PIN-2023, projet, decontaminators

Contents

Introduction

Après la réalisation du travail préparatoire « ramping up », ce document introduit maintenant le cahier des charges pour le projet « decontaminators ». La réalisation et la validation du travail préparatoire est un prérequis à l'acceptation des livrables décrits ici.

NB : voir la donnée ramping up pour l'introduction des termes spécifiques de ce document, ainsi que pour une explication du contexte et des objectifs du projet ainsi que pour l'organisation du cours et de l'évaluation des étudiants.

Decontaminators

Le projet consiste à transformer la solution basique de simulation du monde 2D des robots et particules en un jeu de stratégie (assez) réaliste. Pour ce faire il manque principalement 3 choses à la solution réalisée dans le travail préparatoire :

- L'objectif du jeu
- Des règles pour jouer
- Une stratégie pour gagner

Objectif du jeu

Le jeu decontaminators, comme son nom l'indique, vise à faire décontaminer le monde par les robots. En d'autres termes, les robots doivent aspirer toutes les particules qui s'y trouvent. Nous appelons ainsi « décontamination » le résultat de l'action d'un robot qui aspire une particule, provoquant sa disparition.

Bien évidemment, les particules finissant par se désintégrer d'elles-mêmes, on pourrait simplement laisser passer le temps pour atteindre l'objectif précédemment cité. Pour rendre le jeu plus compétitif, néanmoins, nous allons chercher à décontaminer le monde 2D des robots et particules en un minimum de temps, ou plus exactement, de manière optimale.

Score

Nous allons attribuer à chaque particule une énergie proportionnelle à la surface du disque qui la symbolise. Chaque fois qu'un robot aspire une particule il va augmenter son score individuel de la quantité d'énergie absorbée. Ceci va nous permettre de calculer un score total, comme la somme des scores individuels de tous les robots.

Afin d'optimiser le score total, les robots vont devoir se « gaver » d'énergie, i.e. aspirer un maximum de grosses particules. Comme la surface des particules diminue à chaque explosion, il convient de les déplacer rapidement, mais pas seulement.

Décontaminations

La décontamination n'est possible que selon un scénario bien précis, celui où la bouche aspirante du robot est positionnée de manière adéquate. La figure 1 ci-dessous présente les conditions à remplir et qui peuvent être résumées ainsi : le rayon qui part du centre du robot et aboutit au point de contact avec la particule doit être contenu dans l'angle de la bouche aspirante (voir ramping up).

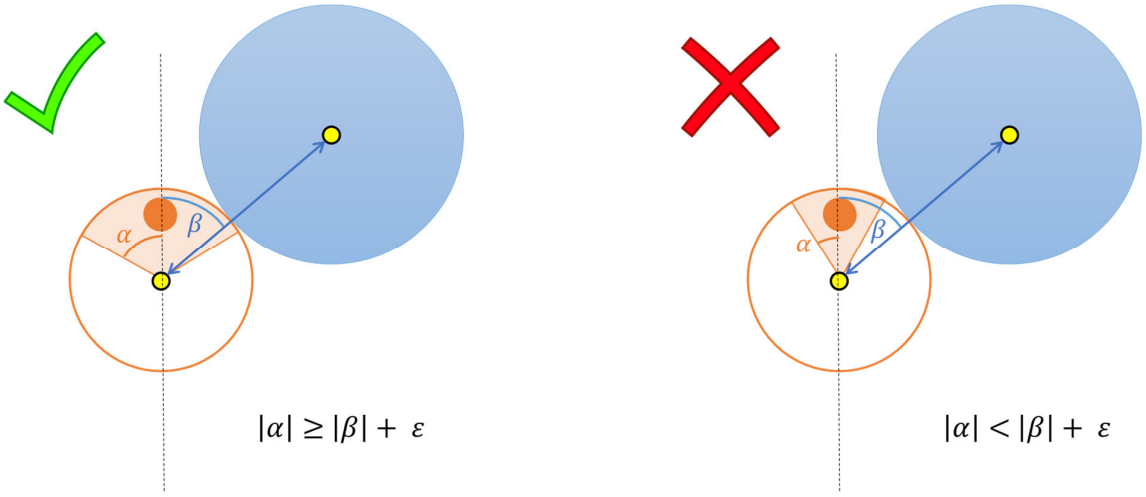

figure 1 Conditions de décontamination

Stratégie

Nous allons piloter les robots pour leur permettre d'être efficaces. Nous allons pour cela, contrôler leurs trajectoires et leurs vitesses de déplacement en modifiant leurs vitesses latérales gauches et droites. La stratégie à mettre en place consiste in fine à calculer ces trajectoires et à mettre à jour ces vitesses au fil du temps avec l'objectif de permettre aux robots d'absorber les particules, si possible avant qu'elles n'explosent.

Attention cependant, foncer tête baissée vers la particule la plus proche ne sera pas forcément la meilleure stratégie, ne serait-ce que du fait du risque de collisions entre robots. Et si une collision se produit, pas question de laisser le robot à l'arrêt comme dans le travail préparatoire. Il faut le remettre en marche à la prochaine opportunité d'envoyer des commandes au robots, évidemment dans une direction qui évitera le contact avec l'autre robot, voire les autres robots.

A l'inverse, si une collision se produit avec une particule, il vaut mieux que celle-ci se produise bouche en avant, prête à aspirer la particule. Il faudra sinon faire pivoter le robot jusqu'à atteindre cet objectif.

Règles

Pour réduire encore un peu plus l'intérêt des stratégies simplistes, nous allons compliquer le jeu avec un certain nombre de contraintes sur les déplacements des robots. Ces contraintes sont de 2 types :

- Vitesses maximales : les vitesses des robots peuvent être modifiées à volonté en restant dans un intervalle [-*maxBackwardSpeed*, *maxForwardSpeed*] où :
  - *maxBackwardSpeed* = vitesse en marche arrière minimale
  - *maxForwardSpeed* = vitesse en marche avant maximale

- Réactivité minimale : sauf collision[1], les vitesses des robots ne peuvent être modifiées que lorsque le temps est un multiple entier de *CommandTimeInterval*

Applications

A l'issue du travail préparatoire, vous avez réalisé 2 applications : l'une en ligne de commande générant des fichiers JSON et l'autre graphique affichant le contenu des fichiers JSON. Il s'agit maintenant d'étendre ces 2 applications pour en faire le jeu décrit ci-avant.

La principale modification à l'application en ligne de commande consiste en l'ajout du module de stratégie qui va piloter les robots lors de la génération des timelines. Il va

[1] Lors des collisions, la vitesse des robots est mise à 0 avec effet immédiat (voir ramping up)

également falloir compléter ces timelines en renseignant le champ prévu pour le score de décontamination des robots. Les règles du jeu sont en principe déjà largement implémentées avec le traitement des contraintes du travail préparatoire ; reste à s'assurer de leur efficacité et à les completer avec la règle réactivité minimale.

L'application graphique reste quasiment inchangée par rapport au rendu préparatoire. Seule modification perceptible, l'affichage du score dans le panneau de contrôle. Celui-ci affiche le score total / l'énergie totale restante (i.e., la surface totale des particules restantes).

Livrables

Comme pour le travail préparatoire, vous réaliserez votre travail dans votre fork du GitHub du cours (voir ramping up). **Le résultat final doit être taggé avec le nom « rendu »**.

Sources

Tous les sources **.cpp*, **.h, *.ui*, ainsi que les directives de build (*CMakeLists.txt*) sont contenus dans le répertoire */Src*, lui-même structuré par application. **Le build doit être fonctionnel** et des commentaires doivent être ajoutés dans le *CMakeLists.txt* pour indiquer la plateforme utilisée ainsi que les éventuelles configurations spécifiques effectuées.

Données de test

En plus des données fournies dans le dossier */Explications*, vous trouverez dans le dossier */Data*, les données de test pour vérifier la bonne implémentation de vos applications. Le sous-dossier */Projet* contient les données de test applicables pour évaluer le rendu du projet. Il contient 2 types de données. Celles dédiées à la validation du rendu sont dans */validation*. Un autre sous-dossier */competition* est vide pour le moment.

Rendu

Voici les étapes à effectuer pour le rendu final :

- N'oubliez pas de commit/merge/push toutes vos modifications dans la branche "main"
- Utilisez votre programme pour générer les timelines demandées pour la validation du rendu et pour la compétition
    - les timelines générées doivent être déposées dans le même dossier que celui des données de test utilisées pour les générer

      /Projet/validation pour la validation du rendu

      /Projet/compétition pour la compétition

    - les noms des fichiers produits doivent reprendre ceux des fichiers state et contraints utilisés, et rappeler le numéro de votre groupe sur 2 chiffres (01, 02, …) :
      <state>_<constraints>_<nn>.tlin
- Faites un commit et un push pour avoirs les fichiers de log dans votre repo
- Ajoutez le tag « rendu » sur le tout dernier commit (l'ajout peut se faire directement sur github ou via git)

Directives

Votre projet doit être fonctionnel sur les jeux de données de validation fournis. L'équipe enseignante se réserve le droit de la tester sur tout ou partie du jeu de test fourni. **Les découvertes suivantes sont disqualificatives pour la compétition et susceptibles de pénalité pour l'évaluation du projet :**

- **mauvais fonctionnement avec le jeu de test de validation**
- **incohérence avec les traces rendues**

Lors de la journée finale, vous devrez brièvement présenter votre réalisation ainsi que la manière dont vous avez collaboré au sein de l'équipe. Les consignes pour la présentation seront fournies ultérieurement.

Si votre projet passe les critères de qualification évoqués ci-avant, votre réalisation sera mise en compétition avec celle des autres groupes et avec l'implémentation référence développée par l'équipe enseignante.

Compétition

La veille de la compétition, vous recevrez un nouveau jeu de test à copier dans le dossier */Projet/compétition*. Vous devrez exécuter votre programme sur chacun d'eux de manière à y générer les timelines correspondantes. Ceux-ci doivent être à disposition de l'équipe enseignante avant le début de la compétition et ils doivent pouvoir être générés avec votre programme en un temps raisonnable[2].

Les résultats de la compétition seront publiés sur Teams dans les jours qui suivent et le groupe vainqueur se verra attribué un prix dont la nature sera dévoilée au moment de la remise.

[2] Quelques (~ 5-10) minutes sur un ordinateur portable récent