


# Compte-rendu projet : “*Tout schuss à Courch !*”

## Groupe (Td 01) :

Elie Kanga

Raphael Dupuy

## Préliminaires :

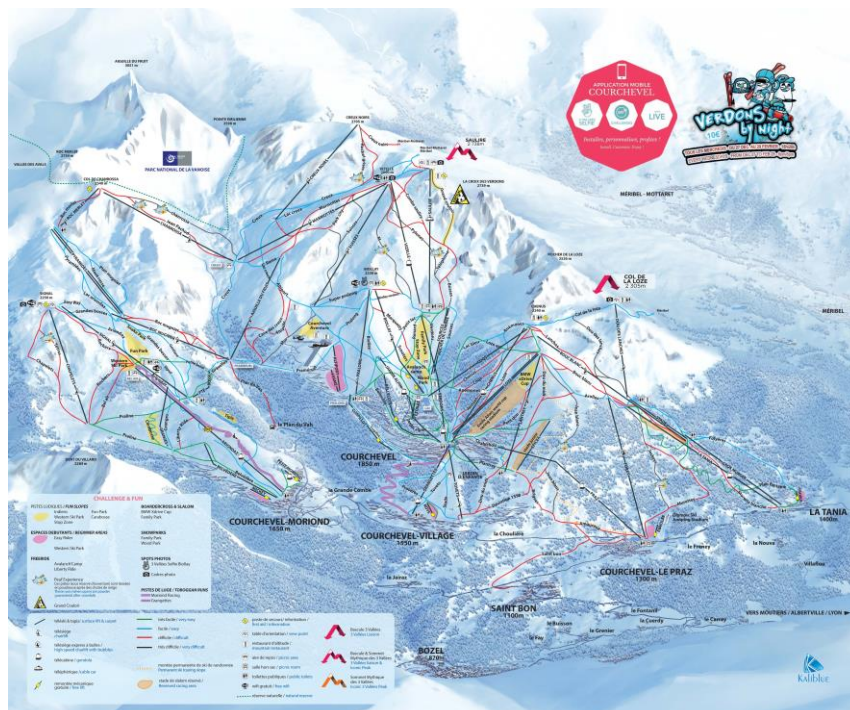
 Bienvenue à Courchevel, la station de ski préférée des riches, des célèbres et des chanceux qui ont gagné au loto ! Ici, vous pouvez skier sur les pistes les plus difficiles du monde tout en étant entouré de Lamborghinis et de yachts de luxe (parce que, pourquoi pas ?). Et si vous avez besoin de vous reposer après une longue journée de ski, vous pouvez vous détendre dans l'un des spas de luxe de la station ou passer la soirée dans l'un des clubs les plus huppés des Alpes.

Et pour les plus téméraires, Courchevel offre également des activités de montagne incroyables, comme la tyrolienne la plus haute des Alpes ou le saut en parachute depuis un hélicoptère. Mais ne vous inquiétez pas, si vous préférez les activités plus tranquilles, vous pouvez toujours admirer les riches en train de skier et vous moquer de leur technique.

En somme, Courchevel est l'endroit idéal pour tous ceux qui veulent dépenser une petite fortune en vacances de ski et se sentir comme un membre de la jet-set. Mais attention, si vous n'avez pas l'air riche, on pourrait vous demander de skier dans une autre station (désolé, mais c'est la vie à Courchevel).

## Généralités et utilisation :

Le projet “*Tout schuss à Courch !*” a pour but de créer un GPS de la station de Courchevel. Pour le faire, nous avons opté pour une implémentation en Python et nous avons pris comme carte la suivante :



En termes d'interface, nous avons choisi une interface graphique implémentée grâce au module Tkinter.py, après avoir lancé le programme une fenêtre s'ouvre affichant la carte de Courchevel ci-dessus avec des points sur les différents nœuds reliant des points de départ, d'arrivée de remontés et d'intersections de pistes.

La carte est navigable en utilisant les scroll bar sur le côté droit et inférieur de la fenêtre. D'origine, le niveau du skieur est initialisé comme Débutant mais il peut être changé pour Confirmé avec le bouton 'Niveau' dans le coin supérieur gauche.

Pour utiliser le GPS, l'utilisateur doit cliquer sur un premier nœud (de départ) puis sur un deuxième nœud (d'arrivée). Après avoir calculé le plus court chemin, l'application affiche le trajet en surbrillance et le temps nécessaire en haut de la fenêtre. Cependant, si les nœuds de départ et d'arrivée sont les mêmes ou s'il n'existe pas de chemin entre les deux, une fenêtre d'information s'ouvre.

Le bouton Reset en haut à droite permet d'effacer le trajet et d'en commencer un autre. Dans le cas où un deuxième trajet est initialisé sans utiliser le bouton Reset, les temps des deux (ou plus) trajets se cumulent, permettant de créer des étapes dans le chemin.

Si toutefois, l'utilisateur voudrait quitter cette belle application, il peut utiliser la croix dans le coin supérieur droit ou le bouton Exit dans le coin supérieur gauche.

#### Implémentation :

Premièrement, pour récupérer les données, nous avons eu recours à une autre application afin de créer un dataset plus facilement via une interface graphique permettant d'ajouter les nœuds et les pistes sur la carte.

Une fois le dataset récupéré et les différentes pistes ainsi que les nœuds enregistrés dans un fichier .json, le programme principal, au lancement utilise le fichier pour afficher les pistes et les nœuds sur la carte en tant que boutons et lignes colorées représentant leur difficulté.

Pour représenter le graphe, nous avons opté pour de la programmation orienté objet. En effet le programme utilise les classes **Application** (héritant de Tk.frame) pour créer la fenêtre et les différents boutons, **Nœuds** pour représenter les nœuds du graphe (avec un nom, des coordonnées et des voisins), **Pistes** pour représenter les pistes (avec une longueur, un nœud de départ et de fin, et une couleur) et enfin la classe **Data** pour gérer l'ensemble des classes du graphe.

La méthode principale de ce projet est la fonction Dijkstra dans la classe application (plutôt la méthode trajet qui appelle Dijkstra). Cette fonction utilise l'algorithme du plus court chemin de Dijkstra en utilisant les listes d'instances des classes Nœuds et Pistes stockées dans l'instance d'Application en vérifiant que les nœuds de départ et d'arrivée soit bien différents.

Merci d'avoir pris le temps de lire ce compte rendu.