Étudiants

Informatique générale

I.U.T.A LYON 1

Alex Mehdi ZAHID
zahid.alexmehdi@gmail.com

Antoine GRÉA
grea09@gmail.com

Blon THO
tho_blon@hotmail.com

Rapport final

Optimisation de trajet pour un robot

DiRIGe

Projet encadré par M. Éric GUÉRIN

mercredi 17 mars 2010

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

Remerciements

Nous tenons à remercier l'ensemble des enseignants de l'IUT A de Bourg-en-Bresse qui ont contribués, directement ou indirectement, à la mise en place du projet DiRIGe par leur enseignement.

Nous remercions tout particulièrement :

- ✔M. GUÉRIN pour nous avoir proposé la mise en place de ce projet et pour les éléments de réponses apportés aussi bien sur le plan technique que sur l'organisation et la mise en place du projet.
- ✔Mme. ESPINASSE et M. BOUTIN pour l'aide qu'ils nous ont apportée dans la résolution de problèmes d'ordre mathématiques.
- ✔M. TELLEZ pour son enseignement du langage de programmation C++ et son introduction à l'utilisation de la bibliothèque Qt au cours des séances de travaux pratiques.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

Sommaire

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

1)Introduction

Dans le cadre de notre cursus à l'IUT A de Bourg-en-Bresse, nous avons réalisé un projet tutoré intitulé « Optimisation du trajet pour un robot. ». Ce projet a été proposé et encadré par M. Éric GUERIN, professeur à l'IUT A de Bourg-en-Bresse dans le département informatique. L'idée de base du projet consiste à développer un logiciel permettant d'éditer un environnement, et de visualiser le trajet le plus optimisé pour un robot partant d'un point de départ jusqu'à un point d'arrivée.

Ce document a pour objectif de résumer la mise en place de ce projet que nous avons baptisé DiRIGe (ou Dijkstra Robot Intelligent Generation). À travers ce document, nous présentons les objectifs de l'application, ainsi que la solution développée en réponse au cahier des charges. Enfin, pour terminer, nous faisons un bref retour sur le déroulement de la mise en place du projet. Nous expliquons entre autres dans cette partie l'organisation du groupe, les difficultés rencontrées et nous étudions les différences entre le planning prévisionnel et le planning réalisé.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

2)Présentation de l'application DiRIGe

Nous allons détailler quelques points essentiels à la compréhension du projet réalisé. Pour plus de renseignements d'ordre technique, nous vous invitons à consulter le cahier des charges, le dossier d'étude de la concurrence, le dossier de conception, ainsi que le code source.

2.1)Objectifs de l'application

•Réaliser une application permettant à un utilisateur de saisir un plan de l'environnement dans lequel est censé évoluer un robot.

•Permettre à un robot de trouver le chemin le plus optimisé en termes de temps de parcours à partir d'un point de départ jusqu'au point d'arrivée.

2.2)Produit du projet

Le produit en question est une application implémentée dans le langage informatique C++.

L'application permet à un utilisateur de saisir un plan d'environnement représenté par une feuille de dessin. Sur ce plan d'environnement, l'utilisateur représente des obstacles en les dessinant. Il place également un point de départ et un point d'arrivée. Les obstacles peuvent être de différentes natures. On identifie la nature d'un obstacle par sa couleur.

L'utilisateur configure les paramètres d'un robot virtuel. Il définit notamment le coût associé à la traversée d'un obstacle suivant sa nature, par le robot. L'utilisateur peut également définir d'autres paramètres comme la largeur du robot, sa direction initiale, ou même le coût associé à une rotation de 180°.

Au moyen de l'actionnement d'un bouton, l'application déterminera le chemin le plus court en termes de temps de parcours pour le robot entre le point de départ et le point d'arrivée. Ce chemin sera représenté sur le plan de l'environnement.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

3)Présentation de la solution développée

La solution que nous avons mis en place utilise la bibliothèque[1] Qt, pour modéliser l'application sous forme graphique et pour réaliser les opérations de dessin. L'application est composée de différents modules qui ont chacun un rôle respectif. Nous avons fait ce choix afin de développer une application souple c'est-à-dire configurable, modulable, réutilisable et maintenable. Nous présentons ici le rôle de chacun des modules ainsi que des extensions et améliorations possibles.

3.1)Module IHM

Le module Interface Homme-Machine (IHM) permet, comme son nom le suggère, d'interagir avec l'utilisateur final.

Il active les différentes parties de l'application, en fonction des actions de l'utilisateur. Il assure également la gestion des fichiers (création, ouverture, fermeture ...) et la fermeture de l'application. Ce module est simple d'utilisation comme le demande le cahier des charges.

Voici quelques images de notre interface graphique :

1 Une bibliothèque correspond à un ensemble de codes déjà écrits et prêts à l'emploi.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

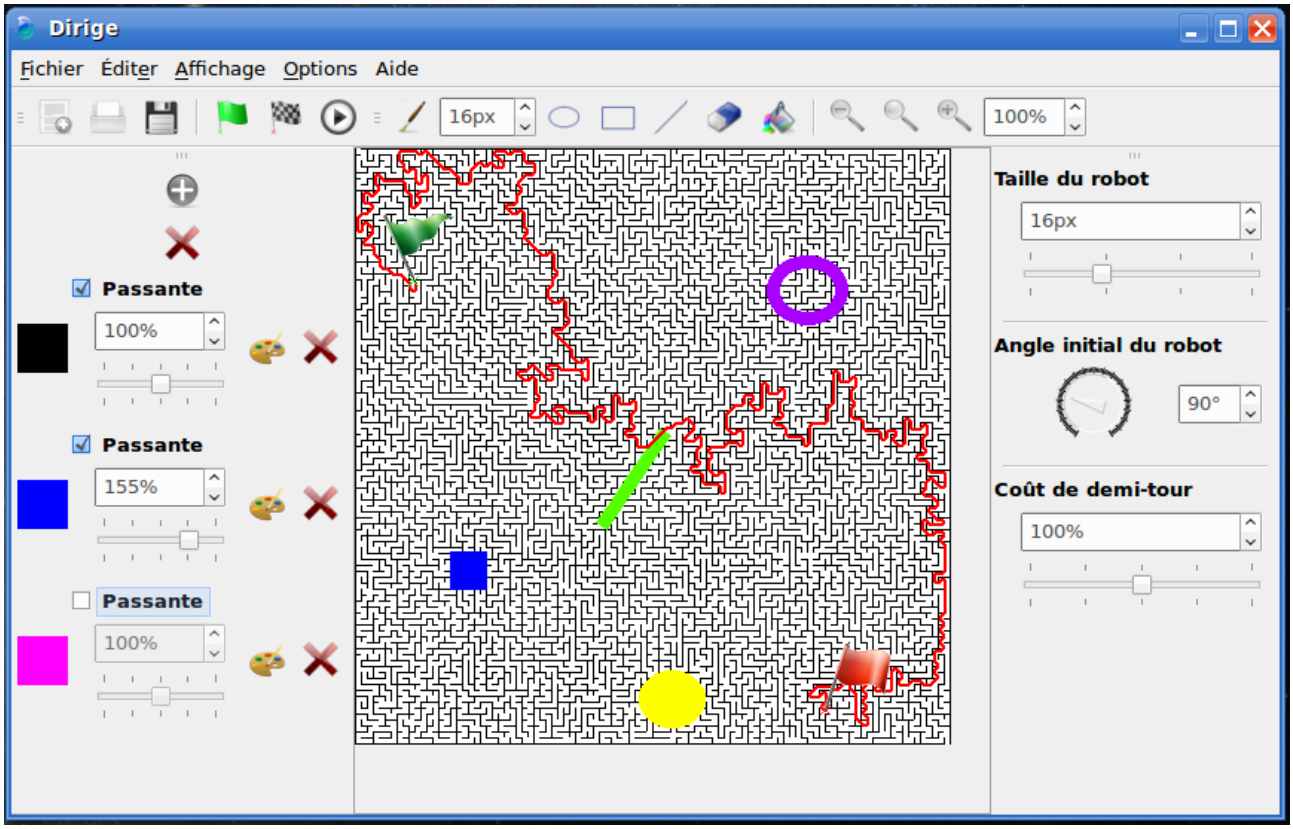

Illustration 1: Apparence générale de l'interface.

L'IHM est composée de différentes barres de tâches. Nous allons essayer d'expliquer brièvement le rôle de chacune d'entre elles.

Barre de menu principal

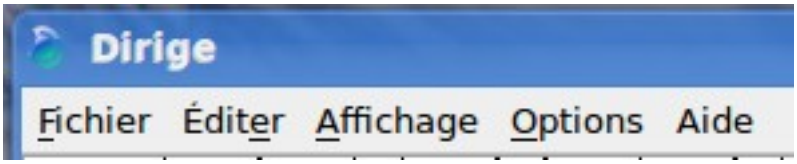

Illustration 2: Barre de menu principal

Ce menu permet d'accéder aux fonctions que l'on retrouve dans la plupart des logiciels.

•**Fichier** : le menu fichier permet entre autres de créer, d'ouvrir, d'enregistrer ou de fermer un document. Il permet également de quitter l'application.

•**Éditer** : à travers ce menu, on peut accéder aux fonctions d'édition de la carte que nous détaillons plus loin.

•**Affichage** : permet d'accéder aux fonctions de zoom sur l'environnement crée.

•**Options** :

•**Aide** : ce menu présente une bref explication de l'utilisation du logiciel, ainsi que des renseignements concernant la version et les auteurs de l'application.

Les icônes de gestion des fichiers

Illustration 3: Barre d'outil pour la gestion des fichiers

Les icônes permettent respectivement de créer un nouveau fichier, d'ouvrir et de sauvegarder un document.

Les icônes de lancement de la recherche du chemin

Illustration 4: Icônes de lancement de la recherche du chemin

Le drapeau vert correspond au point de départ, tandis que le drapeau suivant indique la position du point d'arrivée. Pour placer un drapeau sur un environnement, on clique sur le drapeau correspondant au point que l'on souhaite placer, puis on clique à la position souhaitée sur l'environnement.

Le bouton suivant permet de lancer la recherche du chemin le plus optimisé. Une fois l'environnement défini, les drapeaux précédemment placés ainsi qu'un éventuel paramétrage des configurations du robot, on cliquera sur ce bouton pour afficher le chemin calculé.

Les fonctions d'édition de l'environnement

Illustration 5: fonctions d'édition de l'environnement

Les boutons suivants permettent d'éditer l'environnement.

Pour dessiner des formes non-prédéfinies, on peut utiliser le pinceau, qui correspond au premier bouton.

L'encadré suivant correspond à la taille en pixel de la largeur d'un trait ou de la gomme.

Ensuite nous avons respectivement les boutons permettant de tracer, une ellipse, un rectangle, une ligne. Ces boutons s'utilisent comme dans la majorité des logiciels de dessin, comme par exemple GIMP ou Paint. L'utilisation est très intuitive.

La gomme permet d'effacer un tracé. Sa taille est réglable avec le premier encadré. Par exemple sur l'image ci-dessus, la taille est de 16 pixels.

Pour dessiner un rectangle ou une ellipse rempli à l'intérieur par une couleur, on clique sur la couleur souhaitée dans une barre de tâche que nous détaillons plus loin. Ensuite, on clique sur le pot de peinture avant de tracer une forme remplie.

Les outils de zoom

Illustration 6: Les outils de zoom

Nous avons intégré une fonctionnalité permettant de zoomer sur l'environnement que l'on édite. On peut ainsi éditer l'environnement avec plus de précision, ou avoir une vision plus globale dans le cas d'une carte relativement grande.

La première loupe permet de diminuer le zoom contrairement à la troisième qui permet de l'augmenter. La boîte permet de régler manuellement la valeur du zoom. Une valeur de 100% correspond à une vision à l'échelle réelle de l'environnement.

La gestion des couleurs

Nous avons dans la dernière maquette centralisé la gestion des couleurs dans une barre de tâche pour plus simplicité.

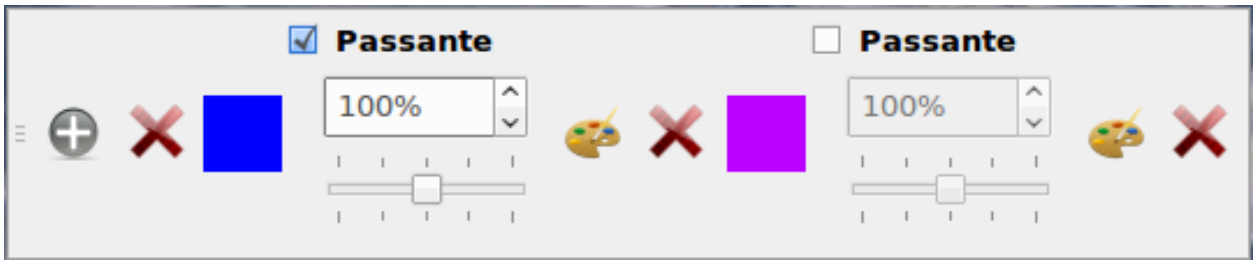

Illustration 7: Palette de couleur

La croix grise sur la gauche permet d'ajouter une couleur ainsi que les outils de gestions de cette couleur.

La croix rouge placée juste après la croix grise permet de supprimer toutes les couleurs déjà définies, après avoir confirmer ce choix dans la boîte de dialogue qui apparaît.

Pour ajouter une couleur, on clique sur la croix grise qui ouvre la palette de couleur de l'environnement de travail de l'utilisateur. Celui-ci choisit et valide la couleur qu'il souhaite utiliser dans la palette de couleur. La couleur apparaît ensuite avec des outils de gestion. Ces outils sont :

•Un carré contenant la couleur choisie.

•Une case permettant de cocher une option « Passante » : si la case est cochée comme par exemple pour la couleur violette sur l'image, cela signifiera qu'un obstacle de cette couleur ne pourra être franchi par le robot.

•Un encadré contenant une valeur en pourcentage : cette valeur signifiera pour le robot le « coût » associé à la traversée d'une unité de surface d'un obstacle de cette couleur. Concrètement, une valeur de 100% peut correspondre par exemple à un terrain plat, une valeur supérieure à 100% à une montée et une valeur inférieure à 100% à une descente. Ces valeurs n'ont pas à proprement parler d'unité afin de généraliser les cas. On peut imaginer que cette valeur correspond à l'énergie nécessaire à la traversée d'une unité de surface de l'obstacle. Par défaut, un environnement vierge est de couleur blanche. La couleur blanche est, par défaut, implicitement associée à la valeur 100%. Cependant, si l'utilisateur souhaite modifier cette valeur, il peut « ajouter » cette couleur est redéfinir sa signification pour le robot.

•Le curseur est un outil intuitif pour augmenter ou diminuer le coût associé à la couleur. Déplacer le curseur vers la gauche pour diminuer la valeur, et vers la droite pour l'augmenter.

•La palette permet de redéfinir la couleur en ouvrant la boîte de couleur propre à l'environnement de travail de l'utilisateur.

•Pour terminer, la croix permet de supprimer la couleur. Ainsi, elle n'aura plus de signification pour le robot. Les couleurs représentées sur l'environnement n'ayant pas de signification pour le robot sont associées par défaut à un coût de 100%.

Le réglages des paramètres du robot

Les paramètres du robot sont centralisés dans la même barre de tâche.

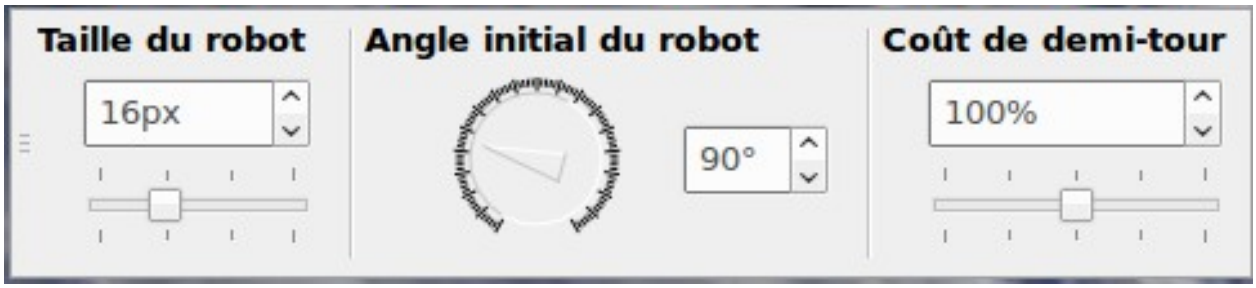

Illustration 8: Réglages des paramètre du robot

La taille de la largeur/longueur du robot peut être définie à partir de l'encadré et/ou du curseur. La largeur peut avoir une importance, notamment pour déterminer si l'espace nécessaire pour effectuer le déplacement est suffisant dans le cas d'obstacle bloquant (obstacle non passant). Si l'obstacle est non bloquant, la traversée totale ou partielle de l'obstacle suivant la largeur du robot, est envisagée par l'algorithme.

L'angle initial du robot par rapport à l'environnement peut être indiqué soit par l'aiguille ou soit par l'encadré. L'utilisation est relativement intuitive. Attention, la direction de la flèche n'indique malheureusement pas la direction initiale du robot. Si le robot a un angle initial de 0°, celui-ci sera tourné vers la gauche. Si le robot possède un angle initiale de 90°, il sera tourné vers le haut, avec un angle de 180° il sera tourné vers la droite, et avec un angle de 270° vers le bas. La valeur de l'angle peut être précisé au degrés près.

L'outil suivant permet d'indiquer le coût associé au robot lorsque celui-ci fait un demi-tour. Le curseur permet d'indiquer la valeur au même titre que l'encadré.

3.2)Module d'interprétation

Le module d'interprétation permet d'interpréter les actions de l'utilisateur. Les différentes parties qui le composent sont appelées par le module IHM précédemment présenté.

Ce module assure plusieurs fonctions :

La saisie de l'environnement : cette fonction consiste à permettre à l'utilisateur de dessiner des formes et de placer les points de départ et d'arrivée sur la feuille de dessin. Conformément au cahier des charges :

➢La taille de l'environnement est paramétrable. Cependant, des limites de tailles sont imposées soit pour des raisons de cohérence (une taille minimum doit permettre de placer les points de départ et d'arrivée), soit pour des raisons de ressources en termes de mémoire.

➢Les points de départ et d'arrivée sont placés par l'utilisateur.

➢Les obstacles sont définis par :

•Une nature : elle correspond au type d'obstacle. Comme nous l'avons évoqué précédemment, on identifie la nature d'un obstacle par sa couleur. L'utilisateur a la possibilité dans l'application de choisir la couleur qu'il souhaite pour dessiner un obstacle. La couleur d'un obstacle peut concerner le contour de celui-ci et/ou son intérieur.

•Une forme : une forme peut être représentée soit en utilisant des motifs prédéfinis (comme par exemple dessiner une droite, un rectangle, une ellipse) ou soit à l'aide d'un crayon ce qui offre une plus grande souplesse. Les formes sont dessinées au moyen de la souris sur la feuille de dessin.

•Une position : les obstacles peuvent être dessinés n'importe où sur la feuille de dessin.

La récupération des informations concernant le robot : le module se chargera de stocker les informations relatives au robot détaillées précédemment (voir la présentation du module IHM 3.1).

La conversion d'une image en données exploitables par le module algorithmique : ce module aura la responsabilité de traduire l'environnement sous forme de données exploitables pour effectuer des calculs. Par exemple, il associe à chaque pixel un coût renseigné par la couleur. Ce coût est défini par l'utilisateur dans le module IHM. Il identifie également les points de départ et d'arrivée placés sur l'environnement.

Le module d'interprétation permet entre autres de traduire les actions de l'utilisateur pour ensuite appeler le module algorithmique qui comme nous allons le voir, va effectuer toute la partie calcul.

3.3)Module algorithmique

Le module algorithmique permet d'effectuer les calculs nécessaires à la résolution du problème, à savoir trouver le chemin le plus optimisé.

Il existe différents algorithmes pour résoudre ce genre de problème. Par exemple, dans notre étude préalable nous avons mentionné et détaillé l'algorithme de Bellman-Ford, l'algorithme de Floyd-Warshall, ou l'algorithme d'A*. Dans notre application, nous avons utilisé l'algorithme de Dijkstra pour plusieurs raisons. En effet il s'avère être le plus adapté à notre cas parmi les algorithmes de recherche de chemin. Pour plus de détails concernant les différents algorithmes étudiés et des différentes raisons qui nous ont poussées à utiliser l'algorithme de Dijkstra, veuillez vous référer au dossier d'études préalables.

Comme nous l'avons dit précédemment, ce module sera activé à la demande du module d'interprétation. L'intérêt d'avoir séparé l'application ainsi est que ce module n'aura aucune notion d'image. Il recevra des données traduites par le module d'interprétation. Ainsi, ce module pourra être utilisé quelque soit le format de l'image. Une fois le travail terminé, il donnera au module d'interprétation la liste des points par lesquels le robot doit passer pour réaliser le trajet le plus optimisé. Le module d'interprétation se chargera ensuite de les représenter sur la feuille de dessin.

3.4)Extensions et améliorations possibles

L'architecture selon laquelle le produit est construit permet facilement d'effectuer des améliorations ou même d'ajouter des extensions sans avoir à modifier profondément le fonctionnement de celui-ci. Le code source est écrit en anglais afin de faciliter la compréhension et l'amélioration du produit. Dans cette partie, nous allons évoquer plusieurs idées d'améliorations et des extensions possibles. Les contraintes de délais ne nous ont malheureusement pas permis de les mettre en place.

Ajouter des paramètres au robot.

Il serait tout à fait envisageable d'ajouter d'autres paramètres au robot. Pour que ces paramètres puissent être ajoutés, il serait nécessaire de modifier les points suivants :

•L'interface graphique du produit devrait être modifiée pour que ces paramètres puissent être visibles et modifiables par l'utilisateur.

•Les nouveaux paramètres devraient être précisés dans la classe[2] Robot.

•Pour que les nouveaux paramètres puissent être pris en compte par le module

2 En programmation objet, une classe est un morceau de code qui permet de préciser les caractéristiques d'un objet. Un objet informatique est une représentation abstraite d'un objet réel (par exemple un robot) ou d'un objet abstrait (par exemple un nœud constituant le chemin).

algorithmique, il serait nécessaire de créer une classe héritant[3] de la classe Cost. La fonction[4] calculateCost devrait être écrite pour préciser comment les nouveaux paramètres vont influencer le calcul du chemin le plus optimisé.

Ajouter une nouvelle méthode de calcul du chemin le plus optimisé utilisable par l'algorithme de Dijkstra.

Pour se faire il faudrait comme précédemment évoqué créer une classe héritant de la classe Cost et écrire la nouvelle fonction de calcul. Il porrait également être nécessaire de modifier l'interface graphique pour rendre cette nouvelle méthode visible par l'utilisateur final.

Ajouter un nouvel algorithme de calcul de chemin le plus optimisé.

Il suffirait d'écrire une nouvelle classe implémentant cet algorithme en reprenant le modèle écrit pour l'algorithme de Dijkstra déjà écrit. On pourrait éventuellement modifier l'interface graphique pour permettre à l'utilisateur de choisir un algorithme parmi ceux existant.

Ajouter un nouveau format d'image pouvant être traité.

Pour l'instant, notre application ne peut gérer qu'une image au format Bitmap (BMP). Nous avons choisi ce format pour plusieurs raisons. La raison principale est que le format BMP n'est pas un format d'image compressé et garde donc l'intégralité des informations. De plus ce format est facilement manipulable au moyen de la bibliothèque Qt.

3 L'héritage en programmation objet permet à un objet "fille" d'hériter des propriétés d'un objet "mère".
4 Une fonction en programmation reçoit des paramètres, effectue des traitement sur ces données et retourne un résultat à l'objet appelant.

Générer un fichier contenant la liste des points faisant partie du chemin.

Il serait possible et intéressant de générer un tel fichier au format XML en plus de représenter le chemin sur la feuille de dessin. Ce fichier pourrait contenir les paramètres du robot définis dans l'interface. De cette manière, le programmeur du robot physique n'aurait qu'à développer sa propre application permettant de mouvoir son robot en fonction de la lecture d'une liste de points. Il s'appuierait sur ce fichier pour indiquer au robot comment atteindre le point d'arrivée de la manière la plus optimisée, à partir du point de départ et de la configuration du robot.

Implémenter une partie du projet dans un robot physique.

Avec la conception que nous avons choisi, il serait possible de réutiliser le module algorithmique ainsi que le module d'interprétation permettant de traduire une image en données exploitables pour effectuer les calculs. Ainsi il suffirait de donner au robot physique une image de l'environnement pour que celui-ci puisse l'interpréter et effectuer le déplacement le plus optimisé. On pourrait également envisager que le robot détecte l'environnement au moyen de capteurs avant de trouver le chemin le plus optimisé.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

4)Déroulement du projet

4.1)Organisation du groupe

4.1.1)Composition du groupe

L'équipe de travail est composée de trois membres :

Chef de projet : Alex-Mehdi ZAHID.

Analystes développeurs : Antoine GREA et Blon THO.

4.1.2)Répartition du travail

La répartition du travail a été faite par le chef de projet suivant différents critères. En effet, il a fallu tenir compte des compétences, des motivations et des exigences de chacun pour optimiser au mieux la durée du projet. Le projet a été séparé en lots entre les différents membres. Si vous souhaitez avoir plus de renseignements sur une partie du projet, pour l'améliorer ou pour tout simplement le comprendre, vous pouvez contacter les personnes correspondantes.

Antoine GREA :

✔Recherche d'une solution pour traiter le format d'image SVG.

✔Recherche d'une solution pour utiliser des greffons indépendants.

✔Graphiste et designer (image et mise en forme des dossiers).

✔Fusion des différentes parties du projet.

✔Développement de l'interface graphique.

Blon THO :

✔Développement de l'interface graphique.

✔Développement du lot permettant d'éditer l'environnement.

Alex-Mehdi ZAHID :

Développement :

✔Développement de l'interface graphique.

✔Développement du module d'interprétation (excepté du lot d'édition de l'environnement).

✔Développement du module algorithmique.

Mission de chef de projet :

✔Mise en place de l'organisation et du planning prévisionnel.

✔Suivi des travaux des développeurs.

✔Dialogue et compte rendu avec le client.

✔Rédaction du rapport final.

Eléments en commun :

✔Réalisation du cahier des charges.

✔Réalisation du dossier de concurrence.

✔Réalisation du dossier de conception.

✔Rédaction du rapport final.

✔Fusion des différentes parties du projet.

4.2)Difficultés rencontrées

Nous avons connu plusieurs difficultés dans l'élaboration du projet. Nous avons classé ces difficultés suivant deux grands axes : les difficultés d'ordre technique et les difficultés liées au travail d'équipe.

4.2.1)Difficultés techniques

Conception

La conception du projet à été un réel défi. Elle nous a pris plus de temps que nous avions prévu. Nous avons étudié de nombreuses stratégies pour mettre en place la solution. Nous avons ensuite fait notre choix parmi ces solutions pour trouver un bon compromis entre modularité et faisabilité. Bien souvent, plus une application est souple et modulable, plus la complexité lors de la réalisation est importante. C'est donc de cette manière que nous avons conçu notre application. Nous avons préféré mettre l'accent sur une facilité à faire évoluer l'application plutôt que de réaliser une application très vite opérationnelle et figée dans le temps.

A l'origine, l'application était constituée uniquement de deux modules : le

module IHM et le module de traitement de données. Le module de traitement devait initialement se charger de traduire l'image retournée par l'IHM, d'effectuer les calculs, de représenter le chemin sur l'image et de la transmettre à l'IHM qui l'afficherait. Cependant, nous avons pris conscience de la nécessité de revoir toute cette conception. En effet, un des inconvénients majeurs était que le format de l'image traitée aurait été figé dans le module algorithmique et qu'il serait par conséquent lié à un algorithme de recherche de chemin. L'application aurait été vite opérationnelle mais relativement limitée. Nous avons donc eu l'idée de scinder notre module de traitement en deux modules. Un module qui se chargerait de traduire l'image sous forme de données exploitables pour les calculs, et un module algorithmique qui serait chargé d'effectuer ces calculs pour trouver le chemin le plus optimisé. Ainsi, il est maintenant envisageable de créer d'autres modules d'interprétation pour d'autres formats d'image en se basant sur celui que nous avons développé. De plus, la partie algorithmique est indépendante. De cette manière, nous pouvons créer d'autres modules algorithmiques utilisant des algorithmes différents de celui de Dijkstra.

Nous vous invitons à consulter le dossier de conception ainsi que le code source pour plus de renseignements.

Module IHM

Le module IHM a demandé une réflexion consciencieuse. Effectivement, les fonctionnalités que l'on pouvait intégrer étaient nombreuses. De plus le cahier des charge stipulait de fournir à l'utilisateur une interface intuitive, personnalisable et adaptée à tous les environnements de travail.

L'initiation à l'utilisation de la bibliothèque Qt que nous avons suivi dans le module C++ à l'IUT nous a permis de nous imprégner des principales fonctionnalités de Qt. Toutefois, tout au long de l'implémentation du module IHM, une documentation approfondie des fonctionnalités de la bibliothèque Qt nous a été nécessaire.

La mise en place du zoom a présenté plusieurs difficultés. Le zoom proposé par la bibliothèque Qt permet seulement de zoomer sur une image. Seule la vue est agrandie. Cependant, lorsqu'on dessine sur l'environnement, avec une valeur du zoom différente de 100%, il se produit un effet de décalage. Les fonctionnalité de dessin de Qt ne prennent pas en compte la valeur du zoom. Il nous a donc fallu trouver une astuce pour corriger ce décalage.

La représentation des drapeaux est différente par rapport aux autres éléments de l'environnement. Effectivement, leur forme ne doit pas être reconnue par le module d'interprétation. Seule la position qu'ils renseignent est détectée. Il en est de même pour l'affichage du chemin trouvé. En effet, si l'utilisateur réutilise ou modifie l'environnement, le chemin précédemment tracé ne doit ni apparaître et ni être reconnu par le module d'interprétation lors du second calcul du chemin. Nous avons finalement trouvé une solution à ce problème. Cependant, dans la version que nous avons développé à ce jour, il persiste encore un bogue d'affichage : lorsqu'on effectue par exemple un zoom sur la carte, le chemin précédemment tracé est effacé en partie. Nous sommes toujours en train de trouver une solution à ce bogue.

Construire la liste des couleurs fût la tâche la plus difficile à mettre en place. Pour faire réaliser cette barre d'outils nous avons dû définir une sous-classe qui gère l'affichage d'une couleur. Ensuite nous avons construit une collection de couleur. Nous avons utilisé un conteneur de type « Map ». Ce conteneur permet de stocker des données en assurant l'unicité de chacune d'entre elles. Nous avons également dû faire face à d'autres problèmes liés à cette gestion des couleurs comme par exemple la suppression des couleurs. Comme vous pouvez le constater, une réflexion minutieuse et progressive fût nécessaire pour permettre à l'utilisateur de personnaliser l'utilisation des couleurs.

Module d'interprétation d'une image au format Bitmap

Les difficultés principales du module d'interprétation résidaient dans la partie permettant la traduction de l'image vers des données exploitables par le module algorithmique. Nous devions choisir une structure permettant de stocker un nombre important de données tout en étant très rapide lors de la lecture par le module algorithmique. C'est principalement pour cette raison que nous nous sommes tournés vers la Standard Template Library[5]. De plus nous avons utilisé le Design Pattern Singleton[6] pour la classe Robot, ce qui permet d'assurer une certaine cohérence et prévient des futurs bogues possibles si des améliorations sont envisagées.

Nous avons initialement commencé à concevoir et développer en parallèle à ce module un autre module permettant d'interpréter une image au format SVG.[7] Ce format d'image possède de nombreux avantages par rapport au format Bitmap (voir le dossier d'étude préalable). Cependant, ce format est relativement difficile à mettre en place tant pour éditer et manipuler une image (avec la bibliothèque Qt), que pour la traduire. En effet, nous avons perdu du temps dans le développement de ce module qui n'aboutira probablement pas avant le délai de livraison.

Module algorithmique

Une fois le module d'interprétation mis au point, nous avons implémenté et adapté l'algorithme de Dijkstra pour trouver le chemin le plus optimisé.

Une première approche a consisté à implémenter l'algorithme. Cependant, cette solution s'avérait très lente à l'exécution. En effet, après avoir réalisé plusieurs tests, nous sommes venus au fait qu'il était nécessaire d'optimiser l'implémentation de l'algorithme.

5 La STL est concrètement composée de diverses classes et fonctions permettant d'effectuer des opérations de bas niveau. Elle est développée par des professionnels qui ont pour objectif d'améliorer au maximum la rapidité des traitements.

6 Un design pattern est un patron de conception ou motif de conception destiné à résoudre les problèmes récurrents suivant le paradigme objet. Le design pattern singleton permet d'assurer l'instanciation unique d'un objet dont la classe utilise ce patron de conception.

7 SVG : *Scalable Vector Graphics*. signifie « graphique vectoriel adaptable », et définit une image à l'aide de formes simples. Peut contenir toute sorte d'informations XML. Il suporte la transparence et est extrèmement léger.

Une seconde approche proposée par M. Guérin a consisté à utiliser les tas de Fibonnacci. Les tas de Fibonnacci ne modifient en rien le fonctionnement de l'algorithme de Dijsktra. Ils nous ont seulement permis de stocker des informations de telle sorte qu'il soit très facile d'avoir accès à l'information que nous recherchons suivant un même critère. La compréhension du fonctionnement de ces tas n'est pas forcement évidente et a impliquée de nombreux efforts de recherche et de compréhension. L'implémentation n'étant pas facile non plus, nous avons préféré risquer de perdre quelques heures à rechercher l'existence d'une implémentation des tas de Fibonnacci pour résoudre ce type de problème. Il n'a pas été simple de rechercher cette information. Cependant nous avons finalement trouvé ce que nous recherchions sur le site Internet http://www.codeproject.com/KB/recipes/Dijkstras_Algorithm.aspx. L'auteur du programme est Max Winkler, un étudiant en modélisation de problèmes mathématiques en C/C++. La licence du programme autorise sa réutilisation dans d'autres projets. Bien sûr, pour pouvoir réutiliser et adapter ce code il a été nécessaire de le comprendre dans son intégralité.

Avec les tas de Fibonnacci, l'exécution du module algorithmique est quasi instantanée, contre plusieurs minutes avant son utilisation. Ils accélèrent considérablement les calculs ce qui permet de les rendre plus complexes en prenant en compte par exemple la largeur du Robot. La largeur du robot est une fonctionnalité relativement intéressante. Elle offre à l'utilisateur plus de possibilités et permet de rendre la simulation plus fidèle à la réalité. Par exemple, cela peut déterminer si le robot peut suivre un chemin sans entrer en collision avec d'autres obstacles du fait de sa largeur. De plus, si dans la définition des paramètres du robot, celui-ci peut chevaucher des obstacles de nature différente, l'algorithme envisagera ces situations. Ainsi, nous sommes certains que le chemin retourné par l'algorithme puisse être parcouru par le robot et qu'il soit le plus optimisé. Cependant, cela implique des calculs supplémentaires. Plus la largeur du robot est importante, plus le temps d'exécution augmente.

4.2.2)Gestion de l'équipe

La gestion de l'équipe n'est pas forcément quelque chose de naturel et évident. Aussi, nous avons rencontré plusieurs difficultés. Nous aborderons ici les points à améliorer dans nos futurs projets.

La répartition du travail

Comme nous l'avons dit, la répartition du travail est relativement délicate. Il est nécessaire de prendre en compte divers éléments (compétences, motivation ...). Comme dans la majorité des travaux en équipe, il est difficile d'avoir une répartition exacte du travail entre les différents membres.

Dans notre cas, nous avons dans un premier temps travaillé en commun pour définir le projet avec plus de précisions, en rédigeant le cahier des charges ainsi que le dossier d'étude préalable. Ensuite, nous avons établi une première approche au niveau de la conception afin de déterminer l'ensemble des tâches à réaliser. Nous avons

ensuite discuté ensemble pour nous les répartir. Dans un second temps, chacun s'est auto-formé plus précisément sur les technologies correspondant aux différentes tâches à réaliser avant de commencer le développement du projet.

Initialement, nous avions prévu de développer un module d'interprétation Bitmap et un module d'interprétation SVG. Cependant, nous avons certainement laissé un peu trop de souplesse aux demandes d'un membre du groupe, dans le développement du module permettant de gérer le format d'image SVG, qui n'aboutira vraisemblablement pas avant les délais prévus. En effet, la réalisation du projet s'est révélée beaucoup plus technique que prévu. Nous n'avions pas envisagé une exécution relativement lente de l'algorithme de Dijkstra. De plus, l'état de santé d'un des membres n'a pas arrangé la situation. Ainsi, le chef de projet a pris la décision de stopper le développement du module SVG pour une durée indéterminée. De plus nous avons été contraints de prendre une charge de travail plus importante de manière à pouvoir compenser ce retard. Il nous faudra donc à l'avenir prévoir une marge de sécurité un peu plus importante de manière à ne pas se retrouver dans ce genre de situation.

La communication dans le groupe

La communication s'est globalement bien passée. La majorité des décisions importantes ont été prises ensemble. Cependant, il a parfois été difficile pour certains d'abandonner leurs idées au profit du groupe, comme par exemple lors de l'arrêt imprévu du développement du module d'interprétation du format SVG.

4.3)Comparaison entre le planning prévisionnel et le planning réel

Il est généralement difficile de suivre exactement le planning prévisionnel. Son élaboration est d'autant plus difficile. Aussi, nous vous proposons dans cette partie d'évaluer les écarts entre le planning prévisionnel[8] et le planning réellement réalisé :

8 Note importante : deux erreurs se sont glissées dans le planning prévisionnel présent dans le cahier des charges. La fin de la rédaction du dossier de conception était prévue pour le 10 janvier et la fin de la tâche concernant la réalisation de la deuxième maquette était prévue pour le 18 janvier et non pas le 14 décembre.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

		Planning prévisionnel		Planning réalisé	
TPÉ	**Nom**	Démarré	Terminé	Retard en jours	Avance en jours
1	Découverte du sujet.	06 Nov	16 Nov		
2	Premier contact avec le client.	17 Nov	17 Nov		
3	**Réalisation du cahier des charges.**	17 Nov	20 Nov		
3.1	▹ Rédaction	17 Nov	17 Nov		
3.2	▹ Réalisation de la première maquette de l'interface graphique.	17 Nov	18 Nov		
3.3	▹ Réalisation de ce planning.	18 Nov	20 Nov		
3.4	▹ Mise en forme.	20 Nov	20 Nov		
4	**Formation aux technologies nécessaires à la réalisation du projet.**	20 Nov	25 Nov		
4.1	▹ Étude des algorithmes de type Dijkstra.	20 Nov	23 Nov		
4.2	▹ **Réalisation du dossier d'étude préalable.**	23 Nov	25 Nov		
4.2.1	◦Recherche d'autres algorithmes.	23 Nov	24 Nov		
4.2.2	◦Précision des choix technologiques.	24 Nov	24 Nov		
4.2.3	◦Mise en forme.	24 Nov	25 Nov		
5	Deuxième contact avec le client pour confirmer le cahier des charges.	14 Déc	14 Déc		
6	**Réalisation du dossier de conception.**	14 Déc	10 Jan	7	
6.1	▹ Réalisation éventuelle de la deuxième maquette de l'interface graphique.	14 Déc	18 Jan	9	
7	***Développement de l'application.***	14 Déc	25 Fév	19	
7.1	▹ Développement de la partie graphique.	14 Déc	17 Déc	22	
7.2	▹ Développement de la partie traitement de données.	14 Déc	19 Fév		17
7.3	▹ Elaboration et développement d'une troisième maquette graphique	01 Mar	16 Mar		
7.4	▹ Tests.	22 Fév	25 Fév	26	
8	Élaboration du rapport final.	25 Fév	02 Mar	15	
9	Présentation de l'application.	25 Mar	25 Mar		

4.3.1)Retard

Nous pouvons remarquer que les délais ont correctement été respectés jusqu'à la réalisation du dossier de conception. Le retard de cette tâche a entraîné le retard d'autres tâches.

Réalisation du dossier de conception :

Nous avons eu un retard d'une semaine. Nous désignons ici le retard de la rédaction de la première version de ce dossier. En effet, le dossier a ensuite évolué au cours du développement de la solution.

Le retard peut s'expliquer par le fait que nous avons conçu notre application sur un modèle plus modulable que celui qui était initialement prévu. Il fût presque nécessaire de revoir une grande partie de la conception déjà établie. La complexité de la conception étant plus importante, cela a nécessité un peu plus de temps de réflexion pour élaborer plusieurs stratégies et sélectionner la meilleure.

Réalisation de la deuxième maquette de l'interface graphique

Nous n'avons pas pris de retard à proprement parlé, puisqu'initialement, nous avions prévu de réaliser cette deuxième maquette que si le temps nous le permettait. Néanmoins, cette deuxième maquette a été créée en parallèle du développement du lot d'édition de l'environnement. La tâche grisée,« Elaboration et développement d'une troisième maquette graphique » est venue se rajoutée en fin de développement du projet. En effet, nous voulions développer une interface plus ergonomique et intuitive que la dernière maquette réalisée.

Développement de la partie graphique

Cette partie concerne le développement des fonctionnalités permettant d'éditer la carte. Nous avons eu un retard de 22 jours. Nous avons rencontré quelques soucis, mais ils n'expliquent pas à eux seuls ce retard important. Nous pensons que nous avons été un peu optimistes dans les délais prévus pour cette tâche dans le planning prévisionnel. En effet, nous ne maîtrisions pas encore la technologie utilisée lors de la réalisation du cahier des charges et les délais pour cette tâche ne représentaient qu'un ordre idée par rapport aux autres tâches du projet.

Tests

Nous désignons ici par tests, l'ensemble des tests effectués sur l'ensemble de l'application. Cela concerne l'IHM, la partie graphique ainsi que le module algorithmique. En réalité les tests effectués pour la partie algorithmique ont été effectués en parallèle à son développement et ont pris fin à la même date. Le retard concerne les tests de communication entre les différents modules. Le développement de la partie graphique a été plus long que prévu, et ce retard a naturellement repoussé les tests.

Élaboration du rapport final

Contenu du fait que nous avions plusieurs retards dans différentes tâches, nous ne pouvions terminer la description de l'application dans son ensemble.

4.3.2)Avance

Développement de la partie traitement de données

Bien que nous avions dû faire face à une difficulté majeure dans le développement du module algorithmique, nous avons eu une avance de 19 jours. En effet, nous avions fini l'implémentation simple de Dijkstra dès le 22 janvier. Ceci nous a permis d'avoir une marge de temps plus importante pour optimiser l'algorithme. De plus la réutilisation d'un code déjà écrit et opérationnel à certainement contribuer à cette avance.

Alex Mehdi ZAHID zahid.alexmehdi@gmail.com
Antoine GRÉA grea09@gmail.com
Blon THO tho_blon@hotmail.com

5)Conclusion

Le produit développé :

L'application que nous avons développé est entièrement fonctionnelle. Les objectifs ont été atteints.

L'utilisateur peut modéliser un environnement relativement poussé à l'aide des fonctionnalités de dessin et définir de lui même un point d'arrivée et un point de départ de manière intuitive. Il peut, s'il le souhaite configurer les paramètres du robot avant de lancer la recherche du chemin le plus optimisé. L'application affichera toujours le chemin le plus optimisé si celui-ci existe, grâce à l'utilisation de l'algorithme de Dijkstra qui prendra en compte à la fois les paramètres du robot ainsi que la configuration de l'environnement. De plus, comme nous l'avons vu, l'architecture que nous avons mis en place permet d'envisager de nombreuses évolutions et réutilisations de l'application relativement intéressantes.

Ce que nous avons appris lors de la réalisation de ce projet :

Le projet DIRiGe est globalement une bonne expérience. Il nous a permis de mener un projet informatique de la conception à la réalisation.

Nous avons pu mettre en pratique les cours de C++, d'UML et de gestion de projet ainsi que des cours de mathématiques discrètes (théorie des graphes). L'élaboration de ce projet, nous a donné une brève simulation du développement en équipe, d'un projet assez conséquent : élaboration d'un dossier d'étude préalable, réalisation d'un dossier de conception, séparation du projet en plusieurs tâches... Au cours de ce projet, nous avons appris à compléter notre formation en étant capable de nous auto-former sur de nouvelles technologies et a devenir autonomes sur un projet relativement complexe. Même si comme nous l'avons vu, certains points peuvent être améliorés, le projet que nous avons mené dans le cadre des projets tutorés restera sans doute une bonne expérience pour notre future vie professionnelle.