Introduction au C++

Robotique/UE 3.1 - TP 1 - Septembre 2019

Supports de cours disponibles sur
`www.simon-rohou.fr/cours/c++`

L'objectif de ce TD est de se familiariser avec les éléments de base du C++. Nous proposons ici une application de robotique simple qui évoluera dans les prochaines séances.

A. Un robot sur une route sans fin

On considère un robot tournant sur une route circulaire de circonférence $l = 100$ et de rayon $r = l/2\pi$. Le robot $\mathcal{R}$ se décrit par les équations d'état suivantes :

$$\begin{cases} \dot{x} &= v, \\ \dot{v} &= u. \end{cases} \tag{1}$$

Le vecteur d'état du robot est $(x, v)^\intercal$ où x correspond à sa position sur le cercle et v à sa vitesse. La commande u correspond à l'accélération du robot. La simulation du système (1) se fera par une simple méthode d'Euler.

Le but de cet exercice est d'animer le robot afin de le voir évoluer sur le cercle, comme représenté sur la Figure 1.

Important : vous devez compiler votre programme C++ à chaque étape.

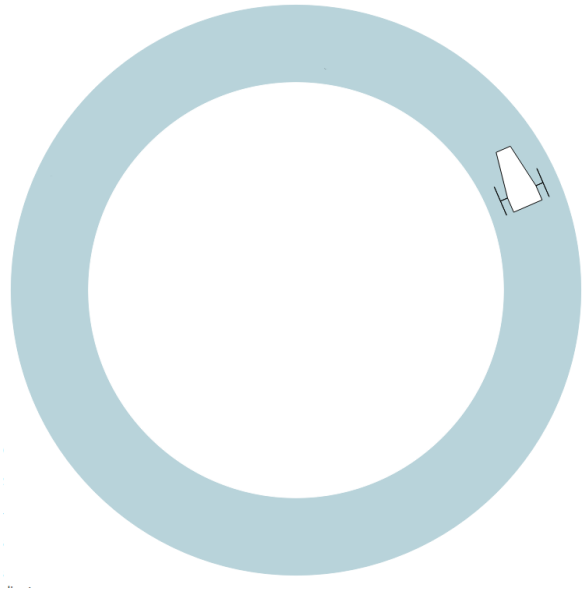

Figure 1: Robot char sur route circulaire.

B. Initialisation du projet

1. Dans un nouveau répertoire, créer un fichier `main.cpp` avec un programme principal déclarant les constantes $\ell = 100$, $\delta = 0.05$, $v_0 = 3$ et $u = 1$. δ correspond au pas d'intégration utilisé pour la simulation du système.

C. Affichages graphiques

Le logiciel VIBes a été développé à l'ENSTA Bretagne afin de fournir des fonctionnalités graphiques simples et décorrélées du programme métier. Il va nous permettre d'afficher la route et le robot s'y trouvant. VIBes est un exécutable à lancer en parallèle d'un projet. Ce dernier devra envoyer des instructions graphiques à VIBes pour afficher des objets.

C1. Installation

Les sources de VIBes se trouvent sur le dépôt `https://github.com/ENSTABretagneRobotics/VIBES`.

2. Dans un autre répertoire, clonez le dépôt avec la commande suivante :
   `git clone https://github.com/ENSTABretagneRobotics/VIBES`
   (utiliser si besoin `git config --global http.proxy http://adresseduproxy:8080`)

3. Installez les paquets nécessaires :
   `sudo apt-get install qt5-default libqt5svg5-dev`

4. L'exécutable `VIBes-viewer` doit être compilé une fois, indépendamment. Placez-vous dans `./VIBES/viewer/` et compilez le projet en utilisant la commande suivante :

   ```
   mkdir make ; cd make ; cmake .. ; make
   ```

   Lancez l'exécutable `VIBes-viewer`.

5. Les instructions graphiques forment une API (Application Programming Interface) permettant de communiquer avec VIBes. Elles se présentent sous la forme de procédures C++ déjà implémentées. Copiez les deux fichiers de `./VIBES/client-api/C++/src/` (`.cpp` et `.h`) dans votre répertoire de projet.
   La compilation de votre projet doit maintenant prendre en compte votre programme principal `main.cpp` ainsi que l'API de VIBes, `vibes.cpp` :

   ```
   g++ main.cpp vibes.cpp -o circularroad
   ```

   Pensez à inclure les *headers* de VIBes dans votre `main.cpp`.

C2. Afficher la route

6. Dans le programme principal, on utilisera les instructions suivantes pour initialiser la vue graphique :

   ```
   vibes::beginDrawing(); // initialisation de VIBes
   vibes::newFigure("Road"); // création d'une figure
   vibes::setFigureProperties("Road",
        vibesParams("x", 100, "y", 100,
        "width", 400, "height", 400)); // propriétés de la figure
   vibes::axisLimits(-20., 20., -20., 20.); // dimensions de la vue graphique
   ```

   et on quittera proprement l'affichage avec la commande `vibes::endDrawing();` en fin de programme.

7. À l'aide de la fonction `vibes::drawCircle(...)`[1], afficher deux disques concentriques pour dessiner la route (d'une largeur de 5). Ces disques seront dessinés dans une nouvelle procédure `draw_road(...)`, à définir dans `main.cpp`. On veillera à réinitialiser la figure avant leur affichage avec `vibes::clearFigure("Road");`

8. Afficher la route dans le programme principal.

C3. Afficher une voiture

9. Une seconde procédure `draw_robot()` dessinera un tank à l'aide de la commande :

   ```
   vibes::drawTank(px, py, theta + (M_PI/2.), length,
                   "black[white]", vibesParams("figure", "Road")); // prendre length = 4
   ```

   Avec `M_PI`, une constante π définie dans `<math.h>`. Si vous ne disposez pas de la dernière version de VIBes, préférez la procédure `vibes::drawVehicle(...)` avec les mêmes paramètres. La position angulaire θ du véhicule sur le cercle est donnée par $\theta = {}^{x}/_{r}$ où r est le rayon de la route. Cette dernière information doit donc être communiquée en paramètre de `draw_robot()`.

10. Afficher la voiture dans le programme principal.

[1] Pour plus d'informations, se référer aux déclarations des fonctions de l'API, accessibles dans le fichier `vibes.h`.

D. Une voiture non régulée

11. Ajouter une méthode représentant la fonction d'évolution du système. Son prototype

    ```
    void f(float x, float v, float u, float& xdot, float& vdot);
    ```

    retournera par arguments les composantes de la dérivée du vecteur d'état $(x, v)^\intercal$ du véhicule. Se référer à l'Équation (1).

12. L'intégration de l'équation différentielle $(\dot{x}, \dot{v})^\intercal = \mathbf{f}((x, v)^\intercal, u)$ se fera par une méthode d'Euler. Créer une fonction `euler(...)` prenant en entrée δ, $\dot{x}$, $\dot{v}$ et en entrées/sorties : x, v.

13. Dans le programme principal, simuler le système de $t = 0$ à $t = 100$ par pas de temps δ en utilisant les procédures déjà implémentées. La commande `usleep()` permettra un affichage fluide à l'écran :

    ```
    #include <unistd.h> // pour usleep
    ...
    usleep(dt * 500000.); // vitesse d'animation
    ```