LE LOGICIEL HIKER PATH ET SON FONCTIONNEMENT :

Conception et ergonomie du logiciel :

Dans cette première sous partie nous allons traiter des choix que nous avons fait en terme d'ergonomie ainsi que de gestion de notre fenêtre.

Le point de départ du stage fut la fenêtre. Nous l'avons implémenté sous forme de classe héritant de « QMainWindow »[1]. Nous avons donc doté notre classe d'un constructeur qui créer notre logiciel, ainsi que de plusieurs « slots ». Un « slot », dans la bibliothèque Qt, est une méthode qui est appelée lorsque c'est produit un événement, celui-ci est relié à un signal au travers d'une méthode « connect() ». Le déroulement d'une méthode « connect() » est la suivante : un événement se produit par l'envoie d'un « emit » du par un clic ou encore une sélection, le slot est appelé et s'exécute.

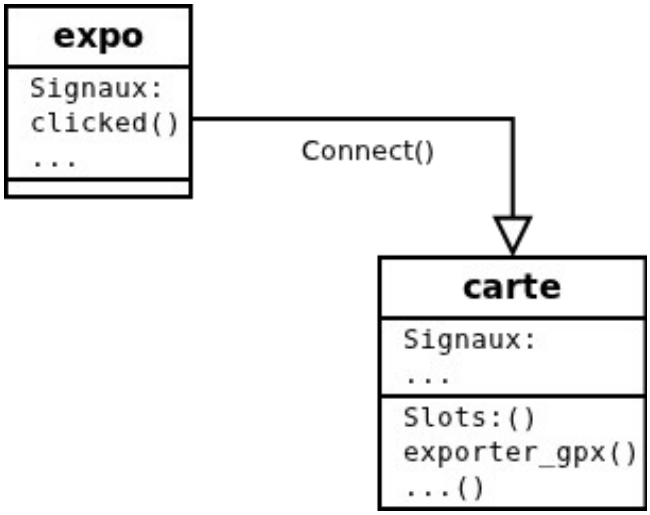

Schéma présentant le fonctionnement de la méthode connect().

1 la classe servant à implémenter les fenêtres principales.

Au niveau de l'organisation de notre fenêtre nous avons décidé de la découper en quatre partie , qui se rapproche de la distribution d'un fenêtre classique :

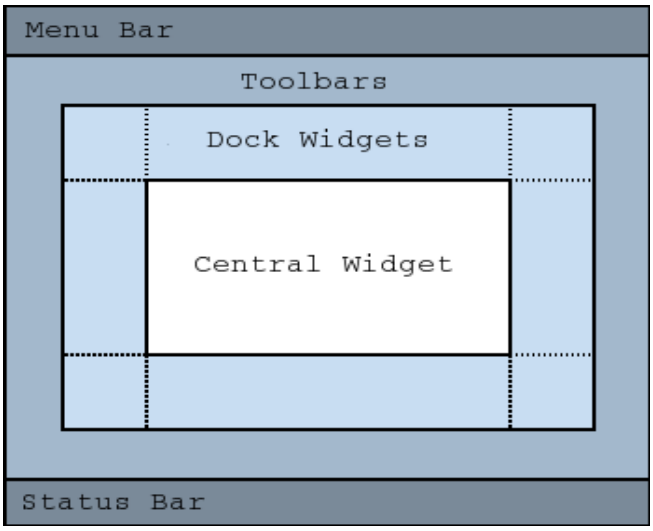

Schéma provenant de la documentation Qt.

La première partie est dédié à la barre de menu où l'on retrouve trois onglets : Fichier, Édition, et Affichage.

Fichier comporte cinq actions (en Qt cela s'appelle un QAction[2]) :

- Ouvrir, pour la sélection de l'image qui va servir de base pour la création du chemin. Le raccourci clavier « Ctrl+O » est-utilisé.
- Charger, qui permet de charger un projet qui a déjà été enregistrer. Le chemin est automatiquement recréé et les coordonnées GPS, s'il y en a, sont de nouveau replacées ainsi que les icônes. Le raccourci clavier « Ctrl+C » est-utilisé.
- Sauvegarder le projet sous, cette action est grisé de base puisqu'il n'y a rien à enregistrer. Elle s'active quand une image est chargée ou encore un projet. Elle permet donc de pouvoir enregistrer un projet à l'emplacement que l'on souhaite le point de départ étant la racine.
- Sauvegarder le projet, permet de sauvegarder les modifications apportées depuis la dernière sauvegarde. Le raccourci clavier « Ctrl+S » est-utilisé.
- Quitter, permet de quitter le programme. Le raccourci clavier « Ctrl+Q» est-utilisé.

Les fonctionnalités de « zoom », sauvegarder, charger sont des méthodes du widget que nous avons développé dans le but de faciliter la portabilité du projet. La sauvegarde se fait au format .xml[3] avec des balises que nous avons nommé et nous laissons l'utilisateur choisir là où il veut enregistre son

2 Un QAction représente interaction avec l'utilisateur permettant l'envoie d'un événement

3 Le XML est un langage informatique sous forme de balise.

projet. En ce qui concerne le chargement du projet, il y a une sécurité pour évité toute mauvaise manipulation, en effet on calcul le md5sum[4] de l'image et lors du chargement on test si le md5sum enregistré est bien le même que celui de l'image (on enregistre le chemin de l'image et l'on calcul à partir de celui-ci).

Ensuite, le menu édition possède une action, celle de pouvoir exporter le projet au format .gpx. A savoir que cette fonction n'est possible que dans le cas où l'utilisateur à déjà entré des coordonnées et tracé un chemin. Le raccourci est « Ctrl+E ».

Affichage, quand à lui, en possède trois :

- Zoom avant, permet de faire un zoom sur l'image à l'endroit où le clic à été enregistrer. Le zoom est de 1,25 avec comme raccourcie « Ctrl+W »
- Zoom arrière, de même mais le zoom à une valeur de 0,8 et comme raccourci clavier « ctrl+alt+W »
- Afficher le gestionnaire GPS, qui permet d'afficher la zone dédié à l'édition des coordonnées. Son raccourcie clavier est « ctrl+D ».

Ces trois zones sont grisées du moment où il n'y a pas de d'image ou de projet chargé.

La fenêtre possède également une barre d'outil reprenant les actions présentées ci-dessus, comme « Ouvrir un projet », « Sauvegarder un projet », les deux types de zoom et le « Gestionnaire GPS », mais aussi fermer le projet et une zone où apparaît la couleur sélectionnée par le clic gauche de la souris (zone blanche par défaut). Les différentes actions sont représentées par des icônes et sont séparées les une des autres de façon à ce que plus lisible. Toutes les actions de ce widget sont reliés par des connect() (QObject::connect(quitter, SIGNAL(triggered()), qApp, SLOT(quit()));), ils sont également tous grisés au lancement dans la fenêtre sauf l'icône « Ouvrir ».

Ensuite, la fenêtre possède un QdockWidget[5] situé sur la gauche du widget. Cette zone est utilisé uniquement pour la gestion GPS, elle est elle même divisée en trois partie : la première pour la saisie des coordonnées en décimales en deux parties pour les deux points. Les points sont divisés entre latitude(de (-90,0000 à +90,0000) et longitude (allant de -180,0000 à +180,0000). La deuxième est dédiée aux coordonnées en format sexagésimales ou degrés minutes secondes divisées en trois zones d'édition correspondant à ce format, toujours divisé entre latitude et longitude. Les deux zones possèdent deux boutons valider, vérifiant avant d'envoyer les données que toutes les zones sont non vide.

La dernière zone du QDockWidget possède également deux boutons, un pour l'export des données au format .gpx (toujours grisés du moment que le format n'est pas correcte) et l'autre pour réinitialisé les deux icônes à positionner sur la carte. Ces trois parties sont organisées de façon

4 Le md5sum une fois calculer permet de donner ce que l'on peut appeler l'empreinte du fichier. C'est est une valeur de 128 bits correspondant à une somme de contrôle calculée à partir de l'archive.

5 Les docks sont des mini-fenêtres que l'on peut généralement déplacer à notre guise dans la fenêtre principale.

suivante les zones d'insertions sont dans une « box » verticale, placé dans une « box » horizontale, le tout dans une verticale ce qui permet une bonne lecture de ce widget. Nous avons choisi d'utilisé un QdockWidget de façon à bien séparé cette partie, puisque l'utilisateur peut très bien construire son projet sans même avoir besoin d'y mettre des points et ne sert qu'à l'export.

Enfin la dernière zone est le widget que nous avons créé, qui se trouve au centre de la fenêtre.

Développement d'un widget sous Qt :

Dans notre programme il est possible, je le rappel, de détecter le tracé d'un chemin et d'exporter ce même tracer en données GPS. Cette partie nous permet d'expliquer les principaux algorithmes cachés derrière l'interface utilisateur.

Pour la reconnaissance d'un chemin et sa construction il nous faut 3 éléments . Tout d'abord un point de départ puis une couleur de chemin et enfin une direction à suivre qui emprunte la même couleur. Un utilisateur va donc dans un premier temps cliquer sur le point de départ. Ce clique permet de donner non seulement les coordonnées de départ mais également la couleur de référence pour la construction du chemin suivi.

La reconnaissance de couleur joue donc un rôle de garde fou. SI la couleur est différente alors c'est que le chemin suivi n'est pas le bon, à l'inverse si la couleur est identique alors il faut continuer dans cette direction. L'algorithme récupère le code RGB (une union des 3 valeurs des couleurs primaires rouge, vert et bleu) du point de départ puis le fixe en référence. Pour vérifier si 2 couleurs sont identiques, la fonction implémenté compare les 3 couleurs primaires de 2 points différents. Les valeurs sont décomposés en 3 (une pour chaque couleur primaire) et le programme fait la différence de chaque éléments. Partant du principe que 2 couleurs identiques possèdent le même code RGB si la somme des différences est éloignés de 0 alors les couleurs sont différentes. Nous avons établi un seuil après avoir testé sur une palette de couleur plusieurs chiffres et notre choix s'est porté sur la valeur de 100. //(vérifié si elle n'a pas été modifié)

L'utilisateur lors de l’exécution d'un projet doit également donner la direction initiale à suivre en faisant un appuie long sur la souris et la relâcher dans la direction du trajet souhaité. Une fois l'impulsion donnée les calculs tracent le chemin automatiquement et s'arrête quand les conditions d'arrêt sont atteintes. La direction initiale est donnée par la différence entre les coordonnées du point de « release »(le point relâché) et le point de clique. L'algorithme va ensuite chercher une nouvelle direction, pour cela il aura le choix entre 5 directions.

Voyons tout d'abord les 8 directions possibles existantes, elles correspondent au 8 direction de la boussole page précédente :

- Ouest : nommée la direction gauche
- Nord : nommée la direction la direction haute
- Est: nommée la direction la direction droite
- Sud : nommée la direction la direction basse
- Nord Ouest : nommée la direction la direction haute gauche
- Nord Est : nommée la direction haute droite
- Sud Ouest : nommée la direction basse gauche
- Sud Est : nommée la direction basse droite

Lorsque l'algorithme recherche une nouvelle direction à suivre il prend en compte l'ancienne direction. Par exemple, il n'est pas possible de choisir la direction faisant un demi tour par rapport à la direction d'où l'on vient car on reviendrait sur nos pas. Cependant il est possible d'effectuer un virage. Les directions possibles sont donc, dans l'ordre de priorité de recherche, la même direction que celle qui précède, ses 2 voisines directes et pour finir ses 2 voisines encore un pas plus loin.

Prenons pour illustrer la direction Ouest, le chemin va rechercher à aller dans la direction Ouest tout d'abord puis Nord Ouest ou Sud Ouest puis Nord ou Sud.

Comment se choix va-t-il se faire ? Comme dit précédemment le chemin doit contenir la couleur de référence. La fonction utilisée travaille donc sur une zone définie qui va compter le nombre de points qui sont de la même couleur que celle définie au clique de départ. La zone qui s'avère être celle qui contient le plus de points identiques sera celle qui donnera la nouvelle direction. En dessous d'un seuil défini, on considère que le chemin comportant le plus de points identiques est terminé et que le tracé est fini ou que l'utilisateur doit donner une nouvelle instruction de direction.

Le choix de la zone de traitement est également très important car il va donner la précision du chemin. Si on choisi une zone petite, le chemin est plus affiné mais il y a le risque du demi-tour traité précédemment. En effet, un tracé aura la place de faire un allé et un retour dans un même chemin en effectuant plusieurs virages.

Dans le cas d'une zone trop grande, le chemin détecté peut « deborder ». Par exemple, dans un cas où le chemin s'arrête ou effectue un virage serré, si un chemin proche comporte la même couleur, l'algorithme va considérer les points dans la zone même s'ils appartiennent à un autre chemin. Ce cas peut encore avoir lieu avec 2 chemins parallèles, la détection pourrait partir sur le chemin parallèle à celui traité. Il faut donc calibrer de manière correct la zone de recherche. De plus, une

fonction modifie légèrement la zone de recherche selon la direction du tracé. Par exemple, si la direction va vers le Nord alors la zone sera un rectangle dont le plus long coté sera dans la direction Nord alors que si on prend la direction Est le plus long côté sera dans la direction Ouest ainsi on affine la zone de recherche.

Liaison avec les coordonnées GPS :

La dernière partie est intégrée au widget. Dans un premier temps l'utilisateur est doit se placer dans le gestionnaire pour pouvoir exporter le projet. Il y a deux méthodes, pour un même résultat, la première est d'utiliser le placement des deux icônes sur la carte, permettant de relever la position du clic avant la transformation en coordonnée GPS, puis en faisant le parallèle avec les coordonnées saisies par l'utilisateur qu'elles soient décimales ou sexagésimales, l'autre méthode est l'inverse de la première : saisir les coordonnées et après placer les icônes sur la carte.

Au niveau du placement des icônes, il faut se trouver dans le gestionnaire. Le placement ne peut s'effectuer qu'une fois pour les deux points. Il faut réinitialisé ces points pour pouvoir les replacés par la suite mais cela ne réinitialise pas pour autant les coordonnées GPS saisies. Le placement de ces deux icônes se fait par l'intermédiaire de deux fonctions utilisant une gestion de drapeaux. Cela permet d'éviter qu'il y ait une possibilité infinie d'icônes positionnés mais aussi qu'ils se sélectionnent l'un après l'autre. Nous avons dû aussi jouer sur les coordonnées entre le clic et l'affichage, puisque le positionnement se fait en haut à gauche, il a fallut retrancher la hauteur de l'image à la coordonnée ainsi que quelques pixels de façon à ce que la pointe de la punaise soit sur alignée avec le clic

La saisie se fait donc par l'intermédiaire du QdockWidget, qui par validation envoie les points aux widgets. Les points réceptionnées sont de deux types différents, nous avons donc été amené à créer une classe « coord_decimal » qui possède deux variables de type réelle qui correspondent aux coordonnées décimales. Nous avons décidé de créer également une classe pour les points sexagésimales mais très vite n'envoyant pas l'utilité nous l'avons supprimer. En effet, les données GPS ne sont utilisés qu'au moment de l'export et ce dernier ce fait au format décimal. Pour palier à ce problème, nous avons alors implémenté directement dans la classe « coord_decimal » une fonction de conversion qui convertie les points sexagésimales en décimales :

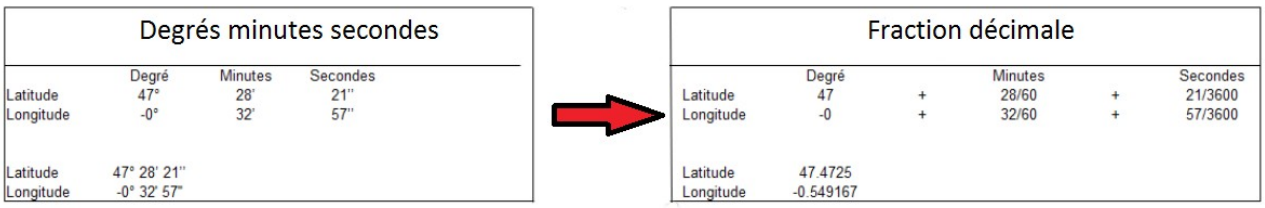

Schéma de conversion d'une coordonnée sexagésimale vers une en décimale

Une fois les points saisie par l'utilisateur, celui-ci peut retourner à la conception du chemin, modifié les valeurs qui ont été rentrées si jamais une erreur avait été faite, ou encore exporter son projet .gpx. Pour la modification, il lui suffit de corriger ou modifier les points et d'utiliser le bouton « valider » pour les mettre à jour. Pour l'export il faut au préalable qu'un chemin soit tracé et que des coordonnées soit validées de façon à ne pas exporter un projet vide ou incomplet. Au moment de l'export, on calcul la coordonnée du premier clic du chemin (correspondant au début du tracé) chargé au préalable dans une pile. Nous avons choisie de faire un calcul de pas, c'est-à-dire que l'on calcul la différence entre les deux points saisies, les deux points du clic (là où se trouve les icônes). On multiplie les coordonnées du clic utilisateur avec le premier résultat et on divise par le second, puis le résultat est ajouté aux coordonnées du point de référence (choisie par défault, le premier point que l'utilisateur entre). De plus nous avons du créé une classe « point_gps » pour sauvegarder le résultat de conversion car Qpoint[6] ne convenait pas, il nous fallait des coordonnées décimales et non entière. Cette classe est juste munie d'un constructeur, d'accesseurs et de mutateurs. Le premier piège avec cette méthode fut que l'échelle était inversée entre notre fenêtre et le site « google maps » qui nous a permit d'avoir accès à des points et servi de base pour nos calcul et le deuxième fut d'ajouter le résultat des premiers calcul à un point de référence.

Au-delà de l'export nous avons implémenté la sauvegarde et le chargement d'un projet de façon à ce que toutes ces informations soient rechargés et disponible. Les coordonnées sont chargées dans le gestionnaire mais des coordonnées au départ en sexagésimales seront converties et placées dans la zone décimale.

Nous avons cherché à travailler sur la simplicité d'utilisation, ainsi que sur la portabilité du logiciel. Cependant, il aurait été intéressant de mettre toutes les possibilités qu'offre le logiciel lors du zoom (qu'il soit avant ou arrière). Actuellement, quand le zoom est utilisé, les autres méthodes sont bloqué car nous n'avions des erreurs au niveau du tracé du chemin ainsi que lors du placement des balisent GPS. Nous n'avons pas pu déterminé la cause de ce décalage, s'il était dû à l'arrondie au niveau de l'échelle (lors du zoom augmente la taille de 1.25%) ou si cela était du à une mauvaise réflexion de l'algorithme. Il aurait été également plaisant de pouvoir nommer les chemins dessinés,

6 QPoint est un point définie dans le plan utilisant des coordonnées entière.

mais cela implique que l'utilisateur pourrait construire plusieurs tracés sur une même carte. En prolongement, l'idée d'imprimer le tracé de façon visuel ou écrite, par exemple au format .xml, aurait pu être envisagée.

CONCLUSION :

Pour conclure, nous avons mis place un logiciel permettant de tracer automatiquement un chemin à partir d'un point sélectionné par l'utilisateur sur une carte qu'il aura chargé. Il permet également de pouvoir exporter le projet au format .gpx à travers la sélection de points GPS sur cette même carte. Nous avons également tenu compte du coté ergonomique de ce logiciel, pour ne pas compliqué les opérations. Nous avons donc mit en place différente méthode comme la sauvegarde d'un projet, (ainsi que le chargement), la possibilité de zoomé (dé-zoomé) sur la carte. Il est aussi possible pour l'utilisateur de fermer le projet sans forcément fermer le logiciel.

Durant ce stage, il a été impératif de mettre en place une bonne organisation comme par exemple un dépôt GIT pour le partage des fichiers sources. Ce stage nous a appris à être rigoureux, précis, dans un échange régulier, par exemple nous avons utilisé le tableau de la salle H004 pour présenter nos visions de la fenêtre ou encore des algorithmes de constructions du chemin. Nous avons également dû être attentif à bien commenté notre code pour qu'il soit compréhensible de l'autre.

Il serait intéressant de pouvoir faire une étude de marché sur le secteur des logicielles GPS. Et ainsi savoir comment pourrions-nous positionner ce produit.

SOURCES :

Liste des sites internet qui nous ont aidé lors de ce stage :

- Nous avons choisi le site Github pour créer notre dépôt et ainsi héberger nos fichiers sources :
  https://github.com/kirby49/stage_2012

- Le site du zéro, qui nous a permit d'apprendre et de comprendre les bases de la bibliothèque Qt, ainsi que la gestion des fichiers :

  http://www.siteduzero.com/tutoriel-3-11250-compiler-votre-premiere-fenetre-qt.html

  http://www.siteduzero.com/tutoriel-3-100412-la-manipulation-des-fichiers.html

- Le site de notre maître de stage M Vincent Barichard, qui nous a permit d'apprendre à développer notre propre widget et de parfaire nos connaissance à travers ses travaux pratique :

  http://www.info.univ-angers.fr/pub/barichar/page_perso/index.php?page=enseignements

- La bibliothèque Qt, qui nous a permit de parfaire notre code à travers les méthodes existantes et les types :

  http://qt-project.org/doc/qt-4.8/

- Le site Wikipedia pour le format .gpx et pour le format GPS :
  http://fr.wikipedia.org/wiki/GPX_(format_de_fichier)
  http://fr.wikipedia.org/wiki/Coordonn%C3%A9es_g%C3%A9ographiques

- Le site GoogleXXL pour la conversion des coordonnées GPS :
  http://googlexxl.blogspot.fr/2007/07/convertir-coordonnees-gps-dans-google.html