

TTPA (Écran)

1.1

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :50 :18

Table des matières

1	Page principale du projet TTPA (Table Tennis Performance Analyser)	1
1.1	Introduction	1
1.2	Table des matières	1
2	Changelog	1
3	Configuration	5
4	Manuel d'installation	6
5	Recette IR	6
6	A propos	6
7	Licence GPL	7
8	Documentation des classes	7
8.1	Référence de la classe Clhm	7
8.1.1	Description détaillée	11
8.1.2	Documentation des constructeurs et destructeur	11
8.1.3	Documentation des fonctions membres	12
8.1.4	Documentation des données membres	27
8.2	Référence de la classe CommunicationBluetooth	29
8.2.1	Documentation des énumérations membres	30
8.2.2	Documentation des constructeurs et destructeur	31
8.2.3	Documentation des fonctions membres	32
8.2.4	Documentation des données membres	38
8.3	Référence de la classe CTable	39
8.3.1	Documentation des constructeurs et destructeur	41
8.3.2	Documentation des fonctions membres	42
8.3.3	Documentation des données membres	50
8.4	Référence de la classe CTrame	52
8.4.1	Documentation des constructeurs et destructeur	53
8.4.2	Documentation des fonctions membres	53
9	Documentation des fichiers	59

9.1	Référence du fichier Changelog.dox	59
9.2	Référence du fichier communicationbluetooth.cpp	59
9.2.1	Description détaillée	59
9.3	Référence du fichier communicationbluetooth.h	59
9.3.1	Description détaillée	60
9.3.2	Documentation des macros	60
9.4	Référence du fichier const.h	60
9.4.1	Documentation des macros	61
9.4.2	Documentation du type de l'énumération	64
9.5	Référence du fichier ihm.cpp	65
9.6	Référence du fichier ihm.h	66
9.6.1	Description détaillée	66
9.6.2	Documentation des macros	66
9.7	Référence du fichier main.cpp	66
9.7.1	Documentation des fonctions	66
9.8	Référence du fichier README.dox	67
9.9	Référence du fichier table.cpp	67
9.10	Référence du fichier table.h	67
9.10.1	Description détaillée	67
9.11	Référence du fichier trame.cpp	67
9.12	Référence du fichier trame.h	67
9.12.1	Description détaillée	68

1 Page principale du projet TTPA (Table Tennis Performance - Analyser)

1.1 Introduction

Le système doit permettre une analyse des performances du joueur (côté relanceur). Il doit proposer une phase d'entraînement adaptée au niveau du joueur, puis de détecter l'impact des balles afin d'afficher le rythme de jeu, la précision, le pourcentage de réussite. La zone d'impact (côté distributeur) est identifiée sur un écran de télévision en fin d'exercice. Le pourcentage de balles dans chacune des zones, le rythme de jeu et le pourcentage de réussite sont disponibles en fin d'exercice. Le joueur lance un exercice spécifique et pourra connaître son évolution individuelle.

Module : Écran (Racamond Adrien)

1.2 Table des matières

- [Configuration](#)
- [Manuel d'installation](#)
- [Changelog](#)
- [Recette IR](#)
- [A propos](#)
- [Licence GPL](#)

Dépôt SVN : <https://svn.riouxsvn.com/ttpa>

2 Changelog

r115 | aracamond | 2018-06-06 11 :03 :25 +0200 (mer., 06 juin 2018) | 1 line

Correction du correctif précédent lié a la taille du texte

r111 | aracamond | 2018-06-05 10 :19 :36 +0200 (mar., 05 juin 2018) | 1 line

Correction du resize, la table causait des problemes dans le layer RECAP, correction de la font sur celle-ci

r110 | aracamond | 2018-06-04 15 :29 :03 +0200 (lun., 04 juin 2018) | 1 line

Ajout de l'affichage du nom du peripherique bluetooth connecte en dessous du nom du joueur, remplacement du nom du joueur sur l'affichage LOGO par le nom du periphe-
rique

r96 | aracamond | 2018-05-25 14 :09 :24 +0200 (ven., 25 mai 2018) | 1 line

ajout de documentation pour Doxygen, deplacement du TARGET du .pro vers /bin

r94 | aracamond | 2018-05-25 12 :22 :35 +0200 (ven., 25 mai 2018) | 1 line

Suppression de fonctions redondantes, déplacement des variables string CSS en tant que Define, correction de bugs mineurs lié aux fonctions redondantes

r88 | aracamond | 2018-05-24 16 :58 :24 +0200 (jeu., 24 mai 2018) | 1 line

correction du mode DEMO pour utiliser le nouveau systeme de HORS-TABLE

r87 | aracamond | 2018-05-24 16 :15 :06 +0200 (jeu., 24 mai 2018) | 1 line

Implementation du compteur d'enchainement maximum par seance

r86 | aracamond | 2018-05-24 16 :06 :23 +0200 (jeu., 24 mai 2018) | 1 line

Modification de l'aspect de la table pour etre plus visible

r85 | aracamond | 2018-05-24 15 :39 :39 +0200 (jeu., 24 mai 2018) | 1 line

Refonte du systeme 'Hors-Table', recoloration de la table pour une couleur plus vraie que nature

r84 | aracamond | 2018-05-23 17 :26 :46 +0200 (mer., 23 mai 2018) | 1 line

Correction de la zone robot ou dans certains cas ROBOT serait remplacé par le nombre de balles dans cette zone

r83 | aracamond | 2018-05-23 17 :15 :42 +0200 (mer., 23 mai 2018) | 1 line

Ajout de la table dans le tableau recapitulatif (deplacement du widget entier), correction pour void [Clhm](#) : [:commencerSeance\(\)](#) les parametres ne sont plus reset

r82 | aracamond | 2018-05-22 16 :00 :10 +0200 (mar., 22 mai 2018) | 1 line

Découpage de bool [CTrame](#) : [:gererTrame\(QString\)](#) en sous parties

r75 | aracamond | 2018-04-18 17 :38 :57 +0200 (mer., 18 avril 2018) | 1 line

changement de taille du logo PAUSE

r74 | aracamond | 2018-04-18 15 :47 :56 +0200 (mer., 18 avril 2018) | 1 line

update de [const.h](#)

r73 | aracamond | 2018-04-18 15 :34 :55 +0200 (mer., 18 avril 2018) | 1 line

ajout de la possibilité de faire une pause et de reprendre, correction d'un bug lié a la lecture des trames à 1 argument

r68 | tvaira | 2018-04-16 14 :51 :35 +0200 (lun., 16 avril 2018) | 1 line

Verification des TODO

r65 | aracamond | 2018-04-12 22 :22 :55 +0200 (jeu., 12 avril 2018) | 1 line

ajout de commentaires pour doxygen

r64 | aracamond | 2018-04-12 21 :39 :51 +0200 (jeu., 12 avril 2018) | 1 line
améliorations au mode -demo, ajout de -norobot pour desactiver le robot dans ce mode

r63 | aracamond | 2018-04-12 21 :02 :50 +0200 (jeu., 12 avril 2018) | 1 line
Remplacement de int en uint8_t pour les zones du fait qu'il est inutile d'avoir plus

r62 | aracamond | 2018-04-12 19 :15 :16 +0200 (jeu., 12 avril 2018) | 1 line
ajout de la fenetre de recapitulatif de la séance ainsi que de la connexion du slot a celle ci

r54 | aracamond | 2018-04-09 16 :00 :09 +0200 (lun., 09 avril 2018) | 1 line
Ajout de la gestion du découpage de trame en cas de trames concaténés

r52 | aracamond | 2018-04-05 17 :56 :55 +0200 (jeu., 05 avril 2018) | 1 line
changement de trames, correction de bugs relatif au trames, restructuration de l'interface TABLE, ajout du timer de seance

r49 | aracamond | 2018-04-04 16 :25 :36 +0200 (mer., 04 avril 2018) | 1 line
Adaptation au nouveau protocole d'impact, correction de bugs lié au nombre de balles

r48 | tvaira | 2018-03-30 09 :38 :09 +0200 (ven., 30 mars 2018) | 1 line
Ajout de la gestion des signaux dans le thread de communication Bluetooth

r44 | aracamond | 2018-03-28 16 :24 :00 +0200 (mer., 28 mars 2018) | 1 line
Correction fermeture/ouverture ports, slot deconnexion

r41 | tvaira | 2018-03-25 14 :23 :38 +0200 (dim., 25 mars 2018) | 1 line
Ajout du Thread pour la communication Bluetooth

r39 | tvaira | 2018-03-24 10 :51 :17 +0100 (sam., 24 mars 2018) | 1 line
Ajout parametrage Doxygen

r38 | aracamond | 2018-03-22 15 :58 :13 +0100 (jeu. 22 mars 2018) | 1 ligne
Refonte de l'interface en termes de couleurs, correction de problemes liées à l'affichage, finalisation des trames

r37 | aracamond | 2018-03-21 17 :59 :36 +0100 (mer. 21 mars 2018) | 1 ligne

Finission des trames, TODO : Fixer le X de connection, thread pour la verification de la connexion

r34 | aracamond | 2018-03-19 18 :06 :23 +0100 (lun. 19 mars 2018) | 1 ligne

Correction du problem lié a la deconnection du port

r32 | aracamond | 2018-03-19 15 :18 :31 +0100 (lun. 19 mars 2018) | 1 ligne

MAJ Documentation

r25 | aracamond | 2018-03-19 11 :36 :20 +0100 (lun. 19 mars 2018) | 1 ligne

Deplacement des fonctions de gestion de trames dans [CTrame](#)

r19 | aracamond | 2018-03-15 16 :57 :29 +0100 (jeu. 15 mars 2018) | 1 ligne

Commancement de la documentation Doxygen, apparition d'un bug causé par l'ouverture/fermeture du bluetooth

r14 | aracamond | 2018-02-22 18 :05 :10 +0100 (jeu. 22 févr. 2018) | 1 ligne

ajout du QextSerialPort

r13 | aracamond | 2018-02-22 17 :54 :43 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout de la zone objectif, refonte de l'interface de la table pour la zone objectif + couleurs, ajout de la capture du port serie (fonctionnel mais non terminé).

r12 | aracamond | 2018-02-15 17 :37 :29 +0100 (jeu. 15 févr. 2018) | 1 ligne

Changement des statistiques, reussite globale a faire

r11 | aracamond | 2018-02-15 15 :51 :05 +0100 (jeu. 15 févr. 2018) | 1 ligne

ajout du batch pour lancement depuis un SSH

r10 | aracamond | 2018-02-15 15 :23 :48 +0100 (jeu. 15 févr. 2018) | 1 ligne

correction des images

r9 | aracamond | 2018-02-15 15 :10 :18 +0100 (jeu. 15 févr. 2018) | 1 ligne

restructuration de l'affichage, debut de l'IHM a gauche, changement du logo

r8 | aracamond | 2018-02-14 17 :57 :18 +0100 (mer. 14 févr. 2018) | 1 ligne
ajout des pourcentages, refonte de la table, ajout du mode -dev

r7 | aracamond | 2018-02-14 10 :50 :06 +0100 (mer. 14 févr. 2018) | 1 ligne
ajout du fichier filet.jpg, fonction rafraichirCSS pour IHM

r6 | tvaira | 2018-02-08 22 :45 :03 +0100 (jeu. 08 févr. 2018) | 1 ligne
Ajout d'un QStackedWidget comme centralWidget

r5 | aracamond | 2018-02-08 17 :54 :55 +0100 (jeu. 08 févr. 2018) | 1 ligne
Ajout de fonctionalitees a Ecran-TTPA

r4 | aracamond | 2018-02-07 18 :37 :07 +0100 (mer. 07 févr. 2018) | 1 ligne
ajout de la base du programme Ecran-TTPA

3 Configuration

Poste de développement :

- Distribution : Ubuntu 12.04.5 LTS
- OS : GNU/Linux
- Noyau : Linux
- Version : 3.8.0-44-generic
- Machine : x86_64
- Processeur : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- Mémoire RAM : 8129984 kB

Liste des paquets Qt nécessaires :

- libqtgui4 libqtcore4 libqt4-svg

Raspberry Pi : voir dossier

Bluetooth : voir dossier

4 Manuel d'installation

Fabrication de l'exécutable :

- `qmake`
- `make`

5 Recette IR

Étudiant 3 : Racamond Adrien

- Le système d'exploitation est installé et fonctionnel
- L'écran est configuré en mode "kiosque"
- La zone d'impact est identifiée et affichée en temps réel
- Les données de la séance sont affichées en temps réel
- Les liaisons sans fil sont opérationnelles
- Les informations sont affichées en fin de séquence

6 A propos

Auteur

Racamond Adrien <adrien.racamond.lasalle@gmail.com>

Version

1.1

Date

2018

7 Licence GPL

This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

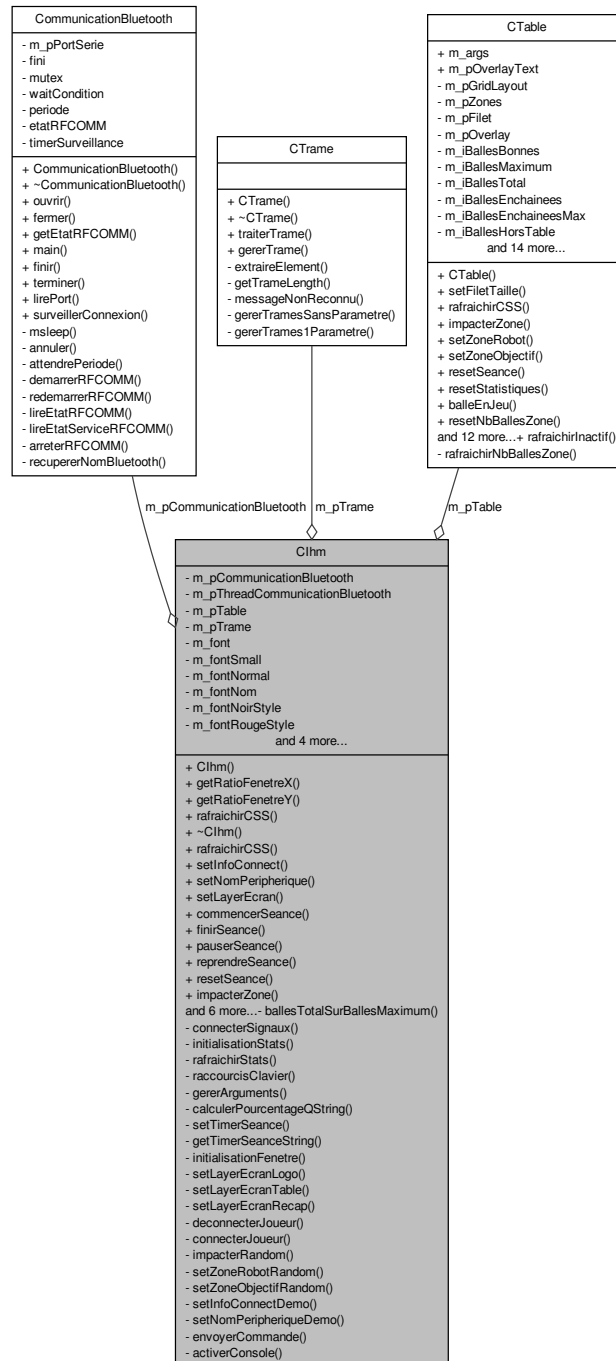
8 Documentation des classes

8.1 Référence de la classe Clhm

Classe principale de l'application (IHM)

```
#include <ihm.h>
```

Graphe de collaboration de Clhm :



Connecteurs publics

- void [rafraichirCSS](#) ()
Rafraichit le CSS lié à l'affichage (fontes, couleurs) utilisant [getRatioFenetreY\(\)](#)
- void [setInfoConnect](#) (QString nom)
Affiche le nom du joueur sur la table.
- void [setNomPeripherique](#) (QString nom)
Affiche le nom du périphérique connecté
- void [setLayerEcran](#) (uint8_t layer)
Change d'écran.
- void [commencerSeance](#) ()
Commencer la seance.
- void [finirSeance](#) ()
Finir la seance et afficher la fenêtre Recap.
- void [pauserSeance](#) ()
- void [reprendreSeance](#) ()
Reprendre la seance suite a une pause.
- void [resetSeance](#) ()
Reset des statistiques et de la configuration.
- void [impacterZone](#) (uint8_t zone)
Calcul et affiche l'impact sur l'IHM et la table.
- void [balleEnJeu](#) ()
La balle a été capté sur le capteur coté joueur, ajout de la balle.
- void [setZoneRobot](#) (uint8_t zone)
Place le robot sur la table.
- void [setZoneObjectif](#) (uint8_t zone)
Place la zone objectif sur la table.
- void [setBallesMaximum](#) (int balles)
Definit le nombre de balles maximum pour la seance.
- void [rafraichirHeure](#) ()
Rafraichit l'heure sur l'IHM (Logo et Table)
- void [rafraichirTimerSeance](#) ()
Rafraichit le timer de la seance.
- void [quitter](#) ()
Quitte l'application (utilisé par le raccourcit CTRL+Q)

Fonctions membres publiques

- [Clhm](#) (QWidget *parent=0)
- float [getRatioFenetreX](#) ()
Récupere le ratio du width par rapport à la résolution par default 960x540.
- float [getRatioFenetreY](#) ()
Récupere le ratio du height par rapport à la résolution par default 960x540.
- void [rafraichirCSS](#) (float ratio)
- [~Clhm](#) ()

Connecteurs privés

- void [initialisationFenetre](#) ()
Initialise la taille des fenetres en fonction de la résolution de l'écran en mode plein-ecran.
- void [setLayerEcranLogo](#) ()
Bascule sur le menu d'attente.
- void [setLayerEcranTable](#) ()
Bascule sur l'interface de la table.
- void [setLayerEcranRecap](#) ()
Bascule sur la récapitulation de la séance.
- void [deconnecterJoueur](#) ()

- Actions nécessaires a la deconnexion du joueur.
- void `connecterJoueur` ()
- Actions nécessaires a la connexion du joueur.
- void `impacterRandom` ()
- [DEBUG] Envoi une balle aléatoire sur la table coté robot
- void `setZoneRobotRandom` ()
- [DEBUG] Définir une position du robot aléatoire (executé depuis un bouton sur l'IHM)
- void `setZoneObjectifRandom` ()
- [DEBUG] Définir un objectif aléatoire (executé depuis un bouton sur l'IHM)
- void `setInfoConnectDemo` ()
- [DEBUG] Execute `setInfoConnect()` avec le nom de demonstration
- void `setNomPeripheriqueDemo` ()
- [DEBUG] Execute `setNomPeripherique()` avec le nom de demonstration
- void `envoyerCommande` ()
- [DEBUG] Envoi une commande de la console, accepte aussi les trames
- void `activerConsole` ()
- [DEBUG] Reset le CSS de la console (nécessaire a cause du delai)

Fonctions membres privées

- QString `ballesTotalSurBallesMaximum` ()
- String affichant le l'état des balles renvoyées sur le nombre de balles paramétrée pour la seance.
- void `connecterSignaux` ()
- Réalise la connexion des slots/signaux.
- void `initialisationStats` ()
- Initialisation des statistiques de la seance.
- void `rafraichirStats` ()
- Rafraichissement des statistiques de la seance exécuté a chaque impact/changement de parametre.
- void `raccourcisClavier` ()
- Implémentation des raccourcis clavier.
- void `gererArguments` ()
- Verification des options de lancement.
- QString `calculerPourcentageQString` (int x, int y)
- Récupere sous forme de QString un pourcentage : "(X%)" utilisé pour les statistiques.
- void `setTimerSeance` (unsigned int iTemps=0)
- Ecriture du temps dans `m_pQLabelTimerSeance`.
- QString `getTimerSeanceString` (unsigned int iTemps)

Attributs privés

- `CommunicationBluetooth` * `m_pCommunicationBluetooth`
- Gestion de la communication Bluetooth.
- QThread * `m_pThreadCommunicationBluetooth`
- Thread pour la classe `CommunicationBluetooth`.
- `CTable` * `m_pTable`
- Association vers la classe `CTable`.
- `CFrame` * `m_pFrame`
- QFont `m_font`
- QFont `m_fontSmall`
- QFont `m_fontNormal`
- QFont `m_fontNom`
- QString `m_fontNoirStyle`
- QString `m_fontRougeStyle`
- QString `m_fontVertStyle`
- QString `m_fontTitreStyle`
- unsigned int `m_iTempsSeance`

- *Temps en seconde de la seance.*
– QTimer * [m_pTimerHeure](#)
Timer rafraichissant l'Heure.
- QTimer * [m_pTimerSeance](#)
Timer incrémentant le temps de la seance.

8.1.1 Description détaillée

Auteur

Racamond Adrien

Version

0.9

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 Clhm (QWidget * parent = 0) [explicit]

Références [connecterSignaux\(\)](#), [DELAI_FIXFENETRE](#), [CommunicationBluetooth](#) : [Evenement](#), [gererArguments\(\)](#), [initialisationFenetre\(\)](#), [LAYER_LOGO](#), [m_fontNoirStyle](#), [m_fontRougeStyle](#), [m_fontTitreStyle](#), [m_fontVertStyle](#), [m_iTempsSeance](#), [m_pCommunicationBluetooth](#), [m_pTable](#), [m_pThreadCommunicationBluetooth](#), [m_pTimerHeure](#), [m_pTimerSeance](#), [m_pTrame](#), [PORT_BLUETOOTH](#), [raccourcisClavier\(\)](#), et [setTimerSeance\(\)](#).

```

                                : QWidget (parent)
{
    setupUi(this);
    raccourcisClavier();

#ifdef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << "PID :" << (int)qApp->applicationPid() << "TID
        :" << QApplication::instance()->thread()->currentThreadId() << qApp->thread();
#endif

    // Gestion de la communication Bluetooth
    m_pCommunicationBluetooth = new CommunicationBluetooth(PORT_BLUETOOTH,
        CommunicationBluetooth::Evenement);
    // ou :
    // m_pCommunicationBluetooth = new CommunicationBluetooth(PORT_BLUETOOTH,
        // CommunicationBluetooth::Scrutation);
    m_pThreadCommunicationBluetooth = new QThread;
    m_pCommunicationBluetooth->moveToThread(m_pThreadCommunicationBluetooth);

    m_pTrame = new CTrame(this);

    m_fontNoirStyle = QString::fromUtf8("QLabel\n{\n        color: #000000;\n}");
    m_fontRougeStyle = QString::fromUtf8("QLabel\n{\n        color: #FF0000;\n}");
    m_fontVertStyle = QString::fromUtf8("QLabel\n{\n        color: #006500;\n}");
    m_fontTitreStyle = QString::fromUtf8("QLabel\n{\n        color: #000000;\n
        background: qlineargradient(spread:reflect, x1:0.5, y1:0, x2:1, y2:0, stop:0
        rgba(255, 255, 255, 255), stop:1 rgba(0, 0, 0, 0));\n}");

    m_pTable = new CTable(this);
    QTimer::singleShot(DELAI_FIXFENETRE, this, SLOT(initialisationFenetre()));
    QTimer::singleShot(DELAI_FIXFENETRE*4, this, SLOT(initialisationFenetre()));
    ;
    m_pHLayoutTable->addWidget(m_pTable);

```

```

    m_pTimerHeure = new QTimer(this);
    m_pTimerHeure->setInterval(3000);

    m_pTimerSeance = new QTimer(this);
    m_pTimerSeance->setInterval(1000);
    m_iTempsSeance = 0;
    setTimerSeance(0);

    gererArguments();

    m_pFenetres->setCurrentIndex(LAYER_LOGO);

    connecterSignaux();

    m_pTimerHeure->start();

    // démarre la communication Bluetooth
    m_pThreadCommunicationBluetooth->start();
}

```

8.1.2.2 Clhm : ~Clhm ()

Références [CommunicationBluetooth : finir\(\)](#), [m_pCommunicationBluetooth](#), et [m_pThreadCommunicationBluetooth](#).

```

{
    // Ferme la communication Bluetooth
    m_pCommunicationBluetooth->finir();
    m_pThreadCommunicationBluetooth->quit();
    m_pThreadCommunicationBluetooth->wait();
    delete m_pCommunicationBluetooth;
    delete m_pThreadCommunicationBluetooth;
#ifdef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << "fin";
#endif
}

```

8.1.3 Documentation des fonctions membres

8.1.3.1 void Clhm : activerConsole () [private, slot]

Référencé par [envoyerCommande\(\)](#).

```

{
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\\n{\\n\\n}");
}

```

8.1.3.2 Clhm : :balleEnJeu () [slot]

Références [CTable : :balleEnJeu\(\)](#), [ballesTotalSurBallesMaximum\(\)](#), [m_fontNormal](#), [m_pTable](#), et [rafraichirStats\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    m_pTable->balleEnJeu();
    m_pQLabelTopMid->setFont(m_fontNormal);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
    rafraichirStats();
}

```

8.1.3.3 QString Clhm : :ballesTotalSurBallesMaximum () [private]

Références [CTable : :getBallesMaximum\(\)](#), [CTable : :getBallesTotal\(\)](#), et [m_pTable](#).

Référencé par [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [impacterZone\(\)](#), [rafraichirCSS\(\)](#), et [setBallesMaximum\(\)](#).

```
{
    QString ballesMax = QString::number(m_pTable->getBallesMaximum());

    if (!m_pTable->getBallesMaximum())
        ballesMax = QString::fromUtf8("");

    return /*QString::fromUtf8(IHM_BALLESENVOYEES) */ QString::number(m_pTable
        ->getBallesTotal()) + " / " + ballesMax;
}
```

8.1.3.4 QString Clhm : :calculerPourcentageQString (int x, int y) [private]

Référencé par [finirSeance\(\)](#), et [rafraichirStats\(\)](#).

```
{
    if (!y)
        return " (0%) ";

    return "(" + QString::number((double(x) / double(y))*100, 'f', 0) + "%) ";
}
```

8.1.3.5 Clhm : :commencerSeance () [slot]

Références [ballesTotalSurBallesMaximum\(\)](#), [CSS_TIMER_ON](#), [LAYER_TABLE](#), [m_iTempsSeance](#), [m_pTable](#), [m_pTimerSeance](#), [rafraichirStats\(\)](#), [CTable : :resetStatistiques\(\)](#), [setLayerEcran\(\)](#), et [setTimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;

    // if (m_pFenetres->currentIndex() != LAYER_TABLE)
    //     return;

    setLayerEcran(LAYER_TABLE);

    m_iTempsSeance = 0;
    m_pTimerSeance->start(1000);

    m_pQLabelTimerSeance->setStyleSheet(CSS_TIMER_ON);

    m_pHBoxLayoutTable->addWidget(m_pTable);
    m_pTable->resetStatistiques();
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());

    setTimerSeance(0);
    rafraichirStats();
}
```

8.1.3.6 void Clhm : :connecterJoueur () [private, slot]

Références [LAYER_LOGO](#), [LOGO_ATTENTECONFIGURATION](#), et [setLayerEcran\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).


```
{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_LOGO);
    // m_pQLabelLogoTexte->setText(LOGO_ATTENTEIDENTIFICATION); Ancien Message
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONFIGURATION);
}
```

8.1.3.7 void Clhm : :connecterSignaux () [private]

Références [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [connecterJoueur\(\)](#), [deconnecterJoueur\(\)](#), [finirSeance\(\)](#), [impacterZone\(\)](#), [m_pCommunicationBluetooth](#), [m_pThreadCommunicationBluetooth](#), [m_pTimerHeure](#), [m_pTimerSeance](#), [m_pTrame](#), [main\(\)](#), [pauserSeance\(\)](#), [quitter\(\)](#), [rafraichirCSS\(\)](#), [rafraichirHeure\(\)](#), [rafraichirTimerSeance\(\)](#), [reprendreSeance\(\)](#), [resetSeance\(\)](#), [setBallesMaximum\(\)](#), [setInfoConnect\(\)](#), [setLayerEcran\(\)](#), [setNomPeripherique\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

Référencé par [Clhm\(\)](#).

```
{
    // Thread Bluetooth
    connect(m_pThreadCommunicationBluetooth, SIGNAL(started()),
        m_pCommunicationBluetooth, SLOT(main()));
    connect(m_pThreadCommunicationBluetooth, SIGNAL(finished()),
        m_pCommunicationBluetooth, SLOT(terminer()));

    // Timers
    connect(m_pTimerHeure, SIGNAL(timeout()),this, SLOT(rafraichirHeure()));
    connect(m_pTimerSeance, SIGNAL(timeout()),this, SLOT(rafraichirTimerSeance(
    )));

    // Gestion des trames
    connect(m_pCommunicationBluetooth, SIGNAL(nouvellesDonneesRecues(QString)),
        m_pTrame, SLOT(traiterTrame(QString)));
    connect(m_pTrame, SIGNAL(setLayerEcran(uint8_t)), this, SLOT(setLayerEcran(
    uint8_t)));
    connect(m_pTrame, SIGNAL(setInfoConnect(QString)), this, SLOT(setInfoConnect(
    QString)));
    connect(m_pTrame, SIGNAL(resetSeance()), this, SLOT(resetSeance()));
    connect(m_pTrame, SIGNAL(commencerSeance()), this, SLOT(commencerSeance()));
    ;
    connect(m_pTrame, SIGNAL(pauserSeance()), this, SLOT(pauserSeance()));
    connect(m_pTrame, SIGNAL(reprendreSeance()), this, SLOT(reprendreSeance()));
    ;
    connect(m_pTrame, SIGNAL(finirSeance()), this, SLOT(finirSeance()));
    connect(m_pTrame, SIGNAL(impacterZone(uint8_t)), this, SLOT(impacterZone(
    uint8_t)));
    connect(m_pTrame, SIGNAL(balleEnJeu()), this, SLOT(balleEnJeu()));
    connect(m_pTrame, SIGNAL(setZoneRobot(uint8_t)), this, SLOT(setZoneRobot(
    uint8_t)));
    connect(m_pTrame, SIGNAL(setZoneObjectif(uint8_t)), this, SLOT(
    setZoneObjectif(uint8_t)));
    connect(m_pTrame, SIGNAL(setBallesMaximum(int)), this, SLOT(setBallesMaximum(
    int)));
    connect(m_pTrame, SIGNAL(rafraichirCSS()), this, SLOT(rafraichirCSS()));
    connect(m_pTrame, SIGNAL(quitter()), this, SLOT(quitter()));

    connect(m_pCommunicationBluetooth, SIGNAL(deconnecterJoueur()), this, SLOT(
    deconnecterJoueur()));
    connect(m_pCommunicationBluetooth, SIGNAL(connecterJoueur()), this, SLOT(
    connecterJoueur()));

    connect(m_pCommunicationBluetooth, SIGNAL(setNomPeripherique(QString)),
        this, SLOT(setNomPeripherique(QString)));
}
```

8.1.3.8 void Clhm : :deconnecterJoueur () [private, slot]

Références [LAYER_LOGO](#), [LOGO_ATTENTECONNEXION](#), [resetSeance\(\)](#), et [setLayerEcran\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_LOGO);
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONNEXION);
    m_pQLabelLogoNom->setText("");
    m_pQLabelLogoNomMessage->setText("");
    resetSeance();
}
```

8.1.3.9 void Clhm : :envoyerCommande () [private, slot]

Références [activerConsole\(\)](#), [m_pTrame](#), [quitter\(\)](#), et [CTrame : :traiterTrame\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    QString texte = m_pConsole->text();

    QTimer::singleShot(300, this, SLOT(activerConsole()));
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n"
                             "{\n"
                             "    background-color: #00FF00;\n"
                             "}");

    if(texte.startsWith("$TPA:"))
    {
        bool retour = m_pTrame->traiterTrame(texte);

        if (!retour)
            m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n{\n"
                                     "background-color: #FF0000;\n}");

        return;
    }
    if(texte.startsWith("hide "))
    {
        QStringList args = texte.split(" ");
        if(args.length() == 2)
        {
            m_pConsole->hide();
            QTimer::singleShot(1000 * args.at(1).toInt(), m_pConsole, SLOT(show
            ()));
        }
        return;
    }
    if (texte.toLowerCase() == "quit")
    {
        quitter();
        return;
    }
    m_pConsole->setStyleSheet("QLineEdit#m_pConsole\n{\nbackground-color:
    #FF0000;\n}");
}
```

8.1.3.10 Clhm : :finirSeance () [slot]

Références [calculerPourcentageQString\(\)](#), [CTable : :finirSeance\(\)](#), [CTable : :getBallesBonnes\(\)](#), [CTable : :getBallesEnchainees\(\)](#), [CTable : :getBallesObjectif\(\)](#), [CTable : :get-](#)

BallesTotal(), CTable : :getZoneObjectif(), LAYER_RECAP, m_pTable, m_pTimerSeance, rafraichirStats(), RECAP_STAT1, RECAP_STAT1_ALT, RECAP_STAT2, RECAP_STAT3, setLayerEcran(), et ZONE_AUCUNE.

Référencé par connecterSignaux(), et gererArguments().

```
{
    qDebug() << Q_FUNC_INFO;
    setLayerEcran(LAYER_RECAP);
    QString decalage = " ";
    m_pTimerSeance->stop();
    m_pTable->finirSeance();

    //=====
    // STAT 1
    if(m_pTable->getZoneObjectif() == ZONE_AUCUNE)
    {
        m_pQLabelLeftStat1TexteRecap->setText(decalage+ RECAP_STAT1_ALT);
        m_pQLabelLeftStat1NbRecap->setText(QString::number(m_pTable->
getBallesBonnes()) + " / " + QString::number(m_pTable->getBallesTotal()) );
        m_pQLabelLeftStat1PerRecap->setText(calculerPourcentageQString(m_pTable
->getBallesBonnes(),m_pTable->getBallesTotal()) );
    }
    else
    {
        m_pQLabelLeftStat1TexteRecap->setText(decalage+ RECAP_STAT1);
        m_pQLabelLeftStat1NbRecap->setText(QString::number(m_pTable->
getBallesObjectif()) + " / " + QString::number(m_pTable->getBallesTotal()) );
        m_pQLabelLeftStat1PerRecap->setText(calculerPourcentageQString(m_pTable
->getBallesObjectif(),m_pTable->getBallesTotal()) );
    }
    //=====
    // STAT 2

    m_pQLabelLeftStat2TexteRecap->setText(decalage+ RECAP_STAT2);
    rafraichirStats();

    //=====
    // STAT 3

    m_pQLabelLeftStat3TexteRecap->setText(decalage+ RECAP_STAT3);
    m_pQLabelLeftStat3NbRecap->setText(QString::number(m_pTable->
getBallesEnchainees()));

    //=====
    // STAT 4

    //      TODO

    m_pHBoxLayoutTableRecap->addWidget(m_pTable);
}
```

8.1.3.11 void Clhm : :gererArguments () [private]

Références balleEnJeu(), commencerSeance(), connecterJoueur(), deconnecterJoueur(), DEV_BALLES_MAX, envoyerCommande(), finirSeance(), impacterRandom(), CTable : :m_args, m_fontRougeStyle, m_pTable, CTable : :setBallesMaximum(), setBallesMaximum(), setInfoConnectDemo(), setLayerEcranTable(), setNomPeripheriqueDemo(), setZoneObjectifRandom(), et setZoneRobotRandom().

Référencé par Clhm().

```
{
    QStringList args = QCoreApplication::arguments(); // Verification des
```

```

arguments

srand(QTime::currentTime().msec());

m_pTable->m_args = args;
//=====
// DEV
if (!args.contains("-dev",Qt::CaseInsensitive)) // Suppression des boutons
pour les tests sans materiel
{
    m_pQLabelLogoNom->setText("");
    m_pQLabelLogoNomMessage->setText("");

    delete m_pTestZoneRobotRandom;
    delete m_pTestZoneObjectifRandom;
    delete m_pTestZoneRandom;
    delete m_pTestZoneReset;
}
else
{
    m_pQLabelTopLeftNomRecap->setText("MODE DEVELOPPEMENT");
    m_pQLabelTopLeftNomRecap->setStyleSheet(m_fontRougeStyle);

    m_pQLabelTopLeftNom->setText("MODE DEVELOPPEMENT");
    m_pQLabelTopLeftNom->setStyleSheet(m_fontRougeStyle);

    m_pQLabelLogoNom->setText("MODE DEVELOPPEMENT");
    m_pQLabelLogoNom->setStyleSheet(m_fontRougeStyle);
    m_pQLabelLogoNomMessage->setText("");

    m_pTable->setBallesMaximum(DEV_BALLESMAX);
}
if (!args.contains("-dev",Qt::CaseInsensitive) && !args.contains("-console"
,Qt::CaseInsensitive))
{
    delete m_pTestOutrepasser;
    delete m_pTestStartRecap;
}
else
{
    connect(m_pTestOutrepasser, SIGNAL(pressed()), this, SLOT(
commencerSeance()));
    connect(m_pTestStartRecap, SIGNAL(pressed()), this, SLOT(commencerSeance
()));
}
//=====
// WINDOWED
if (!args.contains("-windowed",Qt::CaseInsensitive)) // mode fenêtré
{
    this->showFullScreen();
}
//=====
// MODE DEMO
if (args.contains("-demo",Qt::CaseInsensitive)) // Mode de demonstration
{
    setBallesMaximum(DEV_BALLESMAX);

    QTimer::singleShot(2000, this, SLOT(connecterJoueur()));
    QTimer::singleShot(2000, this, SLOT(setInfoConnectDemo()));
    QTimer::singleShot(2000, this, SLOT(setNomPeripheriqueDemo()));
    QTimer::singleShot(3500, this, SLOT(setZoneObjectifRandom()));
    if (!args.contains("-norobot",Qt::CaseInsensitive))
        QTimer::singleShot(4000, this, SLOT(setZoneRobotRandom()));
    QTimer::singleShot(5000, this, SLOT(commencerSeance()));
    QTimer::singleShot(4000, this, SLOT(setLayerEcranTable()));

    for(int i=0 ; i < DEV_BALLESMAX ; i++)
    {
        QTimer::singleShot((6500) + i*1000, this, SLOT(balleEnJeu()));
        if(rand() % 10) // 1 chance sur 10 de rater
            QTimer::singleShot((7000) + i*1000, this, SLOT(impacterRandom()
));
    }
}

```

```

        QTimer::singleShot((7000) + (1+DEV_BALLESMAX)*1000, this, SLOT(
finirSeance()));
        QTimer::singleShot((15000) + (DEV_BALLESMAX)*1000, this, SLOT(
deconnecterJoueur()));
    }
    //=====
    // CONSOLE
    if (!args.contains("--console",Qt::CaseInsensitive))
    {
        delete m_pConsole;
    }
    else
    {
        connect(m_pConsole, SIGNAL(returnPressed()), this, SLOT(envoyerCommande
()));
    }
}

```

8.1.3.12 float Clhm : :getRatioFenetreX()

Références [TAILLE_FENETRE_DEFAULT_WIDTH](#).

```

{
    return ( (float)this->width()/TAILLE_FENETRE_DEFAULT_WIDTH );
}

```

8.1.3.13 float Clhm : :getRatioFenetreY()

Références [TAILLE_FENETRE_DEFAULT_HEIGHT](#).

Référencé par [initialisationFenetre\(\)](#), [rafraichirCSS\(\)](#), [resetSeance\(\)](#), [setInfoConnect\(\)](#), et [setLayerEcran\(\)](#).

```

{
    return ( (float)this->height()/TAILLE_FENETRE_DEFAULT_HEIGHT );
}

```

8.1.3.14 QString Clhm : :getTimerSeanceString (unsigned int iTemps) [private]

Référencé par [pauserSeance\(\)](#), [reprendreSeance\(\)](#), et [setTimerSeance\(\)](#).

```

{
    unsigned int iMinutes   = (unsigned int)((float)iTemps / (float)60);
    unsigned int iSecondes  = iTemps - (iMinutes*60);

    QString zeroSeconde = "";    // CORRECTION LIEE AU ZERO

    if (iSecondes < 10)
        zeroSeconde = "0";

    return QString::number(iMinutes) + ":" + zeroSeconde + QString::number(
iSecondes) + " ";
}

```

8.1.3.15 void Clhm : :impacterRandom () [private, slot]

Références [impacterZone\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    impacterZone(rand() % 9);
}
```

8.1.3.16 Clhm : :impacterZone (uint8_t zone) [slot]

Paramètres

<i>zone</i>	enum voir const.h
-------------	-----------------------------------

Références [ballesTotalSurBallesMaximum\(\)](#), [CTable : :impacterZone\(\)](#), [m_pTable](#), et [rafraichirStats\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [impacterRandom\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    m_pTable->impacterZone(zone);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
    rafraichirStats();
}
```

8.1.3.17 void Clhm : :initialisationFenetre () [private, slot]

Références [getRatioFenetreY\(\)](#), [initialisationStats\(\)](#), [LOGO_ATTENTECONNECTION](#), [m_fontNoirStyle](#), [m_fontNormal](#), [m_fontTitreStyle](#), [m_pTable](#), [rafraichirCSS\(\)](#), [rafraichirHeure\(\)](#), [RATIO_ENTETE](#), [RECAP_TITRE_TEXTE](#), et [CTable : :setFiletTaille\(\)](#).

Référencé par [Clhm\(\)](#).

```
{
    rafraichirCSS(getRatioFenetreY());

    //=====
    // LAYER LOGO
    m_pQLabelLogo->setFixedHeight(this->height() / 2);
    m_pQLabelLogo->setFixedWidth(this->width() / 2);
    m_pQLabelLogoTexte->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelLogoHeure->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONNECTION);

    //=====
    // LAYER TABLE
    qDebug() << Q_FUNC_INFO << QTime::currentTime().toString();
    m_pQLabelTopRightHeure->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTimerSeance->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTimerSeanceRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopMid->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopMid->setStyleSheet(m_fontNoirStyle);
    m_pTable->setFiletTaille(getRatioFenetreY());
    initialisationStats();

    //=====
    // LAYER RECAP
    m_pQLabelTopRightHeureRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopTexteRecap->setFixedHeight(this->height() / RATIO_ENTETE);
    m_pQLabelTopTexteRecap->setStyleSheet(m_fontTitreStyle);
    m_pQLabelTopTexteRecap->setFont(m_fontNormal);
    m_pQLabelTopTexteRecap->setText(RECAP_TITRE_TEXTE);

    //=====
    // HEURE
    rafraichirHeure();
}
```

8.1.3.18 void Clhm : :initialisationStats () [private]

Références [m_font](#), [m_fontNoirStyle](#), [m_fontNormal](#), [rafraichirStats\(\)](#), et [TABLE_STAT1](#).

Référencé par [initialisationFenetre\(\)](#).

```
{
    m_pQLabelLeftStat1Texte->setFont (m_font);
    m_pQLabelLeftStat1Nb->setFont (m_font);

    m_pQLabelLeftStat1Texte->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat1Nb->setStyleSheet (m_fontNoirStyle);

    m_pQLabelLeftStat1Texte->setText (TABLE_STAT1);

    //RECAP
    m_pQLabelLeftStat1TexteRecap->setFont (m_font);
    m_pQLabelLeftStat1TexteRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat2TexteRecap->setFont (m_font);
    m_pQLabelLeftStat2TexteRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat3TexteRecap->setFont (m_font);
    m_pQLabelLeftStat3TexteRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat4TexteRecap->setFont (m_font);
    m_pQLabelLeftStat4TexteRecap->setStyleSheet (m_fontNoirStyle);

    m_pQLabelLeftStat1PerRecap->setFont (m_font);
    m_pQLabelLeftStat1PerRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat2PerRecap->setFont (m_font);
    m_pQLabelLeftStat2PerRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat3PerRecap->setFont (m_font);
    m_pQLabelLeftStat3PerRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat4PerRecap->setFont (m_font);
    m_pQLabelLeftStat4PerRecap->setStyleSheet (m_fontNoirStyle);

    m_pQLabelLeftStat1NbRecap->setFont (m_fontNormal);
    m_pQLabelLeftStat1NbRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat2NbRecap->setFont (m_fontNormal);
    m_pQLabelLeftStat2NbRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat3NbRecap->setFont (m_fontNormal);
    m_pQLabelLeftStat3NbRecap->setStyleSheet (m_fontNoirStyle);
    m_pQLabelLeftStat4NbRecap->setFont (m_fontNormal);
    m_pQLabelLeftStat4NbRecap->setStyleSheet (m_fontNoirStyle);

    m_pQLabelLeftStat1TexteRecap->clear();
    m_pQLabelLeftStat2TexteRecap->clear();
    m_pQLabelLeftStat3TexteRecap->clear();
    m_pQLabelLeftStat4TexteRecap->clear();

    m_pQLabelLeftStat1PerRecap->clear();
    m_pQLabelLeftStat2PerRecap->clear();
    m_pQLabelLeftStat3PerRecap->clear();
    m_pQLabelLeftStat4PerRecap->clear();

    m_pQLabelLeftStat1NbRecap->clear();
    m_pQLabelLeftStat2NbRecap->clear();
    m_pQLabelLeftStat3NbRecap->clear();
    m_pQLabelLeftStat4NbRecap->clear();

    rafraichirStats();
}
```

8.1.3.19 void Clhm : :pauserSeance () [slot]

Références [CSS_TIMER_OFF](#), [getTimerSeanceString\(\)](#), [m_iTempsSeance](#), et [m_pTimerSeance](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;

    m_pTimerSeance->stop();

    m_pQLabelTimerSeance->setStyleSheet(CSS_TIMER_OFF);
    m_pQLabelTimerSeance->setText(QString::fromUtf8(" ") +
        getTimerSeanceString(m_iTempsSeance));
}
```

8.1.3.20 void Clhm : :quitter () [slot]

Référencé par [connecterSignaux\(\)](#), [envoyerCommande\(\)](#), et [raccourcisClavier\(\)](#).

```
{
    close();
}
```

8.1.3.21 void Clhm : :raccourcisClavier () [private]

Références [quitter\(\)](#).

Référencé par [Clhm\(\)](#).

```
{
    // CTRL+Q
    QAction *actionQuitter = new QAction("&Quitter", this);
    actionQuitter->setShortcut(QKeySequence(Qt::CTRL + Qt::Key_Q));
    //actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit)); // Ctrl+Q,
    // NE FONCTIONNE PAS SUR PI
    addAction(actionQuitter);
    connect(actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
}
```

8.1.3.22 void Clhm : :rafraichirCSS (float ratio)

Références [ballesTotalSurBallesMaximum\(\)](#), [m_font](#), [m_fontNoirStyle](#), [m_fontNom](#), [m_fontNormal](#), [m_fontSmall](#), [m_fontTitreStyle](#), [m_pTable](#), [CTable : :rafraichirCSS\(\)](#), [TAILLE_TEXTE](#), [TAILLE_TEXTE_NOM](#), [TAILLE_TEXTE_NORMAL](#), et [TAILLE_TEXTE_SMALL](#).

```
{
    //=====
    // SPECIFIQUE AU LOGO
    //=====

    m_font.setBold(true);
    m_fontSmall.setBold(true);
    m_fontNormal.setBold(true);
    m_fontNom.setBold(true);

    m_font.setPointSize((int)(TAILLE_TEXTE*ratio));
    m_fontSmall.setPointSize((int)(TAILLE_TEXTE_SMALL*ratio));
    m_fontNormal.setPointSize((int)(TAILLE_TEXTE_NORMAL*ratio));
    m_fontNom.setPointSize((int)(TAILLE_TEXTE_NOM*ratio));
    m_pQLabelLogoTexte->setFont(m_font);
    m_pQLabelLogoTexte->setStyleSheet(m_fontTitreStyle);
    m_pQLabelLogoHeure->setFont(m_fontNormal);
    m_pQLabelLogoHeure->setStyleSheet(m_fontNoirStyle);
    m_pQLabelLogoNom->setFont(m_font);
    m_pQLabelLogoNomMessage->setFont(m_font);
    m_pQLabelLogoNomMessage->setStyleSheet(m_fontNoirStyle);
}
```



```

//=====
// SPECIFIQUE A LA TABLE
//=====

m_pQLabelTopMid->setFont (m_fontNormal);
m_pQLabelTopLeftNom->setFont (m_fontNom);
m_pQLabelTopLeftNomPeripherique->setFont (m_fontSmall);
m_pQLabelTopRightHeure->setFont (m_fontNormal);
m_pQLabelTopRightHeure->setStyleSheet (m_fontNoirStyle);
m_pQLabelTimerSeance->setFont (m_fontNormal);
m_pQLabelTimerSeanceRecap->setFont (m_fontNormal);
m_pQLabelTopMid->setText (ballesTotalSurBallesMaximum());

m_pTable->rafraichirCSS (ratio);

//=====
// SPECIFIQUE AU RECAP
//=====
m_pQLabelTopRightHeureRecap->setFont (m_fontNormal);
m_pQLabelTopRightHeureRecap->setStyleSheet (m_fontNoirStyle);
m_pQLabelTopLeftNomRecap->setFont (m_fontNom);
m_pQLabelTopLeftNomPeripheriqueRecap->setFont (m_fontSmall);
}

```

8.1.3.23 Clhm : :rafraichirCSS () [slot]

Rafrachit le CSS lié à l'affichage (fontes, couleurs)

Paramètres

<i>ratio</i>	
--------------	--

Références [getRatioFenetreY\(\)](#).

Référencé par [connecterSignaux\(\)](#), [initialisationFenetre\(\)](#), et [resetSeance\(\)](#).

```
{ rafraichirCSS (getRatioFenetreY()); }
```

8.1.3.24 void Clhm : :rafraichirHeure () [slot]

Référencé par [connecterSignaux\(\)](#), et [initialisationFenetre\(\)](#).

```

{
    QString zeroMinute = "", zeroHeure = "";    // CORRECTION LIEE AU ZERO
    if (QTime::currentTime().hour() < 10)
        zeroHeure = "0";
    if (QTime::currentTime().minute() < 10)
        zeroMinute = "0";

    QString format = zeroHeure + QString::number(QTime::currentTime().hour()) +
        ":" + zeroMinute + QString::number(QTime::currentTime().minute()) + " ";
        // EASTER EGG ( ° °)

    if (QTime::currentTime().hour() == 13 &&
        QTime::currentTime().minute() == 37){ format = "L3:3T";}
    m_pQLabelTopRightHeureRecap->setText (format);
    m_pQLabelTopRightHeure->setText (format);
    m_pQLabelLogoHeure->setText (format);
}

```

8.1.3.25 void Clhm : :rafraichirStats () [private]

Références [calculerPourcentageQString\(\)](#), [CTable : :getBallesHorsTable\(\)](#), [CTable : :getBallesTotal\(\)](#), et [m_pTable](#).

Référencé par [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [finirSeance\(\)](#), [impacterZone\(\)](#), [initialisationStats\(\)](#), [resetSeance\(\)](#), [setZoneObjectif\(\)](#), et [setZoneObjectifRandom\(\)](#).

```
{
    m_pQLabelLeftStat1Nb->setText (QString::number (m_pTable->getBallesHorsTable (
    )) + " " + calculerPourcentageQString (m_pTable->getBallesHorsTable(), m_pTable->
    getBallesTotal() - 1));
    m_pQLabelLeftStat2NbRecap->setText (QString::number (m_pTable->
    getBallesHorsTable()) + " / " + QString::number (m_pTable->getBallesTotal()) );
    m_pQLabelLeftStat2PerRecap->setText (calculerPourcentageQString (m_pTable->
    getBallesHorsTable(), m_pTable->getBallesTotal()) );
}
```

8.1.3.26 void Clhm : :rafraichirTimerSeance () [slot]

Références [CSS_TIMER_ON](#), [m_iTempsSeance](#), et [setTimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    //    qDebug() << Q_FUNC_INFO;           //!< Spam
    m_iTempsSeance++;

    m_pQLabelTimerSeance->setStyleSheet (CSS_TIMER_ON);
    setTimerSeance (m_iTempsSeance);
}
```

8.1.3.27 Clhm : :reprendreSeance () [slot]

Références [CSS_TIMER_RES](#), [getTimerSeanceString\(\)](#), [m_iTempsSeance](#), et [m_p-TimerSeance](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;

    m_pTimerSeance->start (1000);
    m_pQLabelTimerSeance->setStyleSheet (CSS_TIMER_RES);
    m_pQLabelTimerSeance->setText (QString::fromUtf8 (" ") +
    getTimerSeanceString (m_iTempsSeance));
}
```

8.1.3.28 Clhm : :resetSeance () [slot]

Références [getRatioFenetreY\(\)](#), [m_iTempsSeance](#), [m_pTable](#), [m_pTimerSeance](#), [rafraichirCSS\(\)](#), [rafraichirStats\(\)](#), [CTable : :resetSeance\(\)](#), [setInfoConnect\(\)](#), et [set-TimerSeance\(\)](#).

Référencé par [connecterSignaux\(\)](#), et [deconnecterJoueur\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
```

```

m_pTable->resetSeance();
rafraichirCSS(getRatioFenetreY());
rafraichirStats();
setInfoConnect(m_pQLabelLogoNom->text());

m_iTempsSeance = 0;
m_pTimerSeance->stop();
setTimerSeance(0);
}

```

8.1.3.29 Clhm : :setBallesMaximum (int balles) [slot]

Paramètres

<i>balles</i>	int
---------------	-----

Références [ballesTotalSurBallesMaximum\(\)](#), [m_fontNormal](#), [m_pTable](#), et [CTable : :setBallesMaximum\(\)](#).

Référéncé par [connecterSignaux\(\)](#), et [gererArguments\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << "BALLES: " << balles;
    m_pTable->setBallesMaximum(balles);
    //    rafraichirCSS(getRatioFenetreY()); // nécessaire pour refresh l'IHM car
        ballemax est utilisé par elle aussi
    m_pQLabelTopMid->setFont(m_fontNormal);
    m_pQLabelTopMid->setText(ballesTotalSurBallesMaximum());
}

```

8.1.3.30 Clhm : :setInfoConnect (QString nom) [slot]

Paramètres

<i>nom</i>	QString nom du joueur
------------	-----------------------

Références [getRatioFenetreY\(\)](#), [LOGO_ATTENTECONFIGURATION](#), [m_fontNom](#), et [TAILLE_TEXTE_NOM](#).

Référéncé par [connecterSignaux\(\)](#), [resetSeance\(\)](#), et [setInfoConnectDemo\(\)](#).

```

{
    if (nom == "")
        return;

    m_pQLabelTopLeftNomRecap->setText(nom);
    m_pQLabelTopLeftNom->setText(nom);
    //    m_pQLabelLogoNom->setText(nom); affiche le nom du peripherique depuis
        1.1
    m_pQLabelLogoTexte->setText(LOGO_ATTENTECONFIGURATION);
    qDebug() << Q_FUNC_INFO << " nom : " << nom;

    float tailleNomRatio;
    if (nom.length() > 20)
        tailleNomRatio = (4.0 / (float)(nom.length()) + (1-4.0/20));
    else
        tailleNomRatio = 1.0;

    m_fontNom.setPointSize((int)(TAILLE_TEXTE_NOM*getRatioFenetreY()) *
        tailleNomRatio);
    //    m_pQLabelTopLeftNom->setFont(m_fontNom); affiche le nom du
        peripherique depuis 1.1
    m_pQLabelTopLeftNomRecap->setFont(m_fontNom);
}

```

8.1.3.31 void Clhm : :setInfoConnectDemo () [private, slot]

Références [IHM_NOMDETEST](#), et [setInfoConnect\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{ setInfoConnect (IHM_NOMDETEST); }
```

8.1.3.32 Clhm : :setLayerEcran (uint8_t layer) [slot]

Paramètres

<i>layer</i>	enum voir const.h
--------------	-----------------------------------

Références [getRatioFenetreY\(\)](#), [m_pTable](#), et [CTable : :setLayerEcran\(\)](#).

Référencé par [commencerSeance\(\)](#), [connecterJoueur\(\)](#), [connecterSignaux\(\)](#), [deconnecterJoueur\(\)](#), [finirSeance\(\)](#), [setLayerEcranLogo\(\)](#), [setLayerEcranRecap\(\)](#), et [setLayerEcranTable\(\)](#).

```
{
    m_pTable->setLayerEcran(layer,getRatioFenetreY());
    m_pFenetres->setCurrentIndex(layer);
    qDebug() << Q_FUNC_INFO;
}
```

8.1.3.33 void Clhm : :setLayerEcranLogo () [private, slot]

Références [LAYER_LOGO](#), et [setLayerEcran\(\)](#).

```
{ setLayerEcran (LAYER_LOGO); }
```

8.1.3.34 void Clhm : :setLayerEcranRecap () [private, slot]

Références [LAYER_RECAP](#), et [setLayerEcran\(\)](#).

```
{ setLayerEcran (LAYER_RECAP); }
```

8.1.3.35 void Clhm : :setLayerEcranTable () [private, slot]

Références [LAYER_TABLE](#), et [setLayerEcran\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{ setLayerEcran (LAYER_TABLE); }
```

8.1.3.36 Clhm : :setNomPeripherique (QString nom = "Peripherique DEMO")
[slot]

Paramètres

<i>nom</i>	QString nom du périphérique
------------	-----------------------------

Références [LOGO_JOUEUR_CONNECTE](#), et [m_fontVertStyle](#).

Référencé par [connecterSignaux\(\)](#), et [setNomPeripheriqueDemo\(\)](#).

```
{
    m_pQLabelLogoNom->setText (nom);
    m_pQLabelTopLeftNomPeripherique->setText (" " + nom + " ");
    m_pQLabelTopLeftNomPeripheriqueRecap->setText (" " + nom + " ");

    m_pQLabelLogoNom->setStyleSheet (m_fontVertStyle);
    m_pQLabelTopLeftNomPeripherique->setStyleSheet (m_fontVertStyle);
    m_pQLabelTopLeftNomPeripheriqueRecap->setStyleSheet (m_fontVertStyle);

    m_pQLabelLogoNomMessage->setText (LOGO_JOUEUR_CONNECTE);
}
```

8.1.3.37 void Clhm : :setNomPeripheriqueDemo () [private, slot]

Références [IHM_PERIPHERIQUEDETEST](#), et [setNomPeripherique\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{ setNomPeripherique (IHM_PERIPHERIQUEDETEST); }
```

8.1.3.38 void Clhm : :setTimerSeance (unsigned int iTemps = 0) [private]

Paramètres

<i>Minutes,- Secondes</i>	
-------------------------------	--

Références [getTimerSeanceString\(\)](#).

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [rafraichirTimerSeance\(\)](#), et [resetSeance\(\)](#).

```
{
    m_pQLabelTimerSeance->setText ( getTimerSeanceString (iTemps) );
    m_pQLabelTimerSeanceRecap->setText ( getTimerSeanceString (iTemps) );
}
```

8.1.3.39 Clhm : :setZoneObjectif (uint8_t zone) [slot]

Paramètres

<i>zone</i>	enum voir const.h
-------------	-----------------------------------

Références [m_pTable](#), [rafraichirStats\(\)](#), et [CTable : :setZoneObjectif\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug () << Q_FUNC_INFO << "ZONE: " << zone;
    m_pTable->setZoneObjectif (zone);
    rafraichirStats ();
}
```

8.1.3.40 void Clhm : :setZoneObjectifRandom () [private, slot]

Références [m_pTable](#), [rafraichirStats\(\)](#), et [CTable : :setZoneObjectif\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    m_pTable->setZoneObjectif(rand() % 9);
    rafraichirStats();
}
```

8.1.3.41 Clhm : :setZoneRobot (uint8_t zone) [slot]

Paramètres

zone	enum voir const.h
-------------	-----------------------------------

Références [m_pTable](#), et [CTable : :setZoneRobot\(\)](#).

Référencé par [connecterSignaux\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << "ZONE: " << zone;
    m_pTable->setZoneRobot (zone);
}
```

8.1.3.42 void Clhm : :setZoneRobotRandom () [private, slot]

Références [m_pTable](#), et [CTable : :setZoneRobot\(\)](#).

Référencé par [gererArguments\(\)](#).

```
{
    qDebug() << ">> DEV! " << Q_FUNC_INFO;
    int random = -1;

    while (random < 0 || random == m_pTable->getZoneObjectif())
    {
        srand(QTime::currentTime().msec());
        random = rand() % 9;
        m_pTable->setZoneRobot (random);
    }
}
```

8.1.4 Documentation des données membres**8.1.4.1 QFont Clhm : :m_font [private]**

Référencé par [initialisationStats\(\)](#), et [rafraichirCSS\(\)](#).

8.1.4.2 QString Clhm : :m_fontNoirStyle [private]

Référencé par [Clhm\(\)](#), [initialisationFenetre\(\)](#), [initialisationStats\(\)](#), et [rafraichirCSS\(\)](#).

8.1.4.3 QFont Clhm : :m_fontNom [private]

Référencé par [rafraichirCSS\(\)](#), et [setInfoConnect\(\)](#).

8.1.4.4 QFont Clhm : :m_fontNormal [private]

Référencé par [balleEnJeu\(\)](#), [initialisationFenetre\(\)](#), [initialisationStats\(\)](#), [rafraichirCSS\(\)](#), et [setBallesMaximum\(\)](#).

8.1.4.5 QString Clhm : :m_fontRougeStyle [private]

Référencé par [Clhm\(\)](#), et [gererArguments\(\)](#).

8.1.4.6 QFont Clhm : :m_fontSmall [private]

Référencé par [rafraichirCSS\(\)](#).

8.1.4.7 QString Clhm : :m_fontTitreStyle [private]

Référencé par [Clhm\(\)](#), [initialisationFenetre\(\)](#), et [rafraichirCSS\(\)](#).

8.1.4.8 QString Clhm : :m_fontVertStyle [private]

Référencé par [Clhm\(\)](#), et [setNomPeripherique\(\)](#).

8.1.4.9 unsigned int Clhm : :m_iTempsSeance [private]

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [pauserSeance\(\)](#), [rafraichirTimerSeance\(\)](#), [repandreSeance\(\)](#), et [resetSeance\(\)](#).

8.1.4.10 CommunicationBluetooth* Clhm : :m_pCommunicationBluetooth [private]

Référencé par [Clhm\(\)](#), [connecterSignaux\(\)](#), et [~Clhm\(\)](#).

8.1.4.11 CTable* Clhm : :m_pTable [private]

Référencé par [balleEnJeu\(\)](#), [ballesTotalSurBallesMaximum\(\)](#), [Clhm\(\)](#), [commencerSeance\(\)](#), [finirSeance\(\)](#), [gererArguments\(\)](#), [impacterZone\(\)](#), [initialisationFenetre\(\)](#), [rafraichirCSS\(\)](#), [rafraichirStats\(\)](#), [resetSeance\(\)](#), [setBallesMaximum\(\)](#), [setLayerEcran\(\)](#), [setZoneObjectif\(\)](#), [setZoneObjectifRandom\(\)](#), [setZoneRobot\(\)](#), et [setZoneRobotRandom\(\)](#).

8.1.4.12 QThread* Clhm : :m_pThreadCommunicationBluetooth [private]

Référencé par [Clhm\(\)](#), [connecterSignaux\(\)](#), et [~Clhm\(\)](#).

8.1.4.13 QTimer* Clhm : :m_pTimerHeure [private]

Référencé par [Clhm\(\)](#), et [connecterSignaux\(\)](#).

8.1.4.14 QTimer* Clhm : :m_pTimerSeance [private]

Référencé par [Clhm\(\)](#), [commencerSeance\(\)](#), [connecterSignaux\(\)](#), [finirSeance\(\)](#), [pauserSeance\(\)](#), [repandreSeance\(\)](#), et [resetSeance\(\)](#).

8.1.4.15 CFrame* CIhm : :m_pTrame [private]

Référencé par [CIhm\(\)](#), [connecterSignaux\(\)](#), et [envoyerCommande\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

- [ihm.h](#)
- [ihm.cpp](#)

8.2 Référence de la classe CommunicationBluetooth

Assure la réception des trames via le Bluetooth.

```
#include <communicationbluetooth.h>
```

Types publics

- enum [Mode](#) { [Scrutation](#), [Evenement](#) }
Mode de gestion du port de communication.

Connecteurs publics

- void [main](#) ()
Le corps du thread assurant la réception des données via le Bluetooth.
- void [finir](#) ()
Met fin au Thread.
- void [terminer](#) ()
Termine le thread.
- void [lirePort](#) ()
Lit les données disponibles sur le port Bluetooth.
- void [surveillerConnexion](#) ()
Surveille l'état du service RFCOMM.

Signaux

- void [nouvellesDonneesRecues](#) (QString donneesRecues)
Signale les données reçues sur le port Bluetooth.
- void [deconnecterJoueur](#) ()
Signale une déconnexion du Bluetooth.
- void [connecterJoueur](#) ()
Signale une connexion du Bluetooth.
- void [setNomPeripherique](#) (QString nom)
Affiche le nom du périphérique connecté

Fonctions membres publiques

- [CommunicationBluetooth](#) (QString nomPort="rfcomm0", Mode mode=[Scrutation](#), Q-Object *parent=0)
Constructeur.
- [~CommunicationBluetooth](#) ()
Destructeur.
- void [ouvrir](#) ()
Ouvre et configure le port série.

- void `fermer ()`
Ferme le port série.
- int `getEtatRFCOMM ()`
Retourne l'état du port RFCOMM.

Fonctions membres privées

- void `msleep (unsigned long sleepMS)`
Temporise l'exécution du Thread.
- void `annuler ()`
Termine la temporisation en cours.
- void `attendrePeriode ()`
Attend une durée PERIODE_SURVEILLANCE.
- void `demarrerRFCOMM ()`
Démarre le service RFCOMM.
- void `redemarrerRFCOMM ()`
Redémarre le service RFCOMM.
- QString `lireEtatRFCOMM ()`
Lit l'état du service RFCOMM.
- QString `lireEtatServiceRFCOMM ()`
- void `arreterRFCOMM ()`
Arrête le service RFCOMM.
- void `recupererNomBluetooth ()`
Récupère le nom de l'appareil connecté

Attributs privés

- QTextSerialPort * `m_pPortSerie`
Agrégation vers la classe QTextSerialPort.
- bool `fini`
état du Thread.
- QMutex `mutex`
pour gérer les temporisations
- QWaitCondition `waitCondition`
pour les temporisations
- int `periode`
la périodicité du thread
- int `etatRFCOMM`
état du service RFCOMM.
- QTimer * `timerSurveillance`
pour surveiller périodiquement l'état de la connexion Bluetooth

8.2.1 Documentation des énumérations membres

8.2.1.1 enum CommunicationBluetooth : :Mode

Valeurs énumérées :

Scrutation

Evenement

```
{
    Scrutation,
    Evenement
};
```

8.2.2 Documentation des constructeurs et destructeur

8.2.2.1 CommunicationBluetooth : :CommunicationBluetooth (QString *nomPort* = "rfcomm0", CommunicationBluetooth : :Mode *mode* = Scrutation, QObject * *parent* = 0) [explicit]

Paramètres

<i>nomPort</i>	QString le nom du fichier de périphérique Bluetooth (par défaut rfcomm0)
<i>parent</i>	QObject Adresse de l'objet Qt parent (par défaut 0)

Références [demarrerRFCOMM\(\)](#), [m_pPortSerie](#), [periode](#), [Scrutation](#), [surveillerConnexion\(\)](#), et [timerSurveillance](#).

```

                                : QObject(parent), fini(false), periode(
PERIODE_SURVEILLANCE), etatRFCOMM(RFCOMM_ARRETE)
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    demarrerRFCOMM();

    // cf. méthode main()
    if(mode == CommunicationBluetooth::Scrutation)
    {
        m_pPortSerie = new QextSerialPort(QextSerialPort::Polling, this);
        m_pPortSerie->setPortName(QString("/dev/") + nomPort);
        surveillerConnexion();
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Mode Polling (par
scrutation)");
        #endif
    }
    else /* CommunicationBluetooth::Evenement */
    {
        m_pPortSerie = new QextSerialPort(QextSerialPort::EventDriven, this);
        m_pPortSerie->setPortName(QString("/dev/") + nomPort);
        surveillerConnexion();

        timerSurveillance = new QTimer(this);
        timerSurveillance->setInterval(periode);
        connect(timerSurveillance, SIGNAL(timeout()), this, SLOT(
surveillerConnexion()));
        timerSurveillance->start();
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Mode EventDriven (par
événement)");
        #endif
    }
}

```

8.2.2.2 CommunicationBluetooth : :~CommunicationBluetooth ()

Références [arreterRFCOMM\(\)](#), et [fermer\(\)](#).

```

{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    fermer();
    arreterRFCOMM();
}

```

8.2.3 Documentation des fonctions membres

8.2.3.1 CommunicationBluetooth : :annuler () [inline, private]

Référencé par [finir\(\)](#).

```
{  
    waitCondition.wakeAll();  
}
```

8.2.3.2 CommunicationBluetooth : :arreterRFCOMM () [private]

Référencé par [~CommunicationBluetooth\(\)](#).

```
{  
    FILE *resultat;  
    resultat = popen("sudo systemctl stop rfcomm.service 2> /dev/null", "r");  
    pclose(resultat);  
}
```

8.2.3.3 CommunicationBluetooth : :attendrePeriode () [private]

Références [msleep\(\)](#), et [periode](#).

Référencé par [surveillerConnexion\(\)](#).

```
{  
    this->msleep(periode); // en ms  
}
```

8.2.3.4 CommunicationBluetooth : :connecterJoueur () [signal]

Référencé par [surveillerConnexion\(\)](#).

8.2.3.5 CommunicationBluetooth : :deconnecterJoueur () [signal]

Référencé par [surveillerConnexion\(\)](#).

8.2.3.6 CommunicationBluetooth : :demarrerRFCOMM () [private]

Référencé par [CommunicationBluetooth\(\)](#), et [surveillerConnexion\(\)](#).

```
{  
    FILE *resultat;  
    resultat = popen("sudo systemctl start rfcomm.service 2> /dev/null", "r");  
    pclose(resultat);  
}
```

8.2.3.7 CommunicationBluetooth : :fermer ()

Références [lirePort\(\)](#), et [m_pPortSerie](#).

Référencé par [surveillerConnexion\(\)](#), et [~CommunicationBluetooth\(\)](#).

```

{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    if (m_pPortSerie->isOpen())
    {
        if(m_pPortSerie->queryMode() == QextSerialPort::EventDriven)
            disconnect(m_pPortSerie, SIGNAL(readyRead()), this, SLOT(lirePort())
        );
        m_pPortSerie->close();
    }
}

```

8.2.3.8 CommunicationBluetooth : :finir() [slot]

Références [annuler\(\)](#), et [fini](#).

Référencé par [Clhm : :~Clhm\(\)](#).

```

{
    fini = true;
    annuler();
}

```

8.2.3.9 CommunicationBluetooth : :getEtatRFCOMM()

Références [etatRFCOMM](#).

```

{ return etatRFCOMM; }

```

8.2.3.10 CommunicationBluetooth : :lireEtatRFCOMM() [private]

Renvoie

QString

Référencé par [surveillerConnexion\(\)](#).

```

{
    FILE *resultat;
    char ligne[1024];
    QString reponse;

    // lit l'état de la connexion
    resultat = popen("rfcomm -a", "r");
    fgets(ligne, 1024, resultat);
    while (!feof(resultat))
    {
        reponse += ligne;
        fgets(ligne, 1024, resultat);
    }
    pclose(resultat);

    return reponse;
}

```

8.2.3.11 QString CommunicationBluetooth : :lireEtatServiceRFCOMM() [private]

Référencé par [surveillerConnexion\(\)](#).

```

{
    FILE *resultat;
    char ligne[1024];
    QString reponse;

    // lit l'état de la connexion
    resultat = popen("sudo systemctl status rfcomm.service 2> /dev/null", "r");
    fgets(ligne, 1024, resultat);
    while (!feof(resultat))
    {
        reponse += ligne;
        fgets(ligne, 1024, resultat);
    }
    pclose(resultat);

    return reponse;
}

```

8.2.3.12 CommunicationBluetooth : lirePort () [slot]

Références [m_pPortSerie](#), [msleep\(\)](#), et [nouvellesDonneesRecues\(\)](#).

Référencé par [fermer\(\)](#), [main\(\)](#), et [ouvrir\(\)](#).

```

{
    if (!m_pPortSerie->isOpen())
        return;

    QByteArray donnees;

    // lecture des données disponibles
    while (m_pPortSerie->bytesAvailable())
    {
        donnees += m_pPortSerie->readAll();
        this->msleep(20); // en ms
        //::usleep(100000); // cf. timeout
    }

    QString donneesRecues = QString(QString::fromUtf8(donnees.data()));
    if (!donneesRecues.isEmpty())
    {
#ifdef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
        QString::fromUtf8("Données reçues : ") << donneesRecues;
#endif
        emit nouvellesDonneesRecues(donneesRecues);
    }
}

```

8.2.3.13 CommunicationBluetooth : main () [slot]

Références [fini](#), [lirePort\(\)](#), [m_pPortSerie](#), et [surveillerConnexion\(\)](#).

```

{
#ifdef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
    QString::fromUtf8("Communication Bluetooth démarrée");
#endif

    if (m_pPortSerie->queryMode() == QextSerialPort::Polling)
    {
        // On interroge périodiquement le port de communication et on
        // surveille l'état de la (connexion)
        while (!fini)
        {
            lirePort();
        }
    }
}

```

```

        surveillerConnexion();
    }
}
else
{
    // On crée une boucle d'événements nécessaire pour gérer les signaux
    (readyRead() et timeout())
    while(!fini)
    {
        QApplication::processEvents();
        //QApplication::processEvents(QEventLoop::ExcludeUserInputEvents,
        periode);
    }
}
#ifdef QT_NO_DEBUG
qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
    QString::fromUtf8("Communication Bluetooth arrêtée");
#endif
}

```

8.2.3.14 CommunicationBluetooth : msleep (unsigned long *sleepMS*) [inline, private]

Paramètres

<i>sleepMS</i>	unsigned long durée de la temporisation en millisecondes
----------------	--

Référencé par [attendrePeriode\(\)](#), [lirePort\(\)](#), et [surveillerConnexion\(\)](#).

```

{
    waitCondition.wait(&mutex, sleepMS);
}

```

8.2.3.15 CommunicationBluetooth : nouvellesDonneesRecues (QString *donneesRecues*) [signal]

Paramètres

<i>donnees-Recues</i>	
-----------------------	--

Référencé par [lirePort\(\)](#).

8.2.3.16 CommunicationBluetooth : ouvrir ()

Références [lirePort\(\)](#), [m_pPortSerie](#), et [recupererNomBluetooth\(\)](#).

Référencé par [surveillerConnexion\(\)](#).

```

{
    if (m_pPortSerie->isOpen())
    {
        #ifndef QT_NO_DEBUG
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Le port %1 est ouvert").
            arg(m_pPortSerie->portName());
        #endif
        return;
    }
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this;
    #endif
    // ouverture du port
}

```

```

m_pPortSerie->open(QIODevice::ReadWrite);
if (!m_pPortSerie->isOpen())
{
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QString::fromUtf8("Le port %1 n'est pas
ouvert").arg(m_pPortSerie->portName());
    #endif
    return;
}
else
    #ifndef QT_NO_DEBUG
    qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouverture du port %1
réussie").arg(m_pPortSerie->portName());
    #endif
// configuration du port
m_pPortSerie->setBaudRate(BAUD9600);
m_pPortSerie->setDataBits(DATA_8);
m_pPortSerie->setParity(PAR_NONE);
m_pPortSerie->setStopBits(STOP_1);
m_pPortSerie->setFlowControl(FLOW_OFF);
if(m_pPortSerie->queryMode() == QextSerialPort::EventDriven)
{
    if (m_pPortSerie->isOpen())
    {
        connect(m_pPortSerie, SIGNAL(readyRead()), this, SLOT(lirePort()));
        recupererNomBluetooth();
    }
}
}

```

8.2.3.17 CommunicationBluetooth : :recupererNomBluetooth () [private]

Références [setNomPeripherique\(\)](#).

Référencé par [ouvrir\(\)](#).

```

{
    FILE *resultat;
    char ligne[1024];
    QString reponse;

    resultat = popen("rfcomm -a", "r"); // récupération de l'adresse
    fgets(ligne, 1024, resultat);
    while (!feof(resultat))
    {
        reponse += ligne;
        fgets(ligne, 1024, resultat);
    }
    pclose(resultat);

    QString adresseMAC;

    if (reponse.size() == 0)
    {
        qDebug() << Q_FUNC_INFO << "!!\ NE PEUT PAS RECUPERER L'ADRESSE MAC !!
        \\";
        return;
    }

    adresseMAC = reponse.mid(30, 17);

    QString commande = "hcitool name " + adresseMAC;

    reponse = "";

    resultat = popen(commande.toAscii(), "r"); // récupération du nom
    fgets(ligne, 1024, resultat);
    while (!feof(resultat))
    {
        reponse += ligne;
        fgets(ligne, 1024, resultat);
    }
}

```

```

    }
    pclose(resultat);

    if (reponse.size() == 0)
    {
        qDebug() << Q_FUNC_INFO << "!!\\ NE PEUT PAS RECUPERER LE NOM DE
        L'APPAREIL /!\\\";
        return;
    }

    qDebug() << Q_FUNC_INFO << "Nom de l'appareil connecte : " << reponse.
        trimmed();

    emit setNomPeripherique(reponse.trimmed());
}

```

8.2.3.18 CommunicationBluetooth : :redemarrerRFCOMM () [private]

Référencé par [surveillerConnexion\(\)](#).

```

{
    FILE *resultat;
    resultat = popen("sudo systemctl restart rfcomm.service 2> /dev/null", "r")
    ;
    pclose(resultat);
}

```

8.2.3.19 CommunicationBluetooth : :setNomPeripherique (QString nom) [signal]

Paramètres

<i>nom</i>	QString nom du périphérique
------------	-----------------------------

Référencé par [recupererNomBluetooth\(\)](#).

8.2.3.20 CommunicationBluetooth : :surveillerConnexion () [slot]

Références [attendrePeriode\(\)](#), [connecterJoueur\(\)](#), [deconnecterJoueur\(\)](#), [demarrerRFCOMM\(\)](#), [etatRFCOMM](#), [fermer\(\)](#), [lireEtatRFCOMM\(\)](#), [lireEtatServiceRFCOMM\(\)](#), [m_pPortSerie](#), [msleep\(\)](#), [ouvrir\(\)](#), [redemarrerRFCOMM\(\)](#), [RFCOMM_ARRETE](#), [RFCOMM_M_CONNECTE](#), et [RFCOMM_FERME](#).

Référencé par [CommunicationBluetooth\(\)](#), et [main\(\)](#).

```

{
    if(m_pPortSerie->bytesAvailable())
        return;

    attendrePeriode();

    QString etat = lireEtatServiceRFCOMM();

    if(!etat.isEmpty())
    {
        if(etat.contains("inactive"))
        {
            #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth rfcomm0
            inactif");
            #endif
            demarrerRFCOMM();
        }
    }
}

```



```

        //return;
        etat = lireEtatServiceRFCOMM();
    }
    if(etat.contains("active"))
    {
        #ifndef QT_NO_DEBUG
        //qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth rfcomm0
actif");
        #endif
        etat = lireEtatRFCOMM();

        #ifndef QT_NO_DEBUG
        //qDebug() << Q_FUNC_INFO << etat << etatRFCOMM;
        #endif
        if(!etat.isEmpty())
        {
            if(etat.contains("connected"))
            {
                if(etatRFCOMM != RFCOMM_CONNECTE)
                {
                    etatRFCOMM = RFCOMM_CONNECTE;
                    #ifndef QT_NO_DEBUG
                    qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth
rfcomm0 connecté");
                    #endif
                    ouvrir();
                    emit connecterJoueur();
                    return;
                }
            }
            else if(etat.contains("closed"))
            {
                if(etatRFCOMM != RFCOMM_FERME)
                {
                    etatRFCOMM = RFCOMM_FERME;
                    #ifndef QT_NO_DEBUG
                    qDebug() << Q_FUNC_INFO << QString::fromUtf8("Bluetooth
rfcomm0 fermé");
                    #endif
                    fermer();
                    this->msleep(100); // en ms
                    redemarrerRFCOMM();
                    emit deconnecterJoueur();
                    return;
                }
            }
        }
    }
    else
    {
        if(etatRFCOMM != RFCOMM_ARRETE)
        {
            etatRFCOMM = RFCOMM_ARRETE;
            #ifndef QT_NO_DEBUG
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucune
connexion Bluetooth rfcomm0");
            #endif
        }
    }
}
}
}

```

8.2.3.21 CommunicationBluetooth : terminer() [slot]

```

{
}

```

8.2.4 Documentation des données membres

8.2.4.1 `int CommunicationBluetooth : :etatRFCOMM` [private]

Référencé par [getEtatRFCOMM\(\)](#), et [surveillerConnexion\(\)](#).

8.2.4.2 `bool CommunicationBluetooth : :fini` [private]

Référencé par [finir\(\)](#), et [main\(\)](#).

8.2.4.3 `QextSerialPort* CommunicationBluetooth : :m_pPortSerie` [private]

Référencé par [CommunicationBluetooth\(\)](#), [fermer\(\)](#), [lirePort\(\)](#), [main\(\)](#), [ouvrir\(\)](#), et [surveillerConnexion\(\)](#).

8.2.4.4 `QMutex CommunicationBluetooth : :mutex` [private]

8.2.4.5 `int CommunicationBluetooth : :periode` [private]

Référencé par [attendrePeriode\(\)](#), et [CommunicationBluetooth\(\)](#).

8.2.4.6 `QTimer* CommunicationBluetooth : :timerSurveillance` [private]

Référencé par [CommunicationBluetooth\(\)](#).

8.2.4.7 `QWaitCondition CommunicationBluetooth : :waitCondition` [private]

La documentation de cette classe a été générée à partir des fichiers suivants :

- [communicationbluetooth.h](#)
- [communicationbluetooth.cpp](#)

8.3 Référence de la classe CTable

```
#include <table.h>
```

Connecteurs publics

- void [rafraichirInactif](#) ()

Fonctions membres publiques

- [CTable](#) (QWidget *parent=0)
- void [setFiletTaille](#) (float ratio)
- void [rafraichirCSS](#) (float ratio)
Définis la hauteur du filet en utilisant la hauteur de la fenêtre actuelle.
- bool [impacterZone](#) (uint8_t numeroZone)
Rafraichit le CSS en utilisant la hauteur de la fenêtre actuelle.
- void [setZoneRobot](#) (uint8_t zone)
Calcul et affiche l'impact sur l'IHM et la table.
- void [setZoneObjectif](#) (uint8_t zone)
Place le robot sur la table.
- void [resetSeance](#) ()
Place la zone objectif sur la table.
- void [resetStatistiques](#) ()

- *Reset complet des variables des statistiques.*
- void `balleEnJeu ()`
- *Reset des statistiques uniquement.*
- void `resetNbBallesZone ()`
- *La balle a bien été mise en jeu par la machine, BalleTotal est incrémenté*
- void `finirSeance ()`
- *Reset du nombre de balles dans chaque zone.*
- int `getBallesBonnes ()`
- int `getBallesTotal ()`
- *Récupère le nombre de balles ayant été jouées et renvoyé par le joueur.*
- int `getBallesMaximum ()`
- *Récupère le nombre de balles ayant été jouées par la machine (en comptant celles n'ayant pas atteint une zone car le joueur est mauvais (° °))*
- int `getBallesObjectif ()`
- *Récupère le nombre de balles maximum pour la seance.*
- int `getBallesEnchainees ()`
- *Récupère le nombre de balles correspondant à la zone objectif actuel.*
- int `getBallesHorsTable ()`
- *Récupère le nombre de balles enchainées.*
- int `getZoneToucheePrec ()`
- *Récupère le nombre de balles hors table.*
- void `setBallesMaximum (int nb)`
- *Récupère la zone touchée précédente.*
- uint8_t `getZoneObjectif ()`
- *Définis le nombre de balles maximum pour la séance.*
- bool `getBalleCoteTablePrec ()`
- *Recupère le numero de la case Objectif.*
- bool `getBalleCoteTable ()`
- *Récupère le coté de la table ou la balle a frapé en dernier.*
- void `setLayerEcran (uint8_t layer, float tailleFenetreY)`
- *Récupère le coté de la table ou la balle a frapé juste avant la derniere.*

Attributs publics

- QStringList `m_args`
- QLabel * `m_pOverlayText`

Fonctions membres privées

- void `rafraichirNbBallesZone ()`
- *Corrections lié au changement de fenetre.*

Attributs privés

- QGridLayout * `m_pGridLayout`
- *Rafraichit le nombre de balles par zone.*
- QVector< QLabel * > `m_pZones`
- QLabel * `m_pFilet`
- QLabel * `m_pOverlay`
- int `m_iBallesBonnes`
- int `m_iBallesMaximum`
- int `m_iBallesTotal`
- int `m_iBallesEnchainees`
- int `m_iBallesEnchaineesMax`
- int `m_iBallesHorsTable`
- int `m_iBallesDansZone [NB_ZONES]`
- int `m_iZoneTouchee`

- int m_iZoneToucheePrec
- int m_iZoneRobot
- int m_iZoneObjectif
- bool m_bBalleCoteTable
- bool m_bBalleCoteTablePrec
- QFont m_font
- QFont m_fontBig
- QFont m_fontOverlay
- QString m_fondInactif
- QString m_fondActif
- QString m_fondRobot
- QString m_fondObjectif
- QString m_fondRate

8.3.1 Documentation des constructeurs et destructeur

8.3.1.1 CTable : CTable (QWidget * parent = 0) [explicit]

Références [CSS_FOND_INACTIF](#), [m_args](#), [m_font](#), [m_pFilelet](#), [m_pGridLayout](#), [m_pOverlay](#), [m_pZones](#), [NB_ZONES](#), [resetSeance\(\)](#), et [WIDGET_SIZE_MAX](#).

```

                                :
QWidget (parent)
{
    //=====
    //          GRAPHIQUE
    //=====

    m_pGridLayout = new QGridLayout (this);

    //=====
    //          OVERLAY TABLE
    //=====
    m_pOverlay = new QLabel (this);
    m_pOverlay->setStyleSheet (QString::fromUtf8 ("QLabel\n"
        "{\n"
        "    border-image: url (:/images/resources/table.jpg) 0 0 0 0 stretch\n"
        "    stretch;\n"
        "}")");
    m_pOverlay->setMinimumSize (QSize (0, WIDGET_SIZE_MAX));
    m_pOverlay->setMaximumSize (QSize (WIDGET_SIZE_MAX, WIDGET_SIZE_MAX));
    m_pGridLayout->addWidget (m_pOverlay, 0, 0, 3, 3);

    //=====
    //          ZONES
    //=====
    QLabel* pZone;
    for (uint8_t i=0; i < NB_ZONES; i++)
    {
        pZone = new QLabel (this);
        pZone->setText ("Zone " + QString::number (i+1));
        pZone->setFont (m_font);
        pZone->setStyleSheet (CSS_FOND_INACTIF);
        pZone->setAlignment (Qt::AlignCenter);

        //-----
        m_pZones.push_back (pZone);
        m_pGridLayout->addWidget (pZone, (i/3), (i%3));
    }

    //=====
    //          OVERLAY TEXT
    //=====

    /* Affichage SUR la table

```

```

m_pOverlayText = new QLabel(this);
m_pOverlayText->setMinimumSize(QSize(0, WIDGET_SIZE_MAX));
m_pOverlayText->setMaximumSize(QSize(WIDGET_SIZE_MAX, WIDGET_SIZE_MAX));
m_pOverlayText->setText("");
m_fontOverlay.setPointSize(12);
m_pOverlayText->setFont(m_fontOverlay);
m_pOverlayText->setAlignment(Qt::AlignCenter);
m_pGridLayout->addWidget(m_pOverlayText, 0, 0, 3, 3);
m_pOverlayText->setStyleSheet(QString::fromUtf8("QLabel\n"
"{\n"
"  background-color: rgb(0, 0, 0, 0);\n"
"  color: rgba(0,0,0,0);\n"
"}"));
*/

//=====
//          FILET
//=====
m_pFilet = new QLabel(this);
m_pFilet->setStyleSheet(QString::fromUtf8("QLabel\n"
"{\n"
"  background: url(/images/resources/filet.jpg) cover;\n"
"}"));
m_pFilet->setMinimumSize(QSize(4, WIDGET_SIZE_MAX));
m_pFilet->setMaximumSize(QSize(WIDGET_SIZE_MAX, 50));
m_pGridLayout->addWidget(m_pFilet, 4, 0, 1, 0);

//=====
setLayout(m_pGridLayout);
m_pGridLayout->setSpacing(0);

//=====
// LOGIQUE
//=====
resetSeance();

//AUTRE
m_args.clear();
}

```

8.3.2 Documentation des fonctions membres

8.3.2.1 void CTable : :balleEnJeu ()

*

Références [getBalleCoteTable\(\)](#), [getBalleCoteTablePrec\(\)](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), [m_iBallesHorsTable](#), et [m_iBallesTotal](#).

Référencé par [CIhm](#) : [:balleEnJeu\(\)](#).

```

{
    m_iBallesTotal++;
    //    rafraichirNbBallesZone();

    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = false;

    if(getBalleCoteTablePrec() == false && getBalleCoteTable() == false)
    {
        qDebug() << Q_FUNC_INFO << " La Balle n'as pas ete renvoye";
        m_iBallesHorsTable++;
    }
}

```

8.3.2.2 void CTable :finirSeance ()

*

Références [getBalleCoteTable\(\)](#), [getBalleCoteTablePrec\(\)](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), et [m_iBallesHorsTable](#).

Référencé par [Clhm :finirSeance\(\)](#).

```
{
    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = false;

    if (getBalleCoteTablePrec() == false && getBalleCoteTable() == false)
    {
        qDebug() << Q_FUNC_INFO << " La derniere balle n'as jamais ete renvoyé"
        ;
        m_iBallesHorsTable++;
    }
}
```

8.3.2.3 bool CTable :getBalleCoteTable ()

Références [m_bBalleCoteTable](#).

Référencé par [balleEnJeu\(\)](#), et [finirSeance\(\)](#).

```
{return m_bBalleCoteTable;}
```

8.3.2.4 bool CTable :getBalleCoteTablePrec ()

*

Références [m_bBalleCoteTablePrec](#).

Référencé par [balleEnJeu\(\)](#), et [finirSeance\(\)](#).

```
{return m_bBalleCoteTablePrec;}
```

8.3.2.5 int CTable :getBallesBonnes ()

Références [m_iBallesBonnes](#).

Référencé par [Clhm :finirSeance\(\)](#).

```
{ return m_iBallesBonnes; }
```

8.3.2.6 int CTable :getBallesEnchainees ()

*

Références [m_iBallesEnchainees](#).

Référencé par [Clhm :finirSeance\(\)](#).

```
{ return m_iBallesEnchainees; }
```

8.3.2.7 int CTable : :getBallesHorsTable ()

*

Références [m_iBallesHorsTable](#).Référéncé par [Clhm : :rafraichirStats\(\)](#).

```
{ return m_iBallesHorsTable; }
```

8.3.2.8 int CTable : :getBallesMaximum ()

*

Références [m_iBallesMaximum](#).Référéncé par [Clhm : :ballesTotalSurBallesMaximum\(\)](#).

```
{ return m_iBallesMaximum; }
```

8.3.2.9 int CTable : :getBallesObjectif ()

*

Références [m_iBallesDansZone](#), et [m_iZoneObjectif](#).Référéncé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iBallesDansZone[m_iZoneObjectif]; }
```

8.3.2.10 int CTable : :getBallesTotal ()

*

Références [m_iBallesTotal](#).Référéncé par [Clhm : :ballesTotalSurBallesMaximum\(\)](#), [Clhm : :finirSeance\(\)](#), et [Clhm : :rafraichirStats\(\)](#).

```
{ return m_iBallesTotal; }
```

8.3.2.11 uint8_t CTable : :getZoneObjectif ()

*

Paramètres

<i>nombre</i>	de balles en INT
---------------	------------------

Références [m_iZoneObjectif](#).Référéncé par [Clhm : :finirSeance\(\)](#).

```
{ return m_iZoneObjectif; }
```

8.3.2.12 int CTable :getZoneToucheePrec ()

*

8.3.2.13 bool CTable :impacterZone (uint8_t numeroZone)

*

Paramètres

<i>usuellement</i>	getRatioFenetreY() de Clhm
--------------------	----------------------------

Références [CSS_FOND_ACTIF](#), [CSS_FOND_RATE](#), [DELAI_COUP](#), [m_bBalleCoteTable](#), [m_bBalleCoteTablePrec](#), [m_fontBig](#), [m_iBallesBonnes](#), [m_iBallesDansZone](#), [m_iBallesEnchainees](#), [m_iBallesEnchaineesMax](#), [m_iBallesHorsTable](#), [m_iBallesTotal](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [Clhm :impacterZone\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    if (numeroZone >= NB_ZONES || numeroZone == m_iZoneRobot)
        return false;

    if (m_iBallesBonnes + m_iBallesHorsTable >= m_iBallesTotal)
        return false;

    m_bBalleCoteTablePrec = m_bBalleCoteTable;
    m_bBalleCoteTable = true;

    qDebug() << Q_FUNC_INFO << numeroZone;
    m_iZoneToucheePrec = m_iZoneTouchee;
    m_iZoneTouchee = numeroZone;

    m_iBallesBonnes++;
    m_iBallesDansZone[numeroZone]++;

    rafraichirNbBallesZone();

    if (numeroZone != ZONE_AUCUNE)
    {
        if (numeroZone == m_iZoneObjectif || m_iZoneObjectif == ZONE_AUCUNE)
        {
            m_pZones[numeroZone]->setStyleSheet(CSS_FOND_ACTIF);
            m_iBallesEnchainees++;
        }
        else
        {
            m_pZones[numeroZone]->setStyleSheet(CSS_FOND_RATE);
            m_iBallesEnchainees = 0;
        }

        m_pZones[numeroZone]->setFont(m_fontBig);
    }
    else
        m_iBallesEnchainees = 0;

    if (m_iBallesEnchainees > m_iBallesEnchaineesMax)
        m_iBallesEnchaineesMax = m_iBallesEnchainees;

    m_iZoneTouchee = numeroZone;
    QTimer::singleShot(DELAI_COUP, this, SLOT(rafraichirInactif()));

    return true;
}
```


8.3.2.14 void CTable : :rafraichirCSS (float ratio)

*

Paramètres

<i>usuellement</i>	getRatioFenetreY() de Clhm
--------------------	----------------------------

Références [CSS_FOND_INACTIF](#), [m_font](#), [m_fontBig](#), [m_fontOverlay](#), [m_pZones](#), [NB_ZONES](#), [TAILLE_OVERLAY](#), [TAILLE_TEXTE](#), [TAILLE_TEXTE_NB](#), et [TAILLE_TEXTE_NB_BIG](#).

Référencé par [Clhm : :rafraichirCSS\(\)](#).

```
{
    m_font.setPointSize((int)(TAILLE_TEXTE_NB*ratio));
    m_font.setBold(true);
    m_fontBig.setPointSize((int)(TAILLE_TEXTE_NB_BIG*ratio));
    m_fontBig.setBold(true);
    m_fontOverlay.setPointSize((int)(TAILLE_OVERLAY*ratio));
    qDebug() << Q_FUNC_INFO << (TAILLE_TEXTE*ratio);

    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        m_pZones[i]->setFont(m_font);
        m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
    }
    //m_pOverlayText->setFont(m_fontOverlay);
}
```

8.3.2.15 void CTable : :rafraichirInactif () [slot]

Références [CSS_FOND_INACTIF](#), [CSS_FOND_OBJECTIF](#), [m_font](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [m_pZones](#), et [NB_ZONES](#).

Référencé par [impacterZone\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        if (i != m_iZoneRobot)
        {
            if (i == m_iZoneObjectif)
                m_pZones[i]->setStyleSheet(CSS_FOND_OBJECTIF);
            else
                m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
        }
        m_pZones[i]->setFont(m_font);
    }
}
```

8.3.2.16 void CTable : :rafraichirNbBallesZone () [private]

Références [m_iBallesDansZone](#), [m_iBallesTotal](#), [m_iZoneRobot](#), [m_pZones](#), et [NB_ZONES](#).

Référencé par [impacterZone\(\)](#), [resetNbBallesZone\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

```
{
```

```

qDebug() << Q_FUNC_INFO;
for(uint8_t i=0; i < NB_ZONES; i++)
{
    if (i != m_iZoneRobot)
    {
        if (m_iBallesTotal)
            m_pZones[i]->setText(QString::number(((double(m_iBallesDansZone
[i])/double(m_iBallesTotal)*100)), 'f', 0) + '%');
        else
            m_pZones[i]->setText(QString::number(0) + '%');
    }
}
}

```

8.3.2.17 void CTable : :resetNbBallesZone ()

*

Références [m_iBallesDansZone](#), [NB_ZONES](#), et [rafraichirNbBallesZone\(\)](#).

Référencé par [resetSeance\(\)](#), et [resetStatistiques\(\)](#).

```

{
    for(uint8_t i=0; i < NB_ZONES; i++)
    {
        m_iBallesDansZone[i] = 0;
        // m_pZones[i]->setText(QString::number(0) + '%');
        // m_pZones[i]->setStyleSheet(CSS_FOND_INACTIF);
    }
    rafraichirNbBallesZone();
}

```

8.3.2.18 void CTable : :resetSeance ()

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [DEV_BALLESMAX](#), [m_args](#), [m_iBallesEnchainees](#), [m_iBallesEnchaineesMax](#), [m_iBallesMaximum](#), [m_iZoneObjectif](#), [m_iZoneRobot](#), [resetNbBallesZone\(\)](#), [resetStatistiques\(\)](#), [setBallesMaximum\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [CTable\(\)](#), et [CIhm : :resetSeance\(\)](#).

```

{
    m_iBallesMaximum      = 0;
    m_iBallesEnchainees   = 0;
    m_iBallesEnchaineesMax = 0;
    m_iZoneRobot = m_iZoneObjectif = ZONE_AUCUNE;
    resetNbBallesZone();

    resetStatistiques();

    if (m_args.contains("-dev"))
        setBallesMaximum(DEV_BALLESMAX);
}

```

8.3.2.19 void CTable : :resetStatistiques ()

*

Références `m_bBalleCoteTable`, `m_bBalleCoteTablePrec`, `m_iBallesBonnes`, `m_iBallesEnchainees`, `m_iBallesHorsTable`, `m_iBallesTotal`, `m_iZoneTouchee`, `m_iZoneToucheePrec`, `resetNbBallesZone()`, et `ZONE_AUCUNE`.

Référencé par `Clhm : :commencerSeance()`, et `resetSeance()`.

```
{
    m_iZoneToucheePrec = m_iZoneTouchee = ZONE_AUCUNE;
    m_iBallesBonnes      = 0;
    m_iBallesTotal       = 0;
    m_iBallesEnchainees  = 0;
    m_iBallesHorsTable   = 0;
    m_bBalleCoteTable     = true;
    m_bBalleCoteTablePrec = true;
    resetNbBallesZone();
}
```

8.3.2.20 void CTable : :setBallesMaximum (int nb)

*

Références `m_iBallesMaximum`.

Référencé par `Clhm : :gererArguments()`, `resetSeance()`, et `Clhm : :setBallesMaximum()`.

```
{ m_iBallesMaximum = nb; }
```

8.3.2.21 void CTable : :setFiletTaille (float ratio)

Références `HAUTEUR_FILET`, `m_pFilet`, et `WIDGET_SIZE_MAX`.

Référencé par `Clhm : :initialisationFenetre()`.

```
{
    m_pFilet->setMinimumSize(QSize(4, (float)HAUTEUR_FILET*ratio));
    m_pFilet->setMaximumSize(QSize(WIDGET_SIZE_MAX, (float)HAUTEUR_FILET*ratio));
    qDebug() << Q_FUNC_INFO << ((float)HAUTEUR_FILET*ratio);
}
```

8.3.2.22 void CTable : :setLayerEcran (uint8_t layer, float tailleFenetreY)

Références `LAYER_RECAP`, `m_font`, `m_pZones`, `NB_ZONES`, et `TAILLE_TEXTE_NB`.

Référencé par `Clhm : :setLayerEcran()`.

```
{
    qDebug() << Q_FUNC_INFO << " layer: " << layer << "tailleFenetreY: " <<
        tailleFenetreY;
    int tailleTexte;

    if (layer == LAYER_RECAP)
        tailleTexte = TAILLE_TEXTE_NB*(tailleFenetreY/1.75);
    else
        tailleTexte = TAILLE_TEXTE_NB*(tailleFenetreY);

    m_font.setPointSize((int)tailleTexte);

    for(uint8_t i=0; i < NB_ZONES; i++)
```

```

    {
        m_pZones[i]->setFont(m_font);
    }
}

```

8.3.2.23 void CTable : :setZoneObjectif (uint8_t zone)

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [CSS_FOND_INACTIF](#), [CSS_FOND_OBJECTIF](#), [m_iZoneObjectif](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [Clhm : :setZoneObjectif\(\)](#), et [Clhm : :setZoneObjectifRandom\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    if ((numeroZone >= NB_ZONES && numeroZone != ZONE_AUCUNE))
    {
        qDebug() << "!!\\ Erreur Zone !!\\";
        return;
    }
    if (m_iZoneObjectif < NB_ZONES)
        m_pZones[m_iZoneObjectif]->setStyleSheet(CSS_FOND_INACTIF);

    m_iZoneObjectif = numeroZone;

    if (m_iZoneTouchee == numeroZone)
        m_iZoneTouchee = ZONE_AUCUNE;
    if (m_iZoneToucheePrec == numeroZone)
        m_iZoneToucheePrec = ZONE_AUCUNE;

    if (m_iZoneObjectif < NB_ZONES)
    {
        m_pZones[numeroZone]->setStyleSheet(CSS_FOND_OBJECTIF);
    }

    rafraichirNbBallesZone();
}

```

8.3.2.24 void CTable : :setZoneRobot (uint8_t zone)

*

Paramètres

<i>enum</i>	zones_e, voir const.h
-------------	---------------------------------------

Références [CSS_FOND_INACTIF](#), [CSS_FOND_ROBOT](#), [m_iZoneRobot](#), [m_iZoneTouchee](#), [m_iZoneToucheePrec](#), [m_pZones](#), [NB_ZONES](#), [rafraichirNbBallesZone\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [Clhm : :setZoneRobot\(\)](#), et [Clhm : :setZoneRobotRandom\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    if ((numeroZone >= NB_ZONES && numeroZone != ZONE_AUCUNE))

```

```

{
    qDebug() << "!\\ Erreur Zone !\\";
    return;
}

if (m_iZoneRobot < NB_ZONES)
    m_pZones[m_iZoneRobot]->setStyleSheet(CSS_FOND_INACTIF);

m_iZoneRobot = numeroZone;

if (m_iZoneTouchee == numeroZone)
    m_iZoneTouchee = ZONE_AUCUNE;
if (m_iZoneToucheePrec == numeroZone)
    m_iZoneToucheePrec = ZONE_AUCUNE;

if (m_iZoneRobot < NB_ZONES)
{
    m_pZones[numeroZone]->setStyleSheet(CSS_FOND_ROBOT);
    m_pZones[numeroZone]->setText("ROBOT");
}

    rafraichirNbBallesZone();
}

```

8.3.3 Documentation des données membres

8.3.3.1 QStringList CTable : :m_args

Référencé par [CTable\(\)](#), [Clhm : :gererArguments\(\)](#), et [resetSeance\(\)](#).

8.3.3.2 bool CTable : :m_bBalleCoteTable [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBalleCoteTable\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.3 bool CTable : :m_bBalleCoteTablePrec [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBalleCoteTablePrec\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.4 QString CTable : :m_fondActif [private]

8.3.3.5 QString CTable : :m_fondInactif [private]

8.3.3.6 QString CTable : :m_fondObjectif [private]

8.3.3.7 QString CTable : :m_fondRate [private]

8.3.3.8 QString CTable : :m_fondRobot [private]

8.3.3.9 QFont CTable : :m_font [private]

Référencé par [CTable\(\)](#), [rafraichirCSS\(\)](#), [rafraichirInactif\(\)](#), et [setLayerEcran\(\)](#).

8.3.3.10 QFont CTable : :m_fontBig [private]

Référencé par [impacterZone\(\)](#), et [rafraichirCSS\(\)](#).

8.3.3.11 QFont CTable : :m_fontOverlay [private]

Référencé par [rafraichirCSS\(\)](#).

8.3.3.12 int CTable : :m_iBallesBonnes [private]

Référencé par [getBallesBonnes\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.13 int CTable : :m_iBallesDansZone[NB_ZONES] [private]

Référencé par [getBallesObjectif\(\)](#), [impacterZone\(\)](#), [rafraichirNbBallesZone\(\)](#), et [resetNbBallesZone\(\)](#).

8.3.3.14 int CTable : :m_iBallesEnchainees [private]

Référencé par [getBallesEnchainees\(\)](#), [impacterZone\(\)](#), [resetSeance\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.15 int CTable : :m_iBallesEnchaineesMax [private]

Référencé par [impacterZone\(\)](#), et [resetSeance\(\)](#).

8.3.3.16 int CTable : :m_iBallesHorsTable [private]

Référencé par [balleEnJeu\(\)](#), [finirSeance\(\)](#), [getBallesHorsTable\(\)](#), [impacterZone\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.17 int CTable : :m_iBallesMaximum [private]

Référencé par [getBallesMaximum\(\)](#), [resetSeance\(\)](#), et [setBallesMaximum\(\)](#).

8.3.3.18 int CTable : :m_iBallesTotal [private]

Référencé par [balleEnJeu\(\)](#), [getBallesTotal\(\)](#), [impacterZone\(\)](#), [rafraichirNbBallesZone\(\)](#), et [resetStatistiques\(\)](#).

8.3.3.19 int CTable : :m_iZoneObjectif [private]

Référencé par [getBallesObjectif\(\)](#), [getZoneObjectif\(\)](#), [impacterZone\(\)](#), [rafraichirInactif\(\)](#), [resetSeance\(\)](#), et [setZoneObjectif\(\)](#).

8.3.3.20 int CTable : :m_iZoneRobot [private]

Référencé par [impacterZone\(\)](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), [resetSeance\(\)](#), et [setZoneRobot\(\)](#).

8.3.3.21 int CTable : :m_iZoneTouchee [private]

Référencé par [impacterZone\(\)](#), [resetStatistiques\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

8.3.3.22 `int CTable : :m_iZoneToucheePrec` [private]

Référencé par [impacterZone\(\)](#), [resetStatistiques\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

8.3.3.23 `QLabel* CTable : :m_pFilet` [private]

Référencé par [CTable\(\)](#), et [setFiletTaille\(\)](#).

8.3.3.24 `QGridLayout* CTable : :m_pGridLayout` [private]

*

Référencé par [CTable\(\)](#).

8.3.3.25 `QLabel* CTable : :m_pOverlay` [private]

Référencé par [CTable\(\)](#).

8.3.3.26 `QLabel* CTable : :m_pOverlayText`

8.3.3.27 `QVector<QLabel*> CTable : :m_pZones` [private]

Référencé par [CTable\(\)](#), [impacterZone\(\)](#), [rafraichirCSS\(\)](#), [rafraichirInactif\(\)](#), [rafraichirNbBallesZone\(\)](#), [setLayerEcran\(\)](#), [setZoneObjectif\(\)](#), et [setZoneRobot\(\)](#).

La documentation de cette classe a été générée à partir des fichiers suivants :

- [table.h](#)
- [table.cpp](#)

8.4 Référence de la classe CTrame

```
#include <trame.h>
```

Connecteurs publics

- bool [traiterTrame](#) (QString donneesRecues)
voir [quitter\(\)](#) de [Clhm](#)
- bool [gererTrame](#) (QString donneesRecues)
découpe les trames de la reception

Signaux

- void [setInfoConnect](#) (QString nom)
- void [setLayerEcran](#) (uint8_t layer)
voir [setInfoConnect\(QString nom\)](#) de [Clhm](#)
- void [commencerSeance](#) ()
voir [setLayerEcran\(int layer\)](#) de [Clhm](#)
- void [pauserSeance](#) ()
voir [commencerSeance\(\)](#) de [Clhm](#)
- void [reprendreSeance](#) ()
voir [pauserSeance\(\)](#) de [Clhm](#)
- void [finirSeance](#) ()

- voir [reprenreSeance\(\)](#) de [Clhm](#)
- void [resetSeance](#) ()
 - voir [finirSeance\(\)](#) de [Clhm](#)
- void [impacterZone](#) (uint8_t zone)
 - voir [resetSeance\(\)](#) de [Clhm](#)
- void [balleEnJeu](#) ()
 - voir [impacterZone\(int zone\)](#) de [Clhm](#)
- void [setZoneRobot](#) (uint8_t zone)
 - voir [balleEnJeu\(\)](#) de [Clhm](#)
- void [setZoneObjectif](#) (uint8_t zone)
 - voir [setZoneRobot\(int zone\)](#) de [Clhm](#)
- void [setBallesMaximum](#) (int balles)
 - voir [setZoneObjectif\(int zone\)](#) de [Clhm](#)
- void [setFrequenceRobot](#) (float freq)
 - voir [setBalleMaximum\(int balles\)](#) de [Clhm](#)
- void [rafraichirCSS](#) ()
 - voir [setFrequenceRobot\(int freq\)](#) de [CTable](#)
- void [quitter](#) ()
 - voir [rafraichirCSS\(\)](#) de [Clhm](#)

Fonctions membres publiques

- [CTrame](#) (QObject *parent=0)
- [~CTrame](#) ()

Fonctions membres privées

- QString [extraireElement](#) (QString donneesRecues, const int element)
 - extrait les elements de la trame avec [extraireElement\(QString donneesRecues\)](#) puis effec-
tue les signals en fonction*
- int [getTrameLength](#) (QString donneesRecues)
 - Découpe la trame et retourne l'élément.*
- void [messageNonReconnu](#) (QString donneesRecues, int element)
 - Retourne la longueur de la trame.*
- bool [gererTramesSansParametre](#) (QString donneesRecues)
 - Affiche les chars dans l'element demandé par rapport a la liste d'elements de la trame.*
- bool [gererTrames1Parametre](#) (QString donneesRecues)
 - verrifie, identifie et execute les trames ne possédant pas de parametre*

8.4.1 Documentation des constructeurs et destructeur

8.4.1.1 CTrame : :CTrame (QObject * parent = 0)

```

                                : QObject (parent)
{
}

```

8.4.1.2 CTrame : :~CTrame ()

```

{
}

```

8.4.2 Documentation des fonctions membres

8.4.2.1 void CTrame : :balleEnJeu () [signal]

*

Référencé par [gererTrames1Parametre\(\)](#), et [gererTramesSansParametre\(\)](#).

8.4.2.2 void CTrame : :commencerSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.3 QString CTrame : :extraireElement (QString *donneesRecues*, const int *element*) [private]

*

Paramètres

<i>Trame</i>	en QString
--------------	------------

Renvoie

validité de la trame

Référencé par [gererTrame\(\)](#), [gererTrames1Parametre\(\)](#), [gererTramesSansParametre\(\)](#), et [messageNonReconnu\(\)](#).

```
{
    if(donneesRecues.isEmpty())
        return QString();

    QStringList listeElements;
    listeElements = donneesRecues.split(":");

    return listeElements.at(iElement).trimmed();
}
```

8.4.2.4 void CTrame : :finirSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.5 bool CTrame : :gererTrame (QString *donneesRecues*) [slot]

*

Références [extraireElement\(\)](#), [gererTrames1Parametre\(\)](#), [gererTramesSansParametre\(\)](#), [getTrameLength\(\)](#), [setBallesMaximum\(\)](#), [setZoneObjectif\(\)](#), [setZoneRobot\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [traiterTrame\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << QThread::currentThreadId() << this <<
        QString::fromUtf8("Données reçues : ") << donneesRecues;
```

```

bool trameValide = true;

if (getTrameLength(donneesRecues) == 2) // sans arguments
    trameValide = gererTramesSansParametre(donneesRecues);

else if (getTrameLength(donneesRecues) == 3) // 1 arguments
    trameValide = gererTrames1Parametre(donneesRecues);

// $TTPA:SETSEANCE:[POS_ROBOT]:[POS_OBJECTIF]:[NB_BALLES_MAX]:[FREQ_ENVOI]
else if (getTrameLength(donneesRecues) == 5 && extraireElement(
    donneesRecues,1).startsWith("SETSEANCE"))
{
    if ((extraireElement(donneesRecues,2)).toInt() <= 0) // AUCUN dans
        le cas d'un negatif ou zero
        emit setZoneRobot(ZONE_AUCUNE);
    else
        emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);

    if ((extraireElement(donneesRecues,3)).toInt() <= 0) // AUCUN dans
        le cas d'un negatif ou zero
        emit setZoneObjectif(ZONE_AUCUNE);
    else
        emit setZoneObjectif((extraireElement(donneesRecues,3)).toInt() - 1);

    emit setBallesMaximum((extraireElement(donneesRecues,4)).toInt());
}
else
{
    qDebug() << Q_FUNC_INFO << "!!\\ TRAME NON RECONNUE !!\\";
    trameValide = false;
}
return trameValide;
}

```

8.4.2.6 bool CTrame : :gererTrames1Parametre (QString *donneesRecues*) [private]

*

Références [balleEnJeu\(\)](#), [extraireElement\(\)](#), [impacterZone\(\)](#), [messageNonReconnu\(\)](#), [setBallesMaximum\(\)](#), [setInfoConnect\(\)](#), [setZoneObjectif\(\)](#), [setZoneRobot\(\)](#), et [ZONE_AUCUNE](#).

Référencé par [gererTrame\(\)](#).

```

{
    bool trameValide = true;

    // $TTPA:CONNECT:[nom]
    if (extraireElement(donneesRecues,1).startsWith("CONNECT"))
    {
        emit setInfoConnect(extraireElement(donneesRecues,2));
    }

    // $TTPA:SETROBOT:[POS_ROBOT]
    else if (extraireElement(donneesRecues,1).startsWith("SETROBOT"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0) // Adaptation
            au systeme de la table
            emit setZoneRobot(ZONE_AUCUNE);
        else
            emit setZoneRobot((extraireElement(donneesRecues,2)).toInt() - 1);
    }

    // $TTPA:SETOBJECTIF:[POS_OBJECTIF]
    else if (extraireElement(donneesRecues,1).startsWith("SETOBJECTIF"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0) // Adaptation
            au systeme de la table
            emit setZoneObjectif(ZONE_AUCUNE);
    }
}

```

```

        else
            emit setZoneObjectif((extraireElement(donneesRecues,2)).toInt() - 1
        );
    }
    // $TTPA:SETPALLESMAX:[BALLES]
    else if (extraireElement(donneesRecues,1).startsWith("SETPALLESMAX"))
    {
        emit setBallesMaximum((extraireElement(donneesRecues,2)).toInt());
    }
    // $TTPA:IMPACT:[X]
    else if (extraireElement(donneesRecues,1).startsWith("IMPACT"))
    {
        if ((extraireElement(donneesRecues,2)).toInt() <= 0) //
            Adaptation au systeme de la table
            emit balleEnJeu();
        else
            emit impacterZone((extraireElement(donneesRecues,2)).toInt() - 1);
    }
    else
    {
        messageNonReconnu(donneesRecues,1);
        trameValide = false;
    }
    return trameValide;
}

```

8.4.2.7 bool CTrame : :gererTramesSansParametre (QString donneesRecues) [private]

*

Paramètres

<i>Trame</i>	en QString, element en int
--------------	----------------------------

Références [balleEnJeu\(\)](#), [commencerSeance\(\)](#), [extraireElement\(\)](#), [finirSeance\(\)](#), [messageNonReconnu\(\)](#), [pauserSeance\(\)](#), [quitter\(\)](#), [reprendreSeance\(\)](#), et [resetSeance\(\)](#).

Référencé par [gererTrame\(\)](#).

```

{
    bool trameValide = true;
    // $TTPA:JOUER
    if (extraireElement(donneesRecues,1).startsWith("JOUER"))
    {
        emit balleEnJeu();
    }
    // $TTPA:START
    else if (extraireElement(donneesRecues,1).startsWith("START"))
    {
        emit commencerSeance();
    }
    // $TTPA:FINSEANCE
    else if (extraireElement(donneesRecues,1).startsWith("FINSEANCE"))
    {
        emit finirSeance();
    }
    // $TTPA:PAUSE
    else if (extraireElement(donneesRecues,1).startsWith("PAUSE"))
    {
        emit pauserSeance();
    }
    // $TTPA:RESUME
    else if (extraireElement(donneesRecues,1).startsWith("RESUME"))
    {
        emit reprendreSeance();
    }
}

```

```

    }
    //TTPA:RESET
    else if (extraireElement(donneesRecues,1).startsWith("RESET"))
    {
        emit resetSeance();
    }
    //TTPA:QUIT
    else if (extraireElement(donneesRecues,1).startsWith("QUIT"))
    {
        emit quitter();
    }
    else
    {
        messageNonReconnu(donneesRecues,1);
        trameValide = false;
    }
    return trameValide;
}

```

8.4.2.8 int CTrame : :getTrameLength (QString *donneesRecues*) [private]

*

Paramètres

<i>Trame</i>	en QString, index de l'élément
--------------	--------------------------------

Référencé par [gererTrame\(\)](#).

```

{
    if(donneesRecues.isEmpty())
        return 0;

    QStringList listeElements;
    listeElements = donneesRecues.split(":");

    return listeElements.length();
}

```

8.4.2.9 void CTrame : :impacterZone (uint8_t *zone*) [signal]

*

Référencé par [gererTrames1Parametre\(\)](#).

8.4.2.10 void CTrame : :messageNonReconnu (QString *donneesRecues*, int *element*) [private]

*

Paramètres

<i>Trame</i>	en QString
--------------	------------

Références [extraireElement\(\)](#).

Référencé par [gererTrames1Parametre\(\)](#), et [gererTramesSansParametre\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << "!!\\ TRAME NON RECONNUE /!\\";
}

```

```

QString testChars = " [Element " + QString::number(element) + "]: [|";

for(int i=0; i<extraireElement(donneesRecues,element).length();i++)
{
    testChars += (extraireElement(donneesRecues,element)[i]) + "|";
}
testChars+="]";
QDebug() << testChars;
}

```

8.4.2.11 void CTFrame : :pauserSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.12 void CTFrame : :quitter () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.13 void CTFrame : :rafraichirCSS () [signal]

*

8.4.2.14 void CTFrame : :reprendreSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.15 void CTFrame : :resetSeance () [signal]

*

Référencé par [gererTramesSansParametre\(\)](#).

8.4.2.16 void CTFrame : :setBallesMaximum (int *balles*) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

8.4.2.17 void CTFrame : :setFrequenceRobot (float *freq*) [signal]

*

8.4.2.18 void CTFrame : :setInfoConnect (QString *nom*) [signal]

Référencé par [gererTrames1Parametre\(\)](#).

8.4.2.19 void CTFrame : :setLayerEcran (uint8_t *layer*) [signal]

*

8.4.2.20 void CFrame : :setZoneObjectif (uint8_t zone) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

8.4.2.21 void CFrame : :setZoneRobot (uint8_t zone) [signal]

*

Référencé par [gererTrame\(\)](#), et [gererTrames1Parametre\(\)](#).

8.4.2.22 bool CFrame : :traiterTrame (QString donneesRecues) [slot]

*

Références [gererTrame\(\)](#).

Référencé par [Clhm : :envoyerCommande\(\)](#).

```
{
    if(donneesRecues.isEmpty())
        return false;

    bool retour = false;

    QStringList listeElements;
    listeElements = donneesRecues.split("$",QString::SkipEmptyParts);

    for(uint8_t i = 0; i < listeElements.length(); i++)
    {
        if (listeElements.at(i).startsWith("TPA:"))
            retour = gererTrame("$" + listeElements.at(i));
    }
    return retour;
}
```

La documentation de cette classe a été générée à partir des fichiers suivants :

- [trame.h](#)
- [trame.cpp](#)

9 Documentation des fichiers

9.1 Référence du fichier Changelog.dox

9.2 Référence du fichier communicationbluetooth.cpp

Définition de la classe [CommunicationBluetooth](#).

```
#include "communicationbluetooth.h"
```

9.2.1 Description détaillée

9.3 Référence du fichier communicationbluetooth.h

Déclaration de la classe [CommunicationBluetooth](#).

```
#include <QtCore> #include <QApplication> #include "Qext-
SerialPort/qextserialport.h" #include "const.h"
```

Classes

- class [CommunicationBluetooth](#)
Assure la réception des trames via le Bluetooth.

Macros

- #define [PERIODE_SURVEILLANCE](#) 200
durée de la temporisation périodique en ms

9.3.1 Description détaillée

9.3.2 Documentation des macros

9.3.2.1 #define PERIODE_SURVEILLANCE 200

9.4 Référence du fichier const.h

Macros

- #define [WIDGET_SIZE_MAX](#) 16777215
- #define [TAILLE_FENETRE_DEFAULT_WIDTH](#) 960
- #define [TAILLE_FENETRE_DEFAULT_HEIGHT](#) 540
- #define [RATIO_ENTETE](#) 10
- #define [TAILLE_TEXTE](#) 20
- #define [TAILLE_TEXTE_SMALL](#) 10
- #define [TAILLE_TEXTE_NORMAL](#) 30
- #define [TAILLE_TEXTE_BIG](#) 42
- #define [TAILLE_TEXTE_NOM](#) 25
- #define [DELAI_FIXFENETRE](#) 200
- #define [DEV_BALLESMAX](#) 30
- #define [IHM_BALLESENVOYEEES](#) ""
- #define [IHM_NOMDETEST](#) QString : :fromUtf8("Simon GAUZY")
- #define [IHM_PERIPHERIQUEDETEST](#) QString : :fromUtf8("Périphérique_de_ demonstration")
- #define [NB_ZONES](#) 9
- #define [BALLEES_MAX_DEFAULT](#) 20
- #define [DELAI_COUP](#) 400
- #define [HAUTEUR_FILET](#) 20
- #define [TAILLE_TEXTE_NB](#) 30
- #define [TAILLE_TEXTE_NB_BIG](#) 42
- #define [TAILLE_OVERLAY](#) 128
- #define [TABLE_STAT1](#) QString : :fromUtf8("Hors Table :")
- #define [LOGO_ATTENTECONNECTION](#) "ATTENTE DE CONNEXION"
- #define [LOGO_ATTENTECONFIGURATION](#) QString : :fromUtf8("ATTENTE DE C- ONFIGURATION DE LA SÉANCE")
- #define [LOGO_ATTENTEIDENTIFICATION](#) "ATTENTE D'IDENTIFICATION DU J- OUEUR"
- #define [LOGO_JOUEUR_CONNECTE](#) QString : :fromUtf8(" est connecté")
- #define [RECAP_TITRE_TEXTE](#) QString : :fromUtf8("FIN DE SÉANCE")
- #define [RECAP_STAT1](#) QString : :fromUtf8("Balles Dans L'Objectif :")
- #define [RECAP_STAT1_ALT](#) QString : :fromUtf8("Balles Renvoyées :")
- #define [RECAP_STAT2](#) QString : :fromUtf8("Balles Hors Table :")

```

- #define RECAP_STAT3 QString : :fromUtf8("Série Maximale :")
- #define RECAP_STAT4 QString : :fromUtf8("")
- #define CSS_TIMER_ON QString : :fromUtf8("QLabel{color : #B08000 ;}")
- #define CSS_TIMER_OFF QString : :fromUtf8("QLabel{color : #A00000 ;}")
- #define CSS_TIMER_RES QString : :fromUtf8("QLabel{color : #00A000 ;}")
- #define CSS_FOND_INACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color :
  rgba(50, 150, 255, 0);\\nborder : 3px solid rgba(255,255,255,30);\\ncolor : #FFFFFF-
  F;\\n}")
- #define CSS_FOND_ACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color :
  rgb(0, 150, 50);\\nborder : 3px solid #00FF00;\\ncolor : #FFFFFF;\\n}")
- #define CSS_FOND_RATE QString : :fromUtf8("QLabel\\n{\\nbackground-color :
  rgb(175, 50, 25);\\nborder : 3px solid #FF0000;\\ncolor : #FFFFFF;\\n}")
- #define CSS_FOND_ROBOT QString : :fromUtf8("QLabel\\n{\\nbackground-color :
  rgba(0, 0, 0, 120);\\nborder : 3px solid #000000;\\ncolor : #00FF00;\\n}")
- #define CSS_FOND_OBJECTIF QString : :fromUtf8("QLabel\\n{\\nbackground-
  color : rgba(200, 160, 30, 120);\\nborder : 3px solid #FFEE00;\\ncolor : #FFF-
  F77;\\n}")

```

Énumérations

```

- enum zones_e { ZONE_HAUTGAUCHE = 0, ZONE_HAUTMILIEU, ZONE_HAUT-
  DROITE, ZONE_MILIEUGAUCHE, ZONE_MILIEUMILIEU, ZONE_MILIEUDROITE,
  ZONE_BASGAUCHE, ZONE_BASMILIEU, ZONE_BASDROITE, ZONE_ENJEU =
  15, ZONE_AUCUNE = 20 }
  Enumeration des zones de la table coté robot.
- enum layer_e { LAYER_LOGO = 0, LAYER_TABLE = 1, LAYER_RECAP = 2 }
  Enumeration des fenetres de l'IHM.
- enum etatRFCOMM_e { RFCOMM_ARRETE = 0, RFCOMM_CONNECTE, RFCO-
  MM_FERME }
  Enumeration des etats possible du port RFCOMM.

```

9.4.1 Documentation des macros

9.4.1.1 #define BALLES_MAX_DEFAULT 20

9.4.1.2 #define CSS_FOND_ACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color :
 rgb(0, 150, 50);\\nborder : 3px solid #00FF00;\\ncolor : #FFFFFF;\\n}")

Référencé par CTable : :impacterZone().

9.4.1.3 #define CSS_FOND_INACTIF QString : :fromUtf8("QLabel\\n{\\nbackground-color :
 rgba(50, 150, 255, 0);\\nborder : 3px solid rgba(255,255,255,30);\\ncolor : #FFFFFF;\\n}")

Référencé par CTable : :CTable(), CTable : :rafraichirCSS(), CTable : :rafraichirInactif(),
 CTable : :setZoneObjectif(), et CTable : :setZoneRobot().

9.4.1.4 #define CSS_FOND_OBJECTIF QString : :fromUtf8("QLabel\\n{\\nbackground-
 color : rgba(200, 160, 30, 120);\\nborder : 3px solid #FFEE00;\\ncolor :
 #FFFF77;\\n}")

Référencé par CTable : :rafraichirInactif(), et CTable : :setZoneObjectif().

9.4.1.5 `#define CSS_FOND_RATE QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgb(175, 50, 25);\\nborder : 3px solid #FF0000;\\ncolor : #FFFFFF;\\n}")`

Référencé par `CTable : :impacterZone()`.

9.4.1.6 `#define CSS_FOND_ROBOT QString : :fromUtf8("QLabel\\n{\\nbackground-color : rgba(0, 0, 0, 120);\\nborder : 3px solid #000000;\\ncolor : #00FF00;\\n}")`

Référencé par `CTable : :setZoneRobot()`.

9.4.1.7 `#define CSS_TIMER_OFF QString : :fromUtf8("QLabel{color : #A00000;}")`

Référencé par `Clhm : :pauserSeance()`.

9.4.1.8 `#define CSS_TIMER_ON QString : :fromUtf8("QLabel{color : #B08000;}")`

Référencé par `Clhm : :commencerSeance()`, et `Clhm : :rafraichirTimerSeance()`.

9.4.1.9 `#define CSS_TIMER_RES QString : :fromUtf8("QLabel{color : #00A000;}")`

Référencé par `Clhm : :repandreSeance()`.

9.4.1.10 `#define DELAI_COUP 400`

Référencé par `CTable : :impacterZone()`.

9.4.1.11 `#define DELAI_FIXFENETRE 200`

Référencé par `Clhm : :Clhm()`.

9.4.1.12 `#define DEV_BALLES MAX 30`

Référencé par `Clhm : :gererArguments()`, et `CTable : :resetSeance()`.

9.4.1.13 `#define HAUTEUR_FILET 20`

Référencé par `CTable : :setFiletTaille()`.

9.4.1.14 `#define IHM_BALLESENVOYEEES ""`

9.4.1.15 `#define IHM_NOMDETEST QString : :fromUtf8("Simon GAUZY")`

Référencé par `Clhm : :setInfoConnectDemo()`.

9.4.1.16 `#define IHM_PERIPHERIQUEDETEST QString : :fromUtf8("Périphérique de - démonstration")`

Référencé par `Clhm : :setNomPeripheriqueDemo()`.

9.4.1.17 `#define LOGO_ATTENTECONFIGURATION QString : :fromUtf8("ATTENTE DE CONFIGURATION DE LA SÉANCE")`

Référencé par `Clhm : :connecterJoueur()`, et `Clhm : :setInfoConnect()`.

9.4.1.18 `#define LOGO_ATTENTECONNECTION "ATTENTE DE CONNEXION"`

Référencé par `Clhm : :deconnecterJoueur()`, et `Clhm : :initialisationFenetre()`.

9.4.1.19 `#define LOGO_ATTENTEIDENTIFICATION "ATTENTE D'IDENTIFICATION DU JOUEUR"`

9.4.1.20 `#define LOGO_JOUEUR_CONNECTE QString : :fromUtf8(" est connecté")`

Référencé par `Clhm : :setNomPeripherique()`.

9.4.1.21 `#define NB_ZONES 9`

Référencé par `CTable : :CTable()`, `CTable : :impacterZone()`, `CTable : :rafraichirCSS()`, `CTable : :rafraichirInactif()`, `CTable : :rafraichirNbBallesZone()`, `CTable : :resetNbBallesZone()`, `CTable : :setLayerEcran()`, `CTable : :setZoneObjectif()`, et `CTable : :setZoneRobot()`.

9.4.1.22 `#define RATIO_ENTETE 10`

Référencé par `Clhm : :initialisationFenetre()`.

9.4.1.23 `#define RECAP_STAT1 QString : :fromUtf8("Balles Dans L'Objectif :")`

Référencé par `Clhm : :finirSeance()`.

9.4.1.24 `#define RECAP_STAT1_ALT QString : :fromUtf8("Balles Renvoyées :")`

Référencé par `Clhm : :finirSeance()`.

9.4.1.25 `#define RECAP_STAT2 QString : :fromUtf8("Balles Hors Table :")`

Référencé par `Clhm : :finirSeance()`.

9.4.1.26 `#define RECAP_STAT3 QString : :fromUtf8("Série Maximale :")`

Référencé par `Clhm : :finirSeance()`.

9.4.1.27 `#define RECAP_STAT4 QString : :fromUtf8("")`

9.4.1.28 `#define RECAP_TITRE_TEXTE QString : :fromUtf8("FIN DE SÉANCE")`

Référencé par `Clhm : :initialisationFenetre()`.

9.4.1.29 `#define TABLE_STAT1 QString : :fromUtf8("Hors Table :")`

Référencé par `Clhm : :initialisationStats()`.

9.4.1.30 `#define TAILLE_FENETRE_DEFAULT_HEIGHT 540`

Référencé par `Clhm : :getRatioFenetreY()`.

9.4.1.31 `#define TAILLE_FENETRE_DEFAULT_WIDTH 960`

Référencé par [Clhm : :getRatioFenetreX\(\)](#).

9.4.1.32 `#define TAILLE_OVERLAY 128`

Référencé par [CTable : :rafraichirCSS\(\)](#).

9.4.1.33 `#define TAILLE_TEXTE 20`

Référencé par [CTable : :rafraichirCSS\(\)](#), et [Clhm : :rafraichirCSS\(\)](#).

9.4.1.34 `#define TAILLE_TEXTE_BIG 42`

9.4.1.35 `#define TAILLE_TEXTE_NB 30`

Référencé par [CTable : :rafraichirCSS\(\)](#), et [CTable : :setLayerEcran\(\)](#).

9.4.1.36 `#define TAILLE_TEXTE_NB_BIG 42`

Référencé par [CTable : :rafraichirCSS\(\)](#).

9.4.1.37 `#define TAILLE_TEXTE_NOM 25`

Référencé par [Clhm : :rafraichirCSS\(\)](#), et [Clhm : :setInfoConnect\(\)](#).

9.4.1.38 `#define TAILLE_TEXTE_NORMAL 30`

Référencé par [Clhm : :rafraichirCSS\(\)](#).

9.4.1.39 `#define TAILLE_TEXTE_SMALL 10`

Référencé par [Clhm : :rafraichirCSS\(\)](#).

9.4.1.40 `#define WIDGET_SIZE_MAX 16777215`

Référencé par [CTable : :CTable\(\)](#), et [CTable : :setFileTaille\(\)](#).

9.4.2 Documentation du type de l'énumération

9.4.2.1 `enum etatRFCOMM_e`

Valeurs énumérées :

RFCOMM_ARRETE
RFCOMM_CONNECTE
RFCOMM_FERME

```
{  
    RFCOMM_ARRETE = 0,  
    RFCOMM_CONNECTE,  
    RFCOMM_FERME  
};
```

9.4.2.2 enum layer_e

Valeurs énumérées :

LAYER_LOGO
LAYER_TABLE
LAYER_RECAP

```
{  
    LAYER_LOGO = 0,  
    LAYER_TABLE = 1,  
    LAYER_RECAP = 2  
};
```

9.4.2.3 enum zones_e

Valeurs énumérées :

ZONE_HAUTGAUCHE
ZONE_HAUTMILIEU
ZONE_HAUTDROITE
ZONE_MILIEUGAUCHE
ZONE_MILIEUMILIEU
ZONE_MILIEUDROITE
ZONE_BASGAUCHE
ZONE_BASMILIEU
ZONE_BASDROITE
ZONE_ENJEU
ZONE_AUCUNE

```
{  
    ZONE_HAUTGAUCHE = 0,  
    ZONE_HAUTMILIEU,  
    ZONE_HAUTDROITE,  
    ZONE_MILIEUGAUCHE,  
    ZONE_MILIEUMILIEU,  
    ZONE_MILIEUDROITE,  
    ZONE_BASGAUCHE,  
    ZONE_BASMILIEU,  
    ZONE_BASDROITE,  
  
    ZONE_ENJEU = 15,  
    ZONE_AUCUNE = 20  
};
```

9.5 Référence du fichier ihm.cpp

```
#include "ihm.h" #include "communicationbluetooth.h"
```

9.6 Référence du fichier ihm.h

La fenêtre principale de l'application.

```
#include <QWidget> #include <QThread> #include <QDebug> ×
#include <unistd.h> #include <stdio.h> #include "ui_ihm.-
h" #include "const.h" #include "table.h" #include "trame.-
h" #include <cstdint>
```

Classes

- class [Clhm](#)
Classe principale de l'application (IHM)

Macros

- #define [PORT_BLUETOOTH](#) "rfcomm0"

9.6.1 Description détaillée

Auteur

Racamond Adrien

Version

1.0

9.6.2 Documentation des macros

9.6.2.1 #define [PORT_BLUETOOTH](#) "rfcomm0"

Référencé par [Clhm](#) : [:Clhm\(\)](#).

9.7 Référence du fichier main.cpp

```
#include <QtGui/QApplication> #include "ihm.h"
```

Fonctions

- int [main](#) (int argc, char *argv[])

9.7.1 Documentation des fonctions

9.7.1.1 int [main](#) (int *argc*, char * *argv*[])

Référencé par [Clhm](#) : [:connecterSignaux\(\)](#).

```
{
    QApplication a(argc, argv);

    C Ihm ihm;
    ihm.show();

    return a.exec();
}
```

9.8 Référence du fichier README.dox

9.9 Référence du fichier table.cpp

```
#include "table.h"
```

9.10 Référence du fichier table.h

Classe gérant la table et les calculs lié a la seance, impacts, affichage des zones de la table.

```
#include <QtGui> #include <QVector> #include "ihm.h"
```

Classes

– class [CTable](#)

9.10.1 Description détaillée

Auteur

Racamond Adrien

Version

1.0

9.11 Référence du fichier trame.cpp

```
#include "trame.h"
```

9.12 Référence du fichier trame.h

Classe gérant la table et les calculs lié a la seance, impacts, affichage des zones de la table.

```
#include "ihm.h"
```

Classes

- class [CTrame](#)

9.12.1 Description détaillée

Auteur

Racamond Adrien

Version

1.0