PIN-2022, série 01, Qt & JSON

Contents

Introduction

A la fin du projet d'informatique (ci-après, PIN-2022), vous participerez à une compétition autour d'un jeu de stratégie visant à coordonner les actions d'une série de robots aspirateurs dans le but décontaminer un plan 2D[1] encombrés de particules[2], et ce, le plus efficacement possible[3]. Nous introduisons ici la 1ère des 2 séries d'exercices, dont la réalisation est une condition préalable au projet.

Séries

Dans les 2 séries préalables, nous introduisons les bases mathématiques du problème, ainsi que l'utilisation du framework Qt[4] pour l'affichage graphique en 2D et de la librairie JSON pour la manipulation des fichiers d'interface.

---

[1] 2D = 2 dimensions

[2] Implémentation exemple réalisée par l'équipe enseignante : https://youtu.be/nVcqsTWvftE

[3] Remerciements à Ronan Boulic qui nous a autorisé à adapter le projet « Décontaminators » donné lors du cours « Programmation 2 », à l'EPFL en 2018

[4] https://www.qt.io/

L'objectif des 2 séries d'exercice consiste à préparer la base des applications à développer pendant le projet autant qu'à prendre en main l'environnement de développement qui se résume ainsi :

1- Langage C++
2- Repo git github
3- Framework graphique Qt
4- Librairie JSON
5- IDE[5] CLion[6] ou Qt Creator[7]

Prérequis

Pour suivre ce cours, vous devez maitriser le langage C++ et avoir installé l'un des 2 IDE CLion ou Qt Creator, ainsi qu'un compilateur C++ et les utilitaires make et cmake compatibles.

Vous devez également disposer d'un compte GitHub Pro (gratuit pour les étudiants[8]). Nous vous inscrirons à l'organisation GitHub du cours[9] dans laquelle vous trouverez le(s) dossiers(s) des séries d'exercice et du projet final.

Organisation

Le travail s'effectue en **groupe de 3 personnes**. Les groupes sont constitués en début de projet et sont fixes jusqu'à la fin du cours. De plus, pour les séries, les groupes sont regroupés à leur tour par **paire**. Les paires changeront entre les séries. Les paires sont constituées pour valider le travail de l'autre groupe.

Le travail est obligatoire mais aucun rendu n'est attendu pour les séries. Vous devrez toutefois être en mesure de démontrer les résultats obtenus à la fin de chaque série et répondre aux questions qui vous seront posées sur votre travail. Les délais fixés sont les séances plénières

- série 01 : 25.08.2021.13h15
- série 02 : 31.08.2021.13h15

Toute dérogation à ces consignes nécessite l'accord écrit explicite de l'un des professeurs.

Evaluation

Vous devrez démontrer le résultat obtenu et répondre aux questions de l'autre groupe avec lequel vous évoluez en paire. L'évaluation donne lieu à un compte-

---

[5] IDE = Interface Development Environment

[6] https://www.jetbrains.com/fr-fr/community/education/#students

[7] IDE de Qt

[8] via https://education.github.com

[9] https://github.com/PIN-HEIGVD/PIN-Project-2022

rendu par le groupe évaluateur ; ce dernier est remis à l'équipe enseignante. L'exercice peut être reconduit en cas d'insuffisance jusqu'à validation explicite et écrite par un professeur.

Remarque : l'équipe enseignante se réserve le droit de vérifier, par sondage, la qualité et l'objectivité de l'évaluation.

Série 1

L'objectif de la série 1 consiste à réaliser une application qui affiche dans une fenêtre interactive, l'état du monde 2D des robots tel que défini dans un fichier JSON.

Applications

Il y a principalement 2 types d'application à développer.

- Les applications de « setup » sont des commandes en ligne. Elles permettent de générer des simulations du monde des robots et des particules.
- L'application de « rendu » est une fenêtre graphique. Elle y affiche graphiquement une simulation précédemment générée avec les applications de setup.
- L'interface entre le setup et le rendu est constitué de fichiers au format JSON[10]. Ceux-ci peuvent contenir une simulation entière, « Timeline », ou un état statique à un instant t de la simulation, « State ».

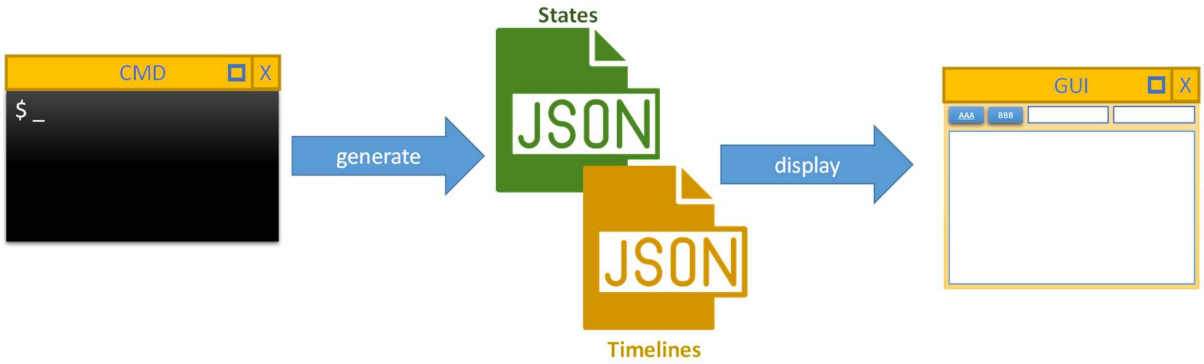

figure 1 Applications à réaliser et interfaces JSON

Objectif

Dans cette série, nous allons réaliser une première version de l'application d'affichage de manière à pouvoir visualiser le monde des robots à un instant t. Dans cette première réalisation, nous n'afficherons que les états instantanés, soit en affichant le contenu d'un fichier state, soit en affichant le dernier état applicable à

[10] https://www.json.org/json-fr.html

l'instant courant en parcourant le contenu d'un fichier timeline. Des exemples de states et timelines à afficher vous sont fournis.

NB : Nous verrons dans la prochaine série, comment animer ce monde virtuel à partir des interfaces timelines.

Simulation

On appelle monde, l'espace bidimensionnel qui contient les robots et les particules virtuels. Les coordonnées (ci-après coordonnées virtuelles) y sont exprimées en 2 dimensions selon un repère orthonormé composé de 2 axes X et Y dont l'origine est le point à l'extrémité bas, gauche. La position et la taille des robots et particules y sont exprimées dans ce repère orthonormé.

Les robots sont représentés par des disques et leur bouche d'aspiration par un secteur de ce disque. La taille du robot est spécifiée par son rayon et la taille de sa bouche par l'angle du secteur couvert. Le rayon qui scinde la bouche en 2 secteurs égaux définit l'orientation du robot. Celle-ci est représentée par un arc de cercle à mi-chemin de ce rayon.

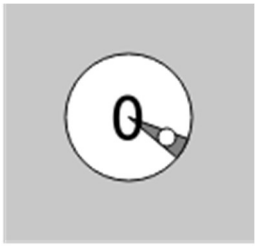

figure 2 Représentation d'un Robot

Les particules sont représentées par des disques simples dont le rayon détermine la taille.

Robots et particules sont identifiés séparément, par un numéro unique qui croît unitairement dans leur ordre de création. Le premier de la série pour chaque type a le numéro 0. L'identifiant est affiché sur le disque représentant l'objet.

NB : vous pouvez assumer que les mondes qui vous sont donnés au format JSON ont été testés et sont donc cohérents.

Affichage du monde

L'affichage du monde est réalisé dans une fenêtre interactive dont un preview est fourni ci-après. La fenêtre est composée de 3 zones organisées verticalement du haut vers le bas et contenant les éléments graphiques (widgets) suivants :

- En haut, un menu composé du seul item « File »

- Au milieu, un panneau de contrôle contient 2 boutons « |<- » et « |> », suivi de 2 champs spin box « Time » et « Speed », et se termine pas un libellé « Score »
- En bas, la zone d'affichage permet d'afficher le rendu du monde

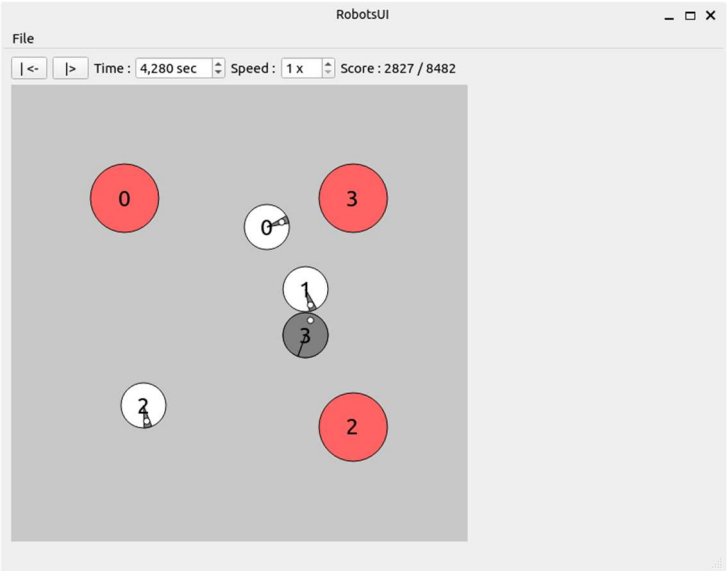

figure 3 Fenêtre d'affichage de la série 01

Le menu « File » permet à l'utilisateur de :

- « Open state » : charger un état depuis un fichier JSON sur le disque
- « Open timeline » : charger une timeline depuis un fichier JSON sur le disque
- « Save state » : sauvegarder l'état courant dans un fichier JSON sur le disque
- « Quit » : quitter l'application

Le panneau de contrôle permet à l'utilisateur de :

- bouton « |<- » : revenir à l'état initial du monde affiché
- bouton « |> » : animer / mettre en pause le monde affiché
- champ « Time » : afficher / changer le temps de l'état affiché par le monde
- champ « Speed » : afficher / accélérer l'animation par un facteur entier
- libellé « Score » : afficher le score de décontamination

La zone d'affichage du monde est définie ainsi :

- par défaut, la zone prend toute la largeur prévue pour le panneau de contrôle pour pouvoir afficher un monde carré avec une hauteur égale à cette largeur
- lorsqu'un monde est affiché, il est mis à l'échelle pour tenir dans la zone d'affichage disponible : i.e., dans le carré d'affichage prévu, par défaut
- la fenêtre peut être redimensionnée ce qui provoque la mise à l'échelle dynamique du monde affiché, le cas échéant

La fenêtre applique les règles de gestion suivantes :

- La détection d'erreur lors du chargement (chemin ou format) du fichier génère un bip, provoque l'effacement de l'affichage
- Les erreurs et warnings sont explicités dans la console (cout <<).
- Le titre de la fenêtre est « PIN-2022 serie 01 » si aucun rendu n'est affiché, il est rafraîchi avec le nom du fichier (sans le chemin) sinon.
- La fenêtre ne peut être redimensionnée que tant que la hauteur du monde affiché atteint au moins 200 pixels.
- Les objets, ou les parties d'objet qui sortent de la zone d'affichage ne sont pas affichés.
- Le temp est affiché en secondes avec une partie décimale jusqu'à la milliseconde.
- En mode animation et sans accélération (« Speed » = 1), le temps s'écoule en secondes réelles ; le temps s'écoule en accéléré avec le ratio indiqué dans « Speed » sinon (p.ex. « Speed = 2 » signifie 2x plus vite).
- L'accélération maximale est 10x (« Speed » <= 10).
- Après activation du bouton « |> », le temps s'écoule sans interruption jusqu'à ce que l'utilisateur provoque la mise en pause ou le retour au temps 0.
- La zone d'affichage affiche le dernier état applicable au temps indiqué dans le champ « Time » (temps de l'état affiché <= « Time »), rien si aucun fichier state ou timeline n'a été ouvert.

Directives

Vous trouverez dans le dossier */Explications* toutes les explications complémentaires (readme.md) et exemples (fichiers de données et images de rendu graphique) nécessaires.

Pour cette série, vous aurez besoin du contenu des sous-dossiers suivants :

- */Json* : ce dossier contient les informations sur la librairie à utiliser ainsi qu'un programme exemple.
- */Robot* : ce dossier contient une explication sur la structure abstraite destinée à représenter un robot ainsi qu'un exemple de fichiers JSON pour la définition du robot et celle de l'état d'un monde peuplé uniquement avec ce robot. Vous y trouverez aussi la timeline animant ce robot.
- */Particule* : ce dossier est l'équivalent du dossier Robot pour une particule.
- */State* : ce dossier présente la structure permettant de définir l'état du monde à un instant t. Il contient également quelques exemples de fichiers JSON représentant différents mondes à leur état initial ainsi que les images des rendus graphiques correspondants.
- */Timeline* : ce dossier présent la structure qui permet de suivre l'évolution des états du monde le long du temps. Il contient un exemple de décontamination d'un monde peuplé de 8 particules pourchassées par 4 robots.

Framework Qt

Pour installer le framework Qt, il faut utiliser l'installeur en ligne[11] qui vous invitera à créer un compte sur le site de Qt. Selon l'OS hôte que vous utilisez, vous pourrez être amené à compléter vos librairies avant[12] et/ou après[13] l'exécution de l'installeur en ligne.

L'installation vous donnera accès à l'IDE Qt Creator. Si vous préférez utiliser CLion, il vous faudra le configurer pour le framework Qt[14].

Remarque : L'IDE Qt Creator présente plusieurs exemples et tutoriaux très utiles pour prendre en main Qt. Ceux particulièrement intéressant pour réaliser le travail demandé sont indiqués dans le sous-dossier */Explications/'User Interface'*.

Livrables

Vous réaliserez votre travail dans un fork du GitHub du cours. L'accès à ce fork en lecture doit être ouvert aux membres de votre groupe et aux professeurs, mais fermé au reste des étudiants. **Le résultat final doit être taggé avec le nom de la série réalisée.**

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**.

NB : Vous êtes autorisés à réutiliser du code de sources externes **à la condition expresse** de citer très clairement la source de tout code qui n'a pas été écrit à 100% par un membre du groupe.

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 dossier est structuré pour distinguer les séries 1 et 2, puis le projet et la compétition finale.

---

[11] https://www.qt.io/download-qt-installer

[12] P.ex. https://askubuntu.com/questions/1298645/shared-library-not-found-for-pyqt5-libxcb-xinerama-so-0-not-found

[13] P.ex. https://askubuntu.com/questions/574217/qt-cant-find-gl-gl-h-but-libgl-so-exists

[14] https://stackoverflow.com/questions/30235175/how-to-configure-clion-ide-for-qt-framework

Directives

La série est valide lorsque vous pouvez démontrer le bon fonctionnement de votre réalisation sur les jeux de données fournis et que vos réponses aux questions des évaluateurs satisfont ces derniers.

L'évaluation vérifie le fonctionnement sur les différents jeux de données ainsi que la conformité du code aux directives présentées ci-avant et aux meilleures pratiques découvertes dans les tutoriaux ou dans d'autres expériences. Le compte-rendu est rédigé en **anglais** et tient sur une **page A4**. Il décrit, en **10 points obligatoires**, les principaux faits, qu'il s'agisse de problèmes lors des tests, d'un fait notoire comme une solution élégante ou, au contraire, à améliorer. Le compte-rendu est déposé sur le git dans le dossier */Documents/Rendus* avant la séance plénière.