Prog – Algo – S1

TP 08 – Programmation graphique

L'objectif de cet TP est de programmer une application basique de tracés géométriques grâce au module **tkinter** de python.

A. Éléments graphiques de l'interface finale

Voici, au terme du TP, ce que sera l'interface graphique :

- un composant ***Canvas*** où a été tracée une grille et qui sera associé, via le clic gauche, à deux actions à programmer :
  - le clic gauche place le point A (couleur rouge ci-dessous) ;
  - le clic droit place le point B (couleur vert ci-dessous) .

- des composants ***Button*** associés à diverses actions :
  - tracer la droite (AB) ;
  - tracer le cercle de centre A et passant par B ;
  - effacer les dessins.

- des composants ***Label*** qui donneront deux informations :
  - les coordonnées de A ;
  - les coordonnées de B.

Voici une capture de l'interface souhaitée :

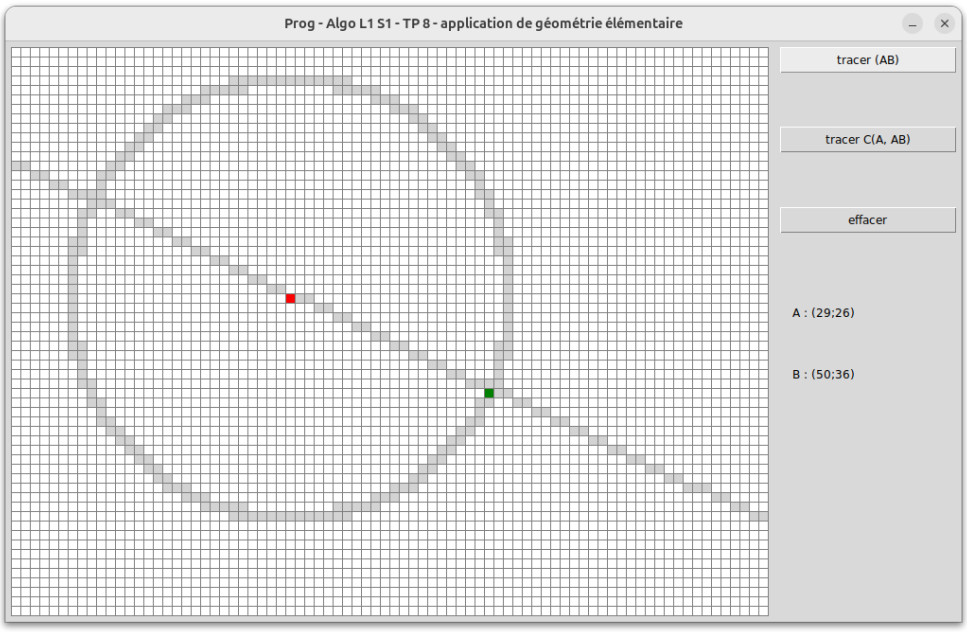

figure 1 : l'interface graphique de l'application

B. Environnement de programmation

Mettez en place l'environnement de programmation :

1. Créez un répertoire **TP8** dans lequel vous copiez le fichier **tp8.py** du cours **Celene** ;
2. Ouvrez **tp8.py** avec **Thonny**, et exécutez le programme. Si tout se passe bien, votre interface (sans grille et statique pour le moment) est créée.

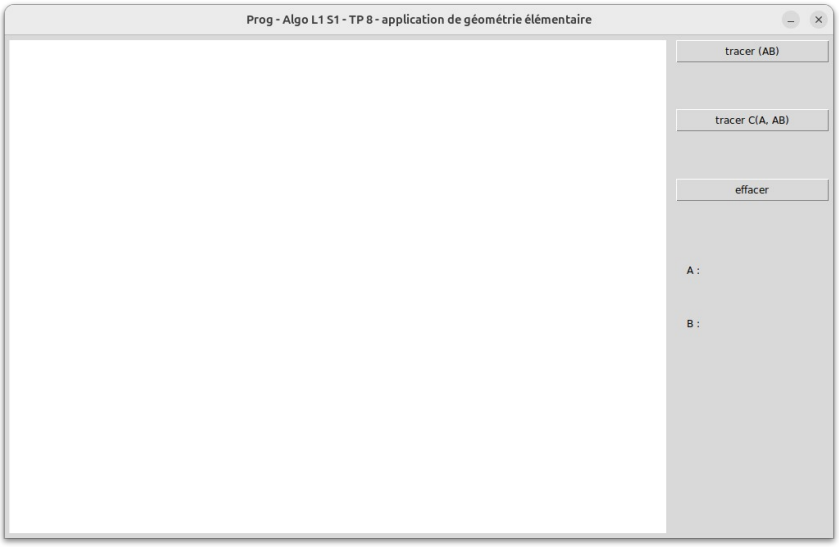

figure 2 : l'interface graphique initiale

C. Programmation des premières fonctions

La méthode **tracer_rectangle** vous est fournie.

```
def tracer_rectangle(X,Y,couleur) :
    x1 = 1 + 10 * X
    y1 = 1 + 10 * Y
    x2 = x1 + 10
    y2 = y1 + 10
    ecran.create_rectangle(x1, y1, x2, y2, outline = couleurTraits, fill = couleur)
```

Elle permet, sur le **Canvas** nommé **ecran**, de créer un rectangle dont les coins supérieur gauche et inférieur droit sont définis par les variables locales x1, y1 et x2, y2. Ceci peut se faire par l'appel à la méthode **create_rectangle** qui peut être appelée par l'objet **ecran** de type **Canvas**.

Le paramètre **outline** définit la couleur de tracé des contours du rectangle, le paramètre **fill** définit la couleur de remplissage du rectangle (ici passée en paramètre).

1. Codez la méthode **initialiser_ecran()** qui utilisera la méthode **tracer_rectangle** pour construire une grille complète (voir figure 1). Il y a en tout 60 lignes et 80 colonnes sur la grille finale. Il faudra passer la valeur adaptée pour le paramètre de couleur de remplissage.

2. Codez la méthode **placerA(event)** qui récupère les coordonnées du clic gauche, les utilise pour mettre à jour les variables globales **xA** et **yA** et enfin appelle la méthode **tracer_rectangle** avec comme paramètres xA, yA et la couleur prévue pour le point A.

   Le calcul de X et de Y permettra concrètement de colorier le carré de la grille où a eu lieu le clic.

   Quelques remarques importantes :

   - Quand une méthode doit modifier une variable du programme principal, cette variable doit être déclarée « global » sinon python créera une variable locale de même nom qui sera sans lien avec la variable globale ;

   - Tout clic gauche doit commencer par « effacer » la position précédente de A. Ceci se fera aussi par un appel à **tracer_rectangle**, avec les valeurs non modifiées de **xA** et **yA**, et la couleur **couleurVide ;**

   - Les coordonnées du clic sont récupérées grâce à l'objet **event** passé en paramètre. Cet « événement » a notamment comme attributs **event.x** et **event.y** qui sont les coordonnées du clic.

3. Codez de même la méthode **placerB(event)** et testez les deux méthodes.

   Vérifiez aussi que le bouton effacer est maintenant opérationnel. Au fait, pourquoi ?

4. Améliorez la méthode placerA(event) pour qu'elle mette à jour le Label **labelA** : le nouveau texte sera de la fome "A (… ; …)". Ainsi l'utilisateur sera renseigné sur la place exacte du nouveau point A.

   Remarque : on change le texte d'un Label **lab** par l'instruction lab['text'] = "……."

5. Faites de même pour le Label **labelB**.

Programmation des fonctions de tracés

Il reste à programmer les fonctions utiles au tracé de la droite (AB) et du cercle de centre A et passant par B.

1. Codez la fonction **tracer_ligne_verticale(X)** qui permet de tracer la ligne passant par tous les carrés de coordonnées X et Y, Y variant de 0 à 59.

   Pour garder la position des points A et B :

   - si le point tracé est confondu avec A, alors la couleur utilisée est **couleurA ;**
   - si le point est confondu avec B, alors la couleur utilisée est **couleurB** ;
   - Dans tous les autres cas, la couleur utilisée est **couleurTraits**.

2. Toujours avec les mêmes précautions de couleur, codez cette fois la fonction **tracer_ligne_horizontale(Y)** qui permet de tracer la ligne passant par tous les carrés de coordonnées X et Y, X variant de 0 à 79.

3. Codez maintenant la fonction **tracer_ligne_oblique(x1, y1, x2, y2)** qui tracera, aussi bien que possible, la droite passant par les points de coordonnées P1(x1, y1) et P2(x2, y2). On partira du prérequis que ces deux points P2 et P2 ne sont ni sur la même horizontale, ni sur la même verticale.

   Remarques importantes :

   - Il va de soi que le tracé sera « pixellisé », autrement dit que le rendu visuel ne sera pas parfait. C'est justement le but : comment choisir le bon « pixel » ?
   - Il peut être bon de déterminer l'équation de la droite. Cela doit faire partie de votre culture de base.
   - La forme générale, pour une droite oblique, est **y = a * x + b**. Mais cette forme suggère que l'on fait varier x, et que l'on détermine alors le y.
   - Dans certains cas (vous le verrez en faisant des essais), il peut être nécessaire d'adopter le point de vue inverse : exprimer x en fonction de y par une formule comme **x = 1/a * (y – b)**.

4. Codez la fonction **tracerAB( )** qui trace la droite (AB) en scindant en trois cas : horizontale, verticale ou oblique. Vérifiez que tout fonctionne en actionnant le bouton correspondant à la fonction. Vous prendrez les mêmes précautions de couleur que pour les questions 1 et 2.

5. Codez enfin la fonction **tracerC( )** qui trace le cercle (toujours pixellisé) de centre A et passant par B. Pour cela, vous pourrez suivre la démarche dont voici les détails :

- calculer la distance r entre A et B ;

- pour toute valeur X entre 0 et 79 et toute valeur Y entre 0 et 59, calculer la distance d entre A et le point M(X ; Y) ;

- si les valeurs de r et d sont égales (ou très peu différentes, à vous de préciser cela, éventuellement avec une constante de valeur à définir), alors on trace le pixel correspondant à (X, Y). Par exemple, on peut convenir que r et d sont de valeur « à peu près » égales si abs(r – d) ≤ 0.75 ou une autre valeur.

Le dessin pourra paraître imparfait. Ce n'est pas grave. Cela peut arriver pour des rayons assez grands ou au contraire assez petits. Vous pourrez chercher à améliorer ceci en offrant, par exemple, la possibilité de changer la précision de la grille et donc d'augmenter le nombre de pixels…