heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

ReDS
Reconfigurable & embedded
Digital Systems

QtrainSim

Kevin Georgy, Jéremie Ecoffey et Daniel Molla

Département TIC
Institut REDS
Version du 5 mars 2012

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Révisions

Date	Version	Ingénieur	Révision
17/01/09	v1.0	KGY	Version initiale
26/03/09	v1.1	KGY	Ajout des plans des maquettes et des fonctionnalités GUI
05/03/12	v1.2	DMO	Mise à niveau de la documentation pour l'application Qt

Table 1 – Révisions

Mise en forme

- Les noms propres sont écrits en Petites Capitales
- Les fichiers, dossiers ou commandes sont en `chasse fixe`
- Les termes étrangers, les nouveaux termes ou les termes techniques sont en *emphasis*
- Le *listing* de code prend la forme suivante :
  - Pour des commandes ou un extrait de code source :

```
./configure
make -j8
make install
```

  - Pour du code sources :

```
1 #include <stdio.h>
2
3 int main(int argc, char ** argv)
4 {
5     print("Un exemple...\n");
6     return 0;
7 }
```

Table des matières

1 Introduction

Qtrainsim est un programme de simulation des maquettes de train Märklin. Grâce à la librairie libtrainsim, il est possible d'écrire des programmes C utilisant ce simulateur.

Ce document indique comment installer le simulateur sur différentes plate-formes disponibles et comment l'utiliser.

Installation 2

2.1 Mise en place

Cette documentation se base sur l'environnement Qt SDK 4.7.3.

Récupérer le projet associé à la plate-forme utilisée. Une fois récupéré, lancer QtCreator à partir du fichier `QtrainSim.pro` du projet ou alors ouvrir le projet à partir de QtCreator. Verifier dans le fichier `QtrainSim.pro` que la ligne suivante est bien commentée :

```
1 CONFIG += MAQUETTE
```

Cette ligne permet de spécifier à l'application si elle doit utiliser la librairie du simulateur ou la librairie des maquettes physiques. Compiler puis lancer l'application. La simulateur est désormais prêt à l'emploi.

2.2 Écriture d'un programme

Les fonctions fournies par la librairie sont définies dans `src/ctrain_handler.h` (*listing* 2.1).

```
1  #ifndef H_CTRAIN_HANDLER
2
3  #define H_CTRAIN_HANDLER
4
5  /*
6   * Fichier          : ctrain_handler.h
7   * Auteur           : Kevin Georgy
8   *
9   * Date de creation : 4.2.2009
10  * But              : Fournit les fonctions de controle du simulateur/maquette de trains.
11  * Revision         : 27.3.2009 (CEZ)
12  *                    27.4.2009 (KGY) Ajout du extern "CE pour les applications c++
13  *                    08.9.2011 (Jérémie Ecoffey) Adaptation au nouveau simulateur.
14  */
15
16 #ifdef __cplusplus
17 extern "C" {
18 #endif
19
20 // Vitesse a l'arret
21 #define VITESSE_NULLE 0
22
23 // Vitesse minimum
24 #define VITESSE_MINIMUM 3
25
26 // Vitesse maximum
27 #define VITESSE_MAXIMUM 14
28
29 // Numero max. d'aiguillage
30 #define MAX_AIGUILLAGES 80
31
32 // Numero max. de contact
33 #define MAX_CONTACTS 64
34
35 // Numero max. de loco
36 #define MAX_LOCOS 80
37
38 // Direction des aiguillages
```

```
39 #define DEVIE 0
40 #define TOUT_DROIT 1
41
42 // Etat des phares
43 #define ETEINT 0
44 #define ALLUME 1
45
46 /*
47  * Initialise la communication avec la maquette/simulateur.
48  * A appeler au debut du programme client.
49  */
50 void init_maquette(void);
51
52 /*
53  * Met fin a la simulation. A appeler a la fin du programme client.
54  */
55 void mettre_maquette_hors_service(void);
56
57 /*
58  * Realimente la maquette. Inutile apres init_maquette().
59  */
60 void mettre_maquette_en_service(void);
61
62 /*
63  * Change la direction d'un aiguillage.
64  *   no_aiguillage : No de l'aiguillage a diriger.
65  *   direction     : Nouvelle direction. (DEVIE ou TOUT_DROIT)
66  *   temps_alim    : Temps l'alimentation minimal du bobinage de l'aiguillage.
67  */
68 void diriger_aiguillage(int no_aiguillage, int direction, int temps_alim);
69
70 /*
71  * Attend l'activation du contact donne.
72  *   no_contact : No du contact dont on attend l'activation.
73  */
74 void attendre_contact(int no_contact);
75
76 /*
77  * Arrete une locomotive (met sa vitesse a VITESSE_NULLE).
78  *   no_loco : No de la loco a arreter.
79  */
80 void arreter_loco(int no_loco);
81
82 /*
83  * Change la vitesse d'une loco par palier.
84  *   no_loco        : No de la loco a stopper.
85  *   vitesse_future : Vitesse apres changement.
86  * Remarque : Dans le simulateur cette procedure agit comme la fonction
87  *            "mettre_vitesse_loco". Son comportement dépend de l'option
88  *            "Inertie" dans le menu ad hoc.
89  */
90 void mettre_vitesse_progressive(int no_loco, int vitesse_future);
91
92 /*
93  * Permettre d'allumer ou d'eteindre les phares de la locomotive.
94  *   no_loco : No de la loco a controler.
95  *   etat    : Nouvel etat des phares. (ETEINT ou ALLUME)
96  * Remarque : Dans le simulateur cette fonction n'a aucun effet.
97  *            Les phares sont toujours allumés, et indiquent le sens de la loco.
98  */
99 void mettre_fonction_loco(int no_loco, char etat);
100
101 /*
102  * Inverse le sens d'une locomotive, en conservant ou retrouvant sa vitesse originale.
103  *   no_loco : No de la loco a inverser.
104  * Remarque : Dans le simulateur, le comportement dépend de l'option "Inertie" dans le menu ad hoc.
105  */
106 void inverser_sens_loco(int no_loco);
107
108 /*
109  * Change la vitesse d'une loco.
110  *   no_loco : No de la loco a controler.
111  *   vitesse : Nouvelle vitesse.
112  * Remarque : Dans le simulateur, le comportement dépend de l'option "Inertie" dans le menu ad hoc.
113  */
114 void mettre_vitesse_loco(int no_loco, int vitesse);
115
116 /*
117  * Indique au simulateur de demander une loco a l'utilisateur. L'utilisateur entre le
118  * numero et la vitesse de la loco. Celle—ci est ensuite placee entre les contacts
119  * "contact_a" et "contact_b".
120  *   contact_a   : Contact vers lequel la loco va se diriger.
121  *   contact_b   : Contact a l'arriere de la loco.
122  *   numero_loco : Numero de loco choisi par l'utilisateur.
123  *   vitesse     : Vitesse choisie par l'utilisateur.
124  * Remarque : cette méthode n'est pas utilisée dans le simulateur.
125  *            Veuillez utiliser assigner_loco(...);
126  */
```

```
127 void demander_loco(int contact_a, int contact_b, int *no_loco, int *vitesse);
128
129 /*
130  * Indique au simulateur d'assigner une loco.
131  * Le numero et la vitesse de la loco sont définies, et elle est ensuite placee
132  * entre les contacts "contact_a" et "contact_b".
133  *   contact_a    : Contact vers lequel la loco va se diriger.
134  *   contact_b    : Contact a l'arriere de la loco.
135  *   numero_loco : Numero de loco.
136  *   vitesse      : Vitesse de la loco.
137  */
138 void assigner_loco(int contact_a, int contact_b, int no_loco, int vitesse);
139
140
141 /**
142   Sélectionne la maquette à utiliser.
143   Cette fonction termine l'application si la maquette n'est pas trouvée.
144   \param maquette Nom de la maquette.
145   */
146 void selection_maquette(const char *maquette);
147
148
149 #ifdef __cplusplus
150 }
151 #endif
152
153 #endif
```

Listing 2.1 – ctrain_handler.h

Le *listing* 2.2 montre un fichier source minimal pour l'utilisation de la librairie. On remarque l'inclusion *<trainsim/ctrain_handler.h>* qui fournit l'accès au fonctions. L'appel à *init_maquette()* et *mettre_maquette_hors_service()* sont à effectuer obligatoirement en début et fin de programme.

```
1  #include <pthread.h>
2  #include "ctrain_handler.h"
3  #include <errno.h>
4
5
6  // structure qui definit une locomotive
7  typedef struct
8  {
9    int no;
10   int vitesse;
11 } Locomotive;
12
13
14 // Declaration des deux locomotives
15 Locomotive loco1;
16
17
18 void emergency_stop()
19 {
20   printf("\nSTOP!");
21
22   // on arrete les locomotives.
23   arreter_loco(loco1.no);
24 }
25
26
27 // Contacts a parcourir
28 #define NB_CTS 7
29 int parcours[] = {6, 11, 10, 13, 14, 19, 3};
30
31
32 void cmain()
33 {
34   int ct;
35
36   loco1.no =1;
37   loco1.vitesse = 12;
38
39   selection_maquette("MAQUET_B");
40
41   // Demande au simulateur de placer une loco entre les contacts 6 et 11
42   // Recupere le numero et la vitesse saisis par l'utilisateur.
43   assigner_loco(  parcours[1],
44                   parcours[0],
45                   loco1.no,
46                   loco1.vitesse);
47
48   // Dirige les aiguillages sur le parcours
49   diriger_aiguillage(7,TOUT_DROIT,0);
50   diriger_aiguillage(8,DEVIE,0);
51   diriger_aiguillage(5,TOUT_DROIT,0);
52   diriger_aiguillage(9,DEVIE,0);
```

```
53   diriger_aiguillage(10,TOUT_DROIT,0);
54   diriger_aiguillage(14,TOUT_DROIT,0);
55   diriger_aiguillage(13,DEVIE,0);
56   diriger_aiguillage(1,TOUT_DROIT,0);
57
58
59   // Demarre la loco
60   mettre_vitesse_progressive(loco1.no, loco1.vitesse);
61
62   // Attend que la loco passe sur les differents contacts de son parcours.
63   for (ct = 1; ct < NB_CTS; ct++) {
64     attendre_contact(parcours[ct]);
65     printf("Loco %d de vitesse %d a atteint le contact %d.\n", loco1.no, loco1.vitesse, ct);
66   }
67
68   // Stoppe la loco
69   arreter_loco(loco1.no);
70
71   // Fin de la simulation (a effectuer une seule fois en fin de programme, sans effet
72   // sur le simulateur, mais necessaire sur les maquettes reelles).
73   mettre_maquette_hors_service();
74 }
```

Listing 2.2 – Fichier source minimal

Utilisation 3

3.1 Généralité

Pour programmer le comportement des locomotives dans QTrainSim, il faut utiliserer le langage C. Il est neamoins possible de créer des objets en C++ (le simulateur lui-même est en C++), et même d'utiliser la librairie Qt. Toutefois, le projet est structuré de telle manière que l'utilisation la plus simple consiste en la complétion en C de la méthode run() utilisée par le thread "User" dans main.cpp. En aucun cas il n'est nécessaire, ni même souhaitable de modifier les autres fichiers du projet. En cas de modification, il faut savoir que le programme pourrait ne pas fonctionner sur la maquette physique. Toutes les fonctions nécessaires se trouvent dans l'API représetée par le fichier `ctrain_handler.h`. Ces fonctions peuvent être appelées directement et sans préfixe depuis `cmain.c`.

Le simulateur se lance directement à partir de QtCreator. Afin d'indiquer au simulateur la maquette et les locomotives à utiliser, les fonction *selection_maquette* et *assigner_loco* doivent être appelées à l'intérieur de la fonction *cmain* du fichier `cmain.c`.

3.2 Interface graphique

La figure 3.1 représente l'interface graphique dans son integralité.

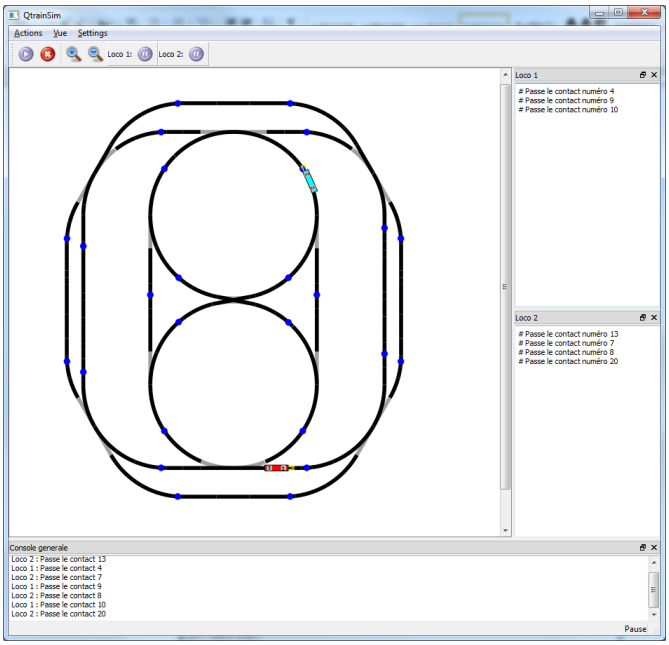

FIGURE 3.1 – Interface graphique du simulateur

3.2.1 Barre de menu

3.2.1.1 Actions

Le menu **Actions** de la barre de menu permet d'interagir directement sur la vue contenant la maquette virtuelle du simulateur. Voici les différentes actions définies :
- "Resume"/"Pause" permet de démarrer ou mettre en pause le système en fonction de l'état de celui-ci.
- "Emergency stop" fait appelle à la fonction *emergency_stop()* située à l'intérieur du fichier `cmain.c`. C'est la responsabilité de l'utilisateur de placer le code spécifique à cet appel.
- "Exit" permet de quitter l'application.
- "Pause loco n"/"Restart loco n" permettent de mettre en pause ou de redémarrer une locomotive spécifique représentée par le numéro "n" en fonction de son état actuel.

3.2.1.2 Views

Le menu **Views** de la barre de menu permet d'intéragir directement sur la vue contenant la maquette virtuelle du simulateur et les vues relatives aux logs des locomotives. Voici les différentes actions définies pour ces vues :
- **Views→Zoom in** permet de réaliser un zoom progressif au sein de la vue contenant la maquette virtuelle du simulateur.
- **Views→Zoom out** permet de réaliser un zoom regressif au sein de la vue contenant la maquette virtuelle du simulateur.
- **Views→Zoom fit** permet de dimensionner la maquette virtuelle du simulateur au sein de sa vue en fonction de la taille de l'application.
- **Views→Rotate +** permet de réaliser une légère rotation de la maquette virtuelle du simulateur sur la droite.
- **Views→Rotate -** permet de réaliser une légère rotation de la maquette virtuelle du simulateur sur la gauche.
- **Views→View Loco logs** permet d'afficher ou non les logs des locomotives dans leur vue respective.
- **Views→View contact number** permet d'afficher à proximité de chaque contact son numéro.
- **Views→View aiguillage number** permet d'afficher à proximité de chaque aiguillage son numéro.

3.2.1.3 Settings

Le menu **Settings** de la barre de menu permet d'interagir sur différents paramètres de l'application. Actuellement, il est existe uniquement l'action **Settings→Inertia** qui permet d'ajouter ou non de l'inertie lors de l'arrêt des locomotives. En pratique, activer l'inertie permet une simulation plus fidèle aux conditions réelles. Les locos changeront alors leurs vitesses de manière progressives. Cela est notamment important dans la gestion des distances de freinage.

3.2.2 Barre d'outils

La barre d'outils offre différents raccourcis pour les actions "Resume"/"Pause", "Emergency stop" , "Zoom in" , "Zoom out" et "Pause loco n"/"Restart loco n".

3.2.3 Maquette virtuelle

La maquette est schématisée par une vue en deux dimension. Les voies variables (aiguillages, aiguillages enroulés, aiguillages triples, traversées-jonctions) sont représentées avec une voie en noir

(qui indique l'orientation actuelle de la voie) et une ou plusieurs voies grisées. La où les locomotives sont représentées par un rectangle de couleur portant le numéro de la locomotive aux deux extrémités. Deux triangles jaunes représentent les phares, et indiquent la direction de la locomotive. Les couleurs des locomotives sont générées dynamiquement de manière à ce qu'elles soient le plus distinguables possible. Les contacts sont indiqués par des disques bleus. Il est possible de modifier l'orientation d'une voie variable en cliquant directement sur celle-ci.

3.2.4 Console générale

La console générale offre un journal des informations de toutes les locomotives dans l'ordre chronologique.

3.2.5 Console locomotive

Chaque locomotive est également dotée de sa propre console sur la droite de la fenêtre. Cette console offre un journal des informations de la locomotive concernée dans l'ordre chronologique.

4 Plan des maquettes

La figure 4.1 indique la composition des différentes maquettes utilisables sur le simulateur.

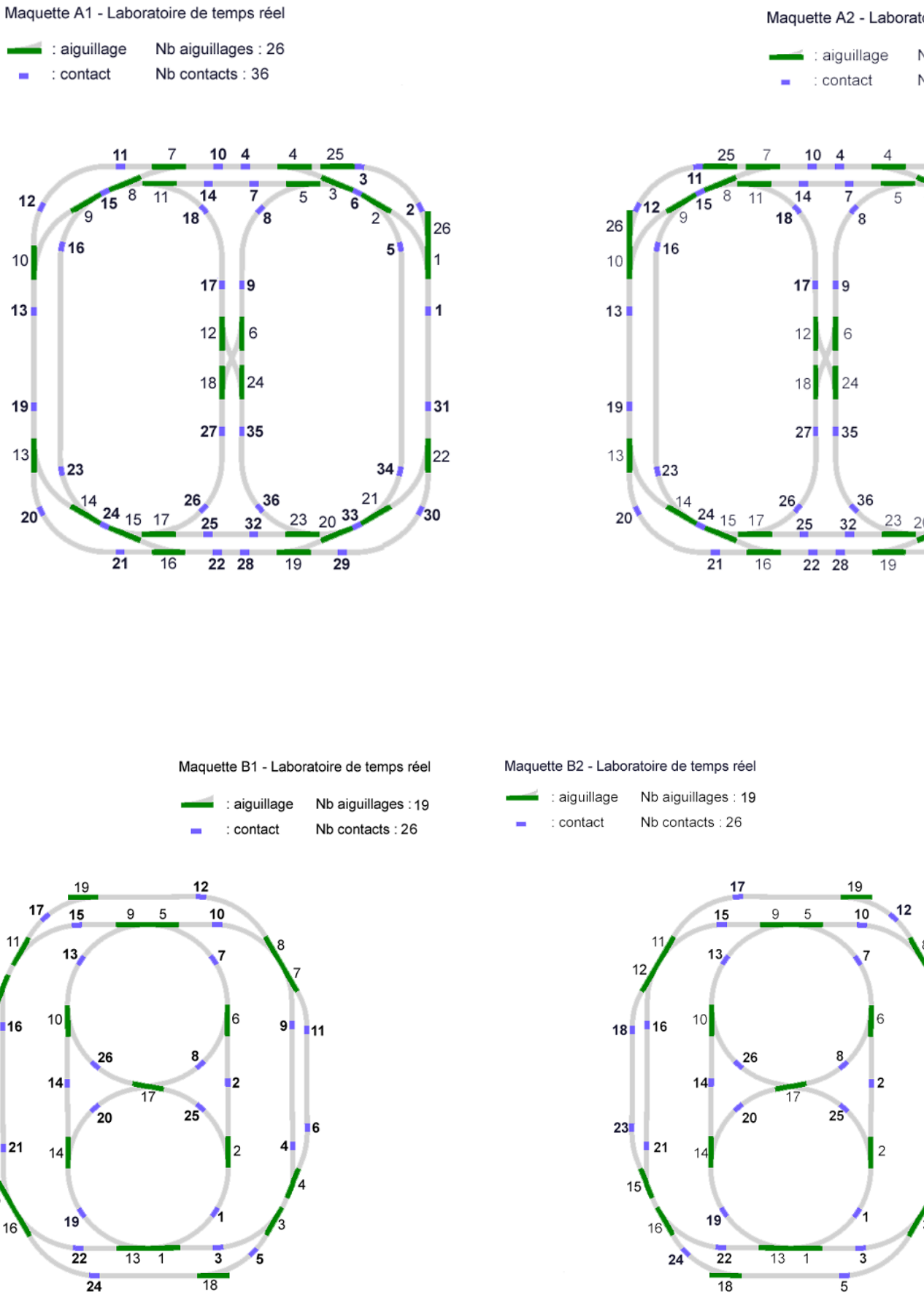

FIGURE 4.1 – Plan des maquettes