INTRODUCTION :

En tant qu'utilisateur, il est possible d'avoir accès à des applications permettant de construire des tracés de chemin que l'on trouve principalement sur Internet. Leur modalité de fonctionnement est de saisir une adresse de départ et une d'arrivé. Cependant, rares sont les applications qui construisent un chemin automatiquement et qui s'arrête dès qu'un croisement est rencontré. C'est pourquoi, durant notre stage, nous avons été amenés à développer un tel outil.

Ainsi, du 23 avril au 30 mai, nous avons effectué un stage pour la faculté des Sciences de l'Université d'Angers. Nous étions sous la direction de M. Vincent Barichard et, pendant son absence, de M. David Genest.

L'objet de notre stage a été de développer un logiciel, que nous avons nommé « Hiker Path[1] », et qui permet de créer automatiquement un chemin sur une carte préalablement chargée. Ainsi, l'utilisateur est amené à sélectionner une carte puis est assisté pour construire son itinéraire. Une fois le chemin construit, l'utilisateur peut exporter son tracé pour l'exploiter sur d'autres plate-formes telles qu'un GPS classique ou encore une montre.

Nous avons été amenés à nous demander : Comment implémenter un programme QT possédant les fonctions de recherche et de création de chemins, produisant les coordonnées GPS associées et répondant aux attentes d'un utilisateur ?

Pour répondre à cette problématique, nous orienterons notre rapport autour de deux pôles. Dans un premier temps, nous présenterons notre organisation autour de ce stage tout en apportant une analyse plus approfondie du sujet. Dans un second temps, nous expliquerons les choix que nous avons été amenés a faire en terme d'ergonomie et d'algorithmie par l'exposé de différentes méthodes.

1 Hicker Path signifie « chemin de randonnée ».

1) ORGANISATION ET ANALYSE DU STAGE

1.1) Notre organisation durant ce stage

Effectuer un stage en binôme a demandé des aménagements. En effet, nous avons modulé nos horaires en fonction de nos travail étudiant. Cela a également demandé l'incorporation d'outils d'échanges.

a) Le dépôt GIT

Nous avons mis en place la création d'un dépôt GIT afin d'échanger nos fichiers plus rapidement. En effet, les conflits entre les versions était gérés automatiquement par GIT. Par exemple, si l'un de nous modifiait un ligne de code déjà existante et la postait sur le dépôt, alors GIT émettait une erreur et plaçait la version modifiée en commentaire. Cela permettait de d'identifier rapidement la modification. Ponctuellement, nous avons donc pu travailler indépendamment sur le code. Nous pouvons résumer le fonctionnement d'un dépôt GIT, et les différentes commandes à utiliser par le schéma suivant :

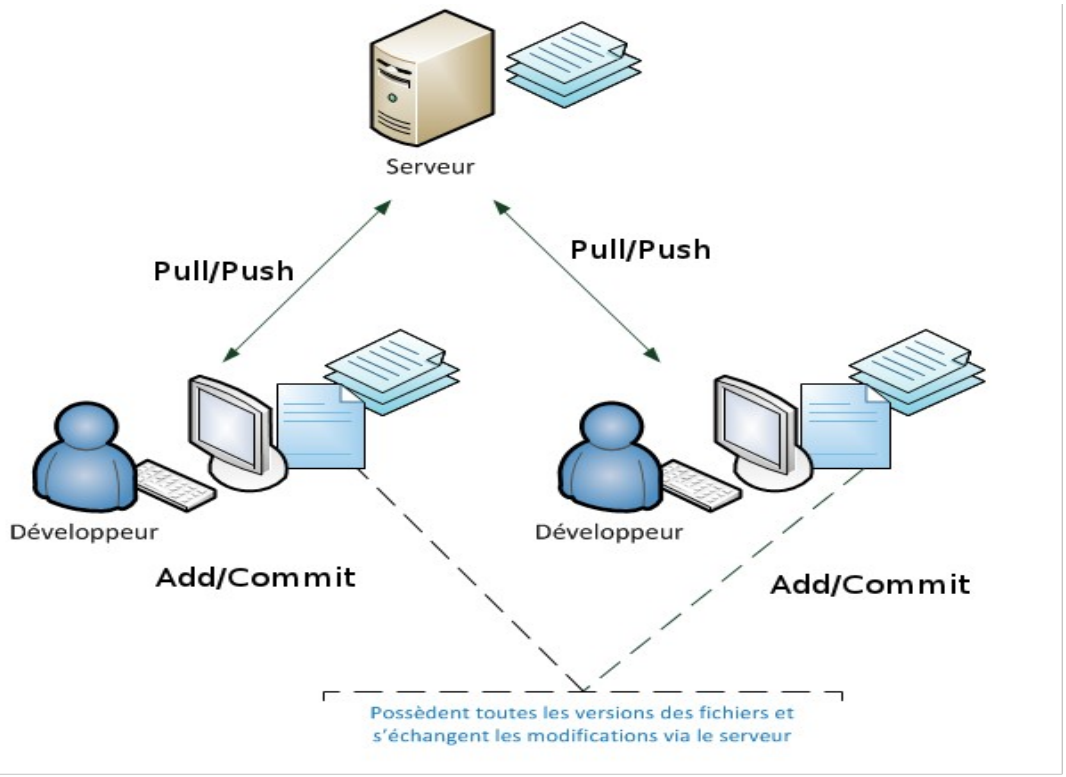

Illustration 1: Schéma illustrant le fonctionnent d'un dépôt GIT avec deux utilisateurs

Source : http://www.siteduzero.com/tutoriel-3-254198-gerez-vos-codes-source-avec-git.html

b) Première approche de la bibliothèque Qt

La deuxième tâche fut d'apprendre et de comprendre le fonctionnement de la bibliothèque Qt. Une bibliothèque est un ensemble de fonctions, de méthodes ou de classes, mises à disposition afin de pouvoir être utilisées, la plus part du temps, sans avoir à les réécrire. Ici, Qt vient apporter une interface graphique au langage orienté objet C++. Nous avons choisi de programmer avec le logiciel QtCreator. Ce logiciel est sous licence GPL, autrement dit c'est un logiciel libre. Un des avantages de ce logiciel est qu'il embarque un débogueur ainsi que la documentation de la bibliothèque Qt. De plus, nous avons essentiellement utilisé le site du zéro[2] afin de nous familiariser avec cette bibliothèque ainsi que les travaux pratiques de M. Vincent Barichard[3].

c) Prise de contact avec notre maître de stage :

Nous avons convenu avec M Vincent Barichard de nous rencontrer au minimum une fois par semaine avant que ce dernier ne parte en Colombie. Nous lui exposions nos interrogations et la progression effective de notre travail. A la suite de son départ, nous avons pu poursuivre par mail cet échange. De plus, nous avons créé un dépôt GIT pour notre tuteur par lequel il pouvait avoir accès à notre projet.

1.2) Analyse de l'objet de notre stage

a) Le sujet

L'objet de notre stage était : « de développer un logiciel permettant de déterminer automatiquement des chemins sur un fond de carte donné sous forme d'image. L'utilisateur endossera le rôle d'un randonneur voulant construire une randonnée passant par des chemins balisés. Après s'être procuré la carte IGN de la zone géographique de la région de sa randonnée, il utilise le logiciel qui tracera une route sur les chemins balisés qu'il a sélectionnés sur la carte. Le résultat sera exporté au format .gpx, format libre et exploitable par un grand nombre de GPS et logiciels de cartographie.»

b) La trame d'un logiciel

2 Adresse du site : http://www.siteduzero.com/tutoriel-3-11250-compiler-votre-premiere-fenetre-qt.html
3 Adresse du site : http://www.info.univ-angers.fr/pub/barichar/page_perso/index.php?page=enseignements

Dans la première phrase de notre sujet, nous trouvons d'ores et déjà les grandes idées de notre projet, c'est-à-dire que nous avions un logiciel à créer ainsi que des algorithmes de constructions de chemins depuis une carte. Ensuite, nous nous sommes penchés sur la création de logiciel et plus particulièrement de la fenêtre Qt. La fenêtre contient les éléments graphique qui seront liés aux algorithmes créés. De même, nous nous sommes demandés quels composants nous pourrions insérer dans le logiciel.

La deuxième phrase nous montre que l'utilisateur est un randonneur, ou tout du moins une personne connaissant déjà le fonctionnement d'une carte (échelle, jeu des couleurs …). Cette phrase complète également la précédente en terme d'algorithmie et nous a conduit à l'utilisation de la couleur dans la construction du chemin. De même, on apprend que l'utilisateur est actif dans la construction de chemin et qu'il aura des sélections ou des choix à faire.

c) La partie algorithmie

La troisième phrase explicite le fonctionnement du logiciel. En effet, un randonneur est amené dans un premier temps à choisir l'image qu'il veut charger et à déterminer le point de départ de sa randonnée, ce qui induit pour nous la notion de clic dans le logiciel. De plus, il nous a fallu interpréter ce clic et également récupérer sa position par rapport à l'image pour pouvoir tracer le chemin. Ainsi, cela nous précise qu'il y aura un dessin. Nous devions donc mettre en avant l'itinéraire en le dessinant sur un calque et non pas sur l'image elle-même. Si le chemin se construit automatiquement cela implique que le tracé doit s'arrêter. Notre maître de stage nous a alors conseillé de s'arrêter à chaque croisement que le tracé pouvait rencontrer.

d*) L'export et des coordonnées GPS*

La dernière partie de l'énoncé est consacrée à l'export du chemin au format .gpx[4]. Ce format doit contenir des coordonnées GPS décimales. Il a donc fallu calculer les points du tracé, voire de les convertir si ces derniers sont en sexagésimales. Pour l'enregistrement des points, nous avions deux choix : soit partir d'un point et une direction, soit de deux points. C'est cette dernière solution que nous avons choisi puisqu'elle nous semblait moins complexe à implémenter. C'est au moment de l'export que tous les calculs des coordonnées de l'itinéraire sont effectués. Toutefois, avant de pouvoir faire l'export au format .gpx, il faut que les coordonnées soient remplies par l'utilisateur. C'est pourquoi, nous avons ajouté une zone d'édition qui sera présentée plus en détail dans la

4 Ce format permet l'échange de coordonnées GPS, à savoir que se format est basé sur le langage xml.

seconde partie.

Nous avons donc orienté notre travail sur trois axes : la fenêtre de notre logiciel, le tracé du chemin, et la liaison avec les coordonnées GPS avec l'export au format .gpx.

2) LE LOGICIEL HIKER PATH ET SON FONCTIONNEMENT

2.1) Conception et ergonomie du logiciel

a) Notre classe fenêtre

Le point de départ de notre programmation a été la fenêtre. Nous l'avons implémentée sous forme de classe héritant de « QMainWindow »[5]. Nous avons donc doté notre classe d'un constructeur qui crée notre logiciel, ainsi que de plusieurs « slots ». Un « slot », dans la bibliothèque Qt, est une méthode qui est appelée lorsque s'est produit un événement, celui-ci est relié à un signal au travers d'une méthode « connect() ». La méthode connect() fait le lien entre le signal et le slot comme le montre le schéma ci-dessous. Le signal se produit au moment où l'utilisateur interagit avec la fenêtre et le slot permet alors d'exécuter l'action définie.

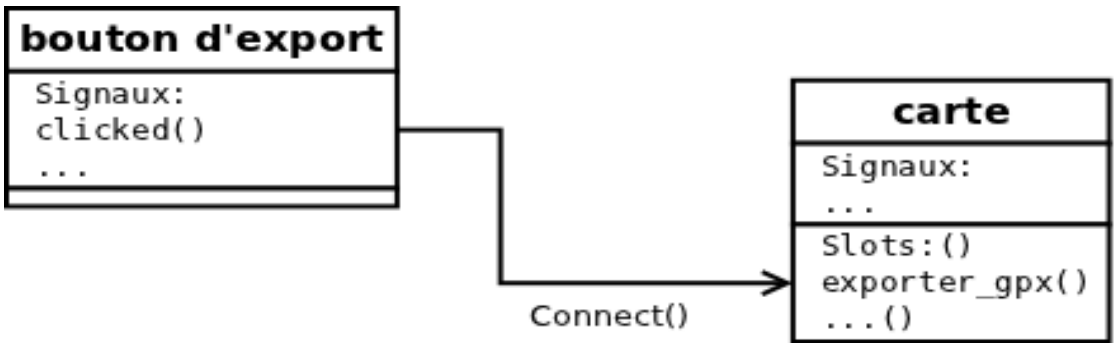

Illustration 2: Schéma présentant le fonctionnement de la méthode connect()

5 la classe servant à implémenter les fenêtres principales.

Au niveau de l'organisation de notre fenêtre nous avons décidé de la découper en quatre parties :

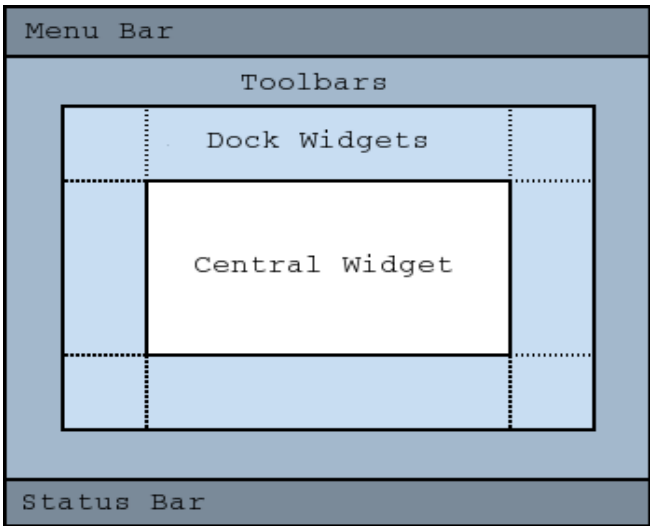

Illustration 3: Schéma présentant l'organisation d'une fenêtre et provenant de la documentation Qt

b) La barre de menu :

On retrouve trois onglets dans la barre d'outils : Fichier, Édition, et Affichage.

Fichier comporte cinq items (en Qt cela s'appelle un QAction[6]) :

- Ouvrir, pour la sélection de l'image qui va servir de base pour la création du chemin. Le raccourci clavier « Ctrl+O » est utilisable.
- Charger, qui permet le chargement d'un projet qui a déjà été enregistré. Le chemin est automatiquement recréé et les coordonnées GPS, s'il y en a, apparaissent de nouveau dans la fenêtre. Il en est de même pour les icônes. Le raccourci clavier « Ctrl+C » peut alors être accessible.
- Sauvegarder le projet sous, cette action est grisée 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 envisageable.
- Quitter, permet de quitter le programme. Le raccourci clavier « Ctrl+Q» est accessible.

Les fonctionnalités de zoom, sauvegarde, et de chargement sont des méthodes du widget que nous

6 Un QAction représente interaction avec l'utilisateur permettant l'envoi d'un événement

avons développées dans le but de faciliter la portabilité du projet. La sauvegarde se fait au format .xml[7] avec des balises que nous avons nommées. Nous laissons l'utilisateur choisir là où il veut enregistré son projet. En ce qui concerne le chargement du projet, il y a une sécurité pour éviter toute mauvaise manipulation. En effet, l'algorithme calcule le md5sum[8] de l'image et lors du chargement, l'item « Charger » teste si le md5sum enregistré est bien le même que celui de l'image.

Ensuite, le menu « édition » peut exporter le projet au format .gpx. Cette fonction n'est possible que dans le cas où l'utilisateur a déjà entré des coordonnées et tracé un chemin. Le raccourci est alors « Ctrl+E ».

L'affichage, quant à lui, possède trois items:

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

Ces trois zones sont grisées dès lors qu'il n'y a pas de d'image ou de projet chargé.

c*) La barre d'outils :*

La fenêtre possède également une barre d'outils reprenant des actions présentées ci-dessus, comme « Ouvrir un projet », « Sauvegarder un projet », les deux types de zoom et le « Gestionnaire GPS ». De plus, on y trouve « fermer le projet », ainsi qu'une zone où apparaît la couleur sélectionnée lors du clic gauche de la souris sur la carte (zone blanche par défaut). Les différentes actions sont représentées par des icônes et sont séparées les une des autres de façon à ce que ce soit plus lisible. Toutes les actions de ce widget sont reliées 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 ».

c) Le QdockWidget : une fenêtre dans une fenêtre

La fenêtre possède un QDockWidget[9] situé sur la gauche du widget. Cette zone est utilisée uniquement pour la gestion GPS. Elle est elle-même divisée en trois parties : la première pour la

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

8 Une fois calculé, le md5sum permet de donner ce que l'on appelle l'empreinte du fichier. C'est est une valeur de 128 bits correspondant à une somme de contrôle calculée à partir de l'archive.

9 Les docks sont des petites fenêtres que l'on peut ordinairement déplacer dans la fenêtre principale.

saisie des coordonnées en décimales. Cette partie comprend deux points pour la latitude (allant de -90,0000 à +90,0000) et la longitude (allant de -180,0000 à +180,0000). La deuxième est liée aux coordonnées (au format sexagésimales ou degrés minutes secondes). Les coordonnée sont divisées en trois zones d'édition, toujours divisées entre latitude et longitude. Les deux zones possèdent deux boutons « valider », qui vérifient avant d'envoyer les données que toutes les zones soient non-vides. La troisième zone du QDockWidget possède également deux boutons, un pour l'export des données au format .gpx (toujours grisées du moment que le format n'est pas correcte) et l'autre pour réinitialiser les deux icônes à positionner sur la carte. Nous avons choisi d'utilisé un QDockWidget de façon à bien séparer cette partie, puisque l'utilisateur peut très bien construire son projet sans même avoir besoin d'y mettre des points. En effet, lors de l'utilisation du logiciel, l'export n'est pas une obligation.

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