

PIN-2023, ramping up

Contents

| | |
|-----------------------------------|----|
| Introduction..... | 1 |
| Ramping up..... | 2 |
| Prérequis..... | 2 |
| Organisation..... | 2 |
| Evaluation | 3 |
| Spécifications | 3 |
| Applications..... | 3 |
| Simulation du monde | 4 |
| Affichage du monde | 5 |
| Simulation des déplacements | 6 |
| Simulation des explosions..... | 7 |
| Simulation des collisions | 8 |
| Animation du monde | 9 |
| Réalisation..... | 9 |
| Code à compléter..... | 9 |
| Explications..... | 9 |
| Livrables | 10 |
| Sources..... | 10 |
| Données de test..... | 10 |
| Directives | 11 |

Introduction

A la fin du projet d'informatique (ci-après, PIN-2023), 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 2D¹ encombré de particules², et ce, le plus efficacement possible³. Nous introduisons ici le développement de base, dont la réalisation est une condition préalable au projet.

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

Ramping up

Dans ce travail préparatoire, nous introduisons les bases mathématiques du problème, ainsi que l'utilisation du framework Qt⁴ pour l'affichage graphique en 2D et de la librairie JSON pour la manipulation des fichiers d'interface.

L'objectif 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- IDE⁵ CLion⁶ ou Qt Creator⁷

Prérequis

Pour suivre ce cours, vous devez maîtriser 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 étudiants⁸). Nous vous inscrirons à l'organisation GitHub du cours⁹ dans laquelle vous trouverez le(s) dossier(s) du travail préparatoire et du projet final.

Organisation

Le travail s'effectue en **groupe de 5 personnes**. Les groupes sont constitués en début de projet et sont fixes jusqu'à la fin du cours. De plus, pour le travail préparatoire, les groupes sont regroupés à leur tour par **paire**. Les paires sont constituées pour valider le travail de l'autre groupe.

Le travail est obligatoire mais aucun rendu n'est attendu pour ce travail préparatoire. Vous devrez toutefois être en mesure de démontrer les résultats obtenus et répondre aux questions qui vous seront posées sur votre réalisation. Le délai fixé est la prochaine séance plénière

- séance plénière 2 : 28.08.2021.13h15 en **F01**

⁴ <https://www.qt.io/>

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-2023>

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-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.

Spécifications

Cette section présente les spécifications des applications qui génèrent, affichent et animent le monde des robots et des particules.

Applications

Il y a 2 types d'application.

- 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 JSON¹⁰. Ceux-ci peuvent contenir une simulation entière, « Timeline », ou un état statique à un instant t de la simulation, « State ».

Le schéma de la figure 1 fait ressortir le fait que les applications de setup consomment également des fichiers JSON en entrée afin de générer les fichiers timelines :

- State : définit l'état de départ au temps 0 à partir duquel on va pouvoir calculer l'évolution du monde au fil du temps
- Constraint : impose des limites sur l'évolution dynamique du monde dans le temps

¹⁰ <https://www.json.org/json-fr.html>

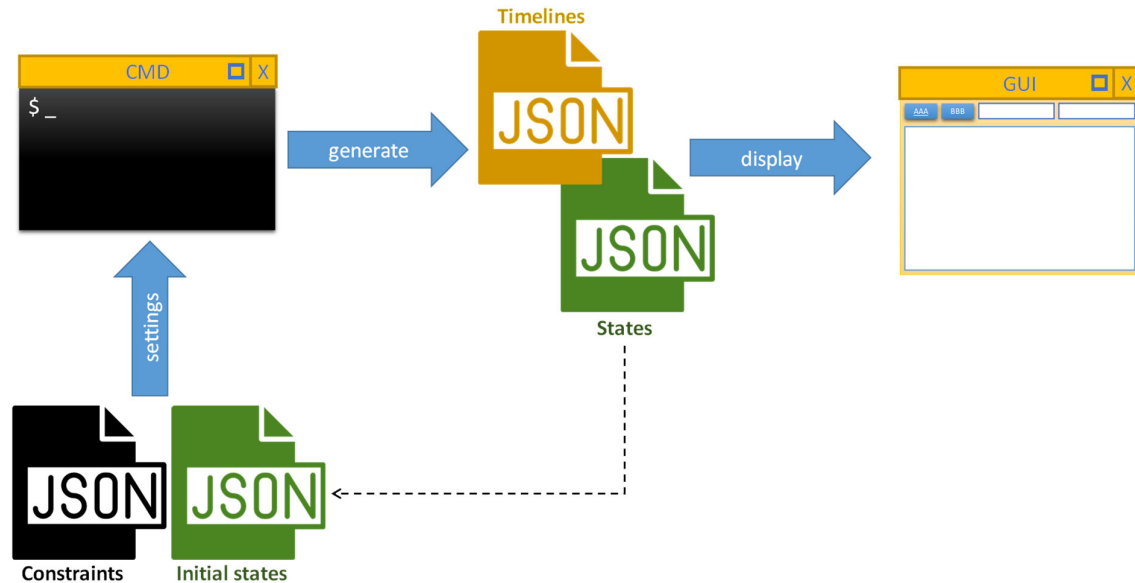


figure 1 Applications à réaliser et interfaces JSON

Simulation du monde

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 haute, 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 cercle à mi-chemin de ce rayon.

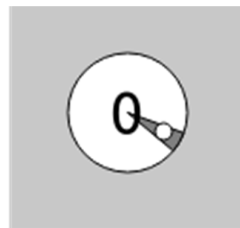


figure 2 Représentation d'un Robot

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 en figure 3. 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 par un libellé « Score »
- En bas, la zone d'affichage permet d'afficher le rendu du monde

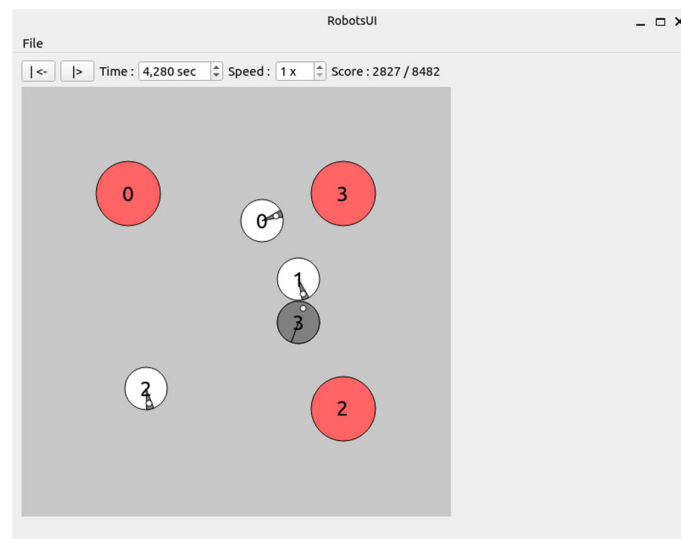


figure 3 Fenêtre d'affichage de la série 01

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 :

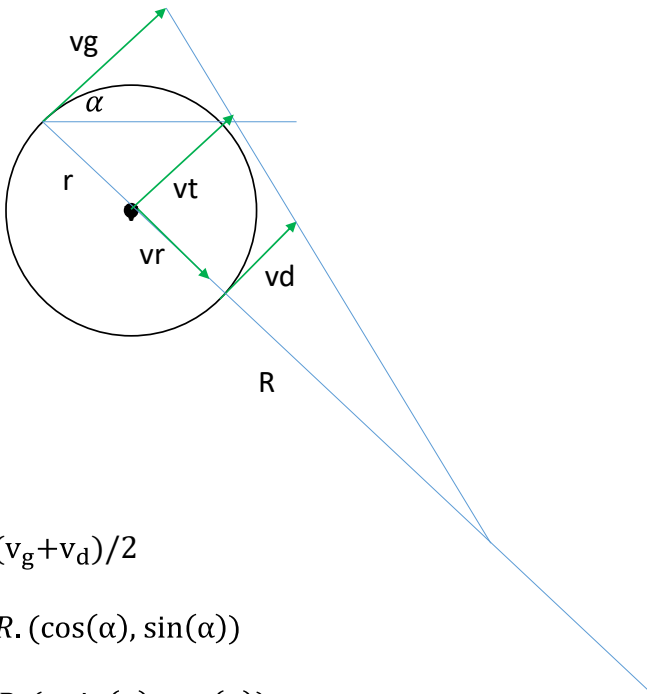
- La détection d'erreur lors du chargement (chemin ou format) du fichier génère un bip, provoque l'effacement de l'affichage
- Les erreurs et warnings sont explicités dans la console (cout <<).
- Le titre de la fenêtre est « PIN-2023 » si aucun rendu n'est affiché, il est rafraîchi avec le nom du fichier (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.

Simulation des déplacements

Les trajectoires des robots sont des courbes qui suivent des arcs de cercle.

L'orientation du robot suit la courbe, elle est constamment tangente à celle-ci. Pour arriver à ce résultat, il faut imaginer, comme illustré sur la figure 4, que les robots ont 2 vitesses, gauche v_g et droite v_d , qui sont des vitesses de déplacement linéaire des points extrêmes du diamètre transversal du cercle quand on se positionne dans le sens d'orientation du robot.

Quand les 2 vitesses sont égales, le déplacement est linéaire. La trajectoire s'incline du côté de la vitesse la plus faible, sinon, et dessine un arc de cercle dont le rayon R , par rapport au centre de l'objet en mouvement, est calculé selon la formule (1) sur la figure 4. L'objet va également tourner sur lui-même pour suivre la trajectoire selon une vitesse de rotation ω qui peut être calculée selon la formule (2). A noter, que l'objet va tourner sur lui-même dès lors que $v_g = -v_d$, les vitesses négatives étant autorisées. Au final, pour un intervalle de temps donné suffisamment court, on obtient la distance parcourue dans le monde virtuel selon la formule (3).



$$\frac{v_g}{v_d} = \frac{R + r}{R - r}$$

$$R \cdot (v_g - v_d) = r \cdot (v_g + v_d)$$

$$\textcircled{1} \quad R = r \cdot \frac{v_g + v_d}{v_g - v_d}$$

$$\textcircled{2} \quad \omega = \frac{v_t}{R} = \frac{v_g - v_d}{2 \cdot r} \quad v_t = (v_g + v_d)/2$$

$$\vec{v}_t = R \cdot (\cos(\alpha), \sin(\alpha))$$

$$\vec{v}_r = R \cdot (-\sin(\alpha), \cos(\alpha))$$

$$\textcircled{3} \quad \vec{d} = \vec{v}_t \cdot \sin(\omega \cdot t) + \vec{v}_r \cdot (1 - \cos(\omega \cdot t))$$

figure 4 calcul des déplacements en arc de cercle

Simulation des explosions

Dans le monde que nous simulons, les explosions de particules se produisent de manière aléatoire. Une particule qui explose est remplacée par 4 sous-particules plus petites (décomposition) ou elle disparaît tout simplement (désintégration).

La décomposition en 4 sous-particules est calculée comme illustré sur la figure 5. Chaque sous-particule occupe l'espace maximal possible en conservant toutes les 4 le même rayon, sans sortir du disque occupé par la particule parente¹¹.

¹¹ https://debart.pagesperso-orange.fr/seconde/empilements_dans_plan.html

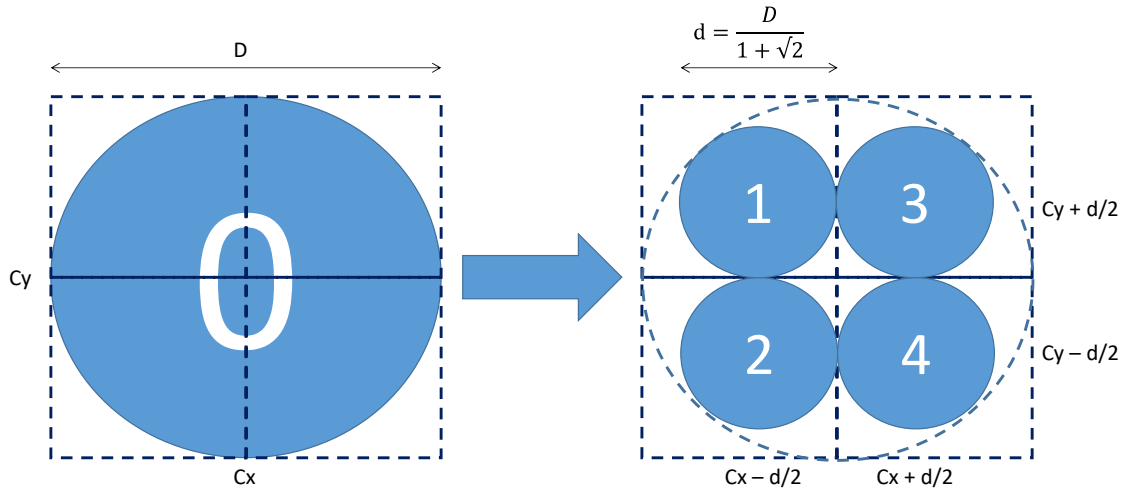


figure 5 décomposition en 4 sous-disques

Nous numérotons les particules séquentiellement dans leur ordre d'apparition, avec l'hypothèse qu'elles apparaissent systématiquement dans l'ordre indiqué sur la figure 5. Pour le reste, le rythme des explosions suit une loi stochastique.

Simulation des collisions

Les trajectoires des robots aboutissent en principe sur les particules, et elles peuvent avant cela se croiser entre elles (voir figure 6), provoquant dans les 2 cas des collisions. Dans notre simulation, cela se traduit par une mise à l'arrêt du ou des robots concernés à la périphérie de l'objet percuté. En d'autres termes les cercles dessinant la périphérie des objets doivent être tangents et les vitesses des robots doivent être mises à 0.

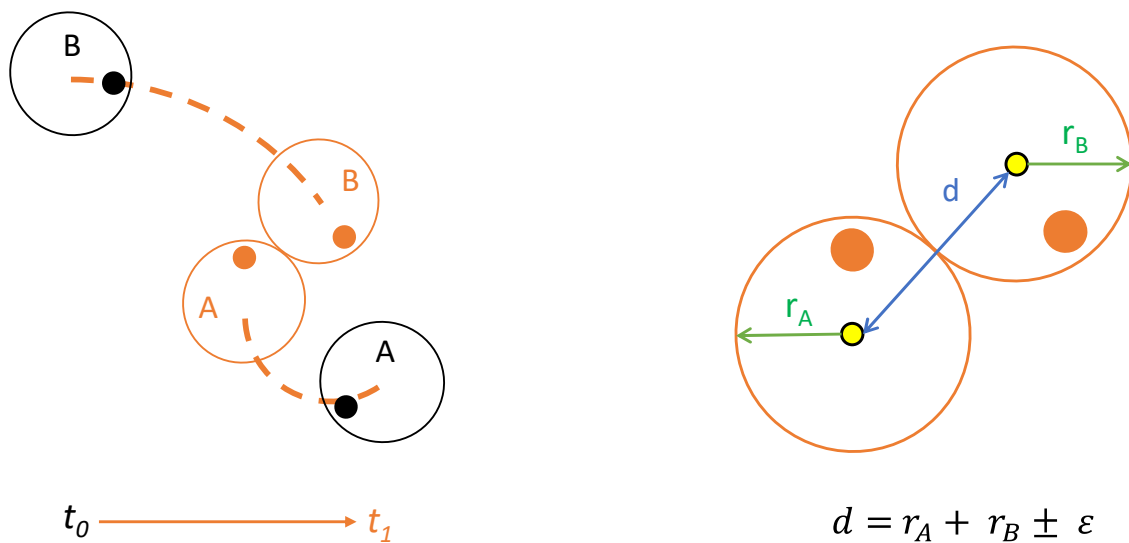


figure 6 collisions

Comme illustré en figure 6, nous allégerons le calcul des collisions en prévoyant une marge d'erreur paramétrable ε . Ce paramètre sera défini comme une constante du code source.

Animation du monde

L'animation est conditionnée par le rythme de rafraîchissement de l'écran. Ce dernier définit le nombre de rafraîchissements de l'image qu'il faut effectuer pour que l'œil humain perçoive un mouvement continu. Il n'est donc pas nécessaire d'aller au-delà, sous peine de saturer la machine avec des traitements inutiles.

Nous définissons un paramètre supplémentaire pour le taux de rafraîchissement sans aucun lien avec l'accélération de l'animation fixée par le champ « Speed ». Ce paramètre sera défini comme une constante du code source et régit seul le rythme de rafraîchissement de la zone d'affichage du monde des robots et particules. Il est fixé à 24 par défaut¹².

Réalisation

L'objectif consiste à réaliser les applications qui génèrent, affichent et animent le monde des robots et des particules.

Code à compléter

Afin de limiter le coût d'apprentissage du framework Qt, le code de l'application de rendu est partiellement fourni. Vous le trouverez dans le dossier */Code*.

Le code mis à disposition a été amputé de la partie animation décrite ci-avant. Celles-ci doivent être rétablies en complétant le code qui vous est fourni.

NB: il n'est pas nécessaire et, en principe non autorisé, de modifier le code fourni; il faut juste le compléter.

Explications

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

¹² https://fr.wikipedia.org/wiki/Images_par_seconde

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 présente 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.
- */Command* : ce dossier contient les informations sur les applications de setup et, en particulier, sur les contraintes à respecter. Les timelines générées doivent respecter ces contraintes.
- */Collisions* : ce dossier contient une explication détaillée de la recherche et du traitement des collisions.

NB : les temps aléatoires auxquels les explosions de particules se produisent ont déjà été calculés dans les fichiers state qui vous sont fournis.

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 "ramping up"**.

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 le livrable préparatoire décrit dans ce document, puis ceux du projet et de la compétition finale.

Directives

Le travail préparatoire est validé 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 final pour validation par l'équipe enseignante est déposé sur le git dans le dossier */Documents/Rendus*. Il est fortement recommandé de l'y déposer **avant la 2ème séance plénière**. Les compte-rendus ne seront plus considérés après le 1er Septembre 2023.