Cégep Limoilou

# TP 3

**420-SF3-RE**

**Consignes :**

- Ce travail doit se réaliser seul.
- Toute utilisation d'une IA doit être ciblée et ponctuelle, faire l'objet d'une citation claire, et être accompagnée d'une compréhension détaillée de chaque instruction

**Remise :**

- Le projet complet sur LEA (sans son venv trop volumineux).

**Date de remise :** mercredi 26 novembre à minuit
**Objectif** : Implémenter les fonctionnalités demandées dans l'application proposée

**Objectifs** :

- Gérer les évènements de souris et de clavier
- Utiliser le framework NetworkX
- Utiliser un QThread pour gérer un long traitement

**Architecture** :

Votre programme doit respecter l'architecture MVC présentée en classe et proposée dans l'application fournie.
Vous disposez d'une application basique qui permet de générer et d'afficher un graphe pondéré (non-orienté).
Remarquez dans la classe de modèle les deux paramètres de la génération, le nombre de sommets et la probabilité d'avoir une arête entre 2 sommets qui caractérisent le type de graphe généré.

**Fonctionnalités à implémenter :**

1) Implémenter **la création d'un nouveau nœud** dont le nom peut être le premier entier disponible, la seule condition est qu'il soit unique. Celle-ci doit se faire en cliquant dans la vue, le nœud doit être placé approximativement à la place ou on a cliqué.
   Attention, la position du `QMouseEvent` récupéré doit d'abord être converti en position `Matplotlib`, la méthode faisant cette conversion vous est fournie : `GraphCanvas._convert_pos(…)`

2) Implémenter **la sélection d'un nœud**. Lorsqu'on clique sur un nœud dans le canvas, celui-ci doit apparaitre comme sélectionné (changement de couleur). Attention le nœud sélectionné doit être stocké dans le modèle!

3) Implémenter **l'effacement d'un nœud sélectionné**, lors de l'appui sur la touche DEL. Les arêtes rejoignant ce nœud doivent être effacées également.

4) Implémenter **le déplacement d'un nœud sélectionné.** Lorsqu'on presse sur le bouton gauche de la souris sur un nœud on commence son déplacement. I au relâchement on a fait un mouvement significatif on va simplement déplacer le nœud.

5) Implémenter **l'ajout d'une arête par clique droit et drag jusqu'à un nouveau nœud**. Si le clique est relâché sur un nœud différent on ajoute une arête de poids 1 et arête est créée, sinon la création est abandonnée.

6) Implémenter **la sélection d'arête**. Utilisez vos connaissances de géométrie vectorielle pour calculer la distance entre le clic et le segment! Attention ce point est clairement le plus délicat! Il est basé sur des connaissances géométriques (distance point-segment)

7) Implémenter **l'effacement** de l'arête sélectionnée

8) Implémentez **la modification du poids de l'arête** sélectionnée, le choix de l'approche est libre

9) Implémentez la recherche du plus court chemin d'un sommet à un autre dans le graphe. Ajouter un mécanisme permettant de sélectionner le sommet de départ et d'arrivée, puis de lancer la recherche du plus court chemin dans un thread séparé. Ajouter une progressBar indéterminée montrant que le chemin est en cours de recherche. (Pour ralentir un eu le mécanisme et voir la progressBar, ajoutez un petit sleep avant la recherche au besoin.). Le plus court chemin doit s'afficher dans une couleur visible.

10) Implémenter le parcours de sommet. L'idée est simplement de parcourir les sommets du graphe et de les colorer en ajoutant un sleep(1) entre chaque nœud. Le déclenchement du parcours doit se faire sur l'appui de la touche « p », et la progression doit s'afficher dans une progress bar deteminée.

**Grille de correction approximative :**

| |
|---|
| Fonctionnalités 1 à 5 : 40% |
| Fonctionnalités 6 à 8 : 10% |
| Fonctionnalité 9 :15% |
| Fonctionnalité 10 :15% |
| Qualité globale : 10% |
| Qualité code et architecture : 10% |