ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

Pilote Tupac pour l'application Grand Prix

Enseignement : Mini-Projet

Références :

- Rakotondramasy Tendry
- Geliot Quentin
- 1A Informatique
- Année universitaire 2014/2015

ENSICAEN
6, boulevard Maréchal Juin – CS 45 053 – F- 14050 Caen Cedex 4
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

Table des matières

INTRODUCTION 3

RÉPARTITION DES TÂCHES 3

1. A*, ALGORITHME DE RECHERCHE DE CHEMIN 3

1.2. Fonctionnement 4

1.3. Optimisation de l'open list 4

2. DÉPLACEMENT DU PILOTE DANS L'ESPACE 5

2.1. Collecte d'informations 5

2.2. Prise de décision 5

3. APPROCHES DIFFÉRENTES 5

3.1. Approche sémantique 5

3.2. Autres approches envisageables 6

CONCLUSION 6

Introduction

L'application Grand Prix est un simulateur de course de formule 1 dans lequel s'affronte des développeurs, dont le but est de concevoir le pilote idéal pour gagner la course.

Il nous a donc été demandé de développer une solution à ce problème. Il fallait toutefois considérer que le pilote se devait de communiquer avec le serveur de jeu, et que celui-ci posait des contraintes et des règles à respecter (limitation de vitesse, collisions entre voitures …)

Nous avons répondu à ce problème en le décomposant en plusieurs tâches à réaliser, ainsi qu'en gardant une approche de « recherche » sur le problème, toujours dans l'optique de s'enrichir et d'appliquer une démarche d'ingénieur afin de trouver la meilleure solution.

Répartition des tâches

Nous avons identifié deux parties essentielles dans le problème :
- Un sous problème de recherche de chemin.
- Un sous problème de déplacement du véhicule dans l'espace.

Le deuxième point étant toutefois aussi complexe nous avons décidé de le résoudre en adoptant différentes approches. Une approche naïve de suivi de trajectoire point par point, ainsi qu'une méthode plus « sémantique » se rapprochant plus du problème de la vie réelle.

Tendry a donc utilisé ses connaissances d'IUT pour se pencher sur la résolution de recherche de chemin. Tandis que Quentin s'est concentré sur la compréhension et l'établissement d'une solution de déplacement du pilote dans l'espace.

Les apports des études des différentes méthodes d'approche ont été non négligeables et nous ont permit d'améliorer notre code à plusieurs reprises, mettant en évidence l'intérêt d'une démarche « scientifique » typée « recherche ».

1. A*, Algorithme de recherche de chemin

1.1. Choix de l'algorithme

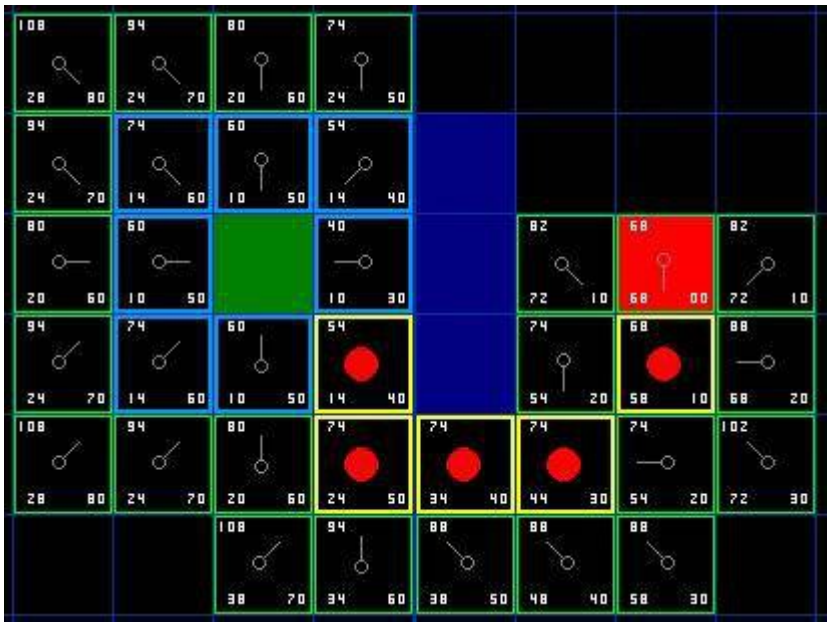
Il existe plusieurs algorithmes de recherche de chemins. Mais les algorithmes de recherche de chemin les plus utilisés sont Diskjtra et A* (prononcé « A star »).

Diskjtra est plus proche d'une approche naïve, plus lent, il garanti en revanche une solution optimale. Toutefois A* est plus rapide, il trouve la solution optimale à chaque fois en moyenne, mais il s'exécute bien plus vite, de plus nous avons considéré le fait que A* peut être optimisé pour aller encore plus vite avec un tas binaire, c'est pourquoi nous l'avons choisi

1.2. Fonctionnement

L'algorithme utilise deux valeurs pour la recherche du chemin, la distance heuristique entre un nœud probable du chemin final et le point d'arrivée. Cette distance est comparable à la distance à vol d'oiseau entre deux points. Toutefois cette seule valeur n'est pas suffisante si jamais les obstacles nous poussent à aller à contre-sens par rapport à la distance heuristique.

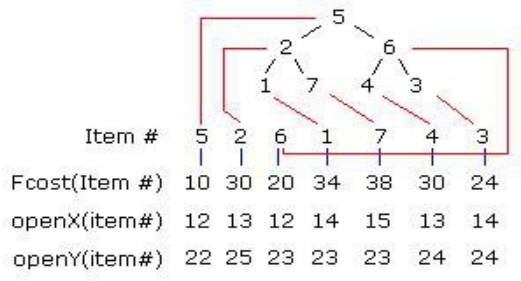
L'algorithme induit ainsi un coût à chaque déplacement, les nœuds les plus intéressants et plus probables d'appartenir à la trajectoire finale sont ceux dont la distance heuristique est minimale ET ceux dont le coût en déplacement est le plus faible.



En vert le point de départ, en rouge le point d'arrivée, les cases entourées en vert sont le chemin. Les autres cases valuées sont celles traitées, en haut à gauche on peut voir leur poids, qui doit être minimal, il est la somme de : en bas à gauche le coût en déplacement pour arriver à ce nœud, et en bas à droite la distance heuristique au point d'arrivée.

1.3. Optimisation de l'open list

Dans le schéma, on voit bien que chaque nœud possède un poids, chaque nœud est stocké dans une open list triée par ordre de poids, les nœuds sont traités dans cet ordre, le souci, quand il y a beaucoup de nœuds, c'est que le tri peut être lent … C'est pourquoi nous avons choisi d'implémenter un arbre binaire sous forme de tas afin de garder la valeur la plus faible au sommet, l'avantage de cette structure est qu'elle peut être modélisée dans un tableau à une dimension.



Fcost = poids du nœud, le tableau stocke les identifiants des nœud dans l'ordre croissant.

Temps d'exécution de A optimisé : 0,00415s*

2. Déplacement du pilote dans l'espace

2.1. Collecte d'informations

Dans un premier temps il a fallu rassembler les informations et les variables nécessaires à la prise de décision. Les informations de déplacement du véhicule à n'importe quel instant de la course donnée par le serveur (vitesse, position, carburant dépensé.).

2.2. Prise de décision

Une fois ces données récoltées, il faut trouver une coordonnée de la trajectoire optimale pour faire déplacer le pilote.

Méthode naïve : Cette méthode consiste à suivre coordonnée par coordonnée la trajectoire trouvée par A*. On choisit la coordonnée dans la trajectoire la plus proche du joueur avec une fonction retournant la distance entre deux points. Comme le pilote suit la trajectoire, la norme du vecteur vitesse est donc compris dans le cercle unité.

Méthode naïve avancée : Le principe reste le même que précédemment. Le pilote va suivre la trajectoire donnée par A*. Le changement s'opère sur la vitesse du pilote.
Nous avons dû repenser à la gestion des virages. Entre deux virages, la voiture accélère. A l'approche du virage, la voiture diminue sa vitesse pour négocier le virage.
Le seul inconvénient de cette méthode est que la voiture rend sa vitesse presque nulle à l'approche des virages. Le pilote n'utilise pas son inertie pour tourner.

3. Approches différentes

3.1. Approche sémantique

Le problème réel est celui de la « trajectoire idéale », Lorsque l'on conduit on se contente d'aller tout droit, et de tourner lorsqu'il y a un virage. Nous avons donc essayé cette approche en exploitant le chemin trouvé par A* pour identifier les virages.

Toutefois, nous nous sommes rendus compte que trop de variables entraient en jeu pour avoir une approche aussi généraliste : il fallait avoir une zone précise correspondant au virage, identifier correctement la fin d'un virage, gérer les cases de sable, prévoir l'instant pour freiner et aborder le virage à une vitesse correcte etc.

Nous n'avons donc pas réussi à mettre en œuvre cette approche. Mais cette étude nous a apporté les outils suffisants pour optimiser le code préexistant.

3.2. Autres approches envisageables

Il existe plusieurs méthodes d'approches qui auraient pu être intéressantes d'étudier.

Une approche s'en tenant exclusivement à l'algorithme A*, recalculant à chaque fois la trajectoire pour s'assurer de trouver la trajectoire optimale à chaque fois, surtout lorsque des perturbations peuvent se produire comme une collision avec un autre pilote, ou une erreur de calcul faisant faire demi-tour au véhicule etc.

Il aurait aussi été possible de voir le problème comme une avancée à l'aveugle sans effectuer de recherche de chemin. En faisant avancer le pilote dans le sens le plus favorable par rapport à une portée d'analyse constante autour de lui.

Conclusion

En développant ce projet, nous avons pu mettre en œuvre plusieurs méthodes de recherche de solutions, et avons eu l'occasion de développer un « module », dépendant d'une application plus importante. Nous avons pu constater que l'étude de plusieurs approches et l'échec de la mise en œuvre de celles-ci faisaient aussi partie intégrante de la recherche de solution.

Nous avons pu être en mesure d'être très exigeants sur ce que nous savions déjà faire afin d'optimiser notre solution au maximum.

Enfin, ce projet nous a montré qu'un problème « courant » de la vie réelle peut être d'une réelle complexité à résoudre informatiquement.