TD et TP L3 EEAR UE5.24

C/C++ et SGBD

2008-2009

TD 1 Présentation de l'étude de cas KheperaVR (Virtual Reality)
TD 2 BD MySQL et interface Web
TD 3 BD MySQL et API C

TP 1 KheperaVR, pilotage du robot virtuel
TP 2 Commande du *Khepera* en mode Terminal via la liaison série
TP 3 K2Robot, interface C++ du Khepera
TP 4 KheperaVR, intégrer K2Robot pour piloter le robot réel
TP 5 KheperaVR, envoi des informations SQL au serveur
TP 6 Visualisation des informations SQL sur un navigateur distant
TP 7 Modèle sémantique du robot et génération du DDL avec MEGA
TP 8 Bilan du projet

Les informations complémentaires, mise à jour, programmes de départ, sont tous disponibles sur ARCHE, L3 UE 5.24 :

http://arche.uhp-nancy.fr/course/view.php?id=1703

TD et TP
C/C++ et SGBD
L3 EEAR - UE 5. 24

Préambule :

Dans cette partie de l'UE 5.24, les 3 TD et les 8 TP ne sont pas indépendants, bien au contraire. Il s'agit de partir d'un système opérationnel de supervision – commande, de comprendre et d'implanter les modifications au sein du système initial. Le déploiement se fera en mode simulation VR et en mode expérimental sur un petit robot, le *Khepera*.

Une grande partie de travail individuel est nécessaire au bon déroulement de l'étude. Elle concerne surtout la lecture des documents, la compréhension des programmes C/C++ fournis par l'équipe pédagogique et la mise en œuvre des moyens de communication entre les différents programmes et Robots.

Environnement de travail

L'environnement choisi pour cette étude utilise Microsoft Visual C++ / Studio 2005. Cet outil de développement est disponible à l'ATELA et à l'AIP, pendant les créneaux d'accès libre aux salles informatiques. Vous pouvez aussi le télécharger gratuitement en tant qu'étudiant de l'UHP dans le cadre du partenariat MSDN.

Une interface de visualisation Web sera mise à disposition pour notre cas d'étude. On disposera donc d'un serveur Web pour visualiser le suivi du Robot.

Robot Virtuel, KheperaVR et mfc_gl-Robot

KheperaVR est une application Windows qui sera le programme central. Il pilotera, via une interface transparente à l'utilisateur (WM : Windows Messages) une autre application : mfc_gl-Robot. Cette dernière est disponible sur ARCHE et sur le serveur de l'AIP. Elle gère l'environnement du robot virtuel RobotK. Elle reçoit les ordres de KheperaVR et lui renvoie les résultats de déplacement (vitesse, position) ainsi que l'état des capteurs.

Robot réel

Le pilotage d'un robot réel, le *Khepera* se fera par une bibliothèque pré-existante en C++ (K2Robot) qui permet de le commander. Elle sera ensuite utilisée comme interface par le programme KheperaVR.

Le diagramme suivant détaille les interactions entre les différents programmes et applications.

Travail demandé

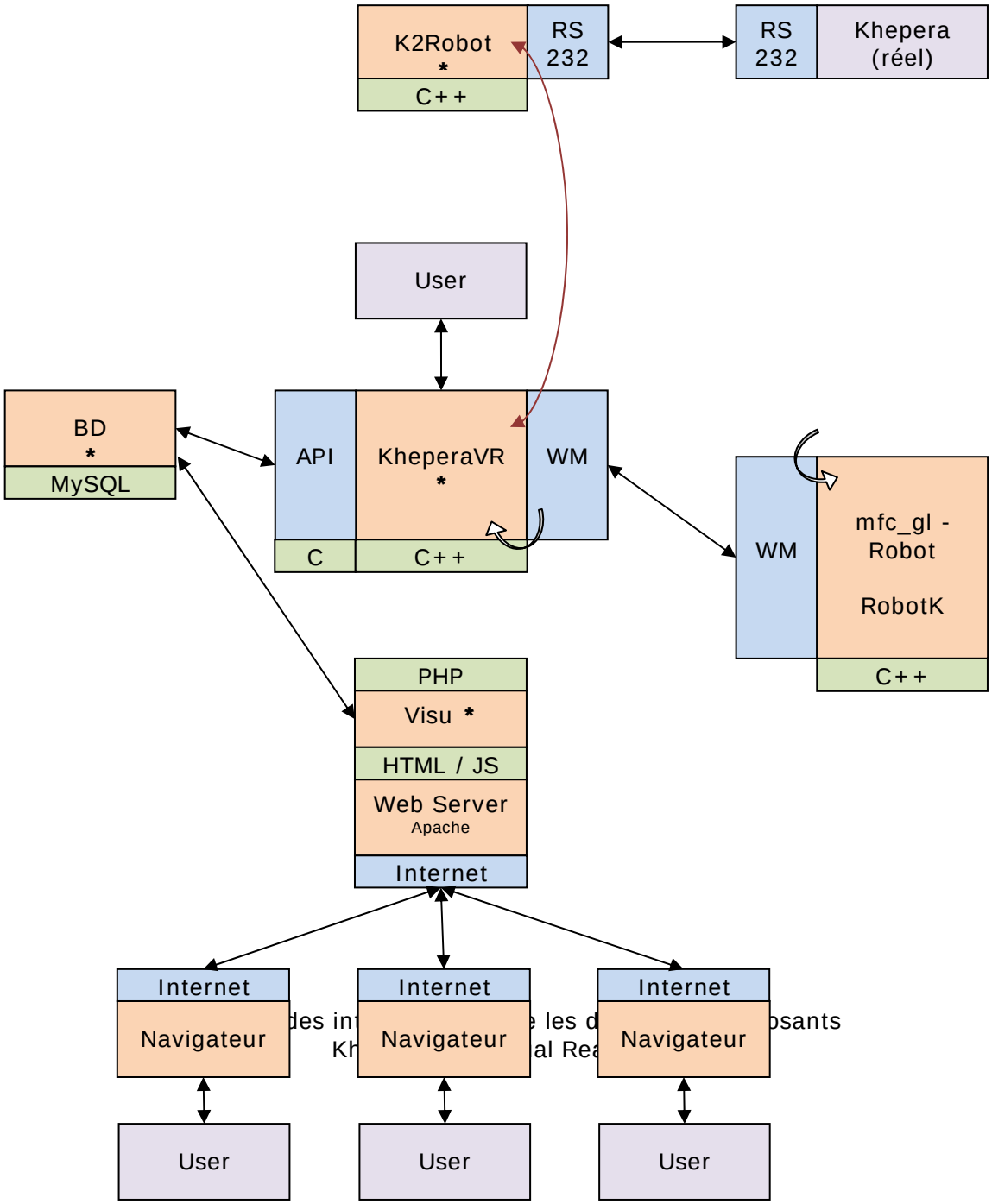
Il ne s'agit pas de réécrire les applications mais plutôt de comprendre leur fonctionnement et leur imbrication. Ensuite, il faut compléter, dans le programme KheperaVR, les trois fonctions demandées.

Pour ce faire, il faut se poser les bonnes questions par rapport à la modélisation et aux variables que l'on doit récupérer et envoyer dans la base de données. Ensuite, établir les requêtes SQL adéquates. La visualisation des résultats est alors aisée et permet une vérification immédiate du fonctionnement du système.

Evaluation de l'étude

L'évaluation se fera par un contrôle continu (préparation du travail demandé, algorithmique et programmation), une évaluation orale et aussi sur un compte rendu final de l'étude ; document

à rendre, à la fin des TP. Le compte rendu doit comporter les explications sur les fonctions que vous avez implémentées et non pas un listing complet des programmes.



Bibliographie en ligne :

Visual C++ et MFC via MSDN
http://msdn2.microsoft.com/en-us/library/ms950410.aspx
http://msdn.microsoft.com/library/fre/default.asp?url=/library/FRE/vccore/html/vcorimfcoverview.asp
http://msdn.microsoft.com/library/fre/default.asp?url=/library/FRE/vccore/html/_core_using_the_classes_to_write_applications_for_windows.asp

http://c.developpez.com/cours/#cb (langage C)
http://cpp.developpez.com/faq/vc/?page=sommaire (FAQ Visual C++ / MFC)

Tutorial MySQL / VC++
http://mysql.developpez.com/cours/
http://www.infres.enst.fr/~danzart/mysql/SQL.phtml

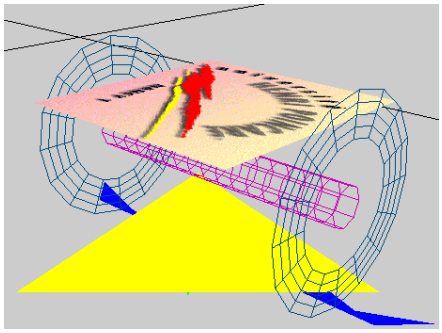
MySQL C API
http://www.siteduzero.com/tutoriel-3-34984-utiliser-l-api-mysql-dans-vos-programmes.html
http://www.cyberciti.biz/tips/linux-unix-connect-mysql-c-api-program.html
http://dev.mysql.com/doc/refman/5.0/en/c.html

Robot Khepera
http://www.k-team.com
http://www.k-team.com/kteam/index.php?site=1&rub=22&page=17&version=EN

TD n°1
Présentation de l'étude de cas KheperaVR
(Virtual Reality)

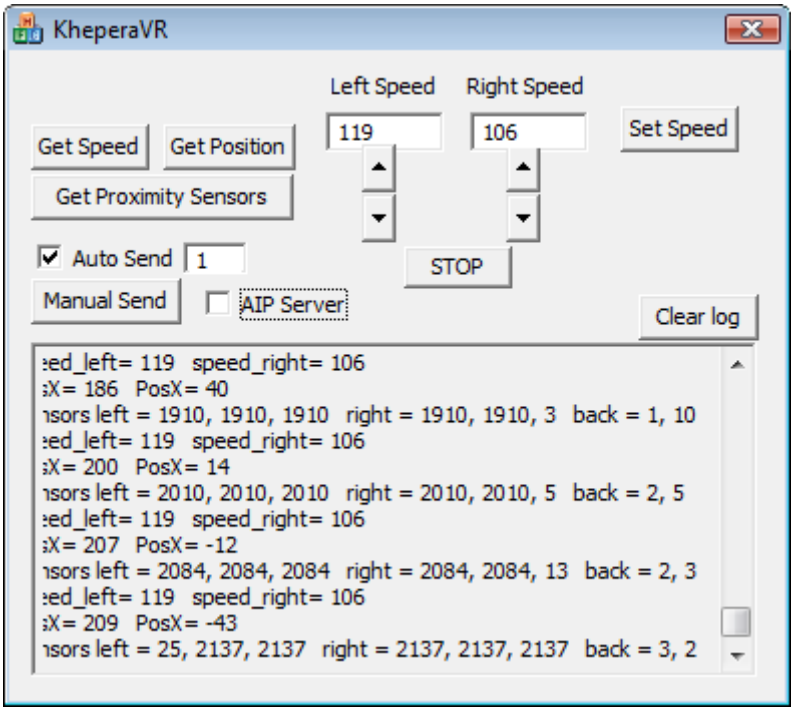
Buts :
Présentation de l'étude d'intégration que l'on va faire autour des robots et qui va servir de support pour l'utilisation des BD et l'interfaçage C/C++.

Présentation :
Voici les éléments que l'on doit apprendre à distinguer :

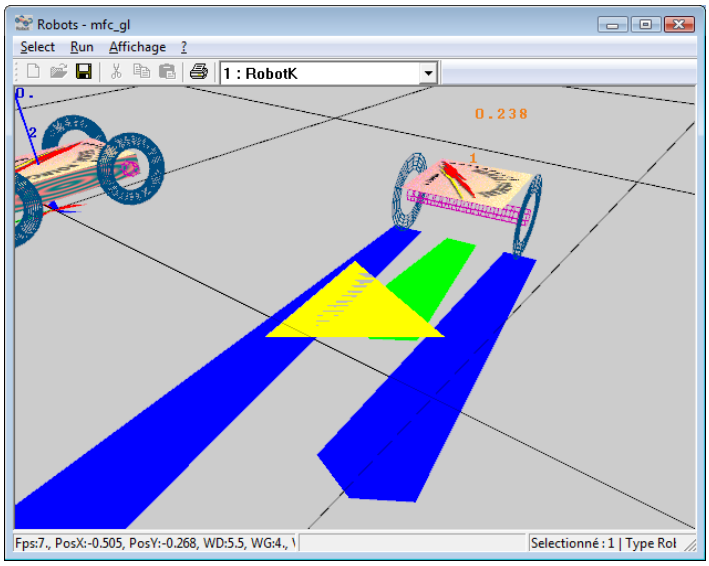


Robot virtuel (*RobotK*)

Robot réel (Khepera)



Vue de **KheperaVR** en cours d'exécution



Vue de **Robots – mfc_gl**, le robot virtuel *RobotK* y est piloté à distance par **KheperaVR**

Manipulation :

Ces programmes doivent être téléchargés à partir de la plateforme ARCHE.

Robots – mfc_gl, version exécutable seulement (http://193.55.104.245/mfc_gl/).

KheperaVR, version code source C/C++ sous Visual Studio 2005.

TD n°2
BD MySQL et interface Web

Buts :

Apprendre à se connecter à une base de données (MySQL), à faire des requêtes MySQL et à récupérer les informations renvoyées par la BD.

Présentation :

L'équipe enseignante vous fournit un exemple de départ et vous montrera comment utiliser l'interface web pour accéder à la base de données et aux enregistrements.

Manipulation :

Utilisez l'utilitaire phpmyadmin (http://193.55.104.245/baghliadmin/index.php) pour créer une table **ex_1_*votrenom*** dans la base de données **test** sur le serveur de l'AIP 193.55.104.245. Pour cela exécuter la requête SQL suivante :

```
-- --------------------------------------------------------
-- Base de données: `test`
-- --------------------------------------------------------
-- Structure de la table `ex_1`
--
DROP TABLE IF EXISTS `ex_1`;
CREATE TABLE `ex_1` (
  `ID` int(11) NOT NULL auto_increment,
  `NOM` tinytext NOT NULL,
  `PRENOM` tinytext NOT NULL,
  `DATE_NAISSANCE` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 COMMENT='ma premiere table' AUTO_INCREMENT=6 ;
--
-- Contenu de la table `ex_1`
--
INSERT INTO `ex_1` VALUES(1, 'Dupont', 'Denis', '1977-09-18');
INSERT INTO `ex_1` VALUES(2, 'Dupont', 'Marie', '1978-02-09');
INSERT INTO `ex_1` VALUES(3, 'Sheen', 'Sylvain', '1970-12-22');
INSERT INTO `ex_1` VALUES(4, 'Sand', 'Kamal', '1971-05-30');
INSERT INTO `ex_1` VALUES(5, 'Fisher', 'Samia', '1973-10-28');
```

- Copier dans le répertoire ex_1, sous votre nom, ex1_***votrenom***_disp.php, le fichier ex1_baghli_disp.php disponible sur le serveur.
- Visualiser la sortie par : http://193.55.104.245/ex1/ex1_***votrenom***_disp.php
- Modifier le fichier afin de classer les enregistrements de la table par ordre alphabétique sur le Nom, sur le prénom.
- Modifier le fichier afin de n'afficher que les 2 premiers enregistrements (classés par Nom, par prénoms et ensuite par ordre décroissant).
- Ajouter au fichier une requête pour ajouter une entrée (enregistrement) supplémentaire.

TD n°3
BD MySQL et API C

But :
Utiliser l'API C MySQL pour accéder aux enregistrements de la base de données.

Présentation :
L'équipe enseignante vous fournit un exemple de départ et vous montrera comment utiliser la bibliothèque C (API MySQL) pour accéder à la base de données et aux enregistrements.

Manipulation :

- Afficher à l'aide du programme en langage C, le nombre des enregistrements de la table ex_1_votrenom que vous aviez créée en TD 2.
- Afficher le contenu des enregistrements de cette table.
  Pour ce faire, entrez les lignes de codes suivantes

```
            //à nouveau la requête qui sélectionne tout dans ma table ex_1
            mysql_query(&mysql, "SELECT * FROM ex_1");
            result = mysql_use_result(&mysql);
             //On récupère le nombre de champs
            num_champs = mysql_num_fields(result);
             while (( row =  mysql_fetch_row(result)))
              {
              //On déclare un pointeur long non signé pour y stocker la taille des
valeurs
              unsigned long *lengths;
              //On stocke cette taille dans le pointeur
              lengths = mysql_fetch_lengths(result);
              //On fait une boucle pour avoir la valeur de chaque champs
              for(i = 0; i < num_champs; i++)
                {
                //On ecrit toutes les valeurs
                printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");
                }
              printf("\n");
              }
```

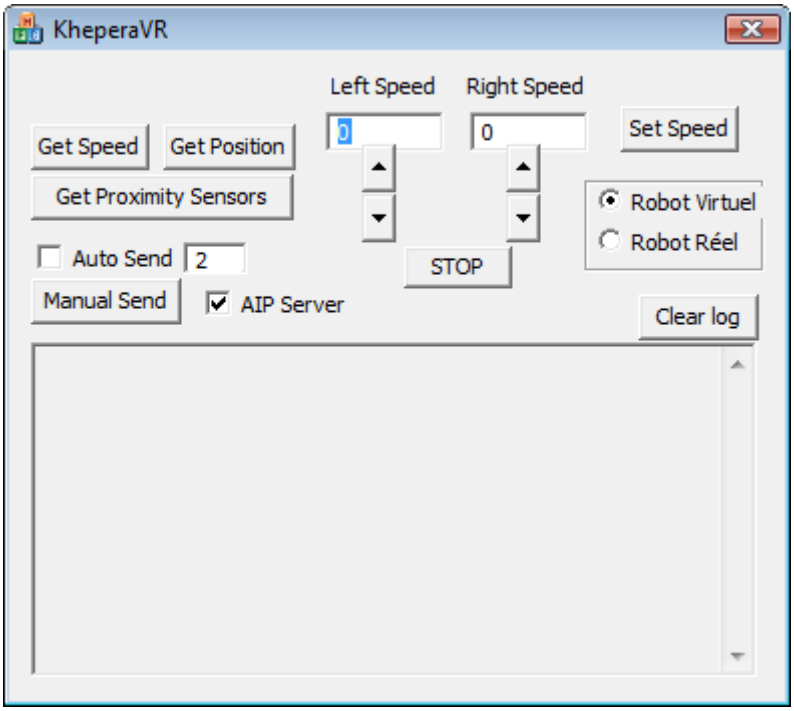
- Ajouter au programme une requête pour ajouter une entrée (enregistrement) supplémentaire, où vous renseignez tous les champs de l'enregistrement.

TP n°1
KheperaVR, pilotage du robot virtuel

Buts :
Prendre en main le code C++ du **KheperaVR**.
Implanter une fonction simple qui envoie un ordre d'arrêt au robot virtuel qui "tourne" sur l'application **Robots – mfc_gl**.

Présentation :
On vous expliquera les différentes fonctions membres de la classe `CKheperaVRDlg`.
C'est une classe qui hérite de `CDialog` et qui représente la fenêtre principale de **KheperaVR**.
Les membres de la classe `CKheperaVRDlg` sont déclarés dans `KheperaVRDlg.h` et définis dans `KheperaVRDlg.cpp`. Ce dernier est le principal fichier sur lequel on interviendra tout au long de cette série de TP.

Manipulation :
Ajouter à `KheperaVRDlg.cpp` des lignes de codes dans la fonction qui est appelée quand l'utilisateur clique sur le bouton **STOP**.
Cette fonction doit envoyer un ordre d'arrêt sous forme d'une chaîne de caractères.
Inspirer vous d'autres fonctions du programme `KheperaVRDlg.cpp`.



Vue de **KheperaVR** en cours d'exécution

TP n°2
Commande du Khepera en mode Terminal via la liaison série

Le but de ce TP est de piloter le robot **réel** à l'aide d'ordres envoyés via la liaison RS-232 à l'aide de l'hyperterminal de Windows.

Le Khépéra est décrit dans l'annexe I. Il contient notamment 2 leds (petites lumières) et 8 capteurs infra-rouge. Ces capteurs servent à détecter, soit des obstacles à proximité de l'objet, soit des sources lumineuses.

Dans ce TP, vous travaillerez seulement sur la détection d'obstacles, par exemple en approchant une main du robot. Notez qu'une détection de sources lumineuses serait aussi envisageable.

Les leds n'ont pas une importance prépondérante. Ce sont des terminaisons qui pourraient servir pour établir des connexions entre le robot et d'autres équipements électroniques.

Hyperterminal Windows

Avant toute chose, activer l'hyperterminal Windows : **(*Démarrer* -> *Programme* -> *Accessoires* -> *Communications* -> *Hyperterminal*).**

L'hyperterminal est l'outil qui permet de gérer directement (sans programme ni bibliothèque C++ pour l'instant) les connexions de votre PC.

Lorsque le Khépéra est branché sur le port série, l'hyperterminal doit détecter la connexion, et vous devez configurer les paramètres de connexion (cf. annexe I, pages 22-23).
Vérifier que le port que vous configurez correspond bien à celui qui est physiquement connecté à l'interface-RS 232 du robot *Khepera*.

On peut ensuite envoyer des commandes dites **«** bas-niveau **»** au Khépéra, qui prennent la forme d'une chaîne de caractères ASCII (cf. annexe I, pages 47-51).

Quelles sont les commandes bas-niveau qui permettent de :

- Déplacer le robot de façon rectiligne ?
- Imprimer au robot un mouvement circulaire ?
- Faire tourner le robot sur lui-même ?
- Arrêter le robot ?
- Allumer/éteindre les leds ?
- Détecter des obstacles ?

Tester quelques commandes sur le robot.

N.B. Sous Windows Vista, qui ne dispose pas de l'Hyperterminal, vous pouvez utiliser le logiciel libre **putty.exe**

TP n°3
K2Robot, interface C++ du Khepera

Lire et comprendre le fichier **« k2Robot.h »** fourni en annexe II. La lecture de **« k2Robot.cpp »** n'est pas nécessaire à ce niveau.

La lecture du seul fichier **« k2Robot.h »** et de ses commentaires doit vous permettre de comprendre la classe **k2Robot**, des attributs du robot jusqu'aux fonctions membres qui permettent de le commander, et de lire les informations en provenance du robot.

Pourquoi la fonction `move` est-elle spécifiée deux fois ?
Quels sont les numéros des capteurs associés aux positions « Left 10 », « Left 45 », « back Left », *etc.* ?
A quoi correspond le paramètre `verbose` dans le constructeur de la classe ?

Fonctions implémentées dans « k2Robot.cpp »

Le but n'est pas de comprendre en détail ce fichier source, mais arrêtons-nous sur quelques fonctions très simples.

Lire le corps des fonctions `move` (deux versions), de la fonction `stop`, et des fonctions qui gèrent les leds. Faîtes le lien avec les commandes bas-niveau du Khépéra.

Quel est le rôle de la fonction `command` ?

Fonction de test « k2Robot_Test.cpp »

On veut réaliser une fonction `main()` qui utilise la bibliothèque **k2Robot** pour commander physiquement le Khépéra.

Projet sous Visual C++

Pour cela, il faut créer un nouveau projet Visual C++, inclure les fichiers **« k2Robot.h »** et **« k2Robot.cpp »,** puis **« k2Robot_Test.cpp »** pour pouvoir compiler et faire l'édition de liens.

Fichier source

Le squelette de la fonction `main()` est fourni dans l'annexe III.

Modifier cette fonction, de façon à appliquer au robot le parcours basique suivant :

1. Déplacement rectiligne pendant 2 secondes.
2. Rotation de 180 degrés sur place.
3. Déplacement rectiligne pendant 2 secondes vers le point de départ.
4. Arrêt sur le point de départ.

Vous effectuerez le test deux fois : en désactivant le mode verbose, puis en l'activant.

Lors de la phase de test avec le Khépéra, vous devez confronter les valeurs imprimées au Khépéra (vitesses de rotation des roues droite et gauche, durée du déplacement) avec la distance réelle parcourue par le robot, pour valider le modèle établi au TP1.

Mesurer « à la main » la distance réelle parcourue (en centimètres) ou l'angle de rotation (en degrés) du robot de façon à établir une correspondance entre les commandes bas-niveau et le déplacement réél.

Détection d'obstacle

Modifier le programme précédent de façon à lire les capteurs d'obstacle.

Concevoir un algorithme qui effectue une détection simple d'obstacle. Ajouter la détection d'obstacle au parcours du robot en surchargeant la classe **k2Robot** en une classe **k2RobotBis**. La nouvelle classe est quasi-identique à k2Robot, à la différence qu'elle contient la fonction `DetectionObstacle()` supplémentaire.

Annexe I : Morceaux choisis de la documentation du Khépéra

- Pages 8, 12 : présentation du robot, de ses accessoires et ses capteurs infra-rouge de proximité;
- Page 22 : Configuration de la communication entre le robot et le PC ;
- Page 23 : Communication avec le robot via l'hyperterminal Windows ;
- Pages 47-51 : liste des commandes du robot.

Annexe II : Fichier « k2Robot.h »

```
#if !defined(K2ROBOT_H__INCLUDED_)
#define K2ROBOT_H__INCLUDED_

#include <stdio.h>
#include <windows.h>
#include <strstream>

class k2Robot
{
  protected:
    bool verbose_flag;
    HANDLE hCom;


  public:
    // Constructor : connect to the robot via the serial port (default is
COM1)
    // An optional verbose flag control debug print statements
    k2Robot(char* serial_port = "COM1", bool verbose = false);

    // Destructor : close the serial connection
    ~k2Robot();

    // Time-framed move method
    // (left,right) velocities are applied to the robot's motors
    // during msec milliseconds
    void move(DWORD msec, DWORD left, DWORD right);

    // Unlimited move method
    // (left,right) velocities are applied to the robot's motors
    void move(DWORD left, DWORD right);

    // Stop current movement
    // Applied a (0,0) velocity to the robot's motors
    void stop();

    // Robot's LED managment
    // n is 0 or 1 (default is 0)
    void led_on(int n = 0);
    void led_off(int n = 0);
    void led_toggle(int n = 0);

    // Read Robot Info
    void read_speed();
    void read_position();
    void read_sensors();
```

```
    // Return individual values
    int get_sensor_prox_left10();
    int get_sensor_prox_left45();
    int get_sensor_prox_left90();
    int get_sensor_prox_backLeft();
    int get_sensor_prox_right10();
    int get_sensor_prox_right45();
    int get_sensor_prox_right90();
    int get_sensor_prox_backRight();
    int get_PosX();
    int get_PosY();
    int get_LeftSpeed();
    int get_RightSpeed();

    // Control debug verbosity (same as constructor argument)
    // level = 0 means no statment (except for error)
    void verboseLevel(int level);

    // Low level method to communicate with the robot
    // The message string is sent via the serial port
    // The response of the robot is returned in res
    // Only maxsize bytes of reponse is placed n res
    // If res is null, nothing is returned
    // res should have at least maxsize allocated
    int command(char* msg, char* res = NULL, DWORD maxsize = 0);

  private:
    int PosX, PosY;
    int LeftSpeed, RightSpeed;
    int sensors_prox[8];
};

#endif // !defined(K2ROBOT_H__INCLUDED_)
```

Annexe III : Fichier « k2Robot_Test.cpp » à compléter

```
#include "k2Robot.h"
#include <conio.h>

int main(int argc, char* argv[])
{
  printf("DEMARRAGE DU TEST DE K2ROBOT...\n");

  // INSTANCIATION : connexion au port série COM1
  k2Robot robot;      // A COMPLETER

  // MOUVEMENTS DE BASE

  // 1. Déplacement rectiligne pendant 2 secondes

  // 2. Rotation de 180 degrés sur place

  // 3. Déplacement rectiligne pendant 2 secondes vers le point de départ

  // ARRET OBLIGATOIRE

  printf("FIN DU TEST.\n");
  getch();
  return(0);
}
```

TP n°4
KheperaVR, intégrer K2Robot pour piloter le robot réel

Buts :

Intégrer, dans le projet KheperaVR, le programme K2Robot afin que l'on puisse piloter le Robot Réel comme on le faisait auparavant avec le Robot Virtuel.

Présentation :

Pour l'instant, on a piloté le robot virtuel à l'aide de KheperaVR. Nous allons ajouter l'interface K2Robot qui permettra à KheperaVR de dialoguer avec le robot réel.

Manipulation :

- Modifier la routine :
  `void CKheperaVRDlg::OnBnClickedRdrobotreel();`
  afin qu'elle puisse créer une instance de la classe **`k2Robot`**.

- Au moment de l'envoi des ordres, la routine adéquate doit, en fonction de la variable booléenne `UseRobotReel`, choisir quel est le mode de communication Robot réel ou virtuel. Compléter la fonction.
  Prévoir la réception de l'information que vous renvoie le Robot réel.
  Regarder la fonction : `int k2Robot::command(…);`

TP n°5
KheperaVR, envoi des informations SQL au serveur

Buts :
Permettre à KheperaVR, d'alimenter une BD avec les données provenant soit du Robot Réel, soit du Robot Virtuel.

Présentation :
On souhaite maintenant alimenter la BD MySQL du serveur de l'AIP avec les données du Robot (réel ou virtuel). Cela nous permettra par la suite de suivre à distance, à l'aide d'un navigateur internet, l'évolution du robot.

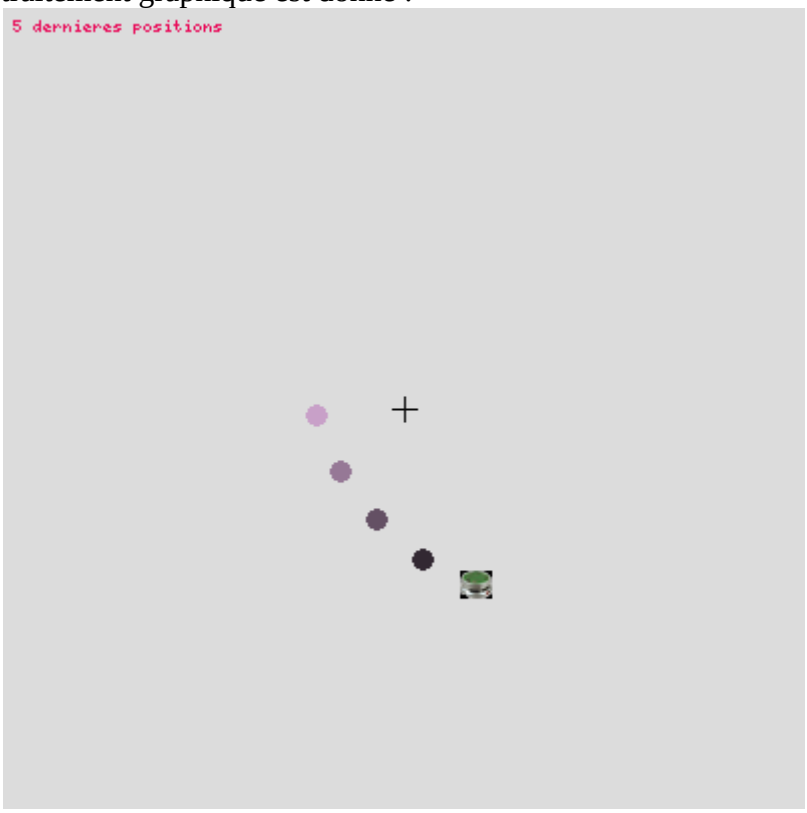
Manipulation :
Il faut choisir les informations à envoyer et le type (int, datetime, tinytext) de chaque champ.
Créer la table adéquate dans la base de données test avec l'appellation : **ex_5_*votrenom***.
Ajouter au projet KheperaVR la fonction `SendSQLRecord()`.
Elle doit être appelée automatiquement par le *timer* de manière à mettre à jour la BD en y ajoutant des enregistrements à chaque fois.
Vérifier à l'aide de *phpmyadmin* que les informations sont bien enregistrées.

TP n°6
Visualisation des informations SQL sur un navigateur distant

Buts :
Manipuler, côté serveur, les enregistrements de la BD envoyés en ligne par KheperaVR.

Présentation :
Le robot a envoyé ou continue à envoyer, en ligne, des informations via KheperaVR qui enrichissent la BD MySQL (TP précédent).
On doit maintenant exploiter ces informations pour les visualiser et les traiter.

Un exemple de traitement graphique est donné :



Il présente les 5 dernières positions du robot.

Manipulation :
Traiter de manière textuelle les informations de la BD pour n'afficher que les 10 derniers enregistrements reçus sous forme de tableau. Inspirez vous du TD n°2.
Rafraîchir le navigateur pour observer le renouvellement des informations (chercher sur google comment : *rafraichir page html*).

TP n°7
Modèle sémantique du robot et génération du DDL avec MEGA

Buts : le but de ce TP est de construire le modèle sémantique du robot Khepera et de son environnement et de générer le script SQL pour MySQL via l'outil MEGA.

Présentation : le but du TP5 est d'alimenter une base de données avec les données provenant soit du robot virtuel soit du robot réel. C'est pourquoi, il est important de concevoir proprement le Système de Gestion de Base de Données du robot même si une seule table sera utilisée.

Manipulation : il faut construire un modèle sémantique du robot Khepera et de son environnement et générer le script SQL à l'aide du logiciel MEGA.
Proposer un modèle sémantique du robot et de son environnement.
Rentrer le modèle dans l'outil MEGA.
Générer le modèle relationnel.
Générer le script SQL pour MySQL.