**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

# Dossier de conception

## *Optimisation de trajet pour un robot*

## DiRIGe

**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

Ce document a pour objectif de décrire les spécifications fonctionnelles du projet afin d'aider au développement de l'application. Pour atteindre cet objectif, ce document détermine les différents acteurs intervenant dans l'application, les cas d'utilisations ainsi que leur exécution. Ce document présente également quelques prototypes d'interfaces graphiques.

Le projet a deux principaux objectifs :

- ✔ Réaliser une application permettant à un utilisateur de saisir un plan de l'environnement dans lequel évoluera 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) Acteur

## 2.1) Utilisateur

L'utilisateur élabore un environnement dans lequel un robot pourra évoluer. Une fois l'environnement définit, il configure les paramètres du robot et lance la recherche du chemin le plus optimisé.

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

# 3) Cas et scénarios d'utilisation

## 3.1) Cas d'utilisation

### 3.1.1) Création de l'environnement

*Acteurs* : ***Utilisateur***

*Intention* : Définir un environnement.

*Scénario* :

✔ Éditer un nouvel environnement.

✔ Modifier un environnement à partir d'un environnement existant.

### 3.1.2) Présentation du trajet le plus court

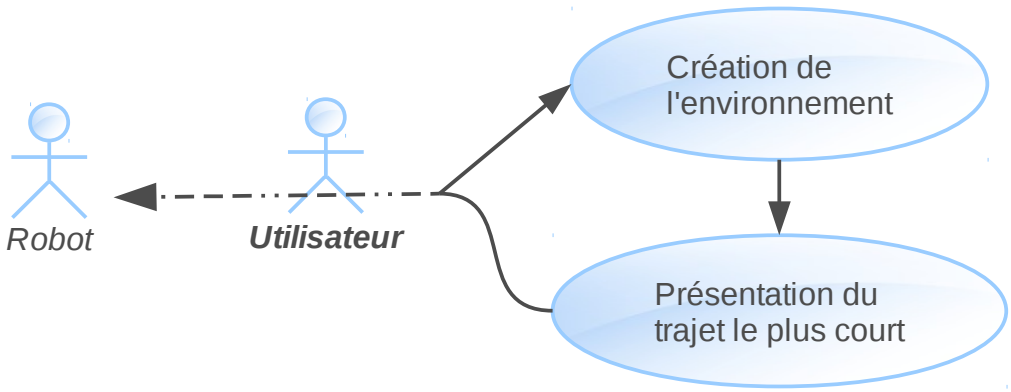
*Acteurs :* ***Utilisateur***

*Intention :* Connaître le chemin le plus optimisé en termes de temps de parcours pour robot ayant une configuration précise dans un environnement définit.

*Scénario* :

✔ Déterminer le chemin le plus optimisé.

## 3.2) Diagrammes des cas d'utilisation



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

## *3.3) Scénarios détaillés*

### *3.3.1) Création de l'environnement*

*But :*

Définir un environnement.

*Résumé :*

✔ Éditer un nouvel environnement.

✔ Modifier un environnement à partir d’un environnement existant.

*Acteurs :*

***Utilisateur***

*Pré-conditions :*

✔ Aucune.

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

*Description :*

***Séquence nominal : Éditer un nouvel environnement***

Étape 1 : Ouvrir un nouveau document.

L'utilisateur ouvre un nouveau document.

Si un document est déjà ouvert, traiter l'exception

Exception 1 : ExistingFile

Étape 2 : Définition de la taille de l'environnement.

L'utilisateur définit la taille de la carte[1].

Étape 3 : Définition des obstacles

L'utilisateur dessine les différentes obstacles. On identifie la nature de l'obstacle part sa couleur. L'utilisateur définit également les points de départ et d'arrivée.

***Séquence alternative : Modifier un environnement à partir d'un environnement existant***

Étape 1 : Ouverture d'un document existant.

L'utilisateur ouvre un document existant.

Si le format du document n'est pas pris en charge par le programme, traiter l'exception

Exception 2 : IncorrectFormat

Les étapes 2 et 3 se déroulent comme dans le scénario principal.

1 *Note* : La taille de la carte ne peut être nulle. Dans la cadre du module bitmap elle aura une taille minimale de deux pixels permettant de déterminer le point de départ et le point d'arrivée.

### *Exceptions*

Exception 1 : ExistingFile :

L'ouverture d'un nouveau document nécessite la fermeture préalable du document existant. Le système envoi un message pour demander la sauvegarde ou non du document existant.

Exception 2 : IncorrectFormat :

Le format de l'image chargé n'est pas pris en charge par le module permettant d'interpréter l'environnement. Le système envoi un message pour avertir que le format est incorrect. L'image n'est pas chargé.

### *Post-conditions*

✔ Un environnement est définit avec un point de départ et un point d'arrivé.

### *Diagrammes*

#### ***Diagramme de séquence***

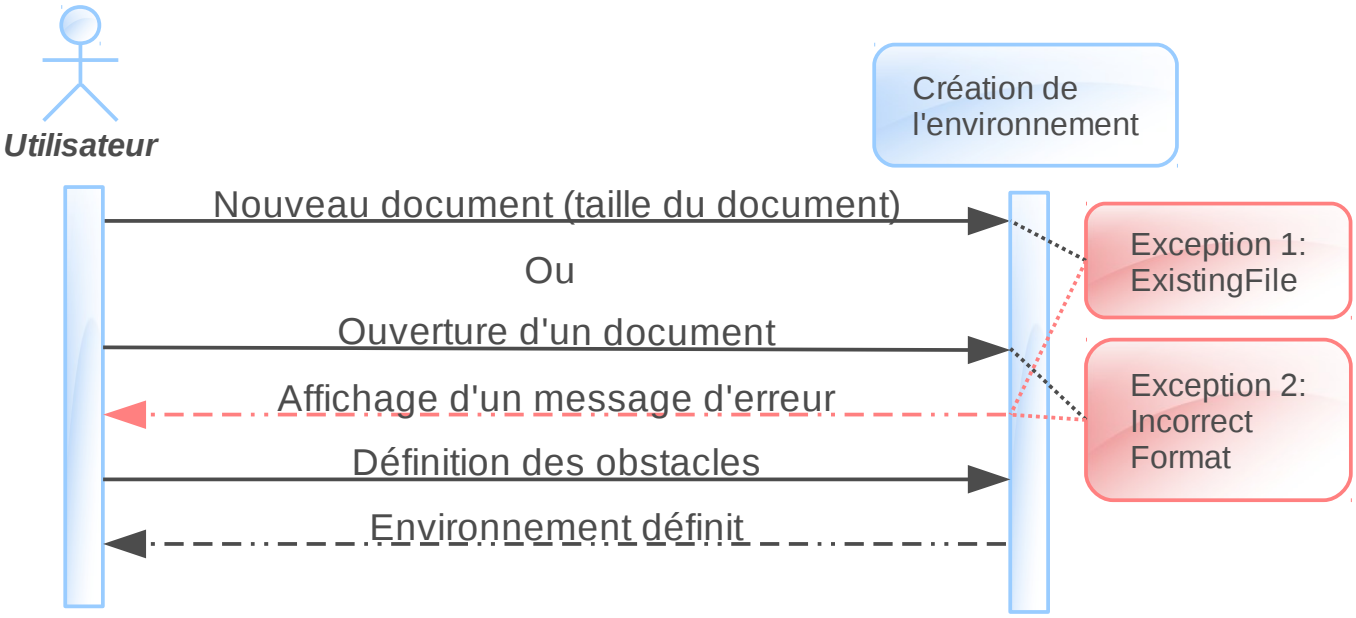



*Diagramme 1: Diagramme de séquence du cas d'utilisation « Création de l'environement »*

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

### *3.3.2) Présentation du trajet le plus court*

#### *But*

Déterminer le chemin le plus optimisé

#### *Résumé*

Connaître le chemin le plus optimisé en termes de temps de parcours pour robot ayant une configuration précise dans un environnement définit et le présenter.

#### *Acteurs*

***Utilisateur***

#### *Pré-conditions*

✔ Un environnement est définit.

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

*Description*

***Séquence nominal : Déterminer le chemin le plus optimisé***

Étape 1 : Configuration des caractéristiques du robot

L'utilisateur configure les caractéristiques du robot.

Étape 2 : Lancement de la recherche du chemin

La procédure de recherche du chemin le plus optimisé en terme de temps de parcours pour le robot précédemment configuré est lancée.

S'il n'existe pas un point de départ, traiter l'exception

Exception 1 : NoStartPoint

S'il n'existe pas un point d'arrivé, traiter l'exception

Exception 2 : NoFinishPoint

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

### *Exceptions*

Exception 1 : NoStartPoint :

L'environnement ne contient pas de point de départ. Le système envoi un message pour avertir qu'il n'existe pas de point de départ. La simulation a échoué.

Exception 2 : NoFinishPoint :

L'environnement ne contient pas de point d'arrivée. Le système envoi un message pour avertir qu'il n'existe pas de point d'arrivé. La simulation a échoué.

### *Post-conditions*

✔ Un chemin est représenté sur l'environnement.

✔ Si une erreur est survenue l'utilisateur est avertit.

### *Diagrammes*

#### ***Diagramme de séquence***

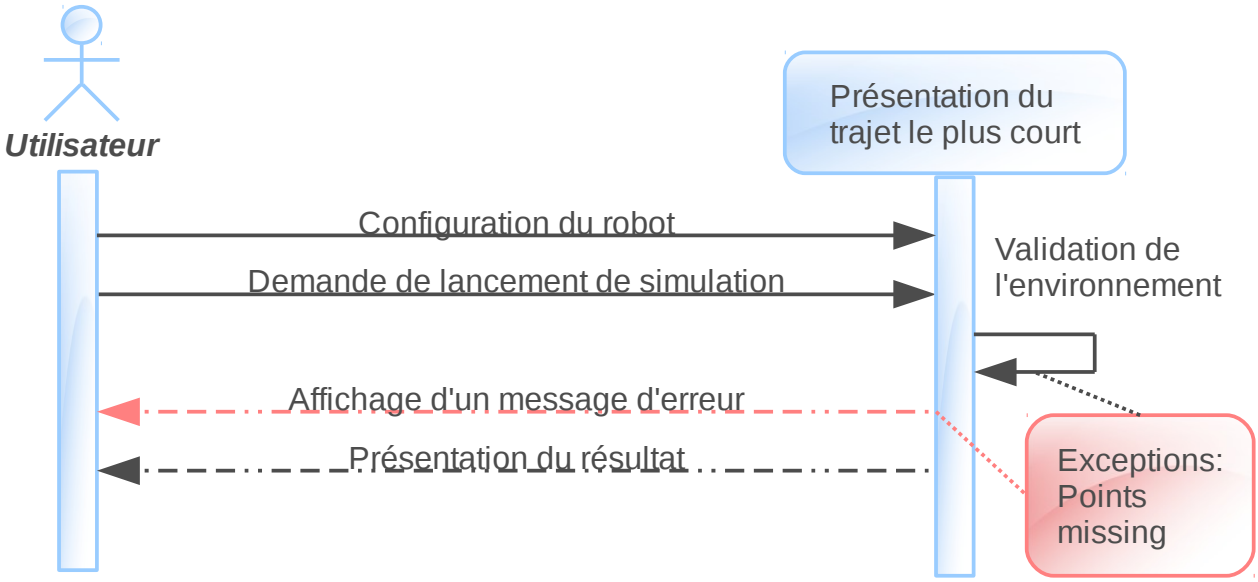



*Diagramme 2: Diagramme de séquence du cas d'utilisation « Présentation du trajet le plus court »*

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

# 4) Architecture

## 4.1) Modules

L'application sera composée de trois modules : IHM, Interprétation, Algorithme.

[2]

*Diagramme 3: Modules du programme*

Cet architecture permet une meilleure lisibilité du code, ainsi qu'une meilleure évolution et maintenance de l'application. Nous avons scindé le programme de manière à pouvoir faire évoluer les parties indépendamment des autres.

### 4.1.1) IHM

Le module Interface Homme Machine, active les différentes parties de l'application concernées, en fonction des actions de l'utilisateur. Elle assure également la gestion des fichiers (création, ouverture, fermeture ...) et la fermeture de l'application.

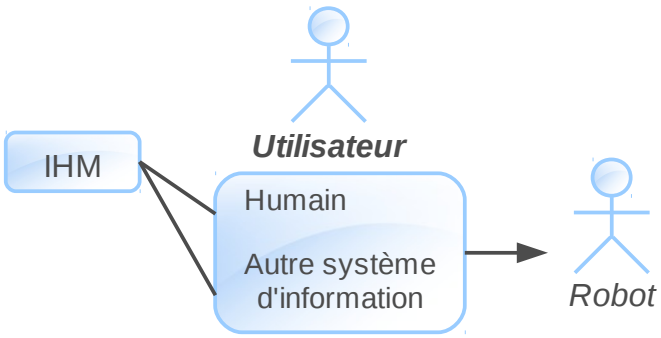



*Diagramme 4: Détail du module IHM*

2 *Note :* Afin de laisser les diagrammes légers on utilise un trait simple pour le liaisons bivalentes.

### *4.1.2) Interprétation*

Ce module est interchangeable. Il peut être décliné en plusieurs versions différentes. Une version permettant de gérer des images Bitmap, une autre permettant de gérer des images vectorielles... Il définit comment l'environnement est interprété et la manière dont il sera stocké. Il peut redéfinir des éléments du module algorithme qui lui sont spécifiques.

#### *Bitmap*

Module du support des images BMP[3]. Ce module travaille avec une notion de correspondance obstacle-couleur pour traiter le problème. Il travaille à l'échelle du pixel. C'est la couleur du pixel qui permettra de déterminer le coût associer à sa traversé par le robot.

#### *Vectoriel*

Module du support des images SVG[4]. Ce module travaille avec une notion de formes pour traiter le problème. Les formes sont extraites du SVG, puis traitées géométriquement à l'aide d'un quadrillage macroscopique pour accélérer le traitement. Les fonctions de calcul de coût sont récursives[5], mais sont dichotomiques[6] et donc restent légères.

### *4.1.3) Algorithme*

Ce module est le cœur du programme. Il contient des fonctions de calcul de coût et une fonction implémentant l'algorithme de Dijkstra. Il retourne le chemin calculé au module Interprétation.

---

3 BMP : *Bitmap.*
L'image est stocké sous forme brute, c'est à dire que chaque pixel a sa couleur stocké sous une forme composite de couleur élémentaire. Ceci est très gourmand en espace mémoire mais très performant en ressource processeur.

4 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. Il est cependant gourmand en ressources processeur pour le calcule des formes complexes.

5 Fonction Récursive :
Fonction qui fait appelle à elle même .

6 Dichotomie :
La dichotomie (« couper en deux » en grec) est un processus itératif ou récursif de recherche où, à chaque étape, on coupe en deux parties (pas forcément égales ) un espace de recherche qui devient restreint à l'une de ces deux parties.

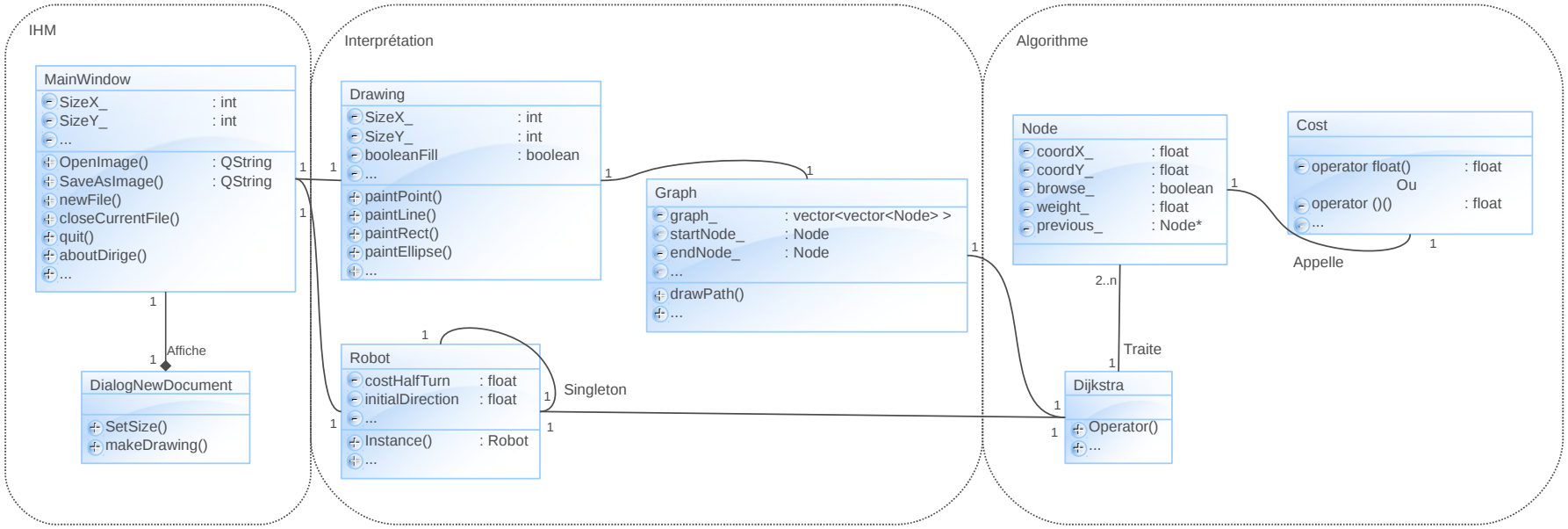
**Alex Mehdi ZAHID**
zahid.alexmehdi@gmail.com

**Antoine GRÉA**
grea09@gmail.com

**Blon THO**
tho_blon@hotmail.com

## *4.2) Diagramme de Classes*

Ce diagramme n'est ici qu'à titre indicatif. Il peut faire l'objet de changements lors du développement de l'application. Nous avons omis volontairement certaines classes, certaines méthodes et attributs pour une meilleure lisibilité.



### *4.2.1) Dans le module IHM*

- ✔ *MainWindow* : cette classe contient l'interface. Elle assure la majorité des responsabilités de l'IHM vu précédemment.
- ✔ *DialogNewDocument* : cette classe représente une boîte de dialogue ouverte lors de la création d'un nouveau document. Elle demande la taille en pixel du document à créer.

### *4.2.2) Dans le module interprétation*

- ✔ *Drawing* : Cette classe assure la partie dessin. Elle permet de définir les obstacles présents sur l'environnement ainsi que le placement des points de départ et d'arrivé.
- ✔ *Robot* : cette classe stockera les paramètres du robot saisis dans le module IHM. Il est important de souligner que la valeur des coûts associés au différentes natures d'obstacle seront définis dans les paramètres du robot. En effet, c'est dans la configuration du robot que seront définis les réactions de celui-ci vis à vis des différents types d'obstacles.
- ✔ *Graph* : la classe Graph traduira l'image édité, sous forme de tableau de nœuds. Elle vérifie la présence des points clés (point de départ et d'arrivé) sur l'environnement. Elle recevra également le chemin calculé par le module algorithmique pour le dessiner le chemin sur l'image affiché par le module IHM.

### *4.2.3) Dans le module algorithmique :*

- ✔ *Node* : un nœud est l'unité de base du graphique. Cette classe contient les attributs nécessaires pour être traité par l'algorithme Dijkstra.
- ✔ *Cost* : on associe à chaque nœud une classe Cost. Cette classe est un foncteur. Elle permet de calculer le coût associer au passage d'un nœud à un autre en prenant en compte la nature du nœud et la direction du robot.
- ✔ *Dijkstra* : la classe va recevoir un ensemble de nœuds. Elle appliquera l'algorithme de Dijkstra pour trouvé le chemin le plus optimisé. Le graphe sera ensuite transmit au module d'interprétation qui dessinera le chemin.

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

## *4.3) Communication inter-modules*

### *4.3.1) Interface IHM-Interprétation*

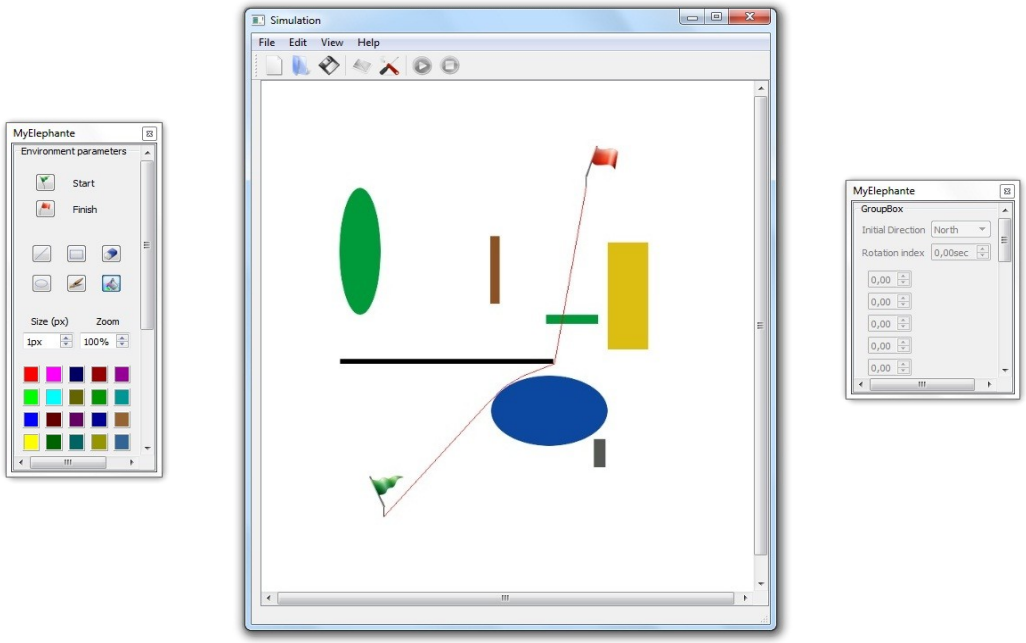
L'IHM se connectera aux slots[7] du module Interprétation. Dans l'autre sens une partie du module Interprétation est incrustée dans le cadre de dessin et affiche l'environnement.

### *4.3.2) Interface Interprétation-Algorithme*

Le module interprétation lancera Dijkstra avec des paramètres comme un ensemble de nœuds. Dijkstra retourne un chemin qui est récupéré par le module appelant.

## *4.4) Prototype d'IHM*

Nous avons définit un prototype d'interface graphique. Cette image est ici uniquement à titre indicatif.




Cette interface permet à l'utilisateur de placer les différents composants graphique de manière personnalisé.

7 Slot :
Fonction appelé automatiquement dés qu'un signal est émit, par exemple quand un bouton est pressé.