**A \* ALGORITHME EN C**

**Installation du projet sous windows**

Installer mingw64 + gcc

Voir le site suivant:

<https://www.msys2.org/>

SDL2 (Interface grapique)

Pour installer SDL2 :

-> <https://www.libsdl.org/download-2.0.php>

-> Dans development Libraries: SDL2-devel-2.0.14-mingw.tar.gz (MinGW 32/64-bit)

-> Dézipper x86\_64-w64-mingw32 dans le projet

SDL2\_TTF (Interface grapique - Gestion des Fonts)

Pour installer SDL2\_TTF:

-> <https://www.libsdl.org/projects/SDL_ttf/>

-> Dans "Development Libraries" : SDL2\_ttf-devel-2.0.15-mingw.tar.gz (MinGW 32/64-bit)

-> Dézipper x86\_64-w64-mingw32 dans le projet

**Architecture du projet**

Les dossiers

/

Le dossier root contient le A\_LIRE.txt & le makefile

/bin

Il contient les .dll des librairies SDL2 & SDL\_TTF

/include

Il contient les .h des librairies SDL2 & SDL\_TTF

/lib

Il contient les .a des librairies SDL2 & SDL\_TTF

/share

Il contient les .m4 des librairies SDL2 & SDL\_TTF

/src

Contient le code & l'exécutable du projet & la font

Makefile

Compiler

La commande make à la racine du projet

L'exécutable généré se trouvera dans le dossier src

Clean

Nettoie le dossier ./src si besoin avec la commande make clean

**Lancement du projet**

Clique droit sur l'exécutable du projet puis "Exécuter en tant qu'admin"

**Comment ça fonctionne**

1. Lorsque le programme démarre il faut renseigner des coordonnées:

Il y a les coordonnées des coffres à renseigner (Les coffres sont des fausses pistes ce n'est pas le vrai trésor!!!). L'algorithme est programmé pour suivre:

point\_de\_départ -> un des coffre -> un autre coffre -> un autre coffre -> vrai trésor

Il faut aussi renseigner le point de départ et le trésor.



Les coordonnées à rentrer sont de la forme "numéro\_ligne numéro\_colonne"

La première et la première colonne comment par 0.

Lors toutes les coordonnées sont correctes le programme démmare.

ATTENTION: il n'y a pas de contrôle stricte sur les données (ce n'est pas le but de l'énoncé!!!) On considère que les données entrées par l'utilisateur sont correctes!!!

2. L'interface graphique:

Lorsque l'interface graphique se lance elle est en pause et elle est blanche, c'est normal!

**Il faut cliquer sur le bouton de la souris gauche pour avancer d'une étape à chaque fois et voir l'algorithme évoluer sur la carte!!!**



Voici la légende des couleurs:

Vert foncé : Fôret

Vert clair : Trésor! Le vrai trésor est ici!

Violet : Coffre vide (fausse piste!)

Bleu : Rivière

Gris : Montagne

Orange : Position de départ

Rouge : Piège

Jaune : Position acuellement en cours d'analyse par l'algorithme c'est le "noeud courant"

Blanc: Terrain plat

A savoir:

- Les pièges sont infranchissables!

- On peut franchir des montagnes des rivières des forêts mais le cout est énorme c'est en dernier recours qu'on les franchis!

**Légende des couleurs des écritures :**



- En vert c'est les positions qui sont actuellement dans la table ouverte.

- En rouge se sont les positions qui ont déjà été étudiées. Elles sont dans la table fermées.

- En rouge sur fond jaune c'est la position en cours d'étude!

Chaque case à 3 nombres:

POIDS:HEURISTIQUE:COUT\_TOTAL

**Le poids:** Le poids est l'addition de la distance entre deux noeuds + le cout de la case

Exemple :

Sur le schema la case grise au dessus de la case de départ en orange :

610:26:636

Le poids est de 610 par qu'on considère que franchir une montagne à un coût de 600 et aller sur cette case c'est une distance de 10 (600 & 10 sont des valeurs que l'on a choisit arbitrairement).

Exemple 2 :

Sur le schema la case grise au dessus à droite de la case de départ en orange :

615:25:640

Le poids est de 615 par qu'on considère que franchir une montagne à un coût de 600 et aller sur cette case c'est une distance de 15 (on considère qu'un déplacement en diagonale est plus coûteux qu'un déplacement en ligne droite, encore une fois on a choisit arbitrairement les valeurs).

Exemple 3 :

Sur le schema la case blanche à droite de la case de départ en orange :

10:26:36

Le poids est de 10 par qu'on considère que franchir un terrain plat à un coût de 0 (aucun coût) et aller sur cette case c'est une distance de 10 (on considère qu'un déplacement en ligne droite est de 10).

Heuristique:

L'heuritique c'est tout simplement la distance entre la case est l'objectif à atteindre:

|Heuristique| = |x\_arrive - x\_depart| + |y\_arrivee - y\_depart|

COUT\_TOTAL:

Le coût total est :

COUT\_TOTAL = POIDS + HEURISTIQUE

**REMARQUE:**

Dans le cadre de l'exercice notre algorithme est optimisée pour priviligier les "terrains plats" des obstacles (montagnes, rivières, fôrets)....Ces derniers ont des coûts très hauts pour éviter de les franchir le terrain plat n'a aucun coût. De ce fait, l'heuristique et la distance entre les cases est mineur par rapport au type de cases dans notre cas pour la démonstration.

**3. Du départ au trésor!**

Lorsque l'algoritheme atteint un coffre et il affiche le meilleur chemin en console et sur l'interface graphique en orange, il se met en pause pendant quelques secondes. Au prochain clique gauche de la souris il continue son exécution avec un nouveau point de départ et un nouveau point d'arrivée et il réinitialise toutes les données.

Dans le cas ou c'est finalement le trésor qu'il trouve. Le programme affiche le dernier meilleur chemin en orange et en console et se met en pause. Un autre clique gauche fermera le programme.



Ici le chemin en orange représente la meilleure solution du point de départ jusqu'au premier coffre!