Barjo-Kart

Projet programmation L3 en C++

29 janvier 2021

Ce sujet est librement inspiré du concours de programmation ICFP 2008.

Introduction

Au cours de la période de projet, vous allez développer un programme pour répondre à un défi de programmation. Ce défi consiste à optimiser la trajectoire de bolides sur des circuits qui vous seront proposés. Les règles qui vous sont données dans ce document sont les règles de base de barjo-kart, des raffinements vous seront proposés au cours de la période de projet, l'enjeu étant alors de pouvoir adapter votre programme à ces changements. Les bonnes pratiques devraient vous être utiles pour cela.

Vous travaillerez par petits groupes, en utilisant les outils de travail à plusieurs comme git. Vous serez évalué sur vos réponses aux défis, mais également sur la qualité de votre code.

Les défis vous seront proposés sur un site web où vous pourrez soumettre vos trajectoires.

1 Le barjo-kart

Chaque défi prend la forme d'un circuit, avec un point de départ et une zone d'arrivée. Pour ce circuit, il faut déterminer une *trajectoire* optimale, c'est à dire une trajectoire qui aille du point de départ à la zone d'arrivée en le plus petit nombre d'étapes possibles. Vous soumettrez pour chaque défi une trajectoire, à laquelle il sera attribuée un score. Pour chaque circuit qui vous sera proposé, l'enjeu est de soumettre la trajectoire avec le meilleur score dans le temps le plus rapide.

Pas de moteurs, pas de volant, pas de freins, pas de ceinture, pas de peur : les véhicules du *barjo-kart* sont des caddies de supermarché que des pilotes téméraires manœuvrent tant bien que mal en donnant des coups de perche sur le sol. Entre deux coups de perches, ils se déplacent donc en ligne droite ; il est tout aussi laborieux d'accélérer, décélérer ou de diriger ces bolides.

Note : toutes les coordonnées, vitesses et accélérations sont représentées par des entiers.

1.1 Principe

Un circuit est donné sous forme d'une image, où les pixels noirs —$(0, 0, 0)$ en rgb— sont des obstacles. La trajectoire que vous proposez ne doit donc pas passer par un pixel noir, sinon c'est la sortie de route.

Une trajectoire est composée de vecteurs, qui représentent des étapes élémentaires. Au début de chaque étape, il est possible de changer —légèrement— la direction et la vitesse du véhicule. Avec le circuit, on vous donne *l'accélération maximale* $a_{\max}$ du véhicule. On note e_i le vecteur qui représente l'étape i. Ainsi, si à l'étape 3, on va du point $(12, 34)$ au point $(45, 67)$, on aura $e_3 = (45 - 12, 67 - 34) = (33, 33)$. On doit alors avoir à toute étape : $||e_k - e_{k+1}|| \leq a_{\max}$. La norme utilisée est la norme 1, dite ≪Manhattan≫ : $||(x, y)|| = |x| + |y|$. Le départ se fait à l'arrêt, on pose donc pour ce calcul $e_{-1} = (0, 0)$. Si à une étape quelconque, $||e_k - e_{k-1}|| > a_{\max}$, c'est le tête à queue voire le tonneau —et accessoirement la fin de la course pour vous.

Étant donné un circuit et une trajectoire $(e_0, e_1, \dots e_n)$, le score obtenu est :

— $+\infty$ si la trajectoire passe par un pixel noir au cours d'une étape quelconque ;
— $+\infty$ si la trajectoire contient deux étapes consécutives e_{k-1}, e_k telles que $||e_k - e_{k-1}|| > a_{\max}$;
— $+\infty$ si le point d'arrivée de la trajectoire est hors de la zone d'arrivée ;
— n sinon —félicitations, on est arrivés à bon port !

1.2 Calcul de trajectoire

Pour déterminer si la trajectoire passe par un pixel interdit, il faut non seulement déterminer où sont situées les extrémités des étapes, mais également quels sont tous les pixels intermédiaires par lesquels ont passe au cours de l'étape. On considère qu'au cours de chaque étape, la trajectoire est rectiligne. Pour calculer les pixels qui sont sur la trajectoire, on utilise l'algorithme de Bresenham entre les extrémités de chaque étape.

1.3 Encodage des circuits et des trajectoires

Les circuits Chaque circuit et donné par deux fichiers :

— un fichier image ; dans ce fichier image, les pixels noirs représentent les zones qui sont en dehors du circuit
— un fichier de configuration du circuit, au format `toml`.

Dans le fichier de configuration, on trouve dans la version de base les informations suivantes :

— un entier, `acc_max`, la valeur de l'accélération maximale
— `depart`, qui contient deux entiers `x` et `y` représentant les coordonnées du départ du circuit
— `couleur_arrivee`, qui contient trois entiers `r,g,b` qui indiquent la couleur de la zone d'arrivée

Voici un exemple de fichier de configuration pour un circuit :

```
acc_max = 30
```

```
depart = { x=12, y=34}
couleur_arrivee = { r=255, g=0, b=0 }
```

Les trajectoires Une trajectoire est encodée par la suite des étapes telles que définies ci-dessus. Chacune de ces étapes est codée par un couple d'entiers. Ces entiers sont écrits directement dans un fichier, chacun sur 32 bits en *little-endian*.[1]

1.4 Quelques indications de départ

Voici quelques indications pour vous aider à démarrer

Méthodologie Commencez par coder un simulateur pour pouvoir vérifier des trajectoires.

L'escargot Avec une vitesse de 1 pixel par étape, vous ne pouvez pas partir dans le décor ! Déterminer le plus court chemin (au sens des graphes) entre le point de départ et la zone d'arrivée vous donne une trajectoire correcte —quoique plutôt lente.

La tortue Avec une vitesse de norme inférieure à `acc_max`, vous pouvez toujours vous arrêter net. Tant que vous ne rasez pas les murs à une distance inférieure à votre vitesse, cela devrait vos mener à bon port. Pour ne pas raser les murs, vous pouvez calculer la distance de chaque point aux obstacles et conserver cette information.

2 Déroulement du projet

Vous aurez des séances de TP pour progresser sur votre code avec notre encadrement. Le serveur pour soumettre vos trajectoires et récupérer de nouveaux circuits sera ouvert en permanence. Au cours de séancess en ligne avec toute la promotion, nous ferons un point sur les scores, présenterons des défis à relever en direct et les évolutions des nouveaux circuits.

2.1 Critères d'évaluation

L'aspect compétitif de la tâche n'est pas un critère d'évaluation central : *l'important, c'est de participer*. Ce qui est exigé, c'est de soumettre des solutions de qualité aux défis qui vous sont proposé. Vous nous donnerez un accès au dépôt *git* dans lequel vous travaillez. Pour la fin du projet, vous rendrez un *post-mortem*, c'est à dire un petit rapport dans lequel vous expliquerez vos choix, ce qui a marché et ce qui n'a pas marché. Les critères d'évaluation seront donc :

— pour 40%, la qualité des solutions proposées, et votre rapidité pour vous adapter aux nouveautés,
— pour 30%, la qualité de votre code,

1. comme dirait l'autre, ≪en binaire, quoi≫

— pour 30%, votre rapport.