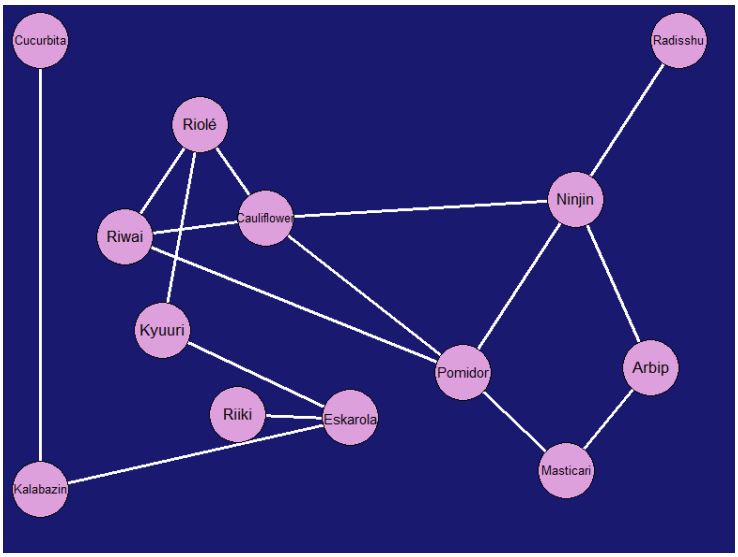
Projet ISN

Baccalauréat 2018

SSB : Swag Spatial Bus

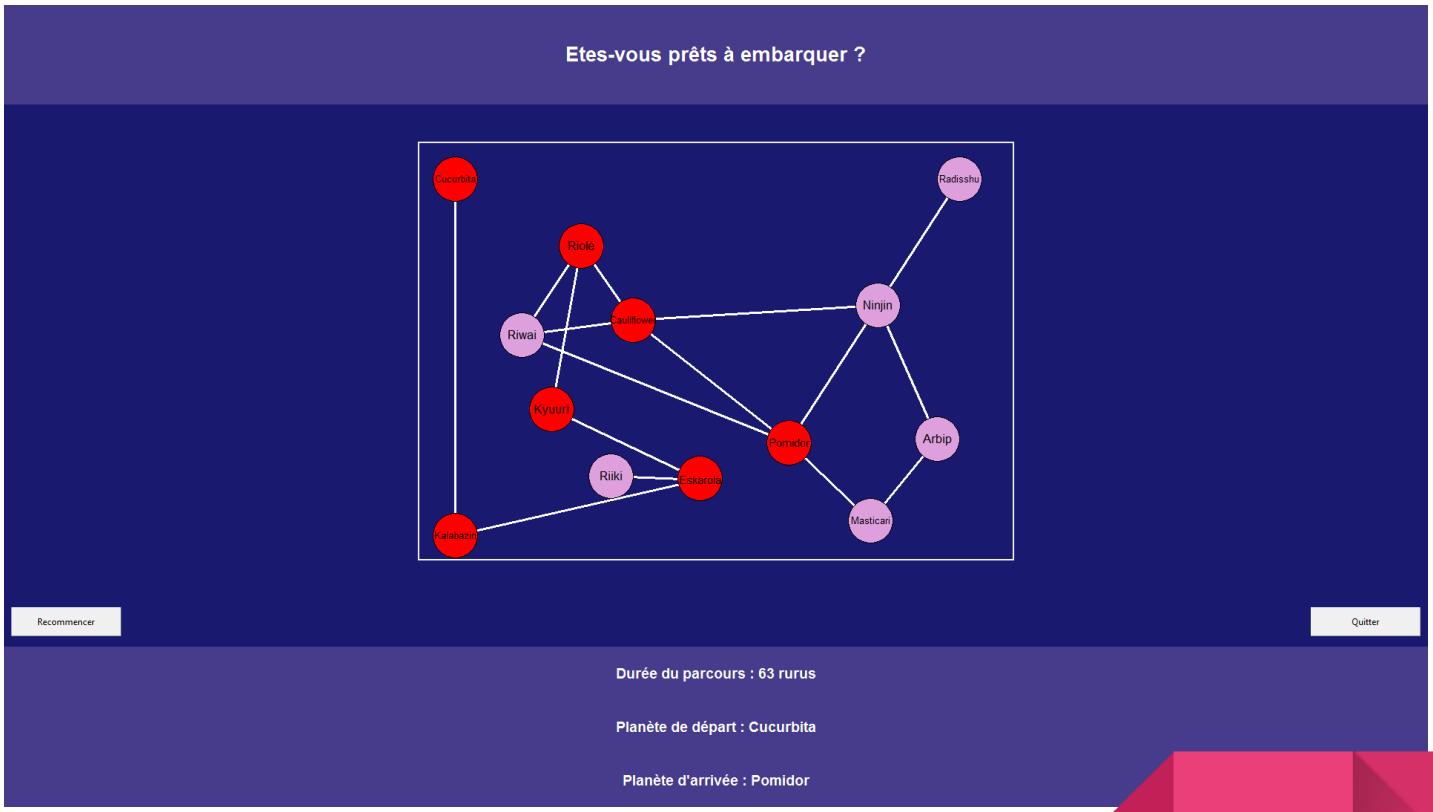
Julie DELISLE & Anne-Soline GUILBERT--LY



Présentation

Habitant dans un système solaire comprenant une bonne dizaine de planètes, nous avons décidé de créer un logiciel qui permet à toute forme de vie (même les bactéries) d'accéder à d'autres planètes que la leur. Notre projet se présente donc sous la forme d'une carte interactive. Vous pourrez choisir l'endroit où vous vous situez, puis la planète où vous aimeriez vous rendre. Notre logiciel indiquera ainsi par quelle planète le bus passera, ainsi que la durée du voyage. Cela vous évitera ainsi de vous perdre dans de nombreuses cartes incompréhensibles et d'arriver sereinement à destination.

Avant-goût de notre projet



Cahier des charges

Le but de notre projet est de créer une carte intéractive. Pour cela, nous voulions utiliser l'algorithme de Dijkstra et de créer une interface grâce à Tkinter. Nous nous sommes inspiré rapidement d'autres cartes interactives de bus/métro comme celui de Dunkerque (www.dkbus.com/index.asp).

Pour se faire, nous avons codé chacune sur un ordinateur séparément, comme nous avons divisé le travail. Nous avons toutes les deux utiliser du Python et nous nous sommes aidés de nombreux sites internet comme :

- Openclassroom (https://openclassrooms.com/)
- Cours en ligne par une professeur en CPGE (http://ressources.unisciel.fr/sillages/informatique/dijkstra/co/Dijkstra_1.html)
- Aide du professeur d'ISN, M. Vors
- pleins d'autres sites divers et variés

Répartition du travail

Quoi ?	Qui ?
Découverte principe de l'algorithme de Dijkstra	Julie & Anne-Soline
Création d'une carte spatiale avec invention des planètes, distance, voisins...	Julie & Anne-Soline
Mise en page du logiciel	Julie & Anne-Soline
Pseudo-code Dijkstra	Anne-Soline
Interface Tkinter	Julie
Algorithme Dijkstra	Anne-Soline
Base de données	Julie
Mise en commun	Julie & Anne-Soline

Moyen de communication

En dehors du lycée (cours et séances au CDI), nous avons utilisé Skype (avec un partage d'écran quand c'était nécessaire), nous nous sommes appelés, et nous nous sommes aussi vus chez l'une et l'autre.

Notre travail (au début)

Découverte du principe de l'algorithme de Dijkstra

Pour cette partie du travail, nous nous sommes vues chez l'une et l'autre.
Nous avons découvert l'algorithme de Dijkstra grâce au site "OpenClassrooms". Ensuite nous avons regardé des vidéos explicatives tout en nous entraînant avec des graphs inventés (un graph est simplement un ensemble de points, reliés entre-eux).

Nous avons aussi commencé à regarder comment marche un pseudo-code sur l'algorithme de Dijkstra.

Création d'une carte spatiale avec invention des planètes, durées, voisins...

Nous avons réalisé cette partie sur une heure de cours. Au début, nous avons fait une carte sur papier, puis nous l'avons créée avec Microsoft Office Word.

Ensuite, pendant un autre cours, nous avons "traduit" notre graph dans une base de données, mais nous nous sommes rendues compte plus tard que cela ne servirait à rien, parce qu'il est plus simple d'utiliser un dictionnaire au début de l'algorithme de Dijkstra.

Mise en page du logiciel

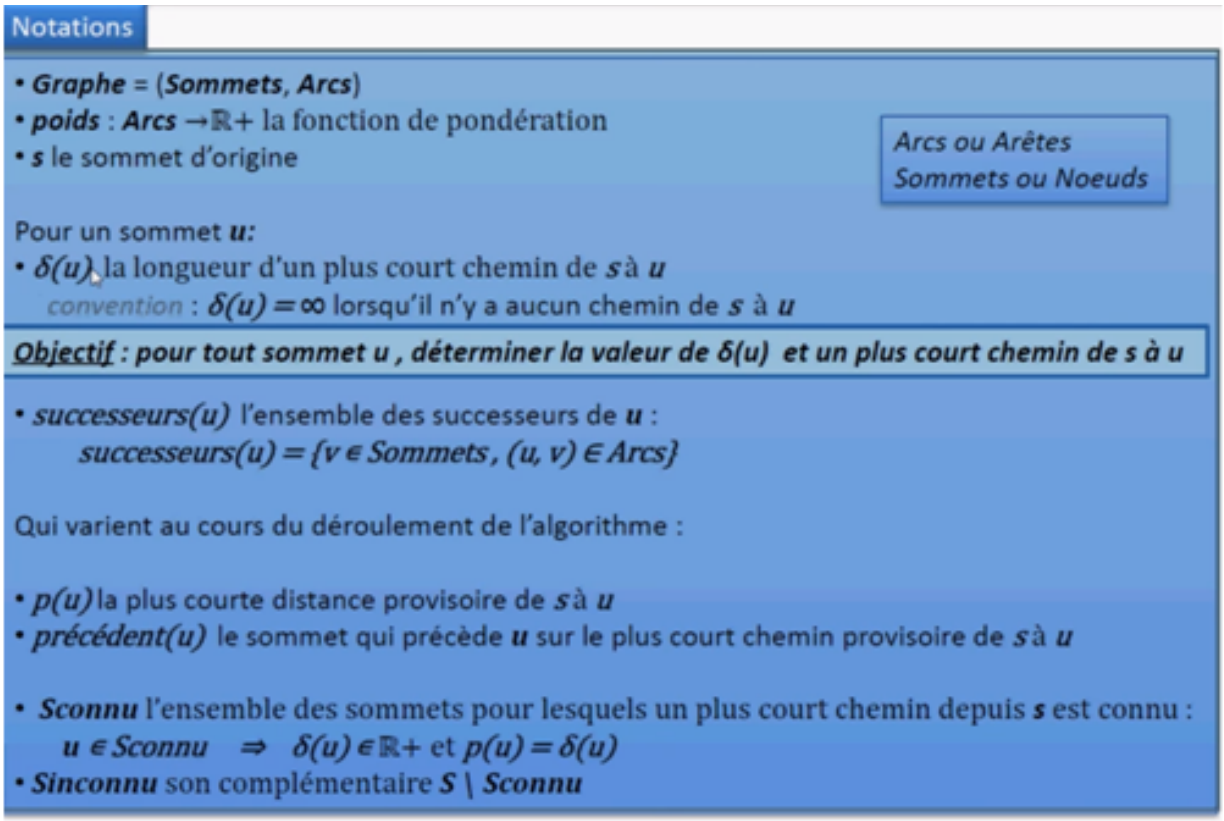
Pendant 2-3 cours et quelques heures au CDI, nous avons réalisé le début de notre projet pour le bac : l'interface du "logiciel". Nous avons réalisé ce travail ensemble mais aussi chacune de notre côté à la maison. Nous avons d'ailleurs réalisé un appel vidéo avec Skype, avec un partage d'écran pour voir nos travaux respectifs.

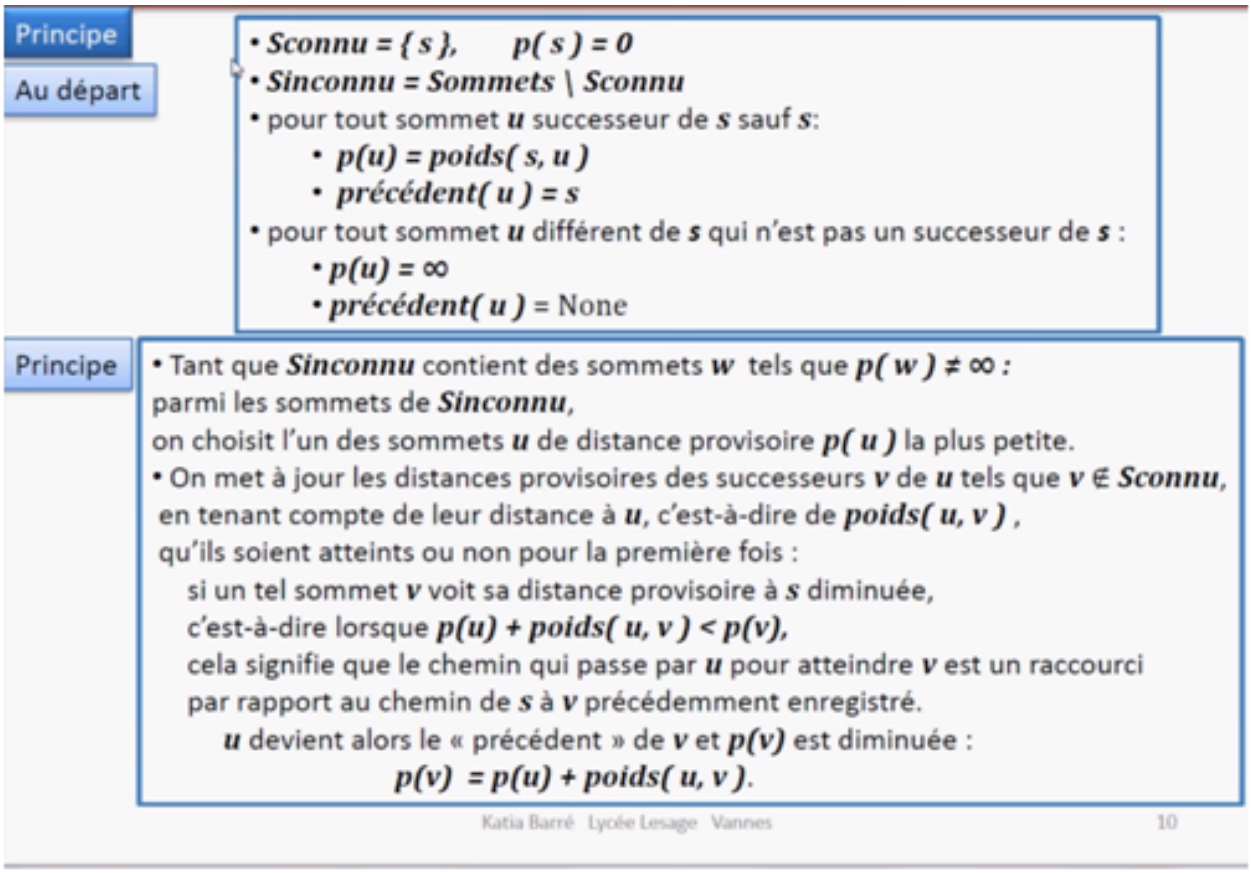
Mon travail

Pseudo-code Dijkstra

Pour cette partie du travail, je l'ai réalisé toute seule. Cette partie servira de base à l'algorithme en lui-même. Un pseudo-code est, d'après *Wikipédia*, "une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier".

Etant donné que je n'avais aucune idée de comment commencer mon pseudo-code, je me suis aidé de cours de prépa que j'ai trouvés sur internet. J'ai donc récupéré le pseudo-code et appris comment il marchait. Comme je le trouvais bien, et logique, j'ai décidé de le garder et de me baser dessus pour coder ensuite. Voici comment il se présente :





La compréhension de ce pseudo-code est donc une application de ce que j'ai appris dans la partie "Découverte du principe de l'algorithme de Dijkstra".

Pseudo-code :

```
ALGORITHME PlusCourtChemin DIJKSTRA(Graphe, s):
    Début
        p(s) ← 0
        Sconnu ← { s }
        Sinconnu ← Sommets \ { s }
        Pour chaque sommet  u ∈ Sinconnu  faire :
            p(u) ← ∞
            précédent(u) ← None
        FinPour
        Pour chaque sommet u ∈ Successeurs(s) \ { s }  faire :
            p(u) ← poids(s, u)
            précédent(u) ← s
        FinPour
        Tant que ∃ u ∈ Sinconnu tel que p(u) ≠∞ :
            choisir u ∈ Sinconnu  tel que p(u) soit minimal
            Pour chaque v ∈ Successeurs(u) ∩ Sinconnu faire:
                Si p(u) + poids(u, v) < p(v) faire:
                    p(v) ← p(u) + poids(u, v)
                    précédent(v) ← u
                FinSi
            FinPour
            Sconnu ← Sconnu ∪ {u}
            Sinconnu ← Sinconnu \ { u }
        FinTantQue
    Fin
```

Algorithme Dijkstra

Cette partie a été réalisée par moi-même, hors des heures de cours. Je me suis appuyée de plusieurs algorithmes de Dijkstra en python que j'ai trouvés sur internet, puis, une fois les principes compris, j'ai réalisé le mien.

Ce travail m'a pris beaucoup de temps, car pour chaque fonction que je ne comprenais pas, je devais faire pleins de recherche pour être sûre d'avoir bien compris. De plus, les algorithmes que je trouvais avaient de nombreuses différences, ce qui prolongeait mon travail. En effet, je voulais avoir plusieurs versions de l'algorithme de Dijkstra pour avoir "plus de liberté" dans l'élaboration de mon propre programme.

Après plusieurs recherches, je suis revenue sur le pseudo-code ci-dessus, et j'ai commencé à programmer en me basant sur celui-ci.

Précédemment, j'ai expliqué que la base de données du graph était inutile. En effet, comme nous n'allons pas effectuer de modifications dans les durées ou noms des planètes du graph, j'ai préféré utiliser un dictionnaire au début du code. Il est simple à comprendre et facil à modifier si on voulait finalement changer notre carte.

Ensuite, pour faciliter le travail dans la création de mon code, j'ai préféré mettre une durée de 1 ruru (unité de mesure inventée) entre chaque planète. Effectivement, cela m'enlevait d'éventuels problèmes dans l'affichage des résultats, concernant le calcul du chemin le plus court. Cela me permettait donc de me concentrer sur le déroulé du code ainsi que ses fonctions.

Une fois le programme fini, j'ai demandé à mon cousin de me corriger quelques erreurs, et d'améliorer "l'affichage des résultats". En effet, au début, mon code affichait une liste de tous les chemins possibles avec comme point de départ, celui qu'on sélectionne. C'était donc compliqué de trouver le chemin avec la destination qu'on voulait. Ainsi, nous avons ajouté un bout de code à la fin, pour que le résultat affiché soit une liste avec seulement le chemin qu'on veut (avec le point de départ et le point d'arrivée souhaités), et la durée que prendra ce chemin.

Explication de l'algorithme de Dijkstra

Explication basée sur le pseudo-code ci-dessus.

Tout d'abord nous allons procéder à l'initialisation. Tout au long de l'algorithme il va y avoir deux listes appelées Sconnu et Sinconnu. Sconnu regroupe tous les sommets (=les planètes) par lesquels on est déjà passé et qui sont validés. Sinconnu est l'ensemble des autres sommets, ceux par lesquels on n'est pas encore passé. Au début, le point de départ (s sur le pseudo-code) est déjà entré dans l'ensemble Sconnu car c'est le point où on se situe pour commencer. Le reste est donc entré dans l'ensemble Sinconnu.

Ensuite, on appelle p(u) (avec u un sommet quelconque, autre que le point de départ s), le poids (=la distance) qu'il y a entre u et s. Ainsi, pour tous les sommets u successeurs de s, p(u) est égale à la distance entre ces deux sommets (prévue dans un dictionnaire représentant le graphe). Et pour tous les autres sommets u, qui ne sont pas successeurs de s, p(u) va être égale à l'infini.

Enfin, s sera un précédent(u), pour tous les u, successeurs de s ; et pour les autres u, on associe "None" (c'est-à-dire aucune valeur, rien) à précédent(u).

L'initialisation est finie, on peut passer au coeur de l'algorithme. Cet algorithme ne se finit que quand l'ensemble Sinconnu est vide. La première étape est de prendre le sommet le plus proche de s, c'est-à-dire, p(u) doit être minimal. Ensuite, pour chaque successeur de u, appartenant à Sinconnu, appelé v, on regarde **si p(u)** (= la distance du début jusqu'au u) **+ poids(u,v)** (= la distance entre u et son successeur v) **est inférieur à p(v)** (= soit la distance est égale à l'infini, soit elle est égale à la distance entre le début et un autre sommet v, déjà calculé auparavant). Et si c'est le cas, alors on remplace p(v) par cette nouvelle distance calculée. Puis, on dit que le précédent de v est u, et on retire u de Sinconnu pour le placer dans Sconnu.

Pour résumer, nous testons tout le long du programme, chaque distance pour voir si elle est plus courte que la précédente. Et nous ne retenons que la distance la plus courte.

Mise en commun de nos parties

Pour les 3 dernières semaines, on a mis en commun notre projet : Julie a fait la partie interface, avec Tkinter, et moi la partie programmation.

Rappel du rendu final qu'on veut avoir

On avait pour but de créer une carte interactive : lorsqu'on clique sur une planète de la carte, puis sur une autre planète, l'ordinateur les enregistre en tant que planète de départ et d'arrivée, puis calcul le chemin le plus court. De plus, on voulait que les lignes entre les planètes du parcours soient mises en valeur, et que la durée du parcours s'affiche.

Parcours, réalisé par Julie

Julie a donc essayé de mettre en valeur les lignes entre les planètes du parcours, mais elle n'y est pas arrivée. Elle a donc décidé de mettre en valeur (d'une autre couleur) seulement les planètes du parcours. Pour cela, elle a fait le lien entre le résultat de l'algorithme de Dijkstra et une fonction de Tkinter.

Durée, réalisé par moi-même

Pour faire afficher la durée du parcours, j'ai aussi fait le lien entre le résultat de l'algorithme de Dijkstra et une fonction qui permet d'afficher des choses, grâce à Tkinter.

Ainsi, on a divisé encore une fois (en toutes petites parties) la mise en commun. Cette mise en commun a donc été faite très rapidement.

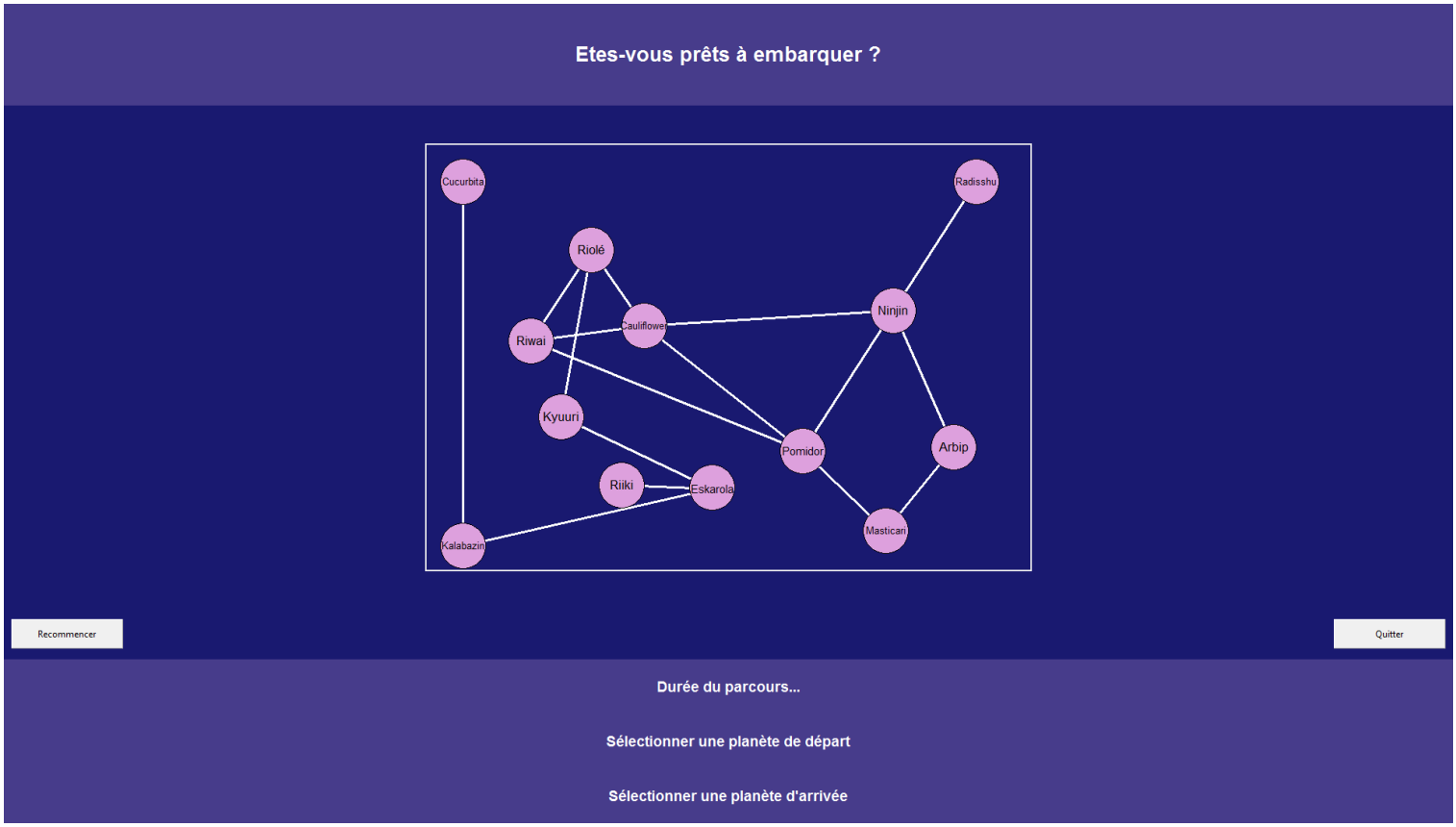
Bilan et perspectives

Le résultat final de notre projet est assez proche de ce qu'on avait prévu de faire. Quelques difficultés nous ont empêché de finir cette carte et de l'améliorer en ajoutant des "options", comme des plages d'horaires pour les bus, les distances, des lignes de bus et des changements… Mais on se doutait qu'on aurait pas le temps de faire tout ça ; et ces "options" en plus seraient au cas où on serait en avance.

On pourrait améliorer aussi la sélection des planètes de départ et d'arrivée pour mieux laisser le temps à l'utilisateur de visualiser vers quelles planètes il se dirige. Mais on arrivait pas à faire marcher une fonction ("time.sleep") correctement.

Finalement, ce projet nous a permis d'avoir un aperçu de ce que sont les "programmes" et de voir à quel point ils demandent une grande rigueur et de la logique. On a bien su départager le travail et le mettre en commun pendant les heures d'ISN ou à d'autres moments. D'ailleurs, on arrivait bien à travailler ensemble parce que quand l'une ne comprenait pas quelque chose, l'autre expliquait, et pareil pour trouver la solution à un problème.

Résultat...



Par Julie DELISLE & Anne-Soline GUILBERT--LY