

REmplissage automatique d'un porte conteneurs

Sommaire

- I) Introduction et présentation générale**
- II) Mise en contexte et cas d'utilisation de l'application**
- III) Présentation et utilisation de l'application**
- IV) Explication du code**
- V) Quelques problèmes rencontrés**

Introduction et présentation générale

Introduction

Contexte

Application

Explications

Problèmes

La logistique et le transport par bateau sont au cœur du XXI ème siècle. En effet, la mondialisation implique de nombreux échanges de marchandises se faisant pour la majorité par bateau. Il est donc nécessaire d'assurer la simplicité du transport et de remplissage des porte conteneurs.

Les conteneurs

Introduction

Contexte

Application

Explications

Problèmes

En 1945, le premier conteneur a été inventé par l'armée américaine afin de faciliter le ravitaillement des troupes en Europe. Un conteneur est un caisson dimensionné pour les transports de tous types. Les conteneurs ont révolutionné le transport de marchandises pour leur facilité d'utilisation, leur sécurité et leur capacité de conservation.

Les conteneurs

Introduction

Contexte

Application

Explications

Problèmes

Il existe de nombreux modèles de conteneurs adaptés au type de marchandises à transporter.

Exemples de conteneurs

Introduction

Contexte

Application

Explications

Problèmes



20 Pieds Flat Rack

Exemples de conteneurs

Introduction

Contexte

Application

Explications

Problèmes



20 Pieds Reefer

Exemples de conteneurs

Introduction

Contexte

Application

Explications

Problèmes



20 Pieds Dry

Exemples de conteneurs

Introduction

Contexte

Application

Explications

Problèmes



20 Pieds Open Top

Les conteneurs connectés

Introduction

Contexte

Application

Explications

Problèmes

Le conteneur est en constante évolution, c'est ainsi que les conteneurs connectés ont fait leur apparition sur le marché. Ces conteneurs permettent par exemple :

Les conteneurs connectés

Introduction

Contexte

Application

Explications

Problèmes

- Un meilleur suivi grâce à un tracker GPS

Les conteneurs connectés

Introduction

Contexte

Application

Explications

Problèmes

- Un meilleur suivi grâce à un tracker GPS
- Une meilleure conservation des marchandises grâce à la gestion à distance de la température, de l'hygrométrie...

Les conteneurs connectés

Introduction

Contexte

Application

Explications

Problèmes

- **Un meilleur suivi grâce à un tracker GPS**
- **Une meilleure conservation des marchandises grâce à la gestion à distance de la température, de l'hygrométrie...**
- **Une sécurité accrue. En effet certains conteneurs sont programmés pour s'ouvrir uniquement lorsqu'ils sont à des coordonnées GPS précises**

Problématique

Introduction

Contexte

Application

Explications

Problèmes

Les conteneurs ont beaucoup évolué grâce aux nouvelles technologies mais qu'en est-il du remplissage des porte conteneurs?

Problématique

Introduction

Contexte

Application

Explications

Problèmes

Comment, à partir d'informations sur le porte conteneurs, avoir une application graphique simulant le remplissage automatique de celui-ci ?

Contexte

Introduction

Contexte

Application

Explications

Problèmes

L'application graphique que je présente s'inscrit dans un contexte plus large que le TIPE.

Essayons de voir comment cette application graphique pourrait être utilisée dans le domaine portuaire.

Du côté du porte conteneurs

Introduction

Contexte

Application

Explications

Problèmes

Il faut imaginer un porte conteneur dans un port, autour duquel des grues porteuses de conteneurs sont disposées. Ces grues sont liées au programme ainsi que les chariots qui amènent les conteneurs aux grues dans l'ordre prédéfini par le programme. Ensuite, les grues n'ont plus qu'à placer les conteneurs selon l'ordre donné par le programme.

Du côté du poste de contrôle

Introduction

Contexte

Application

Explications

Problèmes

Au niveau du poste de contrôle, tout est géré informatiquement. La personne en charge du remplissage du paquebot ouvre l'application, rentre les données propres au porte conteneurs nécessaires et lance ensuite le remplissage. Les grues le remplissent toutes seules et la personne voit sur son écran le remplissage en temps réel.

Application

Introduction

Contexte

Application

Explications

Problèmes

Initialisation caractéristiques du paquebot

Masse maximale supportée par le paquebot : (en tonnes)

Entrer le nombre de ports par lesquels le paquebot va passer

Nombre de conteneurs que l'on peut placer en largeur

Nombre de conteneurs que l'on peut placer en longueur

Nombre de conteneurs total à placer

Valider ces informations

Application

Introduction

Contexte

Application

Explications

Problèmes

Initialisation caractéristiques du paquebot

Masse maximale supportée par le paquebot : (en tonnes)

Entrer le nombre de ports par lesquels le paquebot va passer

Nombre de conteneurs que l'on peut placer en largeur

Nombre de conteneurs que l'on peut placer en longueur

Nombre de conteneurs total à placer

Application

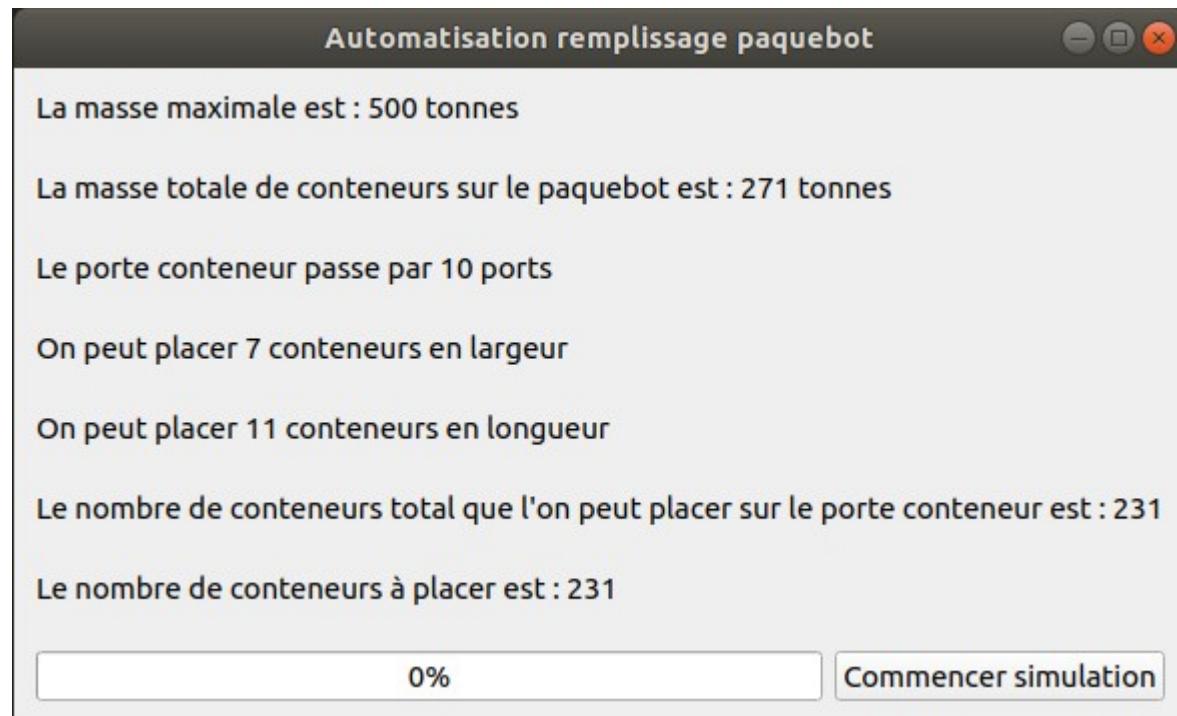
Introduction

Contexte

Application

Explications

Problèmes



Application

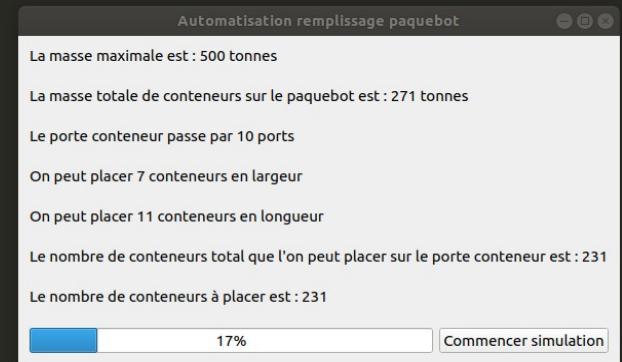
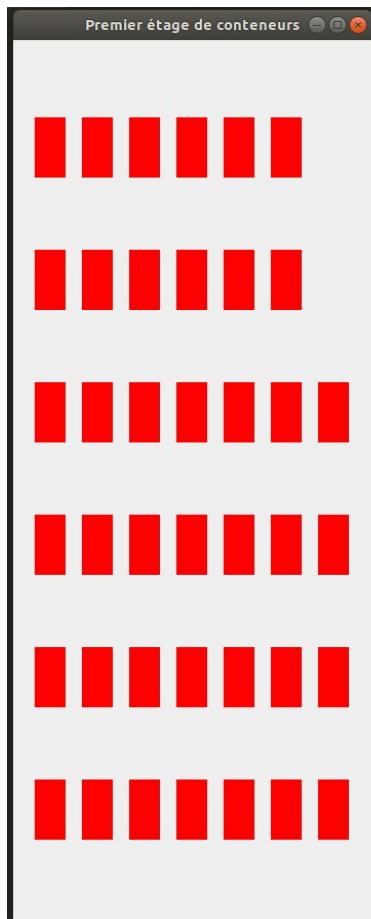
Introduction

Contexte

Application

Explications

Problèmes



Application

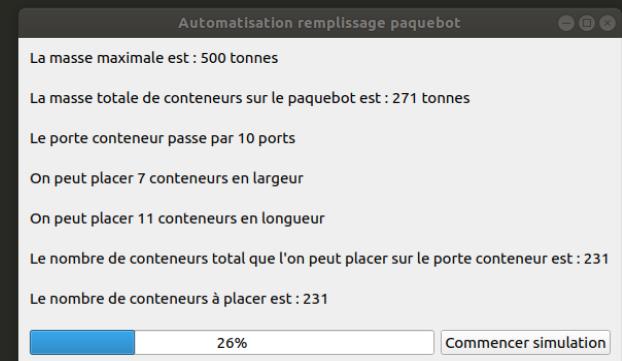
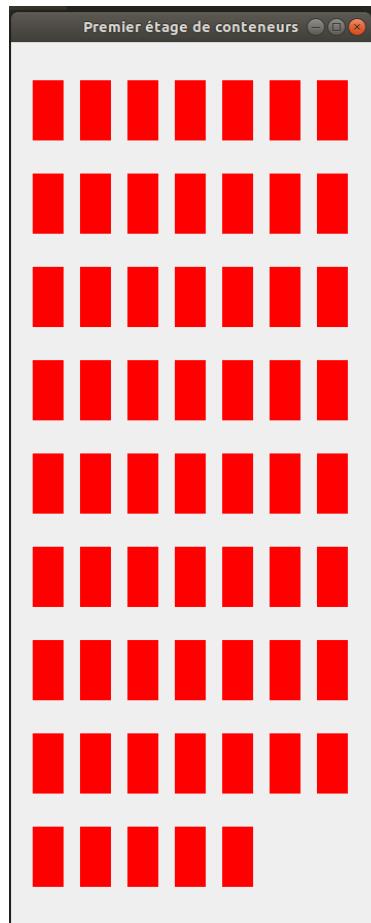
Introduction

Contexte

Application

Explications

Problèmes



Application

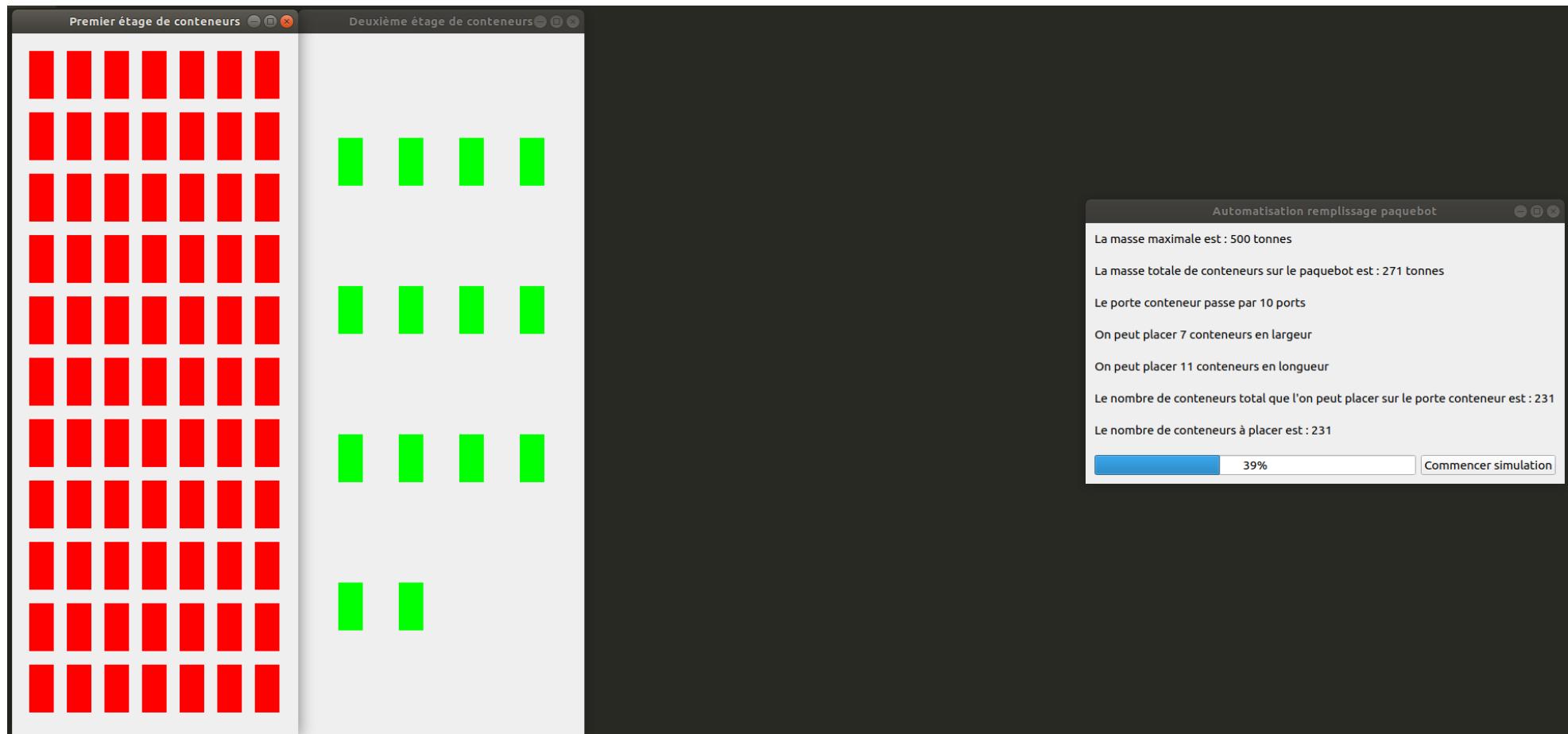
Introduction

Contexte

Application

Explications

Problèmes



Application

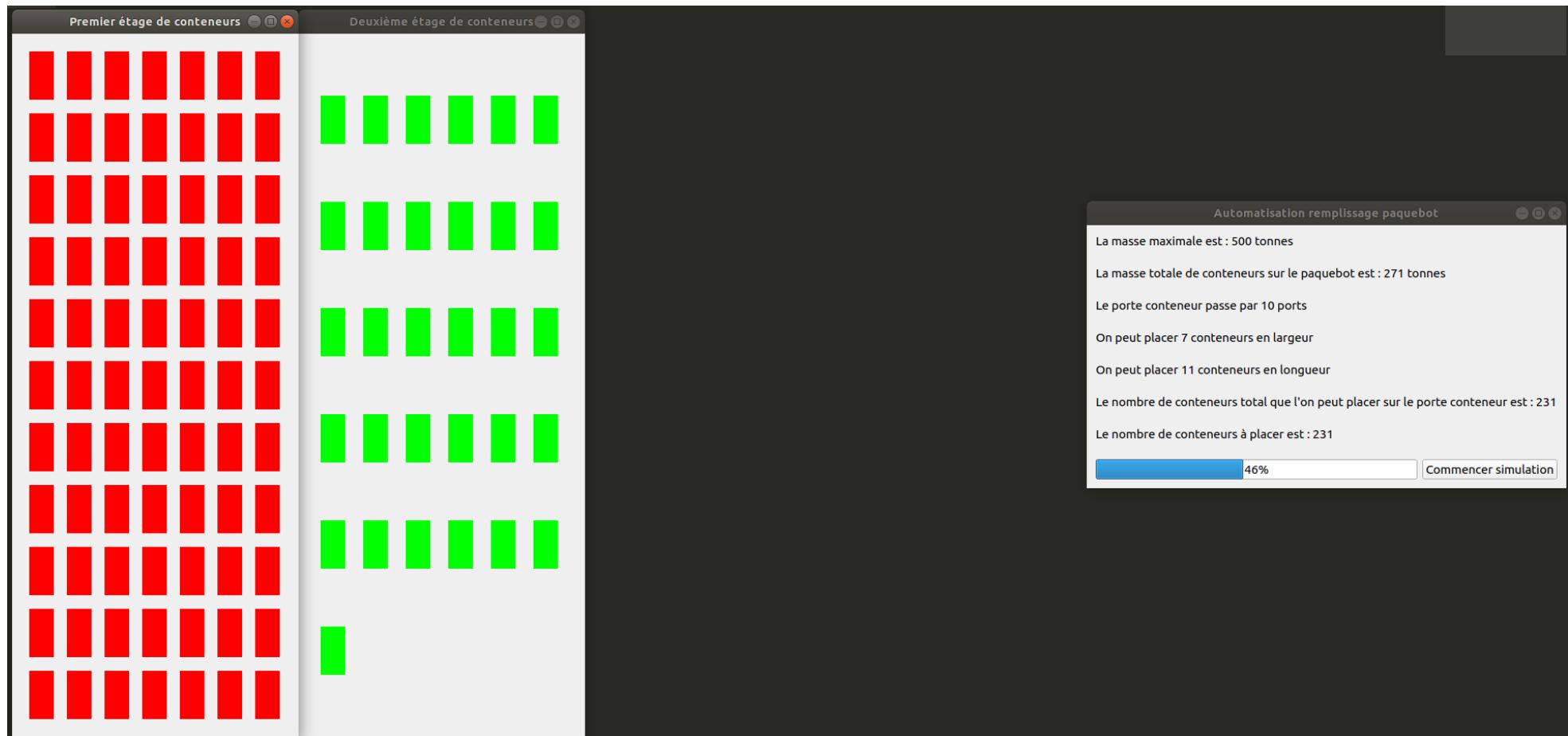
Introduction

Contexte

Application

Explications

Problèmes



Application

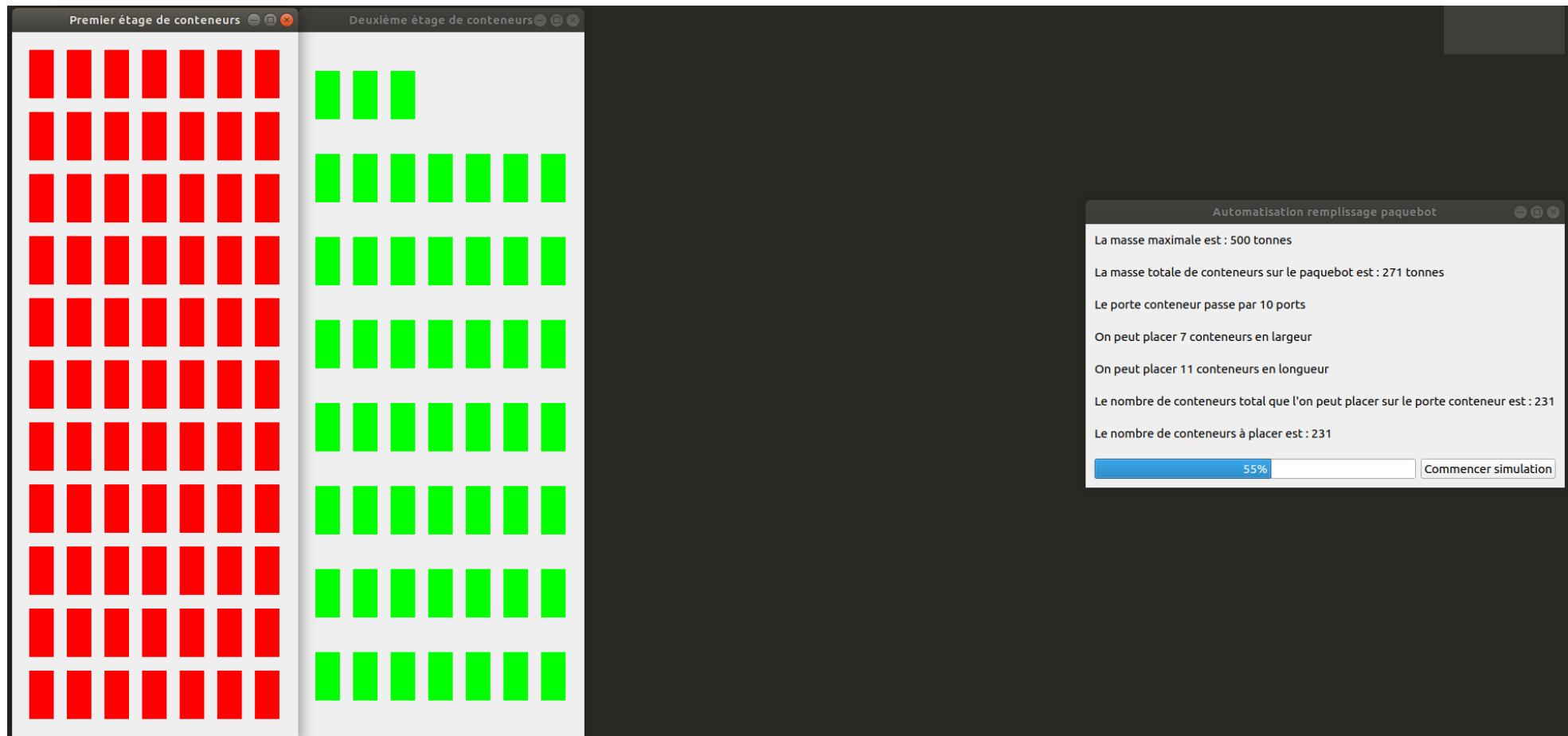
Introduction

Contexte

Application

Explications

Problèmes



Application

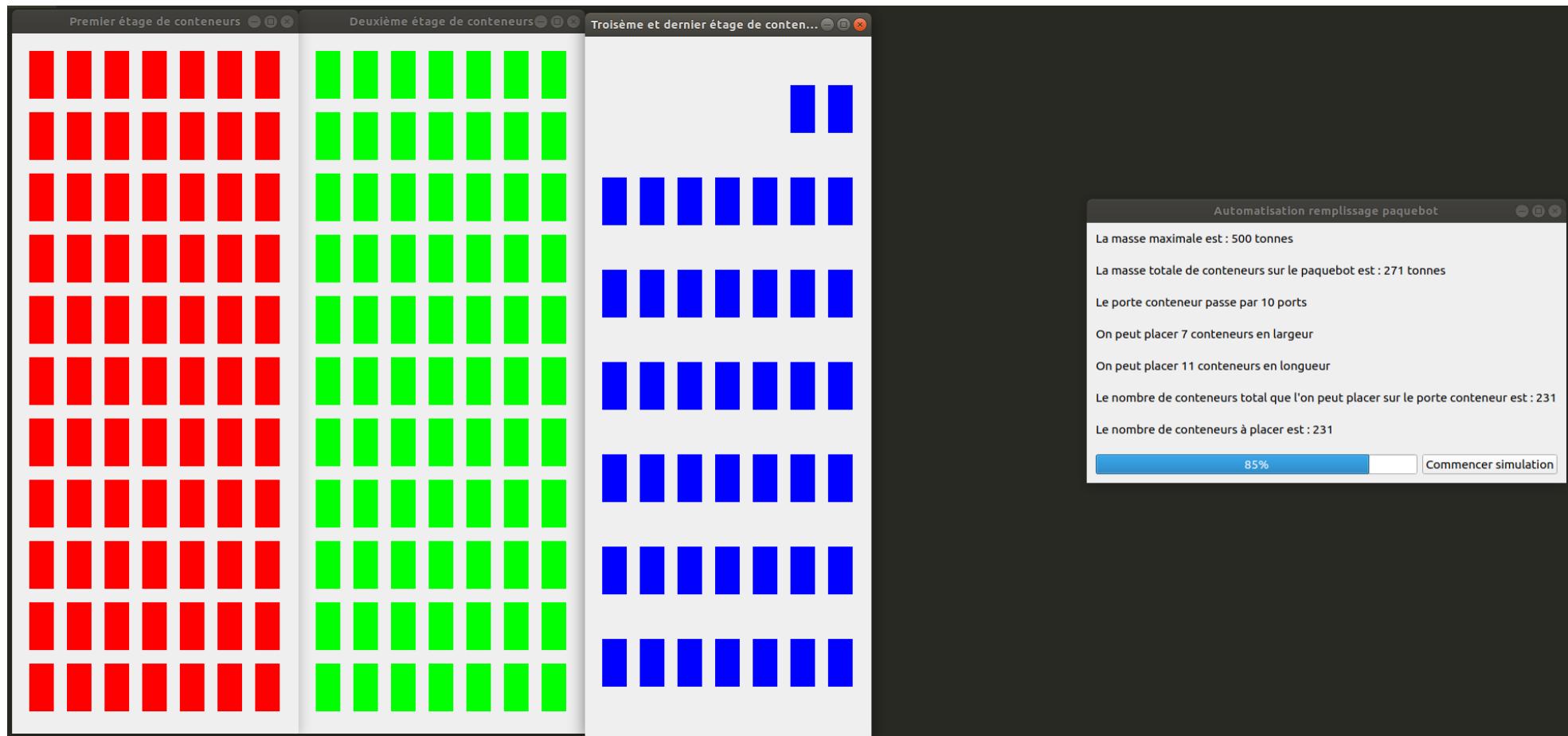
Introduction

Contexte

Application

Explications

Problèmes



Application

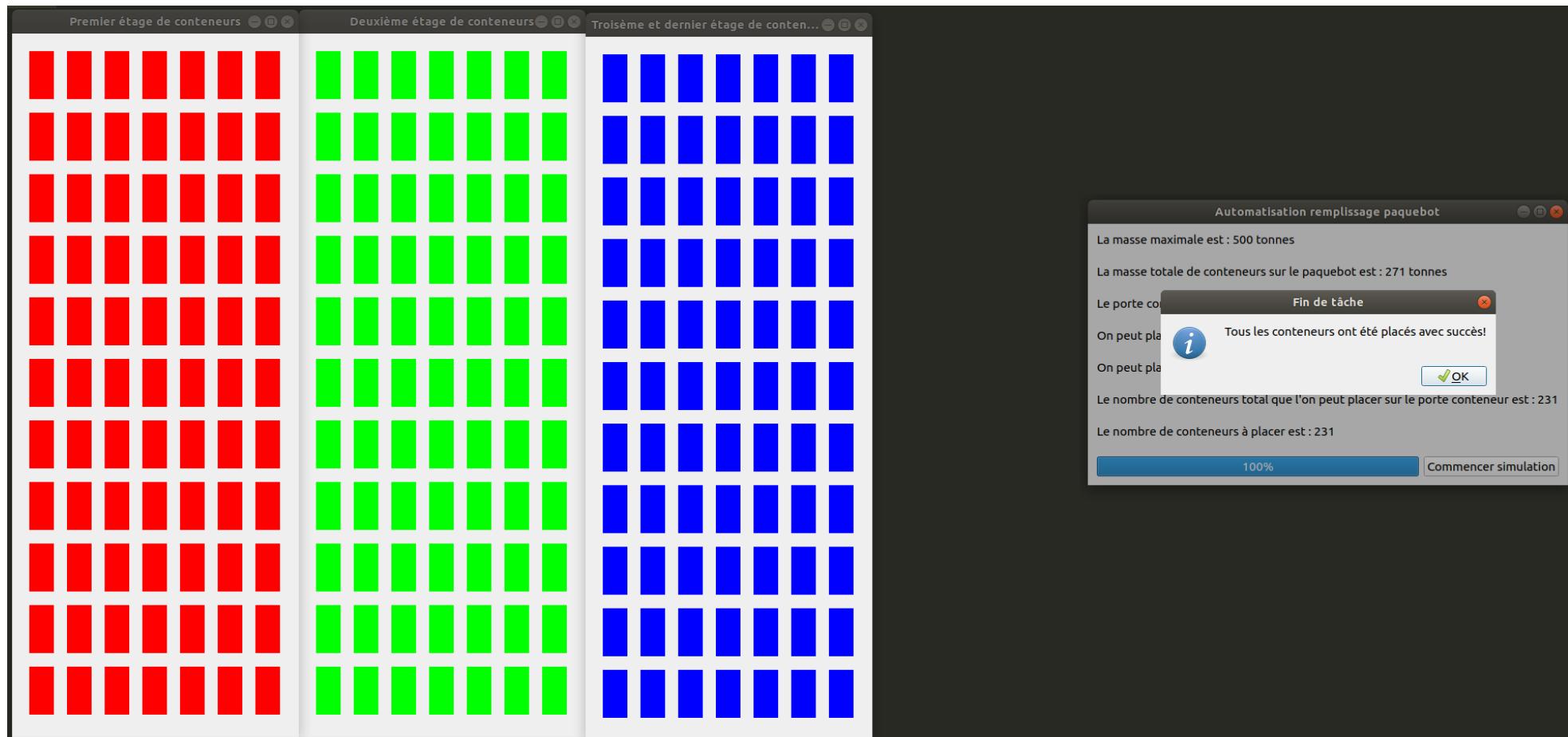
Introduction

Contexte

Application

Explications

Problèmes



Application

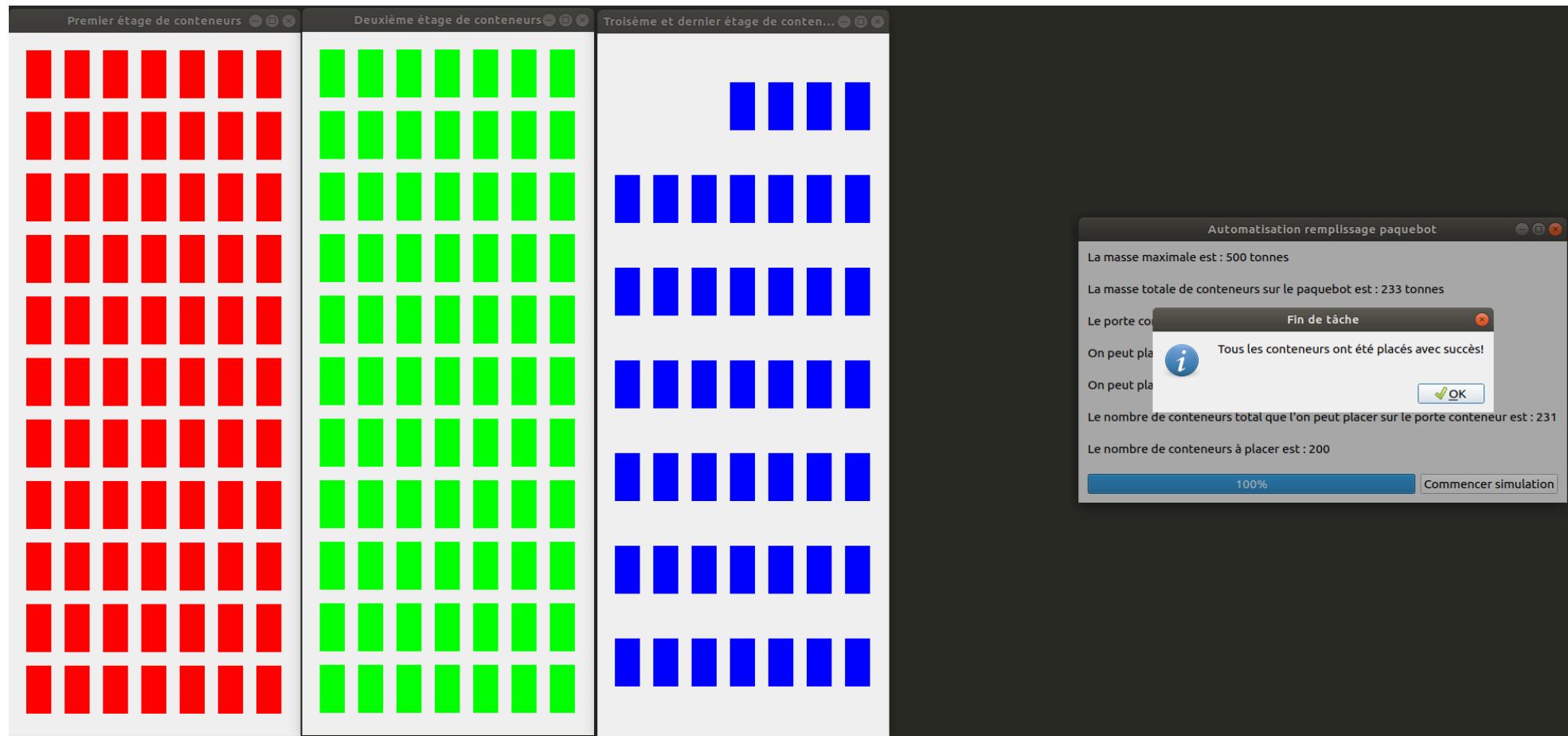
Introduction

Contexte

Application

Explications

Problèmes



Application

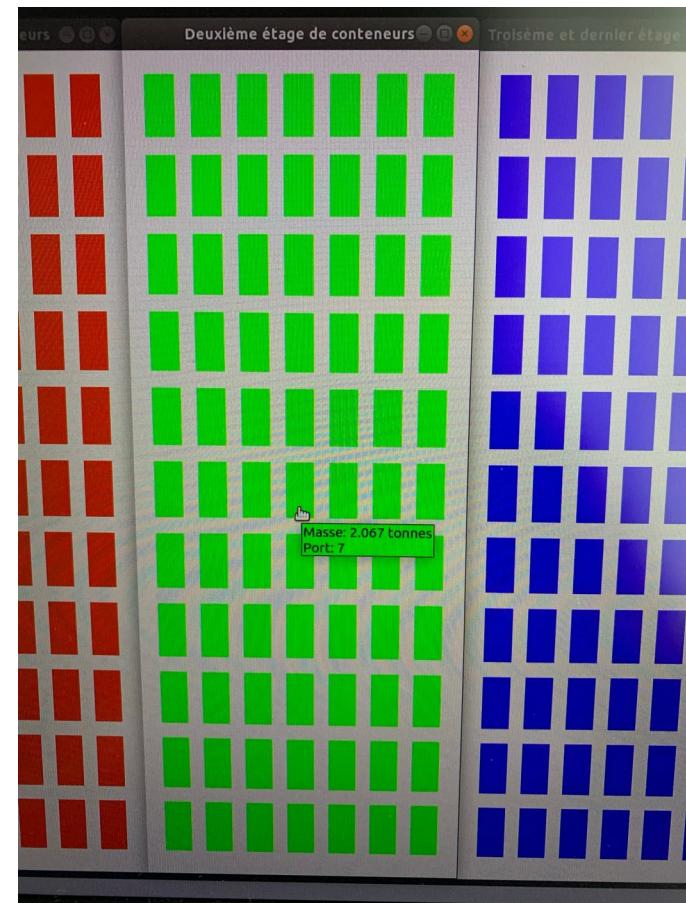
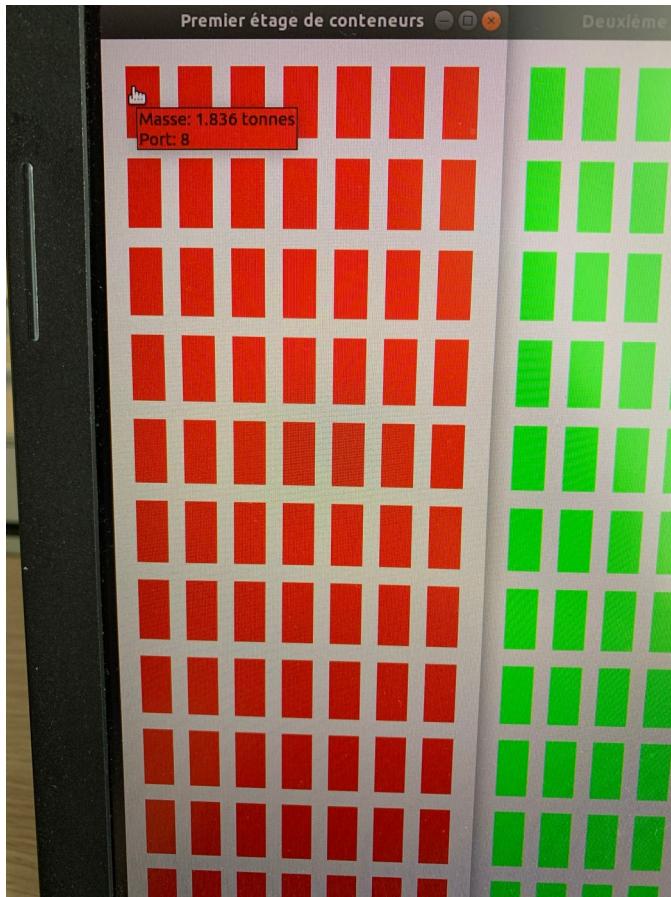
Introduction

Contexte

Application

Explications

Problèmes



Application

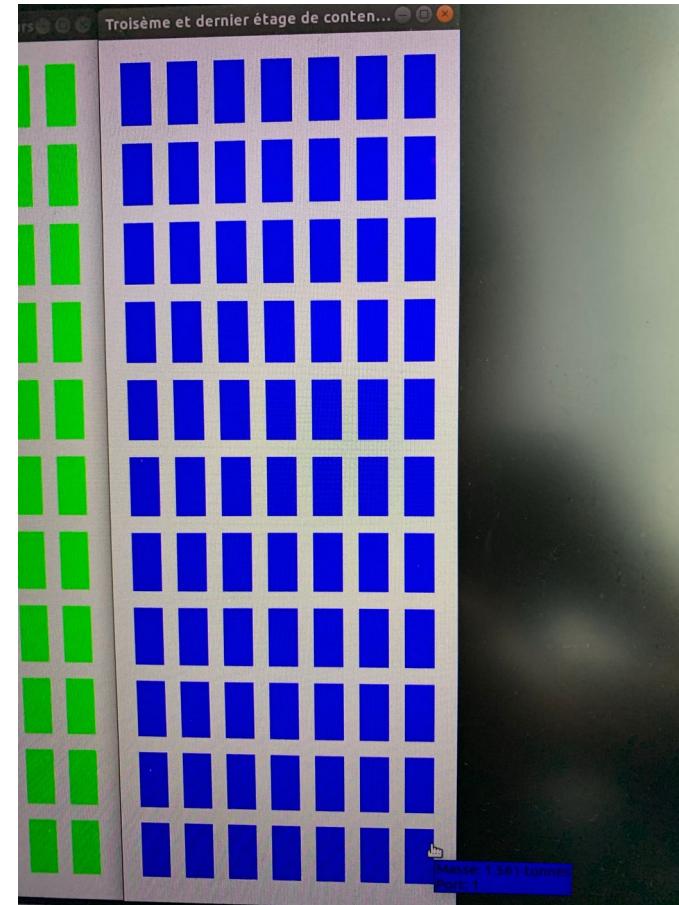
Introduction

Contexte

Application

Explications

Problèmes



Explications et code

Introduction

Contexte

Application

Explications

Problèmes

Main
(main.cpp)



FirstWindow

Fichier d'en-tête
(FirstWindow.h)

Fichier d'implémentation
(FirstWindow.cpp)

Explications et code

Introduction

Contexte

Application

Explications

Problèmes

Main
(main.cpp)



FirstWindow

Fichier d'en-tête
(FirstWindow.h)

Fichier d'implémentation
(FirstWindow.cpp)

Valider ces
informations



Window

Fichier d'en-tête
(Window.h)

Fichier d'implémentation
(Window.cpp)

Explications et code

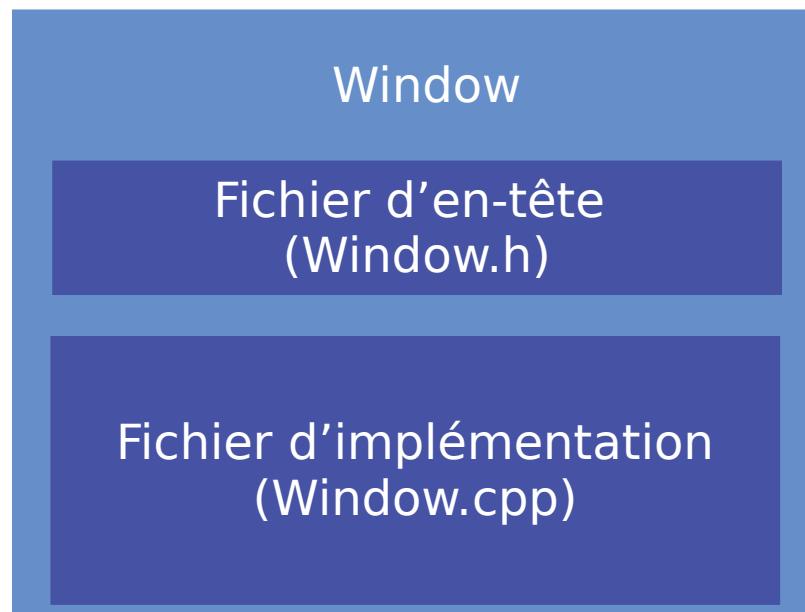
Introduction

Contexte

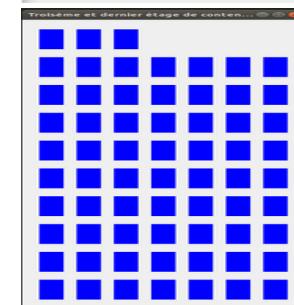
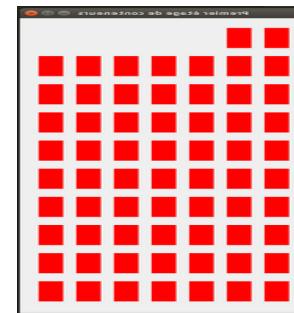
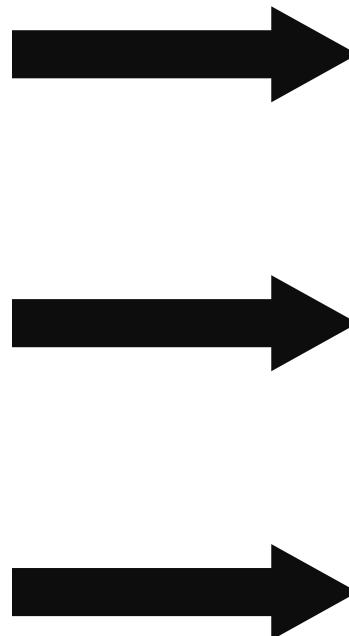
Application

Explications

Problèmes



Commencer la simulation



Ouverture progressive des 3 fenêtres simulant le remplissage

Visuellement

Introduction

Contexte

Application

Explications

Problèmes

FirstWindow

Initialisation caractéristiques du paquebot

Masse maximale supportée par le paquebot : (en tonnes)

Entrer le nombre de ports par lesquels le paquebot va passer

Nombre de conteneurs que l'on peut placer en largeur

Nombre de conteneurs que l'on peut placer en longueur

Nombre de conteneurs total à placer

Valider ces informations

Window

Automatisation remplissage paquebot

La masse maximale est : 500 tonnes

La masse totale de conteneurs sur le paquebot est : 271 tonnes

Le porte conteneur passe par 10 ports

On peut placer 7 conteneurs en largeur

On peut placer 11 conteneurs en longueur

Le nombre de conteneurs total que l'on peut placer sur le porte conteneur est : 231

Le nombre de conteneurs à placer est : 231

0%

Commencer simulation

Main

Introduction

Contexte

Application

Explications

Problèmes

```
1 #include <QApplication>
2 #include "Window.h"
3 #include <FirstWindow.h>
4
5 using namespace std;
6
7
8 //Main habituel pour lancer une application graphique avec QtCreator
9
10 int main (int argc, char *argv[])
11 {
12     QApplication app(argc, argv);
13
14     FirstWindow firstWindow;
15     firstWindow.show();
16
17
18     return app.exec();
19 }
20 }
```

Création d'un objet de type 'FirstWindow' et affichage de celui-ci



FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

```
1 #ifndef FIRSTWINDOW_H
2 #define FIRSTWINDOW_H
3 #include <QWidget>
4 #include <QVBoxLayout>
5 #include <QPushButton>
6 #include <QIntValidator>
7 #include <QLabel>
8 #include <QLineEdit>
9 #include <Window.h>
10 #include <iostream>
11 #include <QMMessageBox>
12 #include <QKeyEvent>
13 |
14 using namespace std;
15
16 - class FirstWindow:public QWidget
17 {
18     Q_OBJECT
19
20 public:
21     FirstWindow();
22     virtual void keyPressEvent(QKeyEvent *event);
23
24 public slots:
25     void validation();
26
27 private:
28
29     QLabel *m_masseMaxLabel;
30     QLineEdit *m_masseMaxEdit;
31     QLabel *m_nombrePortsLabel;
32     QLineEdit *m_nombrePortsEdit;
33     QLabel *m_nombreConteneurLargeurLabel;
34     QLineEdit *m_nombreConteneurLargeurEdit;
35     QLabel *m_nombreConteneurLongueurLabel;
36     QLineEdit *m_nombreConteneurLongueurEdit;
37     QLabel *m_totalNombreConteneursLabel;
38     QLineEdit *m_totalNombreConteneursEdit;
39     int *m_nombreConteneurLargeur;
40     int *m_nombreConteneurLongueur;
41 };
42
43
44 #endif // FIRSTWINDOW_H
```

Inclusion des bibliothèques nécessaires

FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

```
1 #ifndef FIRSTWINDOW_H
2 #define FIRSTWINDOW_H
3 #include <QWidget>
4 #include <QVBoxLayout>
5 #include <QPushButton>
6 #include <QIntValidator>
7 #include <QLabel>
8 #include <QLineEdit>
9 #include <Window.h>
10 #include <iostream>
11 #include <QMMessageBox>
12 #include <QKeyEvent>
13 |
14 using namespace std;
15
16 class FirstWindow:public QWidget
17 {
18     Q_OBJECT
19
20 public:
21     FirstWindow();
22     virtual void keyPressEvent(QKeyEvent *event);
23
24 public slots:
25     void validation();
26
27 private:
28
29     QLabel *m_masseMaxLabel;
30     QLineEdit *m_masseMaxEdit;
31     QLabel *m_nombrePortsLabel;
32     QLineEdit *m_nombrePortsEdit;
33     QLabel *m_nombreConteneurLargeurLabel;
34     QLineEdit *m_nombreConteneurLargeurEdit;
35     QLabel *m_nombreConteneurLongueurLabel;
36     QLineEdit *m_nombreConteneurLongueurEdit;
37     QLabel *m_totalNombreConteneursLabel;
38     QLineEdit *m_totalNombreConteneursEdit;
39     int *m_nombreConteneurLargeur;
40     int *m_nombreConteneurLongueur;
41 };
42
43 #endif // FIRSTWINDOW_H
```

Inclusion des bibliothèques nécessaires

Initialisation des méthodes et slots de la classe

FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

Contrairement à une fonction “normale”, un slot et une fonction réagissant aux actions de l’utilisateur tel qu’un clic ou encore l’appui sur une touche. Ce slot doit être lié par la fonction ‘connect’ à un widget par un signal (clic, modification texte...). Un signal est propre au widget et permet l’exécution du slot.

FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

```
1 #ifndef FIRSTWINDOW_H
2 #define FIRSTWINDOW_H
3 #include <QWidget>
4 #include <QVBoxLayout>
5 #include <QPushButton>
6 #include <QIntValidator>
7 #include <QLabel>
8 #include <QLineEdit>
9 #include <Window.h>
10 #include <iostream>
11 #include <QMMessageBox>
12 #include <QKeyEvent>
13 |
14 using namespace std;
15
16 class FirstWindow:public QWidget
17 {
18     Q_OBJECT
19
20 public:
21     FirstWindow();
22     virtual void keyPressEvent(QKeyEvent *event);
23
24 public slots:
25     void validation();
26
27 private:
28
29     QLabel *m_masseMaxLabel;
30     QLineEdit *m_masseMaxEdit;
31     QLabel *m_nombrePortsLabel;
32     QLineEdit *m_nombrePortsEdit;
33     QLabel *m_nombreConteneurLargeurLabel;
34     QLineEdit *m_nombreConteneurLargeurEdit;
35     QLabel *m_nombreConteneurLongueurLabel;
36     QLineEdit *m_nombreConteneurLongueurEdit;
37     QLabel *m_totalNombreConteneursLabel;
38     QLineEdit *m_totalNombreConteneursEdit;
39     int *m_nombreConteneurLargeur;
40     int *m_nombreConteneurLongueur;
41 };
42
43
44 #endif // FIRSTWINDOW_H
```

Inclusion des bibliothèques nécessaires

Initialisation des méthodes et slots de la classe

Initialisation des attributs de la classe

FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

```
1 #ifndef FIRSTWINDOW_H
2 #define FIRSTWINDOW_H
3 #include <QWidget>
4 #include <QVBoxLayout>
5 #include <QPushButton>
6 #include <QIntValidator>
7 #include <QLabel>
8 #include <QLineEdit>
9 #include <Window.h>
10 #include <iostream>
11 #include <QMessageBox>
12 #include <QKeyEvent>
13
14 using namespace std;
15
16 class FirstWindow:public QWidget
17 {
18     Q_OBJECT
19
20 public:
21     FirstWindow();
22     virtual void keyPressEvent(QKeyEvent *event);
23
24 public slots:
25     void validation();
26
27 private:
28
29     QLabel *m_masseMaxLabel;
30     QLineEdit *m_masseMaxEdit;
31     QLabel *m_nombrePortsLabel;
32     QLineEdit *m_nombrePortsEdit;
33     QLabel *m_nombreConteneurLargeurLabel;
34     QLineEdit *m_nombreConteneurLargeurEdit;
35     QLabel *m_nombreConteneurLongueurLabel;
36     QLineEdit *m_nombreConteneurLongueurEdit;
37     QLabel *m_totalNombreConteneursLabel;
38     QLineEdit *m_totalNombreConteneursEdit;
39     int *m_nombreConteneurLargeur;
40     int *m_nombreConteneurLongueur;
41
42 };
43
44 #endif // FIRSTWINDOW_H
```

Inclusion des bibliothèques nécessaires

Héritage

Initialisation des méthodes et slots de la classe

Initialisation des attributs de la classe

FirstWindow fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

Une des caractéristiques intéressantes de QtCreator est l'héritage. Lors de la création d'une classe, elle peut hériter d'un autre objet lui permettant ainsi d'avoir les mêmes caractéristiques. Ici par exemple la classe 'FirstWindow' hérite de la classe 'QWidget' ; c'est donc un widget elle aussi. On verra ensuite que la classe Window est aussi un widget.

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
3  FirstWindow::FirstWindow():QWidget()
4  {
5
6      this->setWindowTitle("Initialisation caractéristiques du paquebot");
7      QVBoxLayout *layout = new QVBoxLayout(this);           //On définit le layout vertical de la fenêtre
8
9  //Initialisation des widgets
10
11     m_masseMaxLabel = new QLabel("Masse maximale supportée par le paquebot: (en tonnes)");
12     m_masseMaxEdit = new QLineEdit;
13     QIntValidator *validatorMasse = new QIntValidator;
14     m_masseMaxEdit->setValidator(validatorMasse);
15
16     m_nombrePortsLabel = new QLabel("Entrer le nombre de port par lesquels le paquebot va passer");
17     m_nombrePortsEdit = new QLineEdit;
18     QIntValidator *validator = new QIntValidator(1,50);
19     m_nombrePortsEdit->setValidator(validator);
20
21     m_nombreConteneurLargeurLabel = new QLabel("Nombre de conteneur que l'on peut placer en largeur");
22     m_nombreConteneurLargeurEdit = new QLineEdit;
23     QIntValidator *validatorLarg = new QIntValidator(1,30);
24     m_nombreConteneurLargeurEdit->setValidator(validatorLarg);
25
26     m_nombreConteneurLongueurLabel = new QLabel("Nombre de conteneur que l'on peut placer en longueur");
27     m_nombreConteneurLongueurEdit = new QLineEdit;
28     QIntValidator *validatorLong = new QIntValidator(1,60);
29     m_nombreConteneurLongueurEdit->setValidator(validatorLong);
30
31     m_totalNombreConteneursLabel = new QLabel("Nombre de conteneur total à placer");
32     m_totalNombreConteneursEdit = new QLineEdit;
33
34     QPushButton *boutonValiderInitialisation = new QPushButton("Valider ces informations");
35
36 //Ajout des widgets au layout principal de la fenêtre
37
38     layout->addWidget(m_masseMaxLabel);
39     layout->addWidget(m_masseMaxEdit);
40     layout->addWidget(m_nombrePortsLabel);
41     layout->addWidget(m_nombrePortsEdit);
42     layout->addWidget(m_nombreConteneurLargeurLabel);
43     layout->addWidget(m_nombreConteneurLargeurEdit);
44     layout->addWidget(m_nombreConteneurLongueurLabel);
45     layout->addWidget(m_nombreConteneurLongueurEdit);
46     layout->addWidget(m_totalNombreConteneursLabel);
47     layout->addWidget(m_totalNombreConteneursEdit);
48     layout->addWidget(boutonValiderInitialisation);
49
50
51 //Connexion du slot 'valider' au bouton "Valider ces informations"
52
53     connect(boutonValiderInitialisation, SIGNAL(clicked()), this, SLOT(validation()));
```

Implémentation des différents widgets de la fenêtre

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
3  FirstWindow::FirstWindow():QWidget()
4  {
5
6      this->setWindowTitle("Initialisation caractéristiques du paquebot");
7      QVBoxLayout *layout = new QVBoxLayout(this);           //On définit le layout vertical de la fenêtre
8
9  //Initialisation des widgets
10
11     m_masseMaxLabel = new QLabel("Masse maximale supportée par le paquebot: (en tonnes)");
12     m_masseMaxEdit = new QLineEdit;
13     QIntValidator *validatorMasse = new QIntValidator;
14     m_masseMaxEdit->setValidator(validatorMasse);
15
16     m_nombrePortsLabel = new QLabel("Entrer le nombre de port par lesquels le paquebot va passer");
17     m_nombrePortsEdit = new QLineEdit;
18     QIntValidator *validator = new QIntValidator(1,50);
19     m_nombrePortsEdit->setValidator(validator);
20
21     m_nombreConteneurLargeurLabel = new QLabel("Nombre de conteneur que l'on peut placer en largeur");
22     m_nombreConteneurLargeurEdit = new QLineEdit;
23     QIntValidator *validatorLarg = new QIntValidator(1,30);
24     m_nombreConteneurLargeurEdit->setValidator(validatorLarg);
25
26     m_nombreConteneurLongueurLabel = new QLabel("Nombre de conteneur que l'on peut placer en longueur");
27     m_nombreConteneurLongueurEdit = new QLineEdit;
28     QIntValidator *validatorLong = new QIntValidator(1,60);
29     m_nombreConteneurLongueurEdit->setValidator(validatorLong);
30
31     m_totalNombreConteneursLabel = new QLabel("Nombre de conteneur total à placer");
32     m_totalNombreConteneursEdit = new QLineEdit;
33
34     QPushButton *boutonValiderInitialisation = new QPushButton("Valider ces informations");
35
36 //Ajout des widgets au layout principal de la fenêtre
37
38     layout->addWidget(m_masseMaxLabel);
39     layout->addWidget(m_masseMaxEdit);
40     layout->addWidget(m_nombrePortsLabel);
41     layout->addWidget(m_nombrePortsEdit);
42     layout->addWidget(m_nombreConteneurLargeurLabel);
43     layout->addWidget(m_nombreConteneurLargeurEdit);
44     layout->addWidget(m_nombreConteneurLongueurLabel);
45     layout->addWidget(m_nombreConteneurLongueurEdit);
46     layout->addWidget(m_totalNombreConteneursLabel);
47     layout->addWidget(m_totalNombreConteneursEdit);
48     layout->addWidget(boutonValiderInitialisation);
49
50
51 //Connexion du slot 'valider' au bouton "Valider ces informations"
52
53     connect(boutonValiderInitialisation, SIGNAL(clicked()), this, SLOT(validation()));
```

Implémentation des différents widgets de la fenêtre

Ajout de ces widgets au layout de la fenêtre

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

Un layout permet la gestion automatique des widgets de la fenêtre. Il en existe des horizontaux, verticaux, des tableaux... On entre dans un certain ordre les widgets puis le layout les positionne dans la fenêtre. Ces positionnements ne sont pas fixes et s'adaptent aux redimensionnements de la fenêtre.

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
3     FirstWindow::FirstWindow():QWidget()
4 {
5
6     this->setWindowTitle("Initialisation caractéristiques du paquebot");
7     QVBoxLayout *layout = new QVBoxLayout(this);           //On définit le layout vertical de la fenêtre
8
9     //Initialisation des widgets
10
11    m_masseMaxLabel = new QLabel("Masse maximale supportée par le paquebot: (en tonnes)");
12    m_masseMaxEdit = new QLineEdit;
13    QIntValidator *validatorMasse = new QIntValidator;
14    m_masseMaxEdit->setValidator(validatorMasse);
15
16    m_nombrePortsLabel = new QLabel("Entrer le nombre de port par lesquels le paquebot va passer");
17    m_nombrePortsEdit = new QLineEdit;
18    QIntValidator *validator = new QIntValidator(1,50);
19    m_nombrePortsEdit->setValidator(validator);
20
21    m_nombreConteneurLargeurLabel = new QLabel("Nombre de conteneur que l'on peut placer en largeur");
22    m_nombreConteneurLargeurEdit = new QLineEdit;
23    QIntValidator *validatorLarg = new QIntValidator(1,30);
24    m_nombreConteneurLargeurEdit->setValidator(validatorLarg);
25
26    m_nombreConteneurLongueurLabel = new QLabel("Nombre de conteneur que l'on peut placer en longueur");
27    m_nombreConteneurLongueurEdit = new QLineEdit;
28    QIntValidator *validatorLong = new QIntValidator(1,60);
29    m_nombreConteneurLongueurEdit->setValidator(validatorLong);
30
31    m_totalNombreConteneursLabel = new QLabel("Nombre de conteneur total à placer");
32    m_totalNombreConteneursEdit = new QLineEdit;
33
34    QPushButton *boutonValiderInitialisation = new QPushButton("Valider ces informations");
35
36    //Ajout des widgets au layout principal de la fenêtre
37
38    layout->addWidget(m_masseMaxLabel);
39    layout->addWidget(m_masseMaxEdit);
40    layout->addWidget(m_nombrePortsLabel);
41    layout->addWidget(m_nombrePortsEdit);
42    layout->addWidget(m_nombreConteneurLargeurLabel);
43    layout->addWidget(m_nombreConteneurLargeurEdit);
44    layout->addWidget(m_nombreConteneurLongueurLabel);
45    layout->addWidget(m_nombreConteneurLongueurEdit);
46    layout->addWidget(m_totalNombreConteneursLabel);
47    layout->addWidget(m_totalNombreConteneursEdit);
48    layout->addWidget(boutonValiderInitialisation);
49
50
51    //Connexion du slot 'valider' au bouton "Valider ces informations"
52
53    connect(boutonValiderInitialisation, SIGNAL(clicked()), this, SLOT(validation()));
54 }
```

Implémentation des différents widgets de la fenêtre

Ajout de ces widgets au layout de la fenêtre

Connexion du bouton valider avec le slot validation

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
51 //Si l'utilisateur a appuyé sur le bouton valider et que tous les champs sont remplis correctement,  
52 //alors on ouvre la fenêtre principale et on cache la première.  
53  
54 void FirstWindow::validation()  
55 {  
56     //Si l'un des champs n'est pas rempli  
57  
58     if (m_masseMaxEdit->text().isEmpty() || m_nombrePortsEdit->text().isEmpty() || m_nombreConteneurLargeurEdit->text().isEmpty()  
59         || m_nombreConteneurLongeurEdit->text().isEmpty() || m_totalNombreConteneursEdit->text().isEmpty())  
60     {  
61         QMessageBox::warning(0, "Erreur", "Vous devez entrer toutes les informations pour pouvoir continuer");  
62         return;  
63     }  
64  
65     //Si le nombre de conteneur à placer est supérieur au nombre de conteneur maximal supporté par le paquebot  
66     if(m_totalNombreConteneursEdit->text().toInt() > (m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3)  
67     {  
68         QString str ="Vous devez rentrer un nombre total de conteneur inférieur ou égal à "  
69             + QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3);  
70         QMessageBox::warning(0, "Erreur, trop de conteneurs", str );  
71         m_totalNombreConteneursEdit->setText(QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3));  
72         return;  
73     }  
74  
75     //Si la masse maximal supportée par le paquebot renseignée est trop faible  
76     if(m_masseMaxEdit->text().toFloat() < 1.5* m_totalNombreConteneursEdit->text().toInt())  
77     {  
78         QString str = "Vous devez rentrer une masse maximale supérieure à " + QString::number(1.5* m_totalNombreConteneursEdit->text().toInt());  
79         QMessageBox::warning(0, "Erreur, masse maximale trop faible", str );  
80         m_masseMaxEdit->setText(QString::number(1.5* m_totalNombreConteneursEdit->text().toInt() +10));  
81         return;  
82     }  
83  
84     //Sinon: ouverture de la deuxième fenêtre et on masque la première  
85     Window *window = new Window(m_masseMaxEdit->text().toInt(), m_nombrePortsEdit->text().toInt(),  
86                                 m_nombreConteneurLargeurEdit->text().toInt(), m_nombreConteneurLongeurEdit->text().toInt(), m_totalNombreConteneursEdit->text().toInt());  
87     window->show();  
88     this->hide();  
89 }
```

Si l'un des champ n'est pas rempli

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
51 //Si l'utilisateur a appuyé sur le bouton valider et que tous les champs sont remplis correctement,  
52 //alors on ouvre la fenêtre principale et on cache la première.  
53  
54 void FirstWindow::validation()  
55 {  
56     //Si l'un des champs n'est pas rempli  
57  
58     if (m_masseMaxEdit->text().isEmpty() || m_nombrePortsEdit->text().isEmpty() || m_nombreConteneurLargeurEdit->text().isEmpty()  
59         || m_nombreConteneurLongeurEdit->text().isEmpty() || m_totalNombreConteneursEdit->text().isEmpty())  
60     {  
61         QMessageBox::warning(0, "Erreur", "Vous devez entrer toutes les informations pour pouvoir continuer");  
62         return;  
63     }  
64  
65     //Si le nombre de conteneur à placer est supérieur au nombre de conteneur maximal supporté par le paquebot  
66     if(m_totalNombreConteneursEdit->text().toInt() > (m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3)  
67     {  
68         QString str ="Vous devez rentrer un nombre total de conteneur inférieur ou égal à "  
69             + QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3);  
70         QMessageBox::warning(0, "Erreur, trop de conteneurs", str );  
71         m_totalNombreConteneursEdit->setText(QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3));  
72         return;  
73     }  
74  
75     //Si la masse maximal supportée par le paquebot renseignée est trop faible  
76     if(m_masseMaxEdit->text().toFloat() < 1.5* m_totalNombreConteneursEdit->text().toInt())  
77     {  
78         QString str = "Vous devez rentrer une masse maximale supérieure à " + QString::number(1.5* m_totalNombreConteneursEdit->text().toInt());  
79         QMessageBox::warning(0, "Erreur, masse maximale trop faible", str );  
80         m_masseMaxEdit->setText(QString::number(1.5* m_totalNombreConteneursEdit->text().toInt() +10));  
81         return;  
82     }  
83  
84     //Sinon: ouverture de la deuxième fenêtre et on masque la première  
85     Window *window = new Window(m_masseMaxEdit->text().toInt(), m_nombrePortsEdit->text().toInt(),  
86                                 m_nombreConteneurLargeurEdit->text().toInt(), m_nombreConteneurLongeurEdit->text().toInt(), m_totalNombreConteneursEdit->text().toInt());  
87     window->show();  
88     this->hide();  
89 }
```

Si l'un des champ n'est pas rempli

Si le nombre de conteneurs à placer est supérieur au maximum

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
51 //Si l'utilisateur a appuyé sur le bouton valider et que tous les champs sont remplis correctement,  
52 //alors on ouvre la fenêtre principale et on cache la première.  
53  
54 void FirstWindow::validation()  
55 {  
56     //Si l'un des champs n'est pas rempli  
57  
58     if (m_masseMaxEdit->text().isEmpty() || m_nombrePortsEdit->text().isEmpty() || m_nombreConteneurLargeurEdit->text().isEmpty()  
59         || m_nombreConteneurLongeurEdit->text().isEmpty() || m_totalNombreConteneursEdit->text().isEmpty())  
60     {  
61         QMessageBox::warning(0, "Erreur", "Vous devez entrer toutes les informations pour pouvoir continuer");  
62         return;  
63     }  
64  
65     //Si le nombre de conteneur à placer est supérieur au nombre de conteneur maximal supporté par le paquebot  
66  
67     if(m_totalNombreConteneursEdit->text().toInt() > (m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3)  
68     {  
69         QString str ="Vous devez rentrer un nombre total de conteneur inférieur ou égal à "  
70             + QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3);  
71         QMessageBox::warning(0, "Erreur, trop de conteneurs", str );  
72         m_totalNombreConteneursEdit->setText(QString::number((m_nombreConteneurLargeurEdit->text().toInt()* m_nombreConteneurLongeurEdit->text().toInt())*3));  
73         return;  
74     }  
75  
76     //Si la masse maximal supportée par le paquebot renseignée est trop faible  
77  
78     if(m_masseMaxEdit->text().toFloat() < 1.5* m_totalNombreConteneursEdit->text().toInt())  
79     {  
80         QString str = "Vous devez rentrer une masse maximale supérieure à " + QString::number(1.5* m_totalNombreConteneursEdit->text().toInt());  
81         QMessageBox::warning(0, "Erreur, masse maximale trop faible", str );  
82         m_masseMaxEdit->setText(QString::number(1.5* m_totalNombreConteneursEdit->text().toInt() +10));  
83         return;  
84     }  
85  
86     //Sinon: ouverture de la deuxième fenêtre et on masque la première  
87  
88     Window *window = new Window(m_masseMaxEdit->text().toInt(), m_nombrePortsEdit->text().toInt(),  
89                             m_nombreConteneurLargeurEdit->text().toInt(), m_nombreConteneurLongeurEdit->text().toInt(), m_totalNombreConteneursEdit->text().toInt());  
90     window->show();  
91     this->hide();  
92 }
```

Si l'un des champ n'est pas rempli

Si le nombre de conteneur à placer est supérieur au maximum

Si la masse maximale est trop faible

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
51 //Si l'utilisateur a appuyé sur le bouton valider et que tous les champs sont remplis correctement,  
52 //alors on ouvre la fenêtre principale et on cache la première.  
53  
54 void FirstWindow::validation()  
55 {  
56     //Si l'un des champs n'est pas rempli  
57  
58     if (_masseMaxEdit->text().isEmpty() || _nombrePortsEdit->text().isEmpty() || _nombreConteneurLargeurEdit->text().isEmpty()  
59         || _nombreConteneurLongeurEdit->text().isEmpty() || _totalNombreConteneursEdit->text().isEmpty())  
60     {  
61         QMessageBox::warning(0, "Erreur", "Vous devez entrer toutes les informations pour pouvoir continuer");  
62         return;  
63     }  
64  
65     //Si le nombre de conteneur à placer est supérieur au nombre de conteneur maximal supporté par le paquebot  
66  
67     if(_totalNombreConteneursEdit->text().toInt() > (_nombreConteneurLargeurEdit->text().toInt()* _nombreConteneurLongeurEdit->text().toInt())*3)  
68     {  
69         QString str ="Vous devez rentrer un nombre total de conteneur inférieur ou égal à "  
70             + QString::number(_nombreConteneurLargeurEdit->text().toInt()* _nombreConteneurLongeurEdit->text().toInt())*3);  
71         QMessageBox::warning(0, "Erreur, trop de conteneurs", str );  
72         _totalNombreConteneursEdit->setText(QString::number(_nombreConteneurLargeurEdit->text().toInt()* _nombreConteneurLongeurEdit->text().toInt())*3));  
73         return;  
74     }  
75  
76     //Si la masse maximal supportée par le paquebot renseignée est trop faible  
77  
78     if(_masseMaxEdit->text().toFloat() < 1.5* _totalNombreConteneursEdit->text().toInt())  
79     {  
80         QString str = "Vous devez rentrer une masse maximale supérieure à " + QString::number(1.5* _totalNombreConteneursEdit->text().toInt());  
81         QMessageBox::warning(0, "Erreur, masse maximale trop faible", str );  
82         _masseMaxEdit->setText(QString::number(1.5* _totalNombreConteneursEdit->text().toInt() +10));  
83         return;  
84     }  
85  
86     //Sinon: ouverture de la deuxième fenêtre et on masque la première  
87  
88     Window *window = new Window(_masseMaxEdit->text().toInt(), _nombrePortsEdit->text().toInt(),  
89                             _nombreConteneurLargeurEdit->text().toInt(), _nombreConteneurLongeurEdit->text().toInt(), _totalNombreConteneursEdit->text().toInt());  
90     window->show();  
91     this->hide();  
92 }
```

Si l'un des champ n'est pas rempli

Si le nombre de conteneur à placer est supérieur au maximum

Si la masse maximal est trop faible

Sinon, ouverture de la 2ème fenêtre

FirstWindow fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
107 // Cliquer sur le bouton Valider <=> appuyer sur le bouton entrée
108
109 void FirstWindow::keyPressEvent(QKeyEvent *event)
110 {
111
112     if (event->key() == 16777220)
113         validation();
114 }
```

Méthode permettant d'appuyer sur la touche entrée pour valider les informations

Window fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

Création de la classe Window, héritant elle aussi de QWidget. Après inclusion des bibliothèques nécessaires.

```
17 class Window: public QWidget
18 {
19     Q_OBJECT
20
21 public:
22     Window(int masseMax, int nombrePorts, int nombreConteneurLargeur, int nombreConteneurLongueur,
23             int nombreTotalConteneurs);
24     QList<QList<float>> genereListeConteneur();
25     void triInsert();
26     void remplissagePaquebot();
27
28
29 public slots:
30     void simulation();
31     void poseConteneur();
```

Initialisation des méthodes et des slots de la classe

Window fichier d'en-tête

Introduction

Contexte

Application

Explications

Problèmes

```
33  
34     private:  
35  
36         //Les widgets  
37         QPushButton *m_boutonCommencerSimultation;  
38  
39  
40         QHBoxLayout *m_hlayout;  
41  
42         QGridLayout *m_layout1;    //Grille 1er étage  
43         QGridLayout *m_layout2;    //Grille 2ème étage  
44         QGridLayout *m_layout3;    //Grille 3ème étage  
45  
46         QVBoxLayout *m_vlayout;  
47  
48  
49         QProgressBar *m_barre;  
50  
51  
52         QTimer *m_timer;  
53  
54  
55         QWidget *etage1;  
56         QWidget *etage2;  
57         QWidget *etage3;  
58  
59  
60     //Les variables initialisation paquebot:  
61         int *m_masseMax;  
62         float *m_masseTotale;  
63         const int *m_nombrePorts;  
64         const int *m_nombreConteneurLargeur;  
65         const int *m_nombreConteneurLongueur;  
66         const int *m_totalNombreConteneurs;  
67  
68         int *m_conteneursPlaces;  
69         int *indice;  
70  
71  
72  
73     //Liste contenant les informations (masse et destination) de chaque conteneur  
74     QList<QList<float>> *m_listeConteneurs;  
75  
76     //Tableau 3 dimensions contenant les coordonnées de chaque conteneur  
77     QList<QList<QList<int>>> *listeCoordonnees;  
78  
79     };  
80  
81 #endif // WINDOW_H
```

Initialisation des différents widgets de la classe

Initialisation des différents attributs de la classe

Initialisation de 2 listes : l'une contiendra les informations des conteneurs (masse et destination), l'autre leurs coordonnées associées

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
3     Window::Window(int masseMax, int nombrePorts, int nombreConteneurLargeur, int nombreConteneurLongueur,int nombreTotalConteneurs):QWidget ()
4     {
5
6         this->setWindowTitle("Automatisation remplissage paquebot");
7
8         //Initialisation des attributs
9
10        m_masseMax = new int(masseMax);
11        m_masseTotale = new float(0);
12        m_nombrePorts = new int(nombrePorts);
13        m_nombreConteneurLargeur = new int(nombreConteneurLargeur);
14        m_nombreConteneurLongueur = new int(nombreConteneurLongueur);
15        m_totalNombreConteneurs = new int(nombreTotalConteneurs);
16        m_conteneursPlaces = new int(0);
17        m_listeConteneurs = new QList<QList<float>>;
18        *m_listeConteneurs += genererListeConteneur();           //On génère une liste de conteneurs d'après la méthode genererListeConteneur()
19
20
21        // On trie la liste des conteneurs: par destination décroissante, puis pour une même destination par masse décroissante
22
23        triInsert();
24
25
26
27        indice = new int(0);      //Variable nécessaire pour la méthode poseConteneur()
28
29
30        //Liste qui contiendra les coordonnées de chaque conteneur
31
32        listeCoordonnees = new QList<QList<QList<int>> ;
33
34
35
36        //Initialisation des widgets
37
38        m_boutonCommencerSimulation = new QPushButton("Commencer simulation");
39
40        m_barre = new QProgressBar;
41        m_barre->setRange(0,100);
42        m_barre->setValue(0);
43
44        etage1 = new QWidget;
45        etage1->setWindowTitle("Premier étage de conteneurs");
46        etage1->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
47
48        etage2 = new QWidget;
49        etage2->setWindowTitle("Deuxième étage de conteneurs");
50        etage2->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
51
52        etage3 = new QWidget;
53        etage3->setWindowTitle("Troisième et dernier étage de conteneurs");
54        etage3->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
```



Implémentation des attributs

Window fichier d'implémentation

Introduction

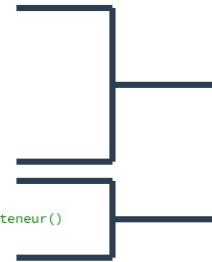
Contexte

Application

Explications

Problèmes

```
3     Window::Window(int masseMax, int nombrePorts, int nombreConteneurLargeur, int nombreConteneurLongueur,int nombreTotalConteneurs):QWidget ()
4     {
5
6         this->setWindowTitle("Automatisation remplissage paquebot");
7
8         //Initialisation des attributs
9
10        m_masseMax = new int(masseMax);
11        m_masseTotale = new float(0);
12        m_nombrePorts = new int(nombrePorts);
13        m_nombreConteneurLargeur = new int(nombreConteneurLargeur);
14        m_nombreConteneurLongueur = new int(nombreConteneurLongueur);
15        m_totalNombreConteneurs = new int(nombreTotalConteneurs);
16        m_conteneursPlaces = new int(0);
17        m_listeConteneurs = new QList<QList<float>>;
18        *m_listeConteneurs += genererListeConteneur();           //On génère une liste de conteneurs d'après la méthode genererListeConteneur()
19
20
21        // On trie la liste des conteneurs: par destination décroissante, puis pour une même destination par masse décroissante
22
23        triInsert();
24
25
26
27        indice = new int(0);      //Variable nécessaire pour la méthode poseConteneur()
28
29
30        //Liste qui contiendra les coordonnées de chaque conteneur
31
32        listeCoordonnees = new QList<QList<QList<int>> ;
33
34
35
36        //Initialisation des widgets
37
38        m_boutonCommencerSimulation = new QPushButton("Commencer simulation");
39
40        m_barre = new QProgressBar;
41        m_barre->setRange(0,100);
42        m_barre->setValue(0);
43
44        etage1 = new QWidget;
45        etage1->setWindowTitle("Premier étage de conteneurs");
46        etage1->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
47
48        etage2 = new QWidget;
49        etage2->setWindowTitle("Deuxième étage de conteneurs");
50        etage2->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
51
52        etage3 = new QWidget;
53        etage3->setWindowTitle("Troisième et dernier étage de conteneurs");
54        etage3->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
```



Implémentation des attributs

Création d'une liste de conteneurs

Window fichier d'implémentation

Introduction

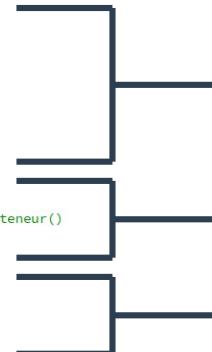
Contexte

Application

Explications

Problèmes

```
3     Window::Window(int masseMax, int nombrePorts, int nombreConteneurLargeur, int nombreConteneurLongueur,int nombreTotalConteneurs):QWidget ()
4     {
5
6         this->setWindowTitle("Automatisation remplissage paquebot");
7
8         //Initialisation des attributs
9
10        m_masseMax = new int(masseMax);
11        m_masseTotale = new float(0);
12        m_nombrePorts = new int(nombrePorts);
13        m_nombreConteneurLargeur = new int(nombreConteneurLargeur);
14        m_nombreConteneurLongueur = new int(nombreConteneurLongueur);
15        m_totalNombreConteneurs = new int(nombreTotalConteneurs);
16        m_conteneursPlaces = new int(0);
17        m_listeConteneurs = new QList<QList<float>>;
18        *m_listeConteneurs += genererListeConteneur();           //On génère une liste de conteneurs d'après la méthode genererListeConteneur()
19
20
21        // On trie la liste des conteneurs: par destination décroissante, puis pour une même destination par masse décroissante
22
23        triInsert();
24
25
26
27        indice = new int(0);      //Variable nécessaire pour la méthode poseConteneur()
28
29
30        //Liste qui contiendra les coordonnées de chaque conteneur
31
32        listeCoordonnees = new QList<QList<QList<int>> ;
33
34
35
36        //Initialisation des widgets
37
38        m_boutonCommencerSimulation = new QPushButton("Commencer simulation");
39
40        m_barre = new QProgressBar;
41        m_barre->setRange(0,100);
42        m_barre->setValue(0);
43
44        etage1 = new QWidget;
45        etage1->setWindowTitle("Premier étage de conteneurs");
46        etage1->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
47
48        etage2 = new QWidget;
49        etage2->setWindowTitle("Deuxième étage de conteneurs");
50        etage2->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
51
52        etage3 = new QWidget;
53        etage3->setWindowTitle("Troisième et dernier étage de conteneurs");
54        etage3->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
```



Implémentation des attributs

Création d'une liste de conteneurs

Tri de la liste

Window fichier d'implémentation

Introduction

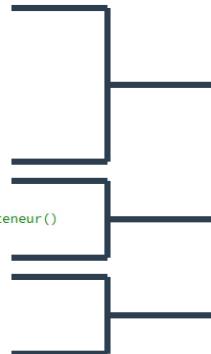
Contexte

Application

Explications

Problèmes

```
3 Window::Window(int masseMax, int nombrePorts, int nombreConteneurLargeur, int nombreConteneurLongueur,int nombreTotalConteneurs):QWidget ()  
4 {  
5  
6     this->setWindowTitle("Automatisation remplissage paquebot");  
7  
8     //Initialisation des attributs  
9  
10    m_masseMax = new int(masseMax);  
11    m_masseTotale = new float(0);  
12    m_nombrePorts = new int(nombrePorts);  
13    m_nombreConteneurLargeur = new int(nombreConteneurLargeur);  
14    m_nombreConteneurLongueur = new int(nombreConteneurLongueur);  
15    m_totalNombreConteneurs = new int(nombreTotalConteneurs);  
16    m_conteneursPlaces = new int(0);  
17    m_listeConteneurs = new QList<QList<float>>;  
18    *m_listeConteneurs += genererListeConteneur();           //On génère une liste de conteneurs d'après la méthode genererListeConteneur()  
19  
20  
21    // On trie la liste des conteneurs: par destination décroissante, puis pour une même destination par masse décroissante  
22  
23    triInsert();  
24  
25  
26  
27    indice = new int(0);      //Variable nécessaire pour la méthode poseConteneur()  
28  
29  
30    //Liste qui contiendra les coordonnées de chaque conteneur  
31  
32    listeCoordonnees = new QList<QList<QList<int>> ;  
33  
34  
35  
36    //Initialisation des widgets  
37  
38    m_boutonCommencerSimulation = new QPushButton("Commencer simulation");  
39  
40    m_barre = new QProgressBar;  
41    m_barre->setRange(0,100);  
42    m_barre->setValue(0);  
43  
44    etage1 = new QWidget;  
45    etage1->setWindowTitle("Premier étage de conteneurs");  
46    etage1->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);  
47  
48    etage2 = new QWidget;  
49    etage2->setWindowTitle("Deuxième étage de conteneurs");  
50    etage2->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);  
51  
52    etage3 = new QWidget;  
53    etage3->setWindowTitle("Troisième et dernier étage de conteneurs");  
54    etage3->setMinimumSize(nombreConteneurLargeur*50, nombreConteneurLongueur*80);
```



Implémentation des attributs

Création d'une liste de conteneurs

Tri de la liste



Implémentation des widgets

Les widgets etage1-2-3 sont les fenêtres représentant les conteneurs placés

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
57 //Initialisation des layouts
58
59 m_hlayout = new QHBoxLayout;
60
61 m_layout1 = new QGridLayout(etage1);
62 m_layout2 = new QGridLayout(etage2);
63 m_layout3 = new QGridLayout(etage3);
64
65 m_vlayout = new QVBoxLayout(this);      //On définit le layout principal de la fenêtre
66
67
68
69
70 //On définit les labels qui donneront les informations du porte-conteneurs
71
72 QLabel* masse = new QLabel;
73 masse->setText("La masse maximale est: " + QString::number(masseMax) + " tonnes\n");
74
75 QLabel *masseCont = new QLabel;
76 masseCont->setText("La masse totale de conteneur sur le paquebot est: " + QString::number(*m_masseTotale) + " tonnes\n");
77
78 QLabel *ports = new QLabel;
79 ports->setText("Le porte conteneur passe par " + QString::number(nombrePorts) + " ports\n");
80
81 QLabel *conteneurLargeur = new QLabel;
82 conteneurLargeur->setText("On peut placer " + QString::number(nombreConteneurLargeur) + " conteneurs en largeur\n");
83
84 QLabel *conteneurLongueur = new QLabel;
85 conteneurLongueur->setText("On peut placer " + QString::number(nombreConteneurLongueur) + " conteneurs en longueur\n");
86
87 QLabel *conteneurTotal = new QLabel;
88 conteneurTotal->setText("Le nombre de conteneur total que l'on peut placer sur le porte conteneur est: "
89                         + QString::number(nombreConteneurLargeur*nombreConteneurLongueur*3) + "\n");
90
91 QLabel *conteneur = new QLabel;
92 conteneur->setText("Le nombre de conteneur à placer est: " + QString::number(nombreTotalConteneurs) + "\n");
93
94
```

Implémentation des layouts. Les 'QGridLayout' sont des tableaux pour y insérer les conteneurs

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
57 //Initialisation des layouts
58
59 m_hlayout = new QHBoxLayout;
60
61 m_layout1 = new QGridLayout(etage1);
62 m_layout2 = new QGridLayout(etage2);
63 m_layout3 = new QGridLayout(etage3);
64
65 m_vlayout = new QVBoxLayout(this);      //On définit le layout principal de la fenêtre
66
67
68
69
70 //On définit les labels qui donneront les informations du porte-conteneurs
71
72 QLabel* masse = new QLabel;
73 masse->setText("La masse maximale est: " + QString::number(masseMax) + " tonnes\n");
74
75 QLabel *masseCont = new QLabel;
76 masseCont->setText("La masse totale de conteneur sur le paquebot est: " + QString::number(*m_masseTotale) + " tonnes\n");
77
78 QLabel *ports = new QLabel;
79 ports->setText("Le porte conteneur passe par " + QString::number(nombrePorts) + " ports\n");
80
81 QLabel *conteneurLargeur = new QLabel;
82 conteneurLargeur->setText("On peut placer " + QString::number(nombreConteneurLargeur) + " conteneurs en largeur\n");
83
84 QLabel *conteneurLongueur = new QLabel;
85 conteneurLongueur->setText("On peut placer " + QString::number(nombreConteneurLongueur) + " conteneurs en longueur\n");
86
87 QLabel *conteneurTotal = new QLabel;
88 conteneurTotal->setText("Le nombre de conteneur total que l'on peut placer sur le porte conteneur est: "
89                         + QString::number(nombreConteneurLargeur*nombreConteneurLongueur*3) + "\n");
90
91 QLabel *conteneur = new QLabel;
92 conteneur->setText("Le nombre de conteneur à placer est: " + QString::number(nombreTotalConteneurs) + "\n");
```

Implémentation des layouts. Les 'QGridLayout' sont des tableaux pour y insérer les conteneurs

Implémentation des 'Qlabel' contenant les informations concernant le paquebot

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
95 //On ajoute les widgets aux layouts
96
97 m_hlayout->addWidget(m_barre);
98 m_hlayout->addWidget(m_boutonCommencerSimulation);
99
100 m_vlayout->addWidget(masse);
101 m_vlayout->addWidget(masseCont);
102 m_vlayout->addWidget(ports);
103 m_vlayout->addWidget(conteneurLargeur);
104 m_vlayout->addWidget(conteneurLongueur);
105 m_vlayout->addWidget(conteneurTotal);
106 m_vlayout->addWidget(conteneur);
107
108
109 //On ajoute le layout secondaire (horizontal) dans le layout principal de la fenêtre(vertical)
110 m_vlayout->addLayout(m_hlayout);
111
112
113
114 //On exécute la méthode remplissagePaquebot()
115 remplissagePaquebot();
116
117
118 //On définit un timer pour plus tard
119
120 m_timer = new QTimer(this);
121
122
123 //Pour finir on connecte le bouton commencer simulation au slot poseConteneur()
124 connect(m_boutonCommencerSimulation, SIGNAL(clicked()), this, SLOT(simulation()));
125
126 }
```

Ajout des
widgets aux
layouts

Window fichier d'implémentation

Introduction

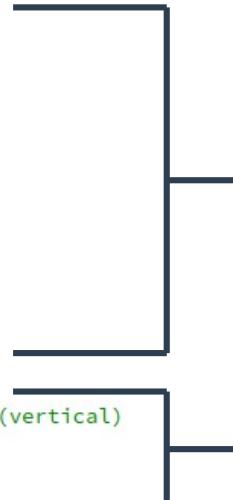
Contexte

Application

Explications

Problèmes

```
95 //On ajoute les widgets aux layouts  
96  
97     m_hlayout->addWidget(m_barre);  
98     m_hlayout->addWidget(m_boutonCommencerSimulation);  
99  
100    m_vlayout->addWidget(masse);  
101    m_vlayout->addWidget(masseCont);  
102    m_vlayout->addWidget(ports);  
103    m_vlayout->addWidget(conteneurLargeur);  
104    m_vlayout->addWidget(conteneurLongueur);  
105    m_vlayout->addWidget(conteneurTotal);  
106    m_vlayout->addWidget(conteneur);  
107  
108  
109 //On ajoute le layout secondaire (horizontal) dans le layout principal de la fenêtre(vertical)  
110  
111     m_vlayout->addLayout(m_hlayout);  
112  
113  
114 //On exécute la méthode remplissagePaquebot()  
115 remplissagePaquebot();  
116  
117  
118 //On définit un timer pour plus tard  
119  
120     m_timer = new QTimer(this);  
121  
122  
123 //Pour finir on connecte le bouton commencer simulation au slot poseConteneur()  
124 connect(m_boutonCommencerSimulation, SIGNAL(clicked()), this, SLOT(simulation()));  
125  
126 }
```



Ajout des widgets aux layouts

Ajout du layout secondaire au layout principal

Window fichier d'implémentation

Introduction

Contexte

Application

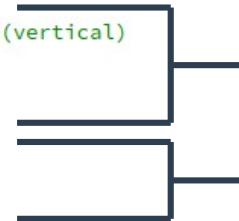
Explications

Problèmes

```
95 //On ajoute les widgets aux layouts  
96  
97     m_hlayout->addWidget(m_barre);  
98     m_hlayout->addWidget(m_boutonCommencerSimulation);  
99  
100    m_vlayout->addWidget(masse);  
101    m_vlayout->addWidget(masseCont);  
102    m_vlayout->addWidget(ports);  
103    m_vlayout->addWidget(conteneurLargeur);  
104    m_vlayout->addWidget(conteneurLongueur);  
105    m_vlayout->addWidget(conteneurTotal);  
106    m_vlayout->addWidget(conteneur);  
107  
108  
109 //On ajoute le layout secondaire (horizontal) dans le layout principal de la fenêtre(vertical)  
110  
111     m_vlayout->addLayout(m_hlayout);  
112  
113  
114 //On exécute la méthode remplissagePaquebot()  
115 remplissagePaquebot();  
116  
117  
118 //On définit un timer pour plus tard  
119  
120     m_timer = new QTimer(this);  
121  
122  
123 //Pour finir on connecte le bouton commencer simulation au slot poseConteneur()  
124 connect(m_boutonCommencerSimulation, SIGNAL(clicked()), this, SLOT(simulation()));  
125  
126 }
```



Ajout des widgets aux layouts



Ajout du layout secondaire au layout principal

Execution de la méthode 'remplissagePaquebot'

Window fichier d'implémentation

Introduction

Contexte

Application

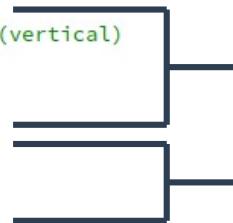
Explications

Problèmes

```
95 //On ajoute les widgets aux layouts  
96  
97     m_hlayout->addWidget(m_barre);  
98     m_hlayout->addWidget(m_boutonCommencerSimulation);  
99  
100    m_vlayout->addWidget(masse);  
101    m_vlayout->addWidget(masseCont);  
102    m_vlayout->addWidget(ports);  
103    m_vlayout->addWidget(conteneurLargeur);  
104    m_vlayout->addWidget(conteneurLongueur);  
105    m_vlayout->addWidget(conteneurTotal);  
106    m_vlayout->addWidget(conteneur);  
107  
108  
109 //On ajoute le layout secondaire (horizontal) dans le layout principal de la fenêtre(vertical)  
110  
111     m_vlayout->addLayout(m_hlayout);  
112  
113  
114 //On exécute la méthode remplissagePaquebot()  
115 remplissagePaquebot();  
116  
117  
118 //On définit un timer pour plus tard  
119  
120     m_timer = new QTimer(this);  
121  
122  
123 //Pour finir on connecte le bouton commencer simulation au slot poseConteneur()  
124 connect(m_boutonCommencerSimulation, SIGNAL(clicked()), this, SLOT(simulation()));  
125  
126 }
```



Ajout des widgets aux layouts



Ajout du layout secondaire au layout principal



Execution de la méthode 'remplissagePaquebot'



Connexion du slot simulation au bouton commencer

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

La méthode ‘remplissagePaquebot’ va ajouter à la liste ‘listeCoordonnees’ des coordonnées. Ensuite le slot ‘poseConteneur’ liera les listes ‘listeCoordonnees’ avec la liste ‘m_listeConteneurs’ contenant les informations sur chaque conteneur.

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

Etage 1

Etage 2

Etage 3

listeCoordonnees

La liste des coordonnées contient 3 listes : celles des coordonnées des conteneurs pour chaque étage.

Window fichier d'implémentation

Introduction

Contexte

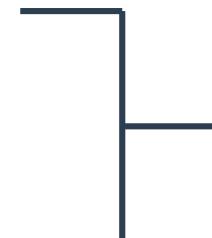
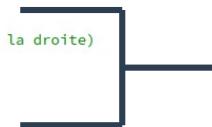
Application

Explications

Problèmes

```
131 void Window::remplissagePaquebot()
132 {
133
134     //soit x0 la ligne du centre de gravité et y0 sa colonne (l'axe des x descendant et l'axe des y allant vers la droite)
135     int x0 = floor(*m_nombreConteneurLongueur/2) ;
136     int y0 = floor(*m_nombreConteneurLargeur/2) ;
137     int nombreConteneursPlaces = 0;
138
139
140     for (int etage(0); etage < 3; etage++)
141     {
142         int nombreConteneursEtage =0;
143         QList<QList<int>> * coordonnees = new QList<QList<int>>;
144         //D'abord remplissage du carré de centre le centre de gravité et de côté le nombre de conteneur
145         //en largeur/2 (partie entière) en 'escargot'
146
147         QList<int> *grav = new QList<int> ;
148         if (nombreConteneursPlaces < *m_totalNombreConteneurs)
149         {
150             grav->append(x0);
151             grav->append(y0);
152
153             //On place le centre de gravité en 1er dans la liste
154             coordonnees->append(*grav );
155             nombreConteneursPlaces++;
156             nombreConteneursEtage++;
157         }
158
159         int k =1;      //indice de la "couronne" autour du centre de gravité sur laquelle on se trouve
160
161         //On remplit le carré de côté le nombre de conteneur en largeur
162         do
163         {
164             for (int i(0); i < 2*k; i++)
165             {
166                 if (nombreConteneursPlaces < *m_totalNombreConteneurs
167                     && nombreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
168                 {
169                     QList<int> *point = new QList<int>;
170                     point->append(x0 -k+1 + i);
171                     point->append(y0 - k);
172                     coordonnees->append(*point);
173                     nombreConteneursPlaces++;
174                     nombreConteneursEtage++;
175                 }
176             }
177         }
```

On définit le « centre de gravité »



Et on le place

Window fichier d'implémentation

Introduction

Contexte

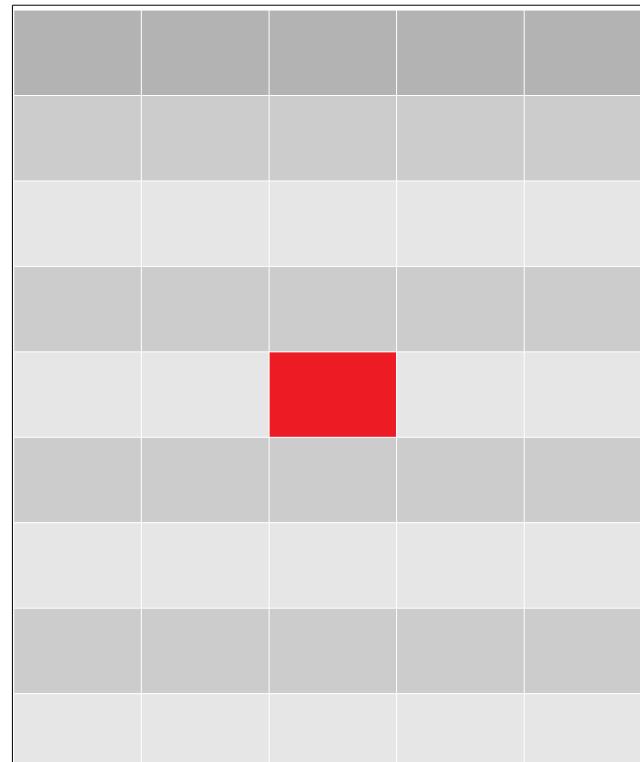
Application

Explications

Problèmes

0

y



x

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
131 void Window::remplissagePaquebot()
132 {
133
134     //soit x0 la ligne du centre de gravité et y0 sa colonne (l'axe des x descendant et l'axe des y allant vers la droite)
135     int x0 = floor(*m_nombreConteneurLongueur/2) ;
136     int y0 = floor(*m_nombreConteneurLargeur/2) ;
137     int nombreConteneursPlaces = 0;
138
139     for (int etage(0); etage < 3; etage++)
140     {
141         int nombreConteneursEtage =0;
142         QList<QList<int>> * coordonnees = new QList<QList<int>>;
143         //D'abord remplissage du carré de centre le centre de gravité et de côté le nombre de conteneur
144         //en largeur/2 (partie entière) en 'escargot'
145
146         QList<int> *grav = new QList<int> ;
147         if (nombreConteneursPlaces < *m_totalNombreConteneurs)
148         {
149             grav->append(x0);
150             grav->append(y0);
151
152             //On place le centre de gravité en 1er dans la liste
153             coordonnees->append(*grav );
154             nombreConteneursPlaces++;
155             nombreConteneursEtage++;
156         }
157
158         int k =1;      //indice de la "couronne" autour du centre de gravité sur laquelle on se trouve
159
160         //On remplit le carré de côté le nombre de conteneur en largeur
161         do
162         {
163             for (int i(0); i < 2*k; i++)
164             {
165                 if (nombreConteneursPlaces < *m_totalNombreConteneurs
166                     && nombreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
167                 {
168                     QList<int> *point = new QList<int>;
169                     point->append(x0 -k+1 + i);
170                     point->append(y0 - k);
171                     coordonnees->append(*point);
172                     nombreConteneursPlaces++;
173                     nombreConteneursEtage++;
174                 }
175
176             }
177         }
```

Ensuite on commence à remplir « en escargot »

Window fichier d'implémentation

Introduction

Contexte

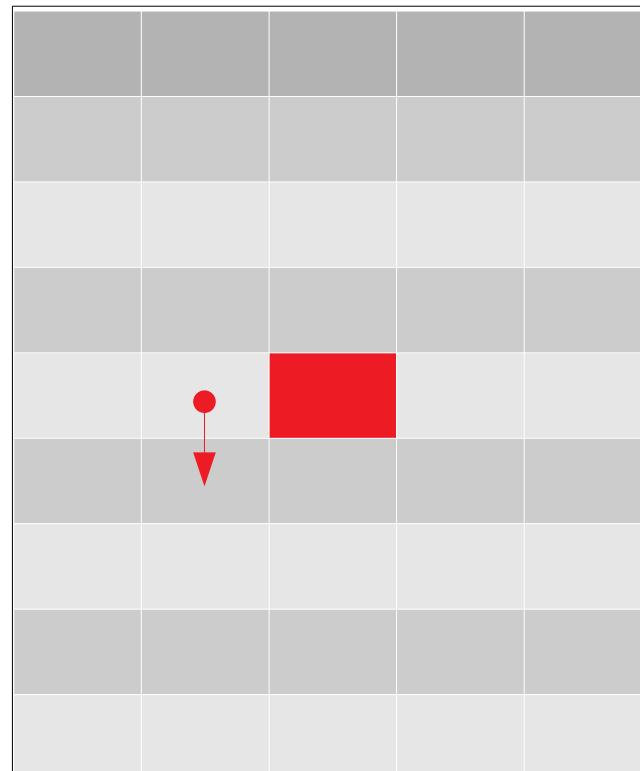
Application

Explications

Problèmes

0

y



x

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
176
177
178 ▼
179
180 | if (nbreConteneursPlaces < *m_totalNombreConteneurs
181 |   && nbreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur )
182 {
183     QList<int> *point = new QList<int>;
184     point->append(x0+k);
185     point->append(y0-k+1 + i);
186     coordonnees->append(*point);
187     nbreConteneursPlaces++;
188     nbreConteneursEtage++;
189 }
190
191 }
192 | for (int i(0); i<2*k; i++)
193 {
194 |   if (nbreConteneursPlaces < *m_totalNombreConteneurs
195 |     && nbreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
196 |
197 |     QList<int> *point = new QList<int>;
198 |     point->append(x0+k-i-1);
199 |     point->append(y0 + k);
200 |     coordonnees->append(*point);
201 |     nbreConteneursPlaces++;
202 |     nbreConteneursEtage++;
203 }
204
205 }
206 | for (int i(0); i<2*k; i++)
207 {
208 |   if (nbreConteneursPlaces < *m_totalNombreConteneurs
209 |     && nbreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur )
210 |
211 |     QList<int> *point = new QList<int>;
212 |     point->append(x0-k);
213 |     point->append(y0+k-i-1);
214 |     coordonnees->append(*point);
215 |     nbreConteneursPlaces++;
216 |     nbreConteneursEtage++;
217 }
218
219
220 }
221
222 k++;
223
224 }while (k <= floor((*m_nombreConteneurLargeur/2)
225 -(*m_nombreConteneurLargeur-1)%2) && nbreConteneursPlaces < *m_totalNombreConteneurs );
```

On continue le remplissage
« en escargot »

Window fichier d'implémentation

Introduction

Contexte

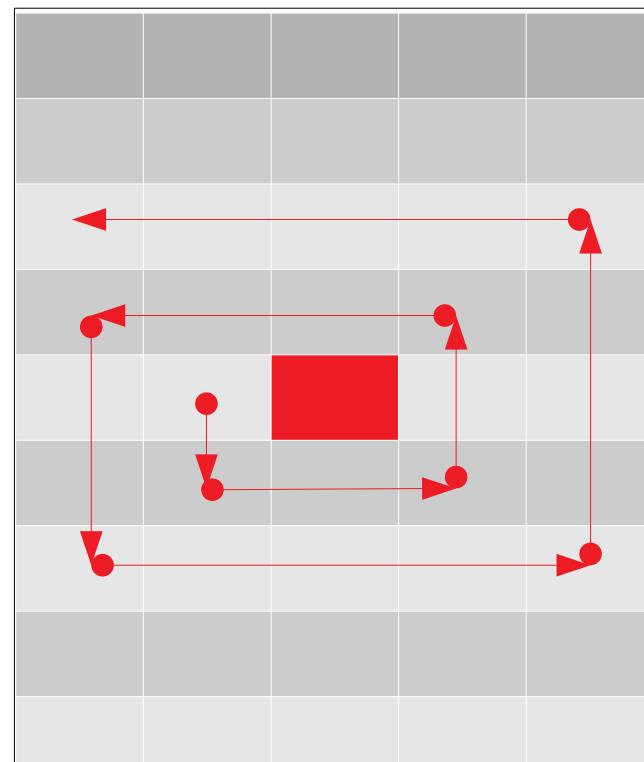
Application

Explications

Problèmes

0

y



x

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
228 if (*m_nombreConteneurLargeur%2 ==0)
229 {
230     for (int j(0); j < 2*k-1; j++)
231     {
232         if (nmbreConteneursPlaces < *m_totalNombreConteneurs
233             && nmbreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
234         {
235             QList<int> *point = new QList<int>;
236             point->append(x0-k+1+j);
237             point->append(0);
238             coordonnees->append(*point);
239             nmbreConteneursPlaces++;
240             nmbreConteneursEtage++;
241         }
242     }
243 }
244 }
```

Si le nombre de conteneurs
en largeur est pair

Window fichier d'implémentation

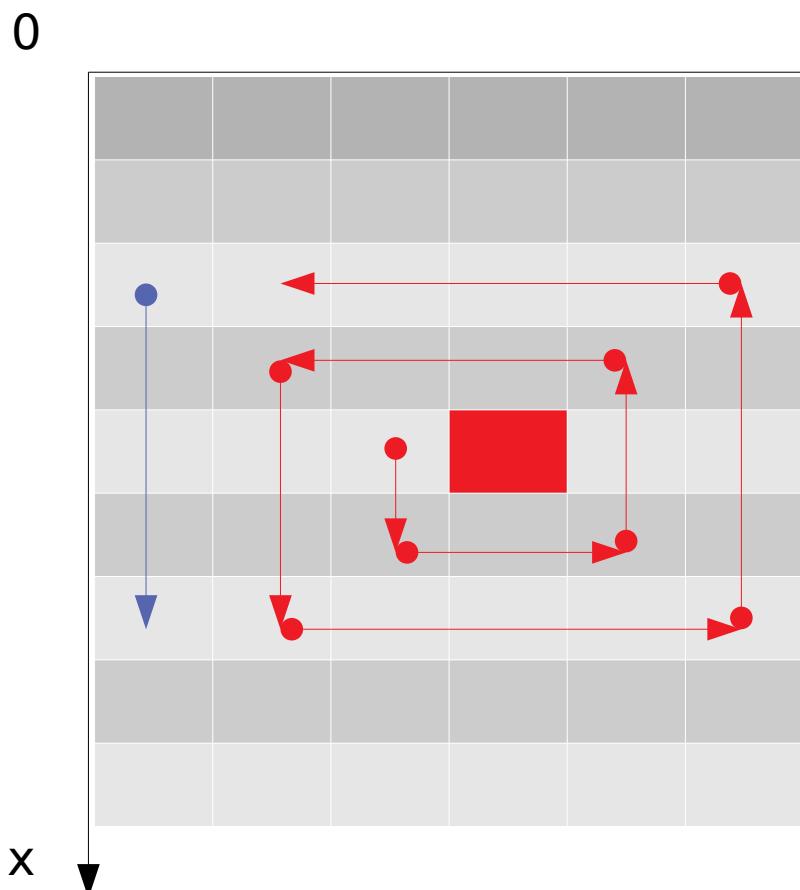
Introduction

Contexte

Application

Explications

Problèmes



Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
245
246     int i =0;
247     do{
248         //On alterne le remplissage par ligne (une en haut, une en bas), x varie de 0 à nombreConteneurLargeur -1
249
250         if (pow(-1,i)>0)
251         {
252             for (int j(0); j < *m_nombreConteneurLargeur; j++)
253             {
254                 if (nombreConteneursPlaces < *m_totalNombreConteneurs
255                     && nombreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
256                 {
257                     QList<int> *point = new QList<int>;
258                     point->append(x0-k);
259                     point->append(j);
260                     coordonnees->append(*point);
261                     nombreConteneursPlaces++;
262                     nombreConteneursEtage++;
263                 }
264
265             }
266
267         if(pow(-1,i) <0)
268         {
269             for (int j(0); j < *m_nombreConteneurLargeur; j++)
270             {
271                 if (nombreConteneursPlaces < *m_totalNombreConteneurs
272                     && nombreConteneursEtage < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
273                 {
274                     QList<int> *point = new QList<int>;
275                     point->append(x0+k);
276                     point->append(j);
277                     coordonnees->append(*point);
278                     nombreConteneursPlaces++;
279                     nombreConteneursEtage++;
280                 }
281
282             }
283
284         }
285
286         k++;
287     }
288
289     i++;
290
291 }while(nombreConteneursPlaces < *m_totalNombreConteneurs
292       && nombreConteneursEtage < *m_nombreConteneurLargeur* *m_nombreConteneurLongueur);
```

Le remplissage se finit
ligne par ligne en
commençant par celle du
haut

Window fichier d'implémentation

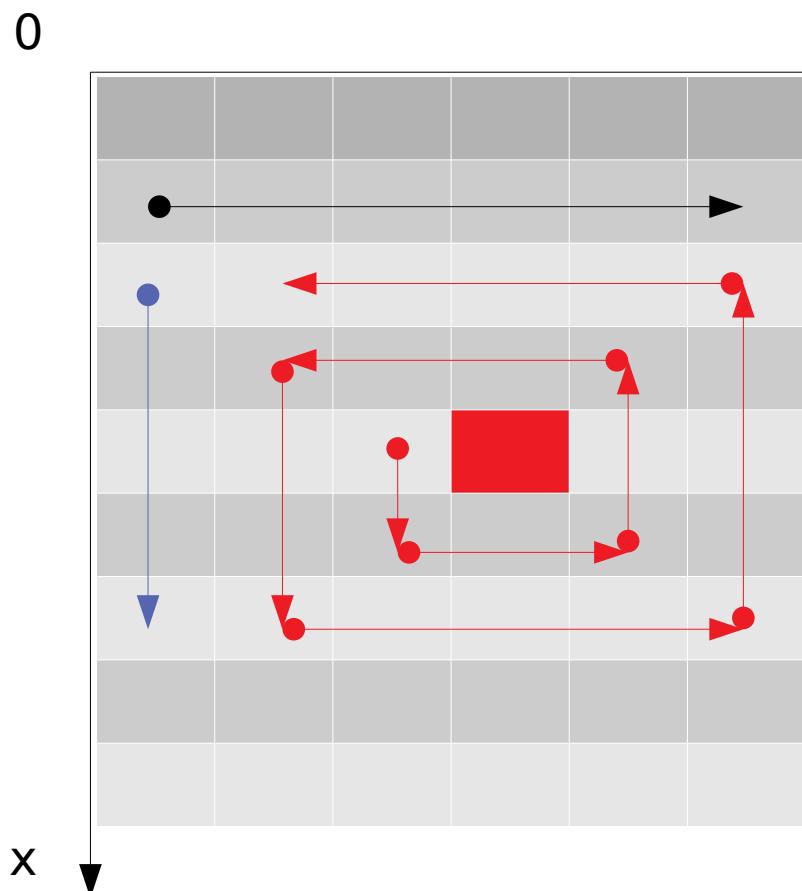
Introduction

Contexte

Application

Explications

Problèmes



Window fichier d'implémentation

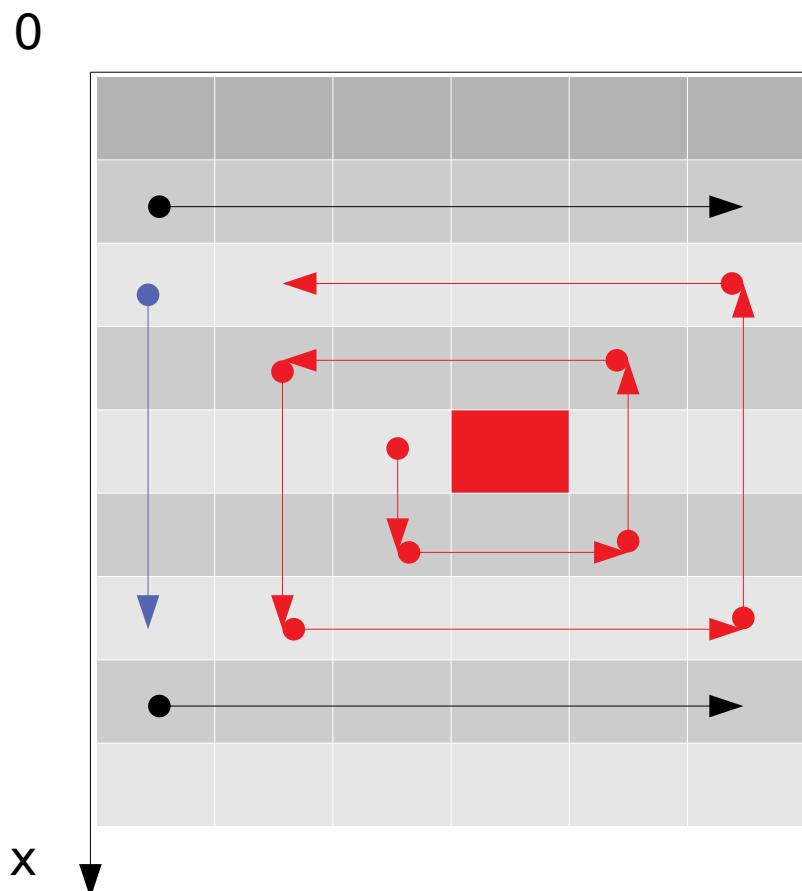
Introduction

Contexte

Application

Explications

Problèmes



Window fichier d'implémentation

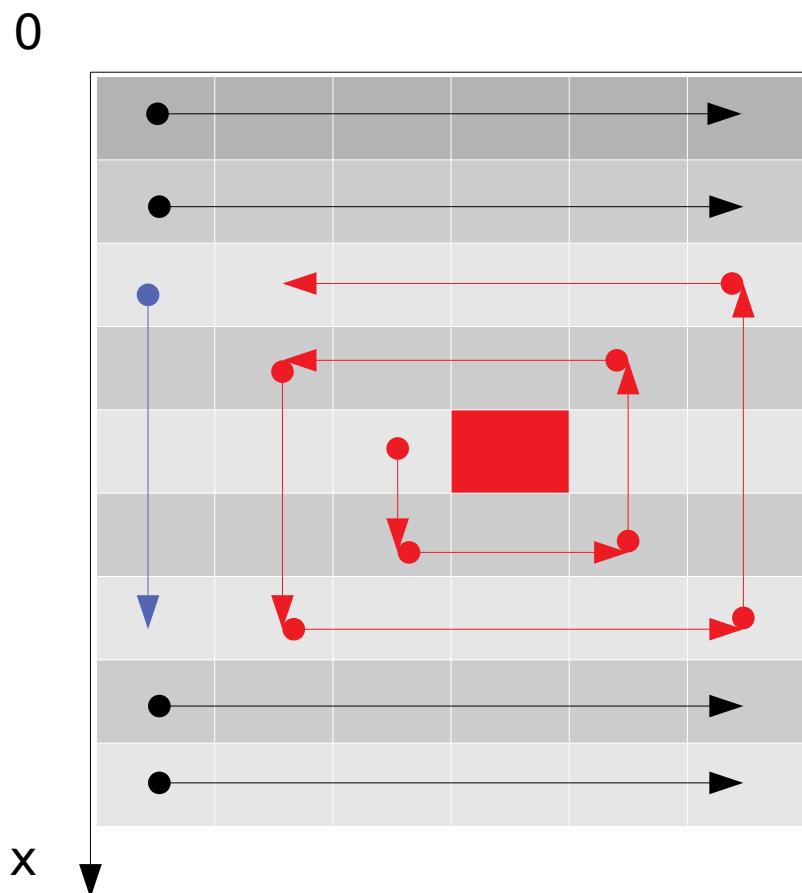
Introduction

Contexte

Application

Explications

Problèmes



Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
295  
296  
297 //Puis on ajoute les coordonnées par étage à la liste correspondante  
298 listeCoordonnees->append(*coordonnees);  
299 }  
300 }  
301 }
```

Pour finir, on ajoute la liste de coordonnées d'un étage à 'listeCoordonnees'

Toute cette opération est répétée 3 fois, une fois pour chaque étage et tant qu'il reste des conteneurs à placer

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
304 QList<QList<float>> Window::genererListeConteneur()
305 {
306
307     QList<QList<float>> listeConteneurs;
308     int i(0);
309     while (i < *m_totalNombreConteneurs && floor(*m_masseTotale + 2.1) < *m_masseMax)
310     {
311         QList<float> listeInformationConteneurs;
312         int masse = rand()%600 + 1500;      //Genere une masse aléatoire entre 1500 et 2100 kg
313         listeInformationConteneurs.append((float)masse/1000); //En tonnes
314         int destination = rand()%*m_nombrePorts + 1;      //Genere une destination aléatoire entre 1 et le nombre de ports
315         listeInformationConteneurs.append(destination);
316         listeConteneurs.append(listeInformationConteneurs);
317         i++;
318         *m_masseTotale += masse/1000;
319     }
320
321
322     return listeConteneurs;
323 }
```

Cette méthode génère et retourne une liste de conteneurs. Pour chaque conteneur une masse aléatoire comprise entre 1,5 et 2,1 tonnes est générée ainsi qu'un numéro de port compris entre 1 et le nombre de port renseigné

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
326 void Window::triInsert()
327 {
328
329     //On trie le conteneurs dans la liste par masse décroissante et par destination décroissante (la fin au début)
330
331     //D'abord par destination
332
333     for (int i(1) ; i < m_listeConteneurs->size(); i++)
334     {
335         float pivot = m_listeConteneurs[0][i][1];
336         QList<float> temp = m_listeConteneurs[0][i];
337         int j = i-1;
338         while (j>=0 && pivot > m_listeConteneurs[0][j][1])
339         {
340             m_listeConteneurs[0][j+1] = m_listeConteneurs[0][j];
341             j--;
342         }
343         m_listeConteneurs[0][j+1] = temp;
344
345
346     // Ensuite par masse en conservant la liste triée par destination:
347     for (int i(1); i< m_listeConteneurs->size(); i++)
348     {
349         float pivot = m_listeConteneurs[0][i][0];
350         QList<float> temp = m_listeConteneurs[0][i];
351         int j = i-1;
352         while ( j >= 0 && m_listeConteneurs[0][j][1] == m_listeConteneurs[0][i][1] && pivot > m_listeConteneurs[0][j][0])
353         {
354             m_listeConteneurs[0][j+1] = m_listeConteneurs[0][j];
355             j--;
356         }
357         m_listeConteneurs[0][j+1] = temp;
358     }
359
360
361 void Window::simulation()
362 {
363     //Créer un QTimer que je connecte au slot et tous les x temps definis, le slot est appellé
364     connect(m_timer, SIGNAL(timeout()), this, SLOT(poseConteneur()));
365     m_timer->start(100);
366 }
```

Double tri par insertion : tout d'abord par numéro de port décroissant, puis pour un même port par masse décroissante

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

La liste des conteneurs est triée par numéro de ports décroissants pour placer en premier, donc en dessous, les conteneurs qui doivent être déchargés au dernier port.

De plus, la liste est triée par masses décroissantes pour placer au centre les conteneurs les plus lourds.

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
326 void Window::triInsert()
327 {
328
329     //On trie le conteneurs dans la liste par masse décroissante et par destination décroissante (la fin au début)
330
331     //D'abord par destination
332
333     for (int i(1) ; i < m_listeConteneurs->size(); i++)
334     {
335         float pivot = m_listeConteneurs[0][i][1];
336         QList<float> temp = m_listeConteneurs[0][i];
337         int j = i-1;
338         while (j>=0 && pivot > m_listeConteneurs[0][j][1])
339         {
340             m_listeConteneurs[0][j+1] = m_listeConteneurs[0][j];
341             j--;
342         }
343         m_listeConteneurs[0][j+1] = temp;
344
345
346     // Ensuite par masse en conservant la liste triée par destination:
347     for (int i(1); i< m_listeConteneurs->size(); i++)
348     {
349         float pivot = m_listeConteneurs[0][i][0];
350         QList<float> temp = m_listeConteneurs[0][i];
351         int j = i-1;
352         while ( j >= 0 && m_listeConteneurs[0][j][1] == m_listeConteneurs[0][i][1] && pivot > m_listeConteneurs[0][j][0])
353         {
354             m_listeConteneurs[0][j+1] = m_listeConteneurs[0][j];
355             j--;
356         }
357         m_listeConteneurs[0][j+1] = temp;
358     }
359
360
361 void Window::simulation()
362 {
363     //Créer un QTimer que je connecte au slot et tous les x temps definis, le slot est appellé
364     connect(m_timer, SIGNAL(timeout()), this, SLOT(poseConteneur()));
365     m_timer->start(100);
366 }
```

Double tri par insertion : tout d'abord par numéro de port décroissant, puis pour un même port par masse décroissante

Slot lié au bouton 'commencerSimulation'

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

Le slot ‘simulation’ définit un timer de 100ms et connecte ce timer au slot ‘poseConteneur’ par le signal ‘timeout’. Ainsi, le slot ‘poseConteneur’ sera appelé toutes les 100ms.

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
368 void Window::poseConteneur()
369 {
370     if (*indice < *m_totalNombreConteneurs )
371     {
372         QLabel *label = new QLabel;
373         label->setFixedSize(30,60);
374         label->setCursor(Qt::PointingHandCursor);
375         QString *infos = new QString;
376         *infos = "Masse: "+QString::number(m_listeConteneurs[0][*indice][0])
377             + " tonnes \nPort: " + QString::number(m_listeConteneurs[0][*indice][1]);
378         label->setToolTip(*infos);
379         if (*indice < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
380         {
381             if (*indice == 0)
382                 etage1->show();
383
384             label->setStyleSheet(QString::fromUtf8("background-color:rgb(255,0,0);")); //Premier étage rempli de conteneurs rouges
385             m_glayout1->addWidget(label, listeCoordonnees[0][0][*indice][0], listeCoordonnees[0][0][*indice][1]);
386
387         }
388         else if (*indice < (*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2)
389         {
390             if (*indice == *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
391                 etage2->show();
392
393             int j = *indice -*m_nombreConteneurLargeur * *m_nombreConteneurLongueur;
394             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,255,0);")); //Deuxième étage rempli de conteneurs verts
395             m_glayout2->addWidget(label, listeCoordonnees[0][1][j][0], listeCoordonnees[0][1][j][1]);
396         }
397         else
398         {
399             if (*indice == (*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2 )
400                 etage3->show();
401
402             int j = *indice -(*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2;
403             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,0,255);")); //Troisième étage rempli de conteneurs bleus
404             m_glayout3->addWidget(label, listeCoordonnees[0][2][j][0], listeCoordonnees[0][2][j][1]);
405         }
406         *indice +=1;
407     }
408
409     m_barre->setValue(floor(*indice * 100/ *m_totalNombreConteneurs));
410
411     if (*indice == *m_totalNombreConteneurs)
412     {
413         QMessageBox::information(this, "Fin de tâche", "Tous les conteneurs ont été placés avec succès!");
414         m_timer->deleteLater();
415     }
416
417
418 }
```

Colore la case associée aux coordonnées de la liste 'listeCoordonnees' pour chacun de ces éléments pour la visualisation et y associe les informations du conteneur

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
368 void Window::poseConteneur()
369 {
370     if (*indice < *m_totalNombreConteneurs )
371     {
372         QLabel *label = new QLabel;
373         label->setFixedSize(30,60);
374         label->setCursor(Qt::PointingHandCursor);
375         QString *infos = new QString;
376         *infos = "Masse: "+QString::number(m_listeConteneurs[0][*indice][0])
377             + " tonnes \nPort: " + QString::number(m_listeConteneurs[0][*indice][1]);
378         label->setToolTip(*infos);
379         if (*indice < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
380         {
381             if (*indice == 0)
382                 etage1->show();
383
384             label->setStyleSheet(QString::fromUtf8("background-color:rgb(255,0,0);")); //Premier étage rempli de conteneurs rouges
385             m_glayout1->addWidget(label, listeCoordonnees[0][0][*indice][0], listeCoordonnees[0][0][*indice][1]);
386
387         }
388         else if (*indice <(*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2)
389         {
390             if (*indice == *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
391                 etage2->show();
392
393             int j = *indice -*m_nombreConteneurLargeur * *m_nombreConteneurLongueur;
394             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,255,0);")); //Deuxième étage rempli de conteneurs verts
395             m_glayout2->addWidget(label, listeCoordonnees[0][1][j][0], listeCoordonnees[0][1][j][1]);
396         }
397         else
398         {
399             if (*indice == (*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2 )
400                 etage3->show();
401
402             int j = *indice -( *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2;
403             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,0,255);")); //Troisième étage rempli de conteneurs bleus
404             m_glayout3->addWidget(label, listeCoordonnees[0][2][j][0], listeCoordonnees[0][2][j][1]);
405
406         }
407         *indice +=1;
408     }
409
410     m_barre->setValue(floor(*indice * 100/ *m_totalNombreConteneurs));
411
412     if (*indice == *m_totalNombreConteneurs)
413     {
414         QMessageBox::information(this, "Fin de tâche", "Tous les conteneurs ont été placés avec succès!");
415         m_timer->deleteLater();
416     }
417 }
418 }
```

Colore la case associée aux coordonnées de la liste 'listeCoordonnees' pour chacun de ces éléments pour la visualisation et y associe les informations du conteneur

Evolution de la barre de chargement

Window fichier d'implémentation

Introduction

Contexte

Application

Explications

Problèmes

```
368 void Window::poseConteneur()
369 {
370     if (*indice < *m_totalNombreConteneurs )
371     {
372         QLabel *label = new QLabel;
373         label->setFixedSize(30,60);
374         label->setCursor(Qt::PointingHandCursor);
375         QString *infos = new QString;
376         *infos = "Masse: "+QString::number(m_listeConteneurs[0][*indice][0])
377             + " tonnes \nPort: " + QString::number(m_listeConteneurs[0][*indice][1]);
378         label->setToolTip(*infos);
379         if (*indice < *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
380         {
381             if (*indice == 0)
382                 etage1->show();
383
384             label->setStyleSheet(QString::fromUtf8("background-color:rgb(255,0,0);")); //Premier étage rempli de conteneurs rouges
385             m_glayout1->addWidget(label, listeCoordonnees[0][0][*indice][0], listeCoordonnees[0][0][*indice][1]);
386
387         }
388         else if (*indice < (*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2)
389         {
390             if (*indice == *m_nombreConteneurLargeur * *m_nombreConteneurLongueur)
391                 etage2->show();
392
393             int j = *indice -*m_nombreConteneurLargeur * *m_nombreConteneurLongueur;
394             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,255,0);")); //Deuxième étage rempli de conteneurs verts
395             m_glayout2->addWidget(label, listeCoordonnees[0][1][j][0], listeCoordonnees[0][1][j][1]);
396         }
397         else
398         {
399             if (*indice == (*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2 )
400                 etage3->show();
401
402             int j = *indice -(*m_nombreConteneurLargeur * *m_nombreConteneurLongueur)*2;
403             label->setStyleSheet(QString::fromUtf8("background-color:rgb(0,0,255);")); //Troisième étage rempli de conteneurs bleus
404             m_glayout3->addWidget(label, listeCoordonnees[0][2][j][0], listeCoordonnees[0][2][j][1]);
405
406         }
407         *indice +=1;
408     }
409
410     m_barre->setValue(floor(*indice * 100/ *m_totalNombreConteneurs));
411
412     if (*indice == *m_totalNombreConteneurs)
413     {
414         QMessageBox::information(this, "Fin de tâche", "Tous les conteneurs ont été placés avec succès!");
415         m_timer->deleteLater();
416     }
417 }
418 }
```

Colore la case associée aux coordonnées de la liste 'listeCoordonnees' pour chacun de ces éléments pour la visualisation et y associe les informations du conteneur

Evolution de la barre de chargement
Message lors de la fin du chargement et arrêt du timer