il n'est pas modifié immédiatement à l'écran. Le système de gestion graphique attend un certain temps pour effectuer ce rafraîchissement, afin de réunir plusieurs modifications et réduire ainsi le nombre de rafraîchissements, opérations assez coûteuses.
Si vous dessinez un simple objet graphique, vous n'aurez donc pas besoin de forcer un rafraîchissement de l'écran. Mais lorsqu'on demande de nombreuses modifications, comme par exemple lorsqu'on déplace de nombreux objets simultanément, cela peut devenir nécessaire.
Si vous ne voyez pas à l'écran les résultats des fonctions graphiques que vous avez appelées, essayez un `actualiser()` !

`pause(sec)`

Parfois, le programme crée se déroule trop rapidement et nécessite d'être ralenti. Cette méthode permet de forcer une pause, d'une durée de `sec` secondes. `sec` est un flottant, on peut donc demander une pause en dixièmes, centièmes, millièmes de secondes. Par défaut, cette méthode crée une pause d'une demi milliseconde.

`fermerFenetre()`

Ferme la fenêtre graphique.

Les couleurs

Il existe deux façons d'indiquer une couleur :

- par son code RVB sous la forme hexadécimale : `#rrvvbb` où `rr,vv,bb` sont les composantes rouges, vertes et bleues de la couleur souhaitée. Exemple : `#ff0000` indique le rouge.
- par une *string* prédéfinie. Les couleurs `'white', 'black', 'red', 'green', 'blue', 'cyan', 'yellow'`, et `'magenta'` sont toujours disponibles. De nombreuses autres couleurs le sont également, en fonction de la configuration locale de votre ordinateur. Faites des tentatives : `"pink", "gold"`,...

Les touches clavier

`Return`	La touche Entrée
`space`	La barre espace
`Tab`	La touche de Tabulation, Tab
`Up`	↑
`Down`	↓
`Left`	←
`Right`	→
`Alt_L`	La touche Alt située à gauche.
`Alt_R`	La touche Alt située à droite.
`Control_L`	La touche Ctrl de gauche
`Control_R`	La touche Ctrl de droite
`Shift_L`	La touche Maj de gauche
`Shift_R`	La touche Maj de droite
`Caps_Lock`	Verr Maj
`Delete`	Suppr
`BackSpace`	La touche Retour Arrière
`Home`	Début
`End`	Fin
`Insert`	Inser
`Escape`	Echap
`F1`	La touche fonction F1
`F2`	La touche fonction F2
`Next`	PageDown
`Prior`	PageUp
`Pause`	Pause
`Num_Lock`	Verr Num
`Print`	ImprÉcran
`KP_0`	0 sur le clavier numérique
`KP_1`	1 sur le clavier numérique
`KP_Up`	↑ sur le clavier numérique
`KP_Down`	↓ sur le clavier numérique
`KP_Left`	← sur le clavier numérique
`KP_Right`	→ sur le clavier numérique
`KP_Add`	+ sur le clavier numérique
`KP_Multiply`	× sur le clavier numérique
`KP_Subtract`	- sur le clavier numérique
`KP_Divide`	/ sur le clavier numérique
`KP_Next`	PageDown sur le clavier numérique
`KP_Prior`	PageUp sur le clavier numérique
`KP_Decimal`	Symbole de la ponctuation décimale (,) sur le clavier numérique
`KP_Delete`	Suppr sur le clavier numérique
`KP_End`	Fin sur le clavier numérique
`KP_Enter`	Entrée sur le clavier numérique
`KP_Home`	Début sur le clavier numérique
`KP_Insert`	Insert sur le clavier numérique