Ecole Publique d'Ingénieurs en 3 ans

Rapport

PROJET GRAND PRIX

le 31 mai 2022

Saad Cheddad
Saad.cheddad@ecole.ensicaen.fr
Mohamed Chergui
Mohamed.chergui@ecole.ensicaen.fr

Enseignant référent : Éric Ziad-Forest

Enseignant référent : Sébastien Fourey

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

www.ensicaen.fr

TABLE DES MATIERES

TABLE DES FIGURES

TABLE DES TABLEAUX

1. Introduction

1.1. Contexte

Afin d'améliorer nos compétences en algorithmique, langage C et gestion de projet, nous nous engageons dans ce challenge de programmation. En fait, cela nous a permis d'appliquer des concepts que nous avons vus dans plusieurs leçons cette année. Le but du projet est de développer un pilote qui peut atteindre la ligne d'arrivée tout en respectant les restrictions associées au pilote et la piste fournie par le gestionnaire de course. Le but du projet est donc d'obtenir un pilote qui respecte toutes les conditions et qui soit le plus rapide possible afin de gagner la course contre les autres pilotes. Ce rapport décrira les étapes franchies pour atteindre nos objectifs et les difficultés rencontrées.

1.2. Présentation d'équipe

En tant que binôme , il a fallu se mettre d'accord assez rapidement sur la manière dont le travail allait être réparti pour avancer le plus efficacement possible. Tout d'abord, il a fallu se familiariser avec le sujet, comprendre ce qui était fournis et notamment le fichier C "droit au but" qui a été la base pour démarrer notre projet, quels étaient les objectifs à atteindre ? Une fois, cela fait, il a fallu décomposer le travail en parties pour travailler séparément. L'élément principal de notre projet étant de trouver le chemin le plus court de la piste, nous avons séparé la création des différentes structures et fonctions développées séparément pour implémenter cet algorithme.

1.3. Présentation du sujet

L'objectif de ce second projet de première année informatique était de créer un pilote qui termine en un minimum de tours et en profitant au maximum du carburant donné des circuits quelconques. Ce second projet mettait en avant d'autres facultés par rapport au premier projet. Nous nous sommes donc heurtés rapidement au problème de coût et donc de réussir à trouver le meilleur chemin en un minimum de temps.

Le gestionnaire de course nous fournit la piste de jeu : une grille de dimension L×l où dans chaque case il y a un caractère (#, ˜ ,.,=) décrivant l'état du terrain. Le but de notre programme est donc de fournir au gestionnaire de course le vecteur accélération de la forme a = (dvx, dvy), qui sera ajouter au vecteur vitesse, à chaque tour. Il s'agit donc de trouver un programme pour trouver le chemin le plus court possible pour chaque piste et de gérer la vitesse de notre pilote ainsi que la consommation de carburant.

2. Définitions et notations

Avant de comprendre ce qu'est un bon pilote , il est normal de se demander quelle est sa définition et comment ce dernier peut se comporter face à une carte quelconque. Le but de cette partie est donc de formaliser le problème en vue de poser des bases solides pour l'implémentation du programme.

- Carte : Une carte de taille h * w est une matrice de caractères parmi {#,~,.,=}

- Pilote : un pilote est la donnée d'un vecteur position p, d'une vitesse v et du nombre de boosts b dont il dispose. Le triplet (p,v,b) constitue un état unique du pilote.

3. Pilote

3.1. Structure de données

Graphe : représentation d'un graphe. Nous avons choisi d'implémenter cette structure par une matrice d'adjacence comme représentation de graphe, et par une liste pour obtenir l'itinéraire suivi par la voiture . Sans oublier aussi des tableaux pour la manipulation de l'algorithme Dijikstra . Cette structure nous a beaucoup aidé dans notre programme . Vu que nous avons procéder par la façon suivante :

Nous avons remplacé tous les '#' qui désignent le chemin par des entiers positifs croissants tandis que tous les autres caractères par des '-1' .

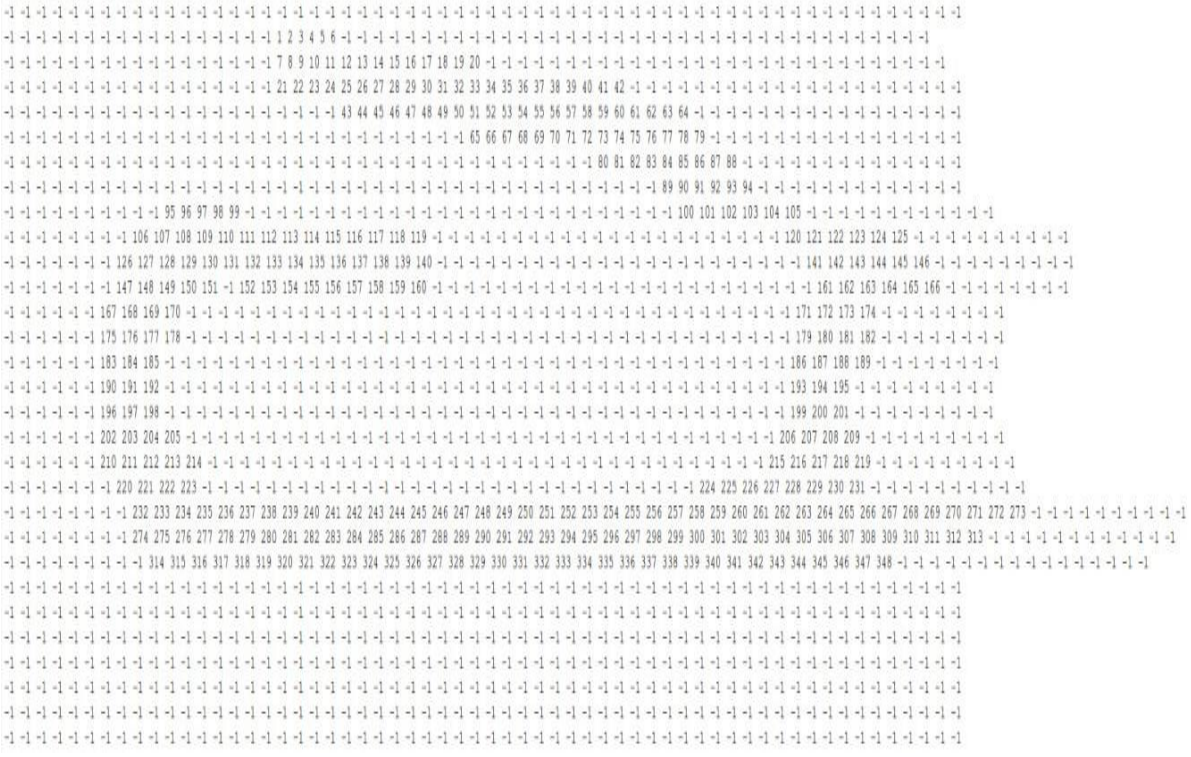

Figure1. Nouvelle représentation de la carte pour calculer le plus court chemin.

Après l'appel à la fonction « plus_court_chemin » qui se base sur l'algorithme de Dijikstra ,qui va être expliqué dans le paragraphe suivant, on désigne par des entiers négatifs croissants le chemin trouvé.

Figure2. Nouvelle représentaion de la carte avec le plus court chemin .

Et le voila le chemin qui va être suivi par notre pilote en suivant les nombres négatifs en ordre croissant.

Tas : structure dédié à la stockage des informations sur les sommets du graphe lors de la recherche en respectant l'ordre d'une tas.

elem_tas : structure qui représente un sommet de graphe avec son potentiel courant lors de la recherche.

3.2. Algorithme implémenté

Parmi les différentes stratégies qui peuvent être mises en place pour trouver un chemin de longueur optimale, nous avons choisi d'implémenter une version développée de l'algorithme de Dijikstra à partir de celle donnée dans notre cours d'Algo avancé . En effet, celle qui est donnée dans le cours ne prend qu'un seul point de départ et un seul point d'arrivée comme paramétres. Par contre, il y'a sur la carte trois arrivées possible. Ainsi et pour que notre pilote soit le premier arrivant , on a développé cet algorithme pour qu'il soit capable de prendre comme paramètre le point de départ et les trois d'arrivés. Pour pouvoir choisir le meilleur point d'arrivée et ainsi retourner le chemin optimal de rendre.

ALGORITHME

(calcul du plus court chemin du sommet n°1 à tous les sommets du graphe)

```
Initialisations
   S = { 1 }          X-S = { 2, 3, … N }
   Π(1) = 0
   Π(i) = v(1,i)    si i est successeur de 1
        = +∞        sinon

Répéter

   1/ Sélectionner le sommet j ∈ X-S tel que :
      Π(j) = minimum Π(i) pour i ∈ X-S

   2/ L'ajouter à l'ensemble S :
      S   = S    + {j}
      X-S = X-S  - {j}

   3/ Si X-S ≠ ∅ alors
      faire pour tout sommet s ∈ X-S et successeur de j
         Π(s) = minimum( Π(s) , Π(j) + v(j,s) )

Jusqu'à ce que X-S = ∅
```

Figure3. L'algorithme de Dijikstra donné dans le cours d'algo avancé.

D'autre part, nous aurions pu garder la même fonction et tester pour les trois points d'arrivée, pour finalement prendre le chemin le plus optimale des trois. Mais on a constaté que notre algorithme va prendre beaucoup de temps ce qu'est pas bien pour notre pilote.

Il faut garder à l'esprit que la course se déroule en présence d'autres pilotes et que l'on peut bien évidemment pas, à un instant donné, aller vers une case occupée par un autre pilote. Cela implique qu'il faut vérifier à chaque tour de jeu la validité de l'accélération que l'on s'apprête à envoyer au gestionnaire. Et s'il est le cas on calcule une autre fois le plus court chemin on prenant ce point comme un obstacle '.', et on repars à nouveau.

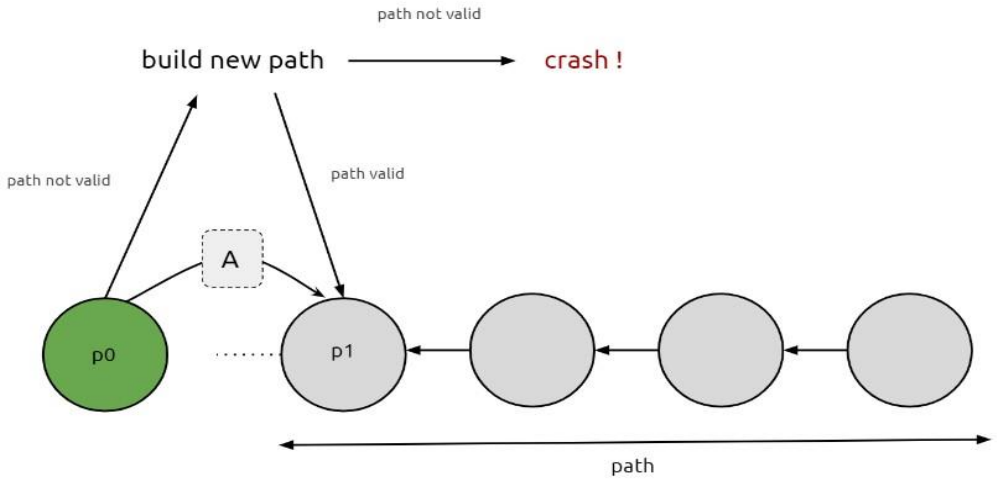

Figure4.Mise à jour dynamique de chemin.

3.3. Gestion de l'acceleration

On a développé plusieurs fonctions pour gérer l'accéleration de notre voiture dont on cite :

- accelerationvalid : Qui valide que notre chemin ne contient aucun obstacle .
- getMinAccelerations : Qui calcule l'acceleration minimale pour passer au point suivant.
- getAccelerations : c'est la fonction qui se charge de calculer la longueur du chemin disponible dans huit sens bien determinés, pour voir après si on va bien accélérer ou pas . Elle fonctionne de la façon suivante :

  Si on note nb = nombre de « # »

  Cas 1 : 4<nb<10 on ne dépasse pas une norme de 2,80 comme vitesse .

  Cas 2 : 10<nb<18 on ne dépasse pas une norme de 4,24 comme vitesse .

  Cas3 : nb>18 on ne dépasse pas une norme de 5 comme vitesse .

Sans oublier ,bien sur, de vérifier si on a assez de carburant dans chaque cas .

4. Gestion du carburant

Pour ce qui concerne la gestion du carburant et vu qu'on doit profiter au maximum du gaz disponible . Nous avons crée deux fonctions à savoir :

- Gasconsumption : Qui calcule le carburant déjà consomé par notre voiture.
- Gasneeded : Qui calcule la quantité de carburant nécessaire pour arriver . Et la on a laisser une marge ça veut dire on peut arriver avec une quantité du carburant restante. Car il faut toujours garder à l'esprit que la course se déroule en présence d'autres pilotes et qu'on ne peut pas accéder à un point occupé par un autre pilote ce qui nécessite de changer le chemin, ainsi plus de consomation du carburant.

5. Résultat

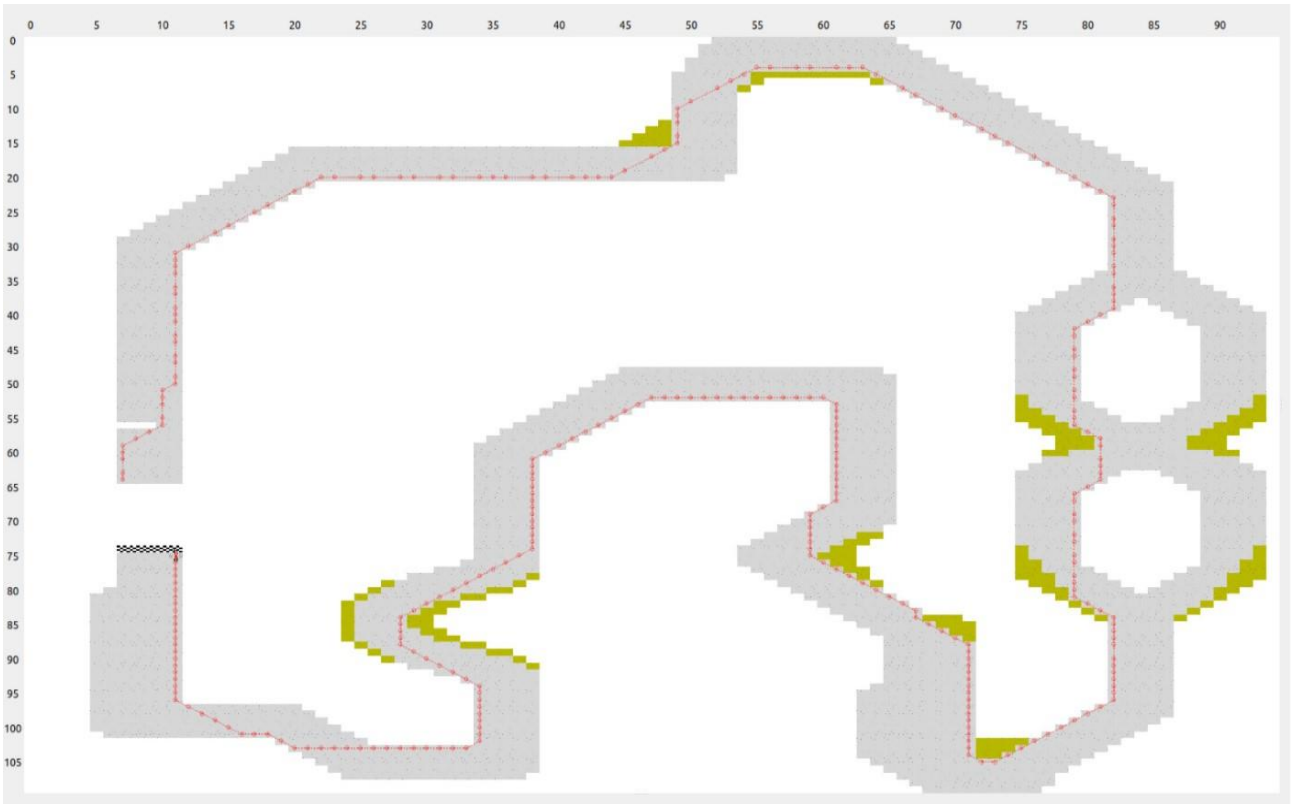

Figure5. un trajet parcouru par notre pilote.

6. Conclusion

Ce projet a été l'occafsion d'implémenter et d'adapter l'algorithme de Dijikstra qui donne à partir d'un graphe des états possibles d'un pilote à l'arrivée, sur tout type de carte. Aussi de rassembler toutes les notions vues cette année, notamment l'utilisation des graphes, l'ODL, et la gestion de projet. Le pilote ainsi obtenu permet de calculer très rapidement un chemin optimal jusqu'à l'arrivée, pour des cartes de taille raisonnable. Il s'adapte en fonction de ses adversaires, en réévaluant le chemin si nécessaire .

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES