INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

INSAMaps

Making navigation simple

Table des matières

Par Guillaume SKMEKTALA, Timothée VIEGAS, Quoc TRAN, Navin RANGA

Encadré par M. VINDAS-YASSINE

ASINSA 2A - Groupe 98

Institut National des Sciences Appliquées de Lyon

I. Contexte du projet et objectifs

Le but de notre projet est de rendre le déplacement sur le campus de l'INSA Lyon plus simple et efficace afin de faciliter la vie des étudiants et surtout des nouveaux arrivants et intervenants.

Nous avons voulu créer un programme qui permet à un utilisateur de se déplacer sur le campus de l'INSA et à ses alentours. L'utilisateur peut saisir son point de départ A et son point d'arrivée B. En réponse, le programme fournira un ensemble d'itinéraires avec des caractéristiques associées tels que le temps, la distance, et le nombre de calories brûlées (puisqu'il s'agit de marche à pied).

Dès le lancement du programme, il y aura la carte du campus sur lequel on doit choisir notre localisation. Ensuite on choisit notre point de départ et notre destination et on lance la recherche. Le programme va alors dessiner sur la carte et afficher le chemin le plus court pour un tel trajet.

Pour notre programme on a choisi de représenter certains bâtiments et lieu prédéfini tel que: la résidence C et D, le Galilée, la Rotonde, les Humanités, la BMC, la direction de l'INSA, le bâtiment Louis Neel, le bâtiment d'Alembert, le bâtiment Pierre de Fermat.

Pour des précisions supplémentaires, veuillez consulter le cahier de charges (dossier « autres » de l'archive).

II. Description de l'IHM

L'interface graphique est constituée de manière à être simple d'utilisation est facilement compréhensible par n'importe qui (Voir Annexe A)

Lorsque le programme est exécuté, nous avons une fenêtre d'animation de début qui s'affiche et qui se ferme, faisant apparaître une carte du campus. Dans le cadre de notre projet, celle-ci est limitée au campus de l'INSA, c'est-à-dire par l'avenue Gaston Berger à l'Ouest, par l'IUT ainsi que par le cimetière militaire à l'Est, par le Boulevard Laurent Bonnevay au Nord, et par l'Avenue Albert Einstein au Sud.

En plus des bâtiments et des routes, la carte indique les voies de tramway, les parkings ainsi que les stations Velo'V. Sur cette carte est superposé un encadré contenant plusieurs éléments. Tout d'abord deux zones de textes dans lesquelles on entre les lieux de départ et d'arrivée, puis un bouton lançant le calcul de l'itinéraire, ainsi que trois étiquettes affichant la durée, la distance et les calories dépensées au cours du trajet. A droite de l'étiquette des calories, se trouve un logo de base de données. En cliquant dessus, on a accès à la base de données contenant toutes les salles, leur étage et leurs détails.

Lorsque le départ et l'arrivé sont choisis, les étiquettes de détails se mettant à jour et le parcours que vous devez faire est tracé sur la carte.

III. Description de l'algorithme

Tout au début du programme, une fenêtre d'animation avec des lettres est créée et disparaît dès que le mouvement des lettres est terminé.

Ensuite, est affiché notre carte et notre espace IHM. En cliquant notre lieu de départ sur le menu déroulant de départ, une instance Dijkstra est créée en collectant l'indice de la destination choisie dans le menu déroulant. En choisissant l'arrivée sur le menu déroulant des lieux d'arrivée, son indice est récupéré et à l'aide de l'algorithme de Dijkstra, on trouve la distance entre les deux points, le chemin qu'on doit effectuer ainsi que la durée et le nombre de calories. Une fois l'étiquette « LET'S GO » cliquée, ces caractéristiques vont se mettre à jour sur leurs étiquettes respectives.

Explication Algorithme de Dijkstra :

Dans cette partie nous allons parler de l'algorithme de Dijkstra, un algorithme qui permet de déterminer le chemin le plus court entre 2 sommets. Dans notre cas, notre graphe représente notre carte avec pour sommets les différents lieux (Résidence C et D, la BMC, la bâtiment Pierre de Fermat…). Entre chaque sommet on a relevé la distance à l'aide de Google Maps afin de constituer notre graphe. Le principe de l'algorithme est de considérer à chaque étape du voyage, la recherche du plus court chemin. Si l'on prend un exemple, on veut déterminer le plus court chemin pour aller de A à G selon le graphe suivant (voir figure 1 de Annexe B).

On représente ensuite dans un tableau les différents sommets en lignes et en colonnes les étapes de notre algorithme (voir figure 2 de Annexe B). Comme on démarre au point A on peut déjà écrire (voir figure 3 de Annexe B). Du village A on peut soit aller au village B soit au village C, on note donc A dans les colonnes B et C pour la 1ère étape, ainsi que le nombre de kilomètres total que nous avons parcouru depuis le début (voir figure 4 de Annexe B).

On remarque qu'ici le chemin le plus court est le chemin en B, on recopie donc la case B1 en B2. On ne va plus revenir au village B on peut donc barrer donc les cases suivantes de la colonne B. On part ensuite de B et on peut soit aller en D soit aller en F. Dans la D2 et dans F2 on peut écrire B puisque l'on vient de B. On écrit aussi la distance totale parcourue (voir figure 5 de Annexe B).

On regarde maintenant, parmi tous les chemins parcourus et on prend le plus court. Ici on voit que c'est le chemin partant de C. On recopie donc la case C1 dans la case C3. Comme précédemment on ne revient plus au sommet C donc on peut barrer toutes les cases suivantes de la colonne C. Du village C on peut aller soit au sommet D soit au village E, on écrit donc C dans la case D3 et E3 ainsi que la distance (voir figure 6 de Annexe B).

Ensuite le schéma est le même, on répète l'algorithme jusqu'à obtenir le chemin le plus court (voir figure 7, 8, 9 de Annexe B).

On voit donc ici que le chemin le plus court est de 6 kilomètres en tout. On veut maintenant retracer le chemin parcouru. On voit en G7 qu'on arrive au sommet G en venant du sommet D, on va ensuite dans la colonne D et on voit que du sommet D on venait de B,

et enfin de la colonne B on peut voir que l'on vient du sommet A. Le chemin final le plus court est donc A-B-D-G et fait 6 de distance.

Dans notre programme nous avons ainsi implémenté cet algorithme afin de trouver la distance la plus courte entre différents bâtiments.

IV. Diagramme UML du programme

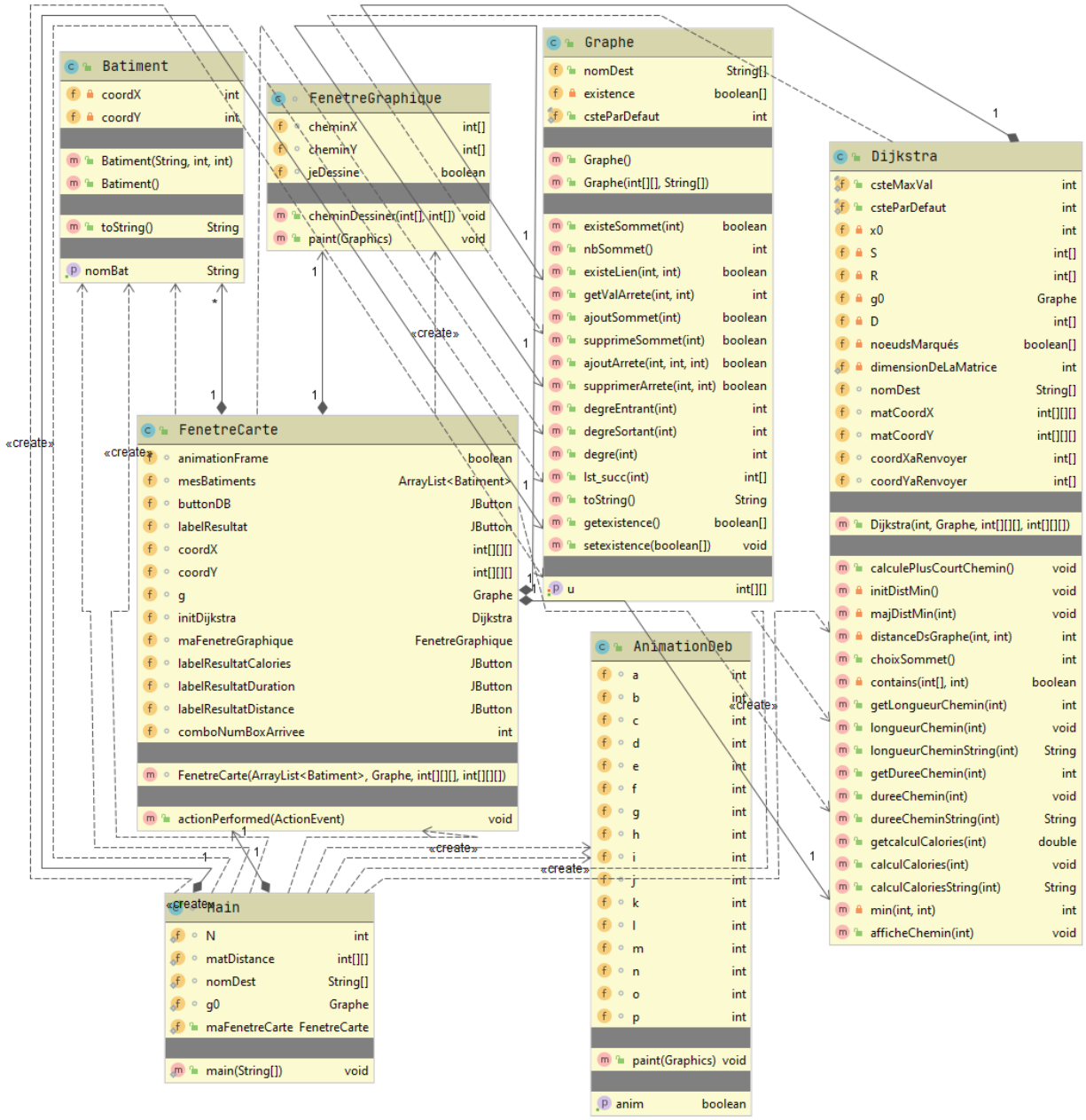

Figure 1 : Diagramme UML généré à l'aide de IntelliJ IDEA

Remarque : Ce diagramme est fourni avec une meilleure lisibilité dans « autres\uml.svg ».

V. Résultats et Discussions

Le programme créé permet bien à un utilisateur de se déplacer sur le campus de l'INSA. L'utilisateur peut rentrer sa destination de départ et sa destination d'arrivée. L'algorithme recherche le chemin le plus court sur le campus, calcule et affiche les caractéristiques associées. Les caractéristiques affichées sont la distance, la durée, et le nombre de calories dépensées. Un seul mode de transport a finalement été retenu, la marche. Les destinations prises en compte sont l'ensemble des bâtiments des étudiants de $2^{ème}$ année ainsi que les résidences.

L'objectif principal du programme est atteint, à savoir permettre à un utilisateur de se déplacer d'un point A à un point B sur le campus en lui communiquant la distance, la durée et le nombre de calories brûlées.

Nous avons été trop ambitieux dans notre programme, nous n'avons pas pu intégrer tout ce que nous voulions. Nous n'avons pas intégré de critères de recherche comme nous le voulions initialement. Les informations calculées et communiquées à l'utilisateur se limitent à la distance, la durée et le nombre de calories dépensées. Nous n'avons pas intégré le coût ou les émissions de CO_2 ou d'autres critères auxquels nous avions pensé. Nous voulions initialement intégrer 7 modes de transport disponibles mais nous nous sommes limités à la marche. Les destinations disponibles se limitent à l'ensemble des bâtiments et des résidences. L'affichage du chemin est parfois mal-alignée et l'algorithme de Dijkstra que nous avons implémenté échoue quand on entre deux bâtiments aux extrêmes du campus (résidence C à Pierre de Fermat par exemple).

Si nous avions été moins ambitieux, on aurait pu coder plus de fonctionnalités du cahier de charge. Car plusieurs séances ont été « perdues » en réfléchissant sur les manières dont on pouvait convertir le langage naturel en langage algorithmique.

VI. Suivi du projet et contribution des membres

Pour ce qui est de la participation, nous nous sommes équitablement réparti le travail. Chaque membre du groupe a pu intervenir sur à peu près chaque partie du programme et d'autant plus au moment de la réflexion sur les méthodes à implanter et de manière générale, la façon dont nous allons faire les choses.

Timothée:

10/03 - Mise en place de la fenêtre générale affichant la carte de l'INSA avec les bâtiments
17/03 - Création d'une méthode permettant de relever les coordonnées de la souris au sein de la fenêtre principale affichant la carte de l'INSA lors d'un clic
24/03 - Création de la base de données relevant la position (coordonnées x et coordonnés y) de chaque bâtiments sur la carte à l'aide de la méthode implémentée la semaine précédente qui relève les coordonnées lors d'un clic de souris
31/03 - Création d'une base de données des routes reliant chaque bâtiment avec tous les

autres. Ainsi entre 2 bâtiments on relève des points qui reliés entre eux permettent de visualiser la route entre les 2 bâtiments.
07/04 - Implémentation de la base de données dans notre algorithme et création d'un programme permettant de créer un rendu visuel des routes reliant les bâtiments entre eux.

Guillaume:

10/03 - Création de l'ihm permettant de visualiser le départ et l'arrivée
17/03 - Ajout du code permettant, après avoir appuyé sur un bouton, de faire apparaître la distance et la durée du trajet
24/03 - Implémentation de ce code au code d'affichage de l'image et d'affichage des coordonnées
31/03 - Résolution des problèmes d'affichage
07/04 - Tentative de création d'une ihm permettant de cliquer sur un bâtiment afin de le définir comme point de départ ou d'arrivée

Navin:

10/03 – Réflexions sur « comment convertir le langage naturel du projet en langage algorithmique ? »
17/03 - – Réflexions sur « comment convertir le langage naturel du projet en langage algorithmique ? » et premiers essais de simplification de la réalité des cartes (voir le fichier Publisher listes v0 bat et rues dans le dossier 'autres')
24/03 – Documentation sur l'algorithme de Dijkstra et code
31/03 – Mise en forme de la base des données et premiers essais de correspondances entre IHM, Base de Données
07/04 – Lien entre IHM, Base de Données, Graphique

Contribution des membres :

Guillaume	Timothée	Quoc	Navin
25%	25%		50%

VII.Bibliography

Utiliser l'algorithme de Dijkstra - PostBac :
https://www.youtube.com/watch?v=rHylCtXtdNs

Problèmes Techniques/Doutes :

https://stackoverflow.com/

VIII. Annexes

A. Annexes A

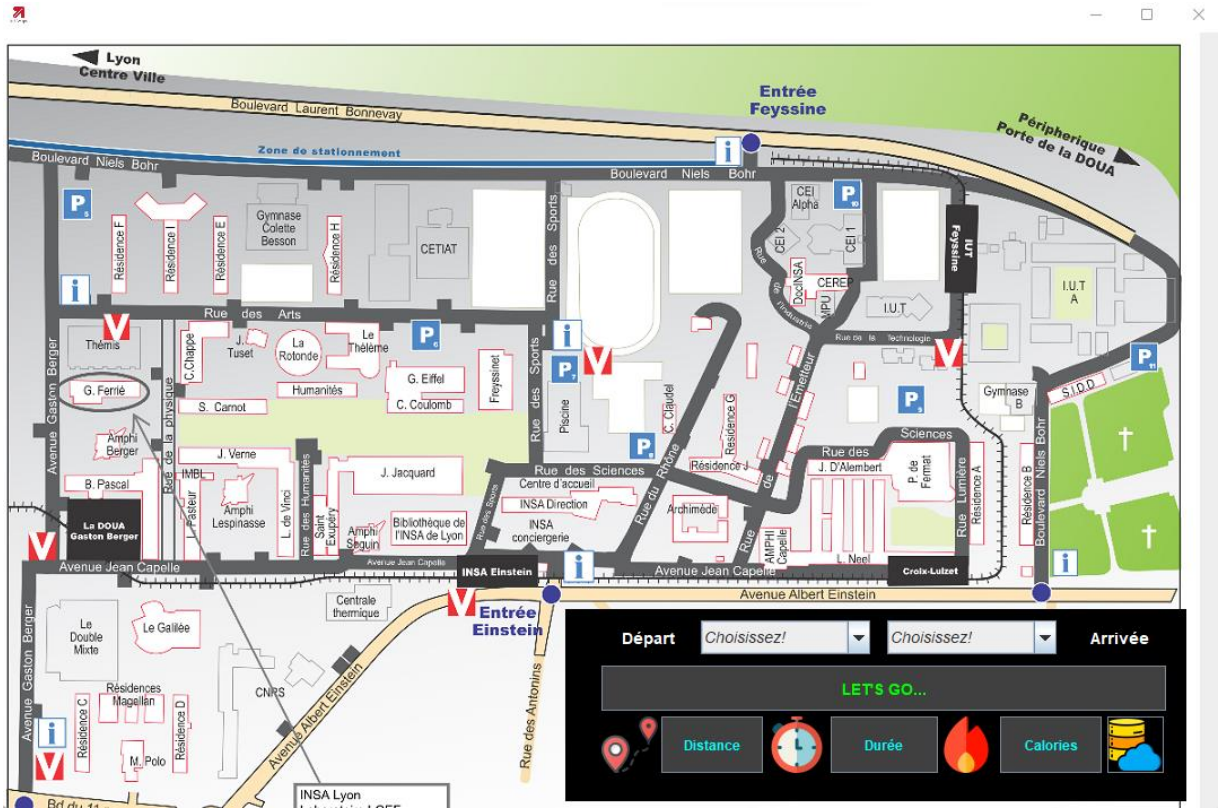

Figure 2 : Interface du programme

B. Annexes B

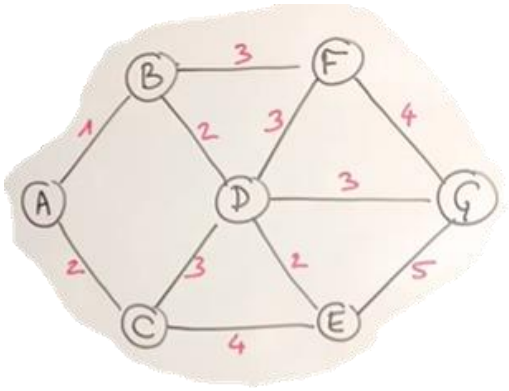

Figure 1

A	B	C	D	E	F	G	Etapes
							1
							2
							3
							4
							5
							6
							7

Figure 2

A	B	C	D	E	F	G	Etapes
0							1
X							2
X							3
X							4
X							5
X							6
X							7

Figure 3

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X							2
X							3
X							4
X							5
X							6
X							7

Figure 4

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X	1-A		3-B		4-B		2
X	X						3
X	X						4
X	X						5
X	X						6
X	X						7

Figure 5

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X	1-A		3-B		4-B		2
X	X	2-A	5-C	6-C			3
X	X	X					4
X	X	X					5
X	X	X					6
X	X	X					7

Figure 6

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X	1-A		3-B		4-B		2
X	X	2-A	5-C	6-C			3
X	X	X	3-B	5-D	6-D	6-D	4
X	X	X	X				5
X	X	X	X				6
X	X	X	X				7

Figure 7

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X	1-A		3-B		4-B		2
X	X	2-A	5-C	6-C			3
X	X	X	3-B	5-D	6-D	6-D	4
X	X	X	X		4-B	8-F	5
X	X	X	X		X		6
X	X	X	X		X		7

Figure 8

A	B	C	D	E	F	G	Etapes
0	1-A	2-A					1
X	1-A		3-B		4-B		2
X	X	2-A	5-C	6-C			3
X	X	X	3-B	5-D	6-D	6-D	4
X	X	X	X		4-B	8-F	5
X	X	X	X	5-D	X	10-E	6
X	X	X	X	X	X	6-D	7

Figure 9

Figure 3 : Illustration de l'algorithme de Dijkstra