Licence Informatique - 2ème Année
Programmation Orientée Objet 1

Rapport : Projet Graphe

Auteurs :

Mathis PIAULT
Matthieu COMME

Année universitaire 2024-2025

SOMMAIRE

I. Introduction

Ce projet a été réalisé dans le cadre de notre apprentissage du Java. Il consiste à développer une application pour dessiner un graphe qui représente l'avancement d'un projet. Chaque tâche est un point du graphe (sommet). Les liens entre les tâches (arcs) indiquent l'ordre dans lequel elles doivent être réalisées. Un arc affiche aussi la durée de la tâche précédente.

L'utilisateur peut créer, modifier, déplacer ou supprimer des sommets et des arcs. Il peut aussi enregistrer son graphe et le recharger plus tard. Seuls AWT et Swing sont autorisés pour la partie graphique.

Ce rapport présente d'abord la conception de l'application. Ensuite, il détaille les choix techniques et les classes principales. Enfin, une analyse du travail réalisé et une prise de recul concluent le rapport.

II. Conception et Implémentation

A. Modélisation

Un sommet représente une tâche dans le projet. Il contient des informations comme son nom, sa position et une description. A chaque sommet est liée une liste de ses successeurs, ce sont les tâches qui viennent après lui.

Un arc représente le lien de précédence entre deux tâches. Il relie deux sommets s1 et s2, et contient un poids qui est la durée de la tâche à partir de s1. Par défaut, les arcs depuis le "Début" ont une durée nulle.

Le graphe est représenté par une liste de Sommet, qui se met à jour à chaque création de Sommet. Il existe aussi les deux sommets immuables Début et Fin. Les relations de précédence sont actualisées au moindre ajout ou suppression d'arc. Enfin, nous verrons l'aspect graphique dans la prochaine partie.

B. Interface Graphique

L'application est composée d'une JFrame principale qui a un CardLayout, ce qui permet d'avoir la page de démarrage puis de switcher à celle de dessin de graphe. La taille de la fenêtre est paramétrée pour prendre tout l'écran.

Dans la fenêtre principale de l'application, on a deux JPanel, GrapheWindow et LignePanel, un où il y a les boutons, et un autre ou il y a les arcs. En effet pour dessiner les arcs on a superposé un JPanel transparent au dessus de notre panel de base, ce qui à eu pour conséquence que tous les clics hors des boutons étaient captés par le second panel, c'est pourquoi on a dû passer par une interface pour que lors d'une action de l'utilisateur, un événement se produit dans la classe Graphe qui implémente l'interface.

Les tâches sont représentées par un bouton de la classe RoundBtn qui est une classe qui étend de JButton. On a fait ce choix pour avoir des boutons personnalisés. La position du sommet et du bouton sont les mêmes et se situe au centre du bouton.

On a choisi de passer par des boutons au lieu de passer par un dessin de cercle pour faciliter les actions, grâce à ça on pouvait directement ajouter des ActionListener aux boutons au lieu de devoir regarder si le clic et sur le bouton puis de faire ce qu'on doit faire, comme on a fait pour les arcs. Pour détecter un clic sur les arcs, on a dû regarder la distance entre le clic et le segment qui constitue l'arc, donc on a dû utiliser des formules mathématiques pour cela. Ce qui est plus complexe que de juste écouter un clic sur un bouton.

C. Sauvegarder et Charger

L'application permet d'enregistrer un graphe dans un fichier pour le recharger plus tard. La sauvegarde se fait dans un fichier .ser grâce à la sérialisation Java. Cette technique permet de transformer un objet afin de le sauvegarder, puis de le recréer.

La classe SaveFileChooser, qui hérite de JFileChooser, affiche une fenêtre pour que l'utilisateur choisisse l'emplacement du fichier. Ensuite, elle récupère la liste des sommets et la liste des arcs du graphe. Cette classe utilise un ObjectOutputStream pour écrire ces listes dans le fichier.

Pour charger le graphe enregistré en .ser, on utilise la classe LoadFileChooser.
En lisant le fichier, on vérifie s'il est bien valide. En effet, on s'assure que les objets lus soient bien des listes, puis si c'est une liste Sommet et une liste Arc.
Si tout est bon, on recrée à partir du fichier les objets, en recréant les boutons graphiques. Nous les avons définis "transient" car les composants Swing ne sont pas fait pour être sérialisés.

D. Pour aller plus loin…

Le calcul de la date au plus tôt permet de connaître à quelle date une tâche peut commencer au plus tôt, en fonction des tâches qui la précèdent.
Pour chaque sommet, on regarde les sommes de la date au plus tôt de ses prédécesseurs et le poids de l'arc les reliant. La somme maximale devient la date au plus tôt du sommet.

Pour les calculer pour le graphe entier, on crée une file et on y insère le sommet "Début".
Puis, tant que la file n'est pas vide, on calcule la date au plus tôt de la tête de file en la défilant, et on enfile tous ses successeurs.

Le calcul de la date au plus tard donne la dernière date possible pour commencer une tâche sans retarder le projet.

Pour chaque sommet, on regarde les soustractions de la date au plus tard de ses succcesseurs et le poids de l'arc les reliant. La différence minimale devient la date au plus tard du sommet.
Même principe que précédemment pour le graphe entier, sauf qu'on commence par le sommet "Fin" et qu'on enfile tous ses prédécesseurs.

Le chemin critique est composé de tâche qui, si elles sont retardées, tout le projet le sera aussi. Pour l'obtenir, on compare la date au plus tôt et au plus tard de chaque tâche. Si elles sont égales, la tâche est critique.

III. Analyse et Prise de Recul

A. Bilan fonctionnel

Fonctionnalités demandées	Réalisée	Commentaire
Placer et nommer des sommets	Oui	Mode 1 ouvre un JDialog (ech pour annuler)
Tracer des arcs entre deux sommets	Oui	Mode 2 crée un arc
Modifier un sommet	Oui	Double-clic sur le sommet ouvre un panel pour le modifier
Déplacer un sommet et mettre à jour l'affichage	Oui	En mode 0: drag and drop déplace le sommet et redessine les arcs
Supprimer un sommet ou un arc	Oui	Mode 3 supprime un arc ou un sommet, mettant à jour le graphe
Enregistrer le graphe réalisé	Oui	via SaveFileChooser
Charger un graphe précédemment enregistré	Oui	via LoadFileChooser

Fonctionnalités supplémentaires	Réalisée	Commentaire
Raccourcis clavier	Oui	Entrée pour valider saisie. & / é / " / ' pour changer de mode
Calculer le chemin critique	Oui	S'actualise à chaque modification du graphe
Multi-sélection	Oui	Sélectionne plusieurs sommets dans une surface pour les déplacer
Zoomer	Oui	Avec la molette de la souris
Menu de démarrage	Oui	Possibilité de charger directement un fichier ou de créer un nouveau graphe ou de quitter l'application

B. Difficultés et solutions

Nos premières difficultés ont été de savoir comment organiser le projet, en effet c'était notre premier projet en Java donc on ne savait pas vraiment comment ranger les fichiers. On avait donc commencé avec une organisation qui était à revoir. Maintenant on a 4 dossiers principaux. Assets, où il y a toutes les images, controller qui rassemble des sortes de listener, model, qui est toutes les classes composants le graphe en interne et enfin on a le dossier ui qui est tout ce qui est graphique.

Ensuite on a également eu un problème concernant les raccourcis claviers. En effet au début on les avait fait via des keyListener, or ces listener ne fonctionnent que lorsque l'objet qui possède le listener est focus, ce qui n'est pas le comportement souhaité. Effectivement ce n'était pas le bon moyen, car en faisant comme cela, les raccourcis ne fonctionnaient pas.
On a alors trouvé un autre moyen plus efficace pour faire des raccourcis clavier qui est les InputMap et ActionMap, qui associe une touche du clavier à une méthode, et cela temps que le panel qui contient ces classes est visible, on avait alors juste à les désactiver à certains moments puis les réactiver pour que tout fonctionne correctement.

Un autre bug notable était que parfois, on se retrouvait avec un bouton "mort" : on ne pouvait plus interagir avec. En fait, lorsqu'on le recréait ou le modifiait, on le plaçait pas dans la bonne couche de panel donc on l'a fix en le corrigeant le zindex.

C. Améliorations possibles

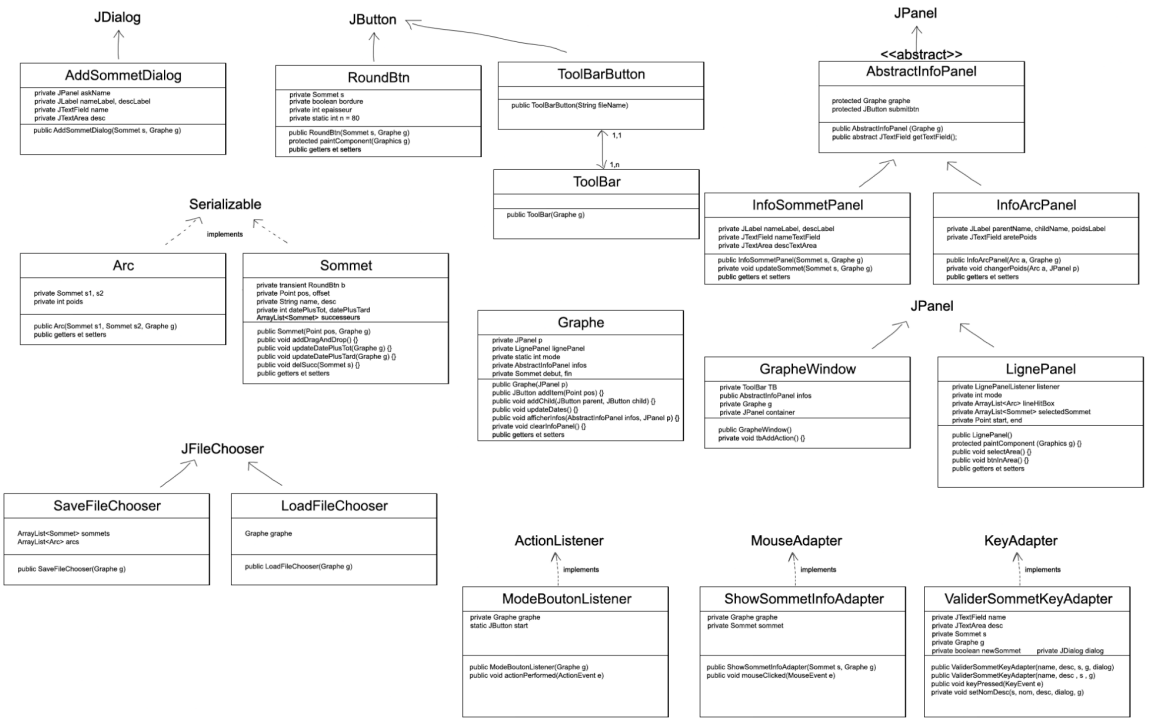
Voici une liste d'améliorations possibles si on avait eu plus de temps.

- Mieux séparer les classes : on a essayé un peu tard de dissocier l'aspect logique du côté graphique des classes.
- Même idée pour une meilleure encapsulation
- Bouton aide : mode d'emploi de l'appli, avec les raccourcis
- Personnalisation : changer couleur de sommets, arc, fond etc
- Interdire les arcs allant vers Début ou provenant de Fin : on a commencé mais ça générait des bugs donc on a commenté cette partie
- Arcs courbés

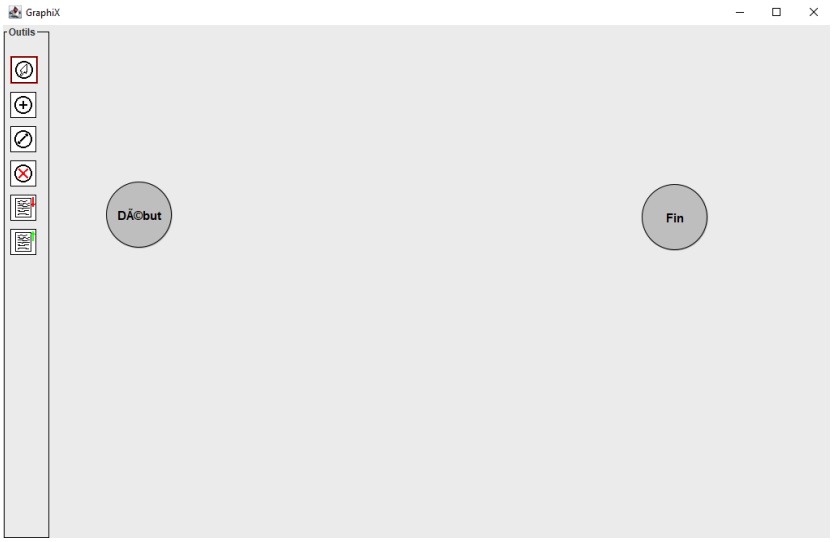
IV. Conclusion

L'application est satisfaisante, elle nous permet de faire ce qui est demandé et même plus.
Le projet nous a permis d'apprendre à travailler en équipe, à s'organiser et à nous améliorer dans notre maîtrise du JAVA.
En effet, en ayant passé plusieurs dizaines d'heures dessus, cela a amélioré nos compétences en programmation orientée objet. Ce projet nous a appris plusieurs fonctionnalités de ce langage.

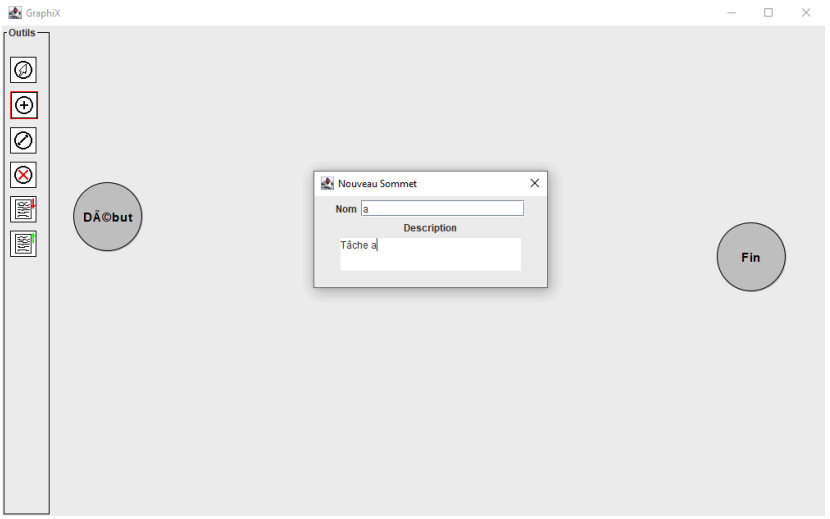
V. Annexes



Menu Principal

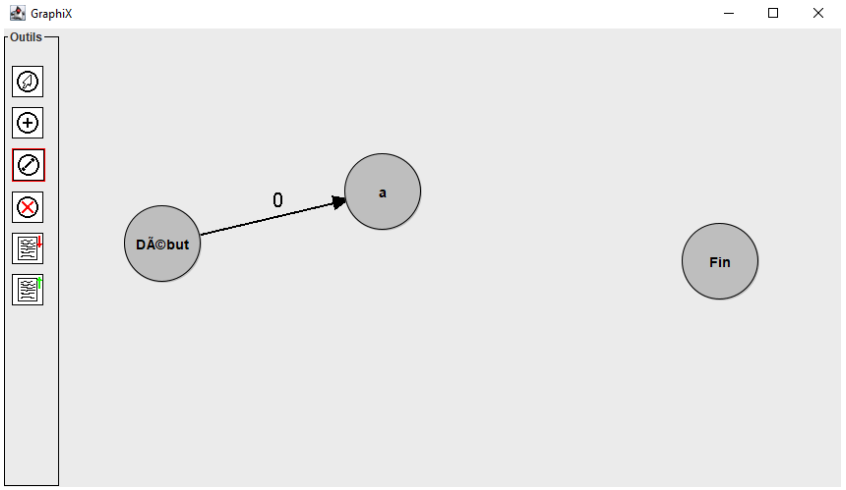


Nouveau graphe

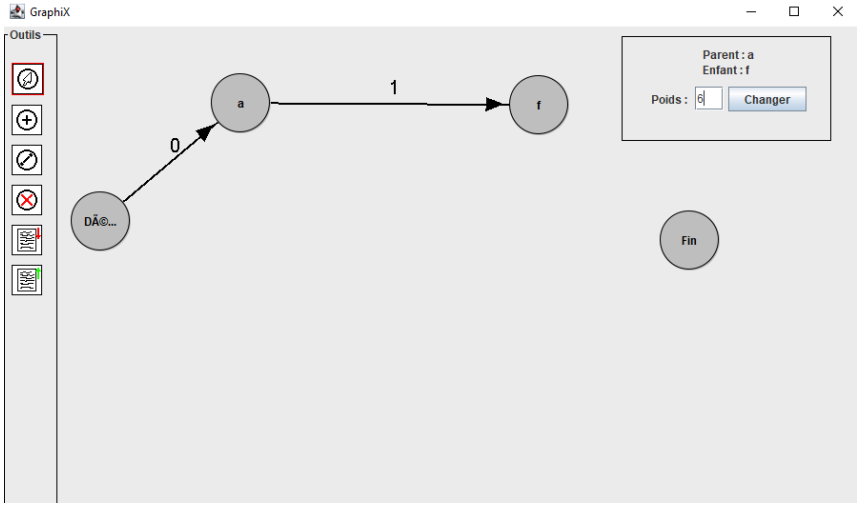


Créer un sommet

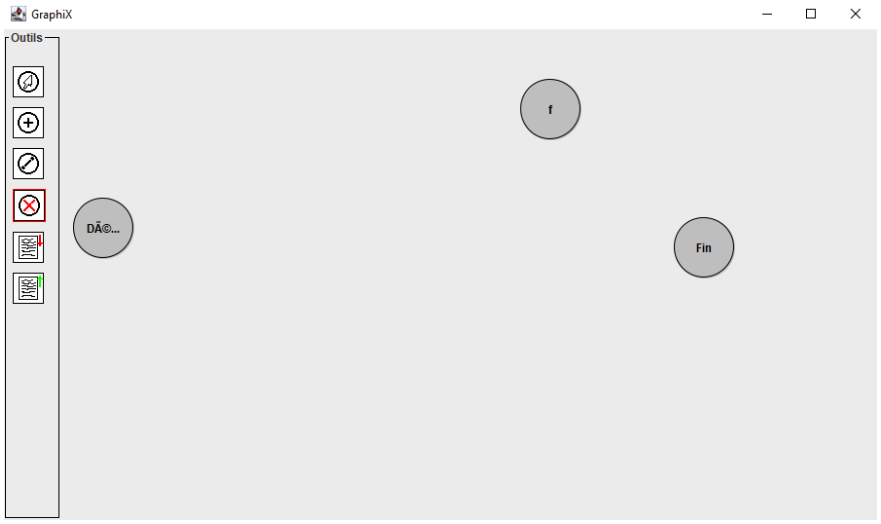
Modifier le sommet (double clic)



Créer un arc en cliquant d'abord sur Début puis sur a

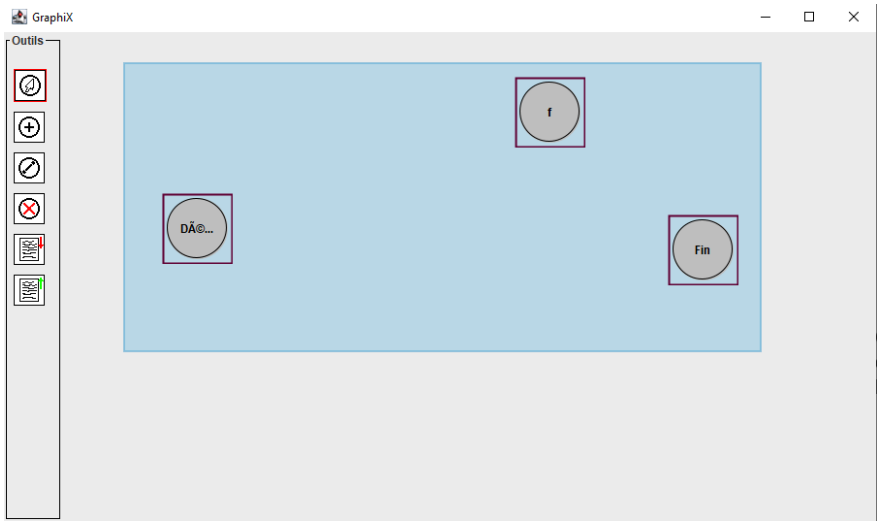


Modifier le poids d'une arête en double-cliquant dessus
Le poids des arêtes partant de Début est bloqué à 0

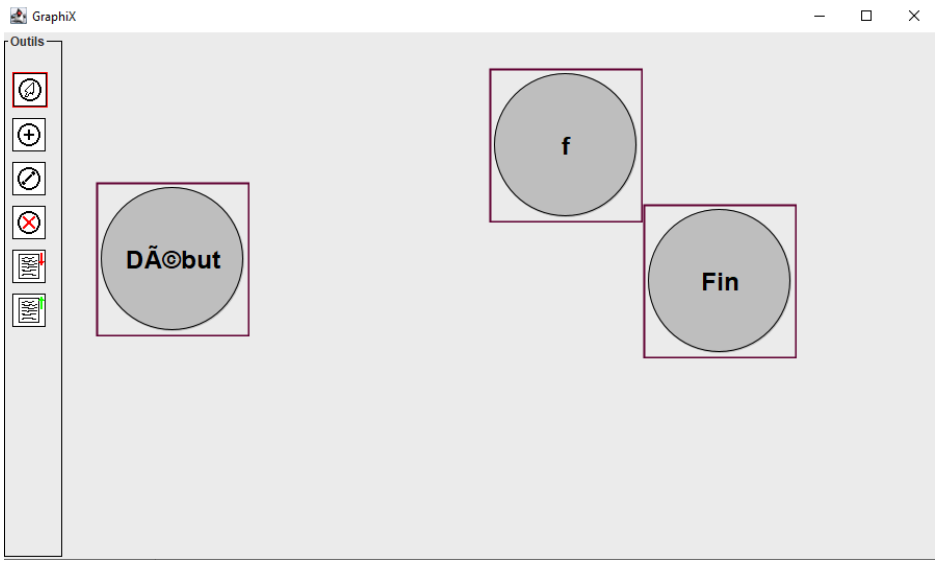


La suppression d'un sommet implique la suppression des arc qui lui sont liés.

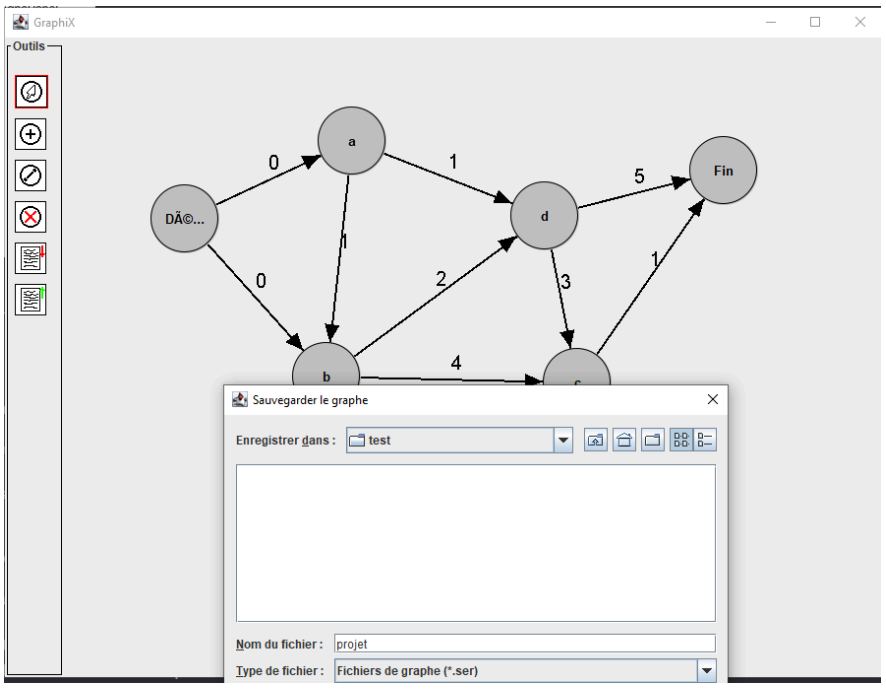
On ne peut pas supprimer Debut et Fin



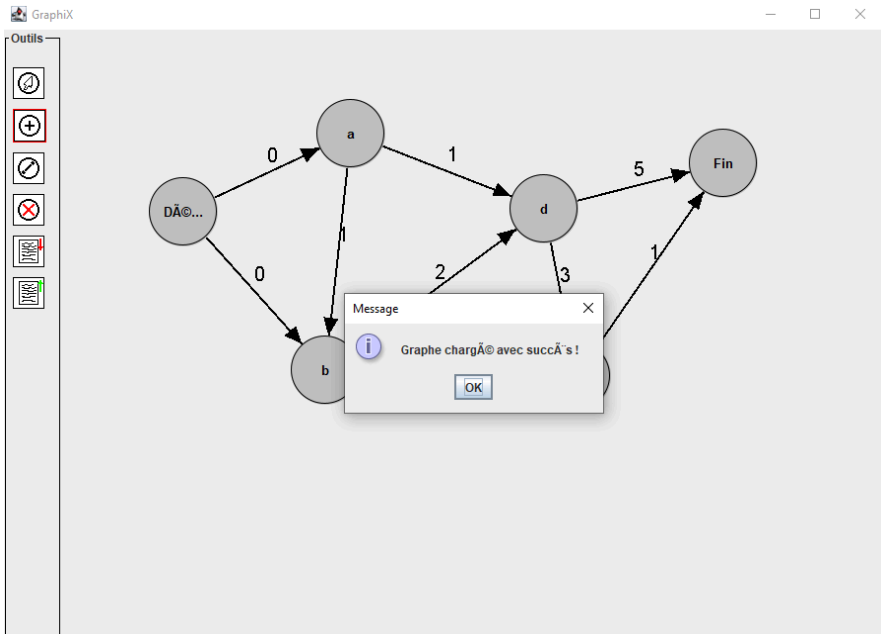
La multi-sélection permet de déplacer plusieurs sommets en une fois avec le drag and drop



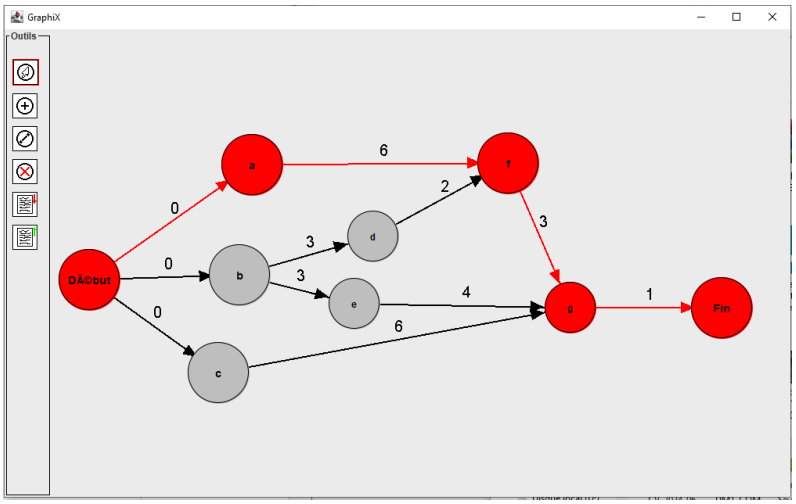
La molette de la souris permet de zoomer / dézoomer le graphe.



Sauvegarder un graphe avec le 5ème bouton



Charger un graphe avec le 6ème bouton



Graphe complet avec chemin critique en rouge