|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Informaticien/-ne CFC**  Travail pratique individuel 2024 (TPI) |  | |

Nom du candidat : Elvin Kuci

Attention : Dernier délai de remise du rapport durant la réalisation du TPI le 18.06.2021 à 17h00 hormis la présentation (ce texte doit être effacé).

**« Road Traffic Simulator »**

### Sommaire

Résumé du rapport du TPI 2

1 Les grandes lignes du projet 3

1.1 Analyse de la situation initiale 3

1.2 Analyse de l’état désiré 3

1.3 Cahier des charges / exigences du système 3

1.4 Organisation du projet 3

2 Travaux préparatoires 5

2.1 Installation de l’environnement 5

2.2 Création d'un projet JavaFX 5

2.3 JavaFX-TT 6

2.4 Moving-TT 8

3 Analyse 11

3.1 Analyse de l’état désiré 11

3.2 Variantes 13

3.3 Sécurité de l’information et protection des données 15

4 Conception 16

4.1 Exigences du système 16

4.2 Architecture du système 16

4.3 Maquettes 16

4.4 Concept d‘implémentation 17

4.5 Concept de tests 19

5 Réalisation 20

5.1 Spécifications détaillées 20

5.2 Design du système 20

5.3 Configuration xyz 20

6 Test 21

6.1 Procédure de test 21

6.2 Protocol de test 21

6.3 Signature du protocole de test 21

7 Conclusion 22

7.1 Améliorations possibles 22

7.2 Auto-évaluation 22

8 Bibliographie: liste des sources et références 23

9 Glossaire 24

10 Signatures 25

11 Annexes 26

# Résumé du rapport du TPI

Le Résumé du rapport du TPI est une présentation conceptuelle du travail effectué et du résultat attendu d’au maximum une page A4 qui permettent au lecteur une rapide compréhension du rapport de travail. . Il contient trois paragraphes : Situation de départ, mise en œuvre, résultats.

Pour plus de détails, veuillez vous référez au Manuel ICT - partie B : Documentation / rapport du TPI - Question 14 - page 81.

# Les grandes lignes du projet

Ajout stratégie de sauvegarde

TODO : Checker c’est où

## Analyse de la situation initiale

Quelle est la situation actuelle? Vous pouvez éventuellement illustrer le processus.

Cette application n’a pas de situation initiale. Ce mandat est le premier mandat sur ce projet et tout est à faire depuis zéro.

## Analyse de l’état désiré

A quoi doit ressembler la solution/le produit fini? Vous pouvez éventuellement illustrer le nouveau processus.

L’objectif final est d’avoir une application pour ordinateur permettant de lancer une simulation d’un carrefour. Il sera possible de modifier et sauvegarder de nombreux paramètres telle que les feux, la vitesse de la route, etc… Cette application permettra de tester et visualiser un ensemble de paramètres appliqué sur un carrefour.

## Cahier des charges / exigences du système

Cahier des charges détaillé découlant du mandat de projet. Définition exacte et mesurable des objectifs à atteindre.

Décrire et structurer les exigences du système désiré.

L’application devra gérer les 4 tronçons de routes, à circulation bidirectionnelle, lié par un carrefour équipé de feux se signalisation. Une extension possible devra être la simulation de ces mêmes tronçons de route avec un rond-point. Ce développement ne devra pas être réalisé, mais pris en compte dans l’analyse et la conception, afin que l’application puisse être modifiée facilement.

L’objectif est aussi de fournir à l’utilisateur une application paramétrable pour agir sur le résultat visuel de la simulation :

* Échelle pour la représentation graphique en pixel par mètres
* Limitation de la vitesse, distance des et densité du trafic
* Temps de réaction, accélération, décélération et temps de sécurité
* Temps des feux et ordre de ceux-ci
* Sauvegarde du jeu de paramètre sous un nom

## Organisation du projet

Citer la méthode de gestion de projet utilisée (Méthode en phases, agile, scrum, …)

Etablir la liste des participants au projet avec leur rôle respectif.

Méthodologie en cascade / séquenciel

Support : planning + journal de travail (blablater dessus)

Décrire comment sont gérées les sauvegardes de la documentation et du code (fréquence, supports utilisés, utilisation de logiciel de gestion de version, …)

*TODO : Check methode with best BUDDY*

Méthode: Séquenciel ? Incrémental ? Cascade ?

Additionellement, voici la liste des participants au projet :

|  |  |
| --- | --- |
| Participant | Rôle |
| Kuci Elvin | Candidat, développeur de l’application |
| Bütschi Daniel | Supérieur professionnel |
| Hertling Frédéric | Formateur en entreprise |
| Perroud Didier | Expert principal |
| Thomet Samuel | Expert secondaire |

Le code et la documentation sont présent dans le Git du projet. Un commit est émis à chaque modification d’un élément dans le code et/ou la documentation.

Chaque groupe de commit majeur est édité sur sa propre branche et est merge sur la branche « main » une fois que le travail sur celle-ci est terminé.

Une fois que l’application atteint un objectif et est stable, une branche « version\_x.y.z » est créer. Cette branche sert de backup pour rapidement revenir à la dernière version stable.

Pour revenir à un ancien commit, exécutez un « git log » pour voir l’identifiant d’un commit et un « git reset IdDuCommit » pour revenir à l’ancien commit.

Pour aller sur une branche, faites un « git switch NomDeLaBranche ». Si vous n’avez pas la branche en local, faites un « git switch remotes/NomDuRemote/NomDeLaBranche –detach ».

*Ajout des exemples, images, et printscreen*

# Travaux préparatoires

## Installation de l’environnement

### Git

Pour installer Git, allez sur le site web [git-scm](https://git-scm.com/download/win) et installer la version voulue.

TODO : Mise en place du compte GitHub

### MySQL

TODO : Installation de MySQL ?

### Préparation et test de l’hébergement

TODO : Test et prep de l’hébergement

### Mise en place d’un projet JavaFX

Pour ce projet, NetBeans va être utilisé comme IDE. Vous pouvez installer NetBeans depuis le [site web officiel](https://netbeans.apache.org/front/main/download/).

Ensuite, pour créer un projet JavaFX, Allez dans « File > New Projet… » Et créer un projet « Java with Maven > FXML JavaFX Maven Archetype (Gluon) ». Maven va installer automatiquement JavaFX et mettre en place un projet de base. Vous pouvez ensuite lancer l’application et tout devrait fonctionner.

La raison pour laquelle nous allons utiliser Maven est car l’installation de JavaFX sera faite automatiquement. Additionnement, beaucoup plus de documentation est écrite sur les projets Maven JavaFX que Ant. Notez que l’architecture arm64 n’est pas supportée par Maven - JavaFX. JavaFX n’a pas un bon support de cette architecture et NetBeans causera de nombreuses difficultés durant la mise en place de l’application. Quelle est la situation actuelle ? Vous pouvez éventuellement illustrer le processus.

Vous pouvez aussi remarquer que Maven ajoute 1 fichier important : « module-info.java ». Ce fichier défini le module (l’application) et ses exports, imports, etc…

## Création d'un projet JavaFX

Le test technologique implémenté a pour but de tester la communication entre le Thread Java et le Thread JavaFX. Voici la liste des technologies qui vont être tester :

* Échange de données entre le Thread JavaFX et le Main Thread
* Expérimentation de Platform.runLater
* Expérimentation de Binding
* Expérimentation de Task

Le choix de sa technologie n’a pas été fait au hasard. La communication entre le « backend (Worker) » et le « frontend (JavaFX) » est une des parties les plus complexe de ce projet.

### Structure MVC avec JavaFX

A screenshot of a computer

Description automatically generatedLe projet est nommé JavaFX-TT et sera une application MVC (Model/Worker – View – Controller) simple.

Les Views sont dans le src/main/ressources/app/views au format FXML. Leur location n’est pas aléatoire, « src/main/ressources » est le dosser depuis lequel les ressources de l’application sont copié avec Maven. Vu que les FXML ne sont pas compiler, ce sont des ressources.

Le contrôleur se trouve dans « app/controller » et le Worker se trouve dans « app/workers ».

Figure 1

(Figure 1 : le projet sur NetBeans)

## JavaFX-TT

### Fonctionnement

Quand vous lancez l’application, vous allez voir un champ texte qui est lié à la « value » du label. Une fois que vous appuyez sur « Fetch data », le « value » affiche « Fetching data… » pendant 1 seconde et va charger des valeurs dans le tableau à droite.

Ce test n’est pas très utile mais permet de bien tester le fonctionnement des différentes technologies utile pour le projet

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |

### Points importants de l’implémentation

Je ne vais pas écrire tout le code de l’application ci-dessous mais uniquement les parties intéressantes.

/\*\*

\* Update the list content

\* @param values The list of values to put in the list

\*/

public void updateList(List<String> values) {

// Use of Platform.runLater because it will be called by an external thread.

Platform.runLater(() -> listTask.getItems().setAll(values));

}

Lors de la mise à jour de la liste de l’application, l’utilisation du Platform.runLater permet à un second thread de modifier le contenu de la liste. Sans ce Platform.runLater, la mise à jour de la liste pourrai créer un java.lang.IllegalStateException. Cette erreur n’apparait pas à chaque clique mais arrivera au bout d’un moment.

/\*\*

\* Bind the label to the input and enable the buttonRunTask.

\*/

private void bindLabelToInputText() {

label.textProperty().bind(textLabel.textProperty());

buttonRunTask.setDisable(false);

}

/\*\*

\* Unbind the label from the input and disable the buttonRunTask.

\*/

private void unbindLabelToInputText() {

label.textProperty().unbind();

buttonRunTask.setDisable(true);

}

Ces 2 fonctions créer/retire le lien entre le texte du label « value » avec le champ texte (textLabel). Sans cette fonctionnalisée, un autre moyen serai d’utiliser un callback ce qui serai beaucoup moins efficient et dure à gérer.

// Create a task to set the new string value

Task<Void> fetchTask = new Task() {

@Override

protected Void call() throws Exception {

// Simulate a long operation (1 second)

wrk.fetchData();

return null;

}

};

// On task success, rebind the text

fetchTask.setOnSucceeded((WorkerStateEvent e) -> {

bindLabelToInputText();

});

// Start the task

new Thread(fetchTask).start();

La création d’une tâche (Task) dans ce cas permet d’ajouter des callbacks pour l’exécution de la tâche. Ici, après que le Worker à récupérer les données, le texte est lié au champ de texte. Comme vous pouvez le voir, la tâche est lancée via la création et exécution d’un nouveau thread.

### Récapitulatifs

* Platform.runLater permet d’interagir avec le thread JavaFX.
* Les Bindings permettent de lier les valeurs avec entre elle.
* Les Task permettent une gestion avancée des Threads et un meilleur contrôle de flux.

## Moving-TT

Ce test technologique a pour but de tester le déplacement d’image sur l’écran. Ceci permet de mieux comprendre comme créer des animations et simulations sur le « backend » et les afficher sur le frontend.

### Fonctionnement

Quand vous lancerez cette application, vous verrez une flèche tourner autour du centre de la fenêtre. Cette application n’a aucune action intéressante mais permet de mieux comprendre comment JavaFX gère ses entités et comment les déplacer sur l’écran.

A screen shot of a computer

Description automatically generated

### Points importants du code

/\*\*

\* Set the arrow position.

\*/

public void setArrowPosition(double x, double y) {

arrow.setLayoutX(x);

arrow.setLayoutY(y);

}

/\*\*

\* Set the arrow rotation in degree.

\*/

public void setArrowRotation(double angle) {

arrow.setRotate(angle);

}

Ceci est le code permettant de déplacer et tourer une ImageView nommée arrow. La modification de ces valeurs ne demande pas de Platform.runLater.

// Update position

// sin(x) = SIN from -1 to 1

// sin(x)+1 = SIN from 0 to 2

// (sin(x)+1)/2 = SIN from 0 to 1

// (sin(x)+1)/2\*n = SIN from 0 to N

double angle = start \* Math.PI \* 2 / SEC\_PER\_ROT;

double xMult = ctrl.getLayerWidth() - ctrl.getArrowWidth();

double yMult = ctrl.getLayerHeight() - ctrl.getArrowHeight();

x = (Math.sin(angle) + 1) / 2 \* xMult;

y = (Math.cos(angle) + 1) / 2 \* yMult;

ctrl.setArrowPosition(x, y);

ctrl.setArrowRotation(-angle / Math.PI / 2 \* 360);

Et voici comment le Worker modifie la position de la flèche. Ce n’est rien de plus qu’un peu de trigonométrie.

// Sleep to reach a FPS goal

double now = getTime();

long sleepMS = Math.round((1.0/FPS - (now - start)) \* 1000);

if (sleepMS > 0) {

try {

Thread.sleep(sleepMS);

} catch (InterruptedException e) {

System.err.println("Unreachable");

}

}

Et voici comment le nombre d’image par second est gérer.

### Récapitulatifs

* ImageView.setLayoutX/Y(double) : Modifie la position d’une image
* ImageView.setRotate(double) : Modifie la direction de l’image en degré

# Analyse

L’analyse du projet à été réalisé le premier jour de projet. Cette phase permet de visualiser les charges de travail ainsi que la direction générale du projet.

## Analyse de l’état désiré

A quoi doit ressembler la solution/le produit fini? Vous pouvez éventuellement illustrer le nouveau processus.

L’objectif du produit fini sera de simuler et visualiser le trafic d’un carrefour à 4 voies en fonctions des différents paramètres éditables. L’application final aura une interface intuitive permettant à l’utilisateur de comprendre visuellement en temps réel les effets de différents paramètres. Il sera ainsi en mesure d’optimiser la séquence des feux de signalisation.

|  |  |
| --- | --- |
| Comme vous pouvez le voir sur la figure ci-présente, le diagramme de cas d’utilisation :  Un « visiteur » peut se connecter et créer un compte.  Un « utilisateur » connecté peut exécuter et visualiser la simulation, modifier et sauvegarder des jeux de paramètres et se déconnecter.  Et la base donnée ne travaille qu’avec le compte et les paramètres. | A screenshot of a computer  Description automatically generated |

### Fonctionnement de la connexion

|  |  |
| --- | --- |
| Quand un visiteur arrive sur l’application, il sera invité à se connecter. Pour se faire, il va entrer son nom de compte, son mot de passe et appuyer sur « Se Connecter ».  Une fois que ceci est fait, l’application dois dévifier que les champs sont bel et bien entré.  Ensuite, l’application va récupérer le compte avec le même nom dans la base de donnée et vérifier si le mot de passe est bon. Si tout se passe bien, l’appliciation changera se vue pour entrer dans le mode de simulation. | A diagram of a computer  Description automatically generated |

### Création d’un jeu de réglage

|  |  |
| --- | --- |
| Lors de la création d’un jeu de réglage, l’utilisateur devra entrer le nom du jeu de réglage et appuyer sur « Créer le jeu de réglage ».  À ce moment, l’application vérifiera l’argument et récupèrera les réglages en cours.  Une fois cela fait, le jeu de réglage sera envoyé à la base de donnée et affichera le résultat de l’action. | A diagram of a diagram  Description automatically generated |

### Sauvegarde de paramètres - détails

L’utilisateur n’est nullement obligé de créer un nouveau jeu de réglage à chaque modification. Il peut aussi sauvegarder de nouveaux réglages sur un jeu de réglage pré-éxistant.

Le diagramme d’activité ci-présent démontre les deux cas et leurs fonctionnements.

|  |  |
| --- | --- |
| Si l’utilisateur souhaite sauvegarder un jeu de réglages, il doit sélectionner le jeu dans la liste et cliquer sur sauvegarder.  Par contre, si l’utilisateur souhaite créer un nouveau jeu de réglages, il devra entrer le nom du jeu de réglage et cliquer sur créer.  Bien qu’il y ait 2 chemins différents dans ce diagramme, nous pouvons remarquer qu’ils ont un fonctionnement assez proche. Il sera donc probablement possible d’utiliser les mêmes fonctionnalités dans le code pour ces deux chemins ! | A diagram of a process  Description automatically generated |

### Exécution de la simulation

|  |  |
| --- | --- |
| Pour débuter la simulation, l’utilisateur n’aura qu’à appuyer sur « Commencer ».  Ensuite, l’application cachera le menu d’édition et démarrera le thread de la simulation.  Celui-ci s’occupera de déplacer et créer les véhicules ainsi que d’afficher le nouvel état et vérifier s’il doit arrêter.  Quand l’utilisateur appuyer sur « Arrêter », la simulation s’arrêtera. | A diagram of a computer  Description automatically generated |

## Variantes

Lors de la conception des structures de données et des flux de l’application, 2 méthode de gestion des données intéressantes me sont venu en tête. La première est plus simple mais moins facile à généraliser et la deuxième est plus complexe mais simple a généraliser.

### Variante 1

enum RoadType =

STRAIGHT\_ROAD

CROSSROAD

type Road:

getType(): RoadType

class StraightRoad impl Road;

getType(): RoadType = return STRAIGHT\_ROAD

// ...

class Crossoad impl Road:

getType(): RoadType = return CROSSROAD

// ...

circuit: List<Road> = { ... }

for rd in circuit:

if rd.getType() == CROSSROAD:

rd = (Crossroad) rd;

// Show crossroad

elif rd.getType() == STRAIGHT\_ROAD:

rd = (StraightRoad) rd;

// Show straight\_road

RoadType est une constante permettant de différencier entre les différents types de routes.

Road est le type parent permettant de définir que les routes doivent implémenter getType.

StraightRoad et Crossroad sont des exemples de routes. Ils ont des propriétés différentes.

Donc dès qu’on utilise le circuit, nous devons récupérer le type de route et la ré-interprété en le bon type de route.

Cette solution est très simple à implémenter mais peut devenir très dure à agrandir. Dès le moment où nous voulons ajouter un nouveau type de route, nous devons ajouter une énumération, une classe et modifier tous les if et switch-cases.

### Variante 2

type Road:

getProps(): List<Pair<Label, Props>>

class StraightRoad impl Road:

getProps(): List<Pair<Label, Props>> =

return // Road props here...

class Crossroad impl Road:

getProps(): List<Pair<Label, Props>> =

return // Crossroad props here ...

circuit: List<Road> = { ... }

for rd in circuit:

(label, property) = rd.getProps()

JFX\_HBox hbox = new JFX\_HBox()

hbox.add(new JFX\_Label(label))

hbox.add(new JFX\_TextField().bindValue(property))

editMenu.add(hbox)

Road est le type parent permettant de définir qu’une route implémentera getProps. Cette méthode est utilisée pour récupérer les propriétés d’une route.

StraightRoad and Crossroad sont des types de routes différent.

Dès le moment où nous devons afficher les champs permettant d’éditer une route, nous avons juste à récupérer les propriétés et les afficher. Le fait que ce soit une Crossroad, une StraightRoad ou un chemin de terre ne change rien, le code devient totalement réutilisable

### Choix de variante

Pourquoi cette variante plutôt qu’une autre? Matrice d’évaluation avec pondération et justification.

*Plus grande est la valeur, plus problématique est la solution. Valeurs de 1 à 5*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pondération | Variante 1 | Variante 2 |
| Difficulté initial | x1 (au début) | 1 (simple) | 3 (complexe) |
| Difficulté future | x3 (extensible) | 4 (immuable) | 2 (constante – se facilite) |
| Risque de bug | x2 (risque + debug) | 2 (normal) | 1 (faible ou simple débugger) |
| Résultats |  | 1 + 12 + 4 = **17** | 3 + 6 + 2 = **11** |

*J’ai donc choisi d’utiliser la deuxième variante car elle est plus extensible malgré que l’implémentation initial soit plus complexe que la première variante.*

## Sécurité de l’information et protection des données

Quelles sont les menaces/dangers pour la protection des données et des systèmes et par quelles mesures peuvent-ils être contournés/supprimés. Les données sont-elles soumises à la protection des données?

Les mots de passes sont hachés dans la base de donnée. Ceci permet de rendre le mot de passe indéchiffrable et toujours utilisable pour la connexion. Perte des données décentralisé, maintenance, …

# Conception

## Exigences du système

Les tronçons ainsi que le carrefour possèderont des identifiants unique et un système de connexion numérotées. La construction logique et visuelle se fera par le développeur mais dois être pensée générique.

Additionnellement, une gestion du flux et de la densité des véhicules par minutes pourra être affecté au tronçon. Le véhicule sera créé avec tous ses paramètres associés ainsi qu’une légère variabilité dans une plage de tolérance permettant de mieux simuler une route. Les véhicules réagiront aux comportements du véhicule suivant en essayant de maintenir une distance de sécurité.

De plus, l’utilisateur doit pouvoir se logger à sa session sur différents PC où l’application est installé et retrouver ses jeux de paramètres. Par conséquent, les données seront stockées sur une base de données hébergée sur emf-infopro-tpi.ch.

Finalement, l’interface graphique représentera de manière simple les tronçons de route et le carrefour avec les feux. L’accès aux paramètres devra être pensé dans une approche ergonomique et les éléments dynamique seront les véhicules et la couleur des feux de signalisation.

## Architecture du système

La structure du projet est relativement simple. L’application Java communique directement avec la base de données SQL distante.

### L’application cliente

L’application cliente est développée en Java avec l’utilisation de la librairie graphique JavaFX. JavaFX est une bibliothéque de code développé par Oracle permettant de créer des interface utilisateur avec Java.

### La base de donnée SQL

TODO : Filll info about the database

## Maquettes

La maquette ci-dessous est la maquette de l’application. Elle permet aussi de représenter le flux de l’application à travers les différentes cliques.

A screenshot of a computer

Description automatically generated

TODO : Explain maquette

## Concept d‘implémentation

### Diagramme de classe

A diagram of a software company

Description automatically generated with medium confidence

TODO : Explain the diagram

### Se connecter

|  |  |
| --- | --- |
| TODO : Complete this | A screenshot of a computer  Description automatically generated |

### Sauvegarder les paramètres

|  |  |
| --- | --- |
| TODO : Complete this | A screenshot of a diagram  Description automatically generated |

### Exécuter la simulation

|  |  |
| --- | --- |
| TODO : Complete this | A screenshot of a computer  Description automatically generated |

## Concept de tests

Comment les tests sont-ils effectués? Blackbox, Whitebox … .

TODO

Vue du client

- Login (succès, erreur)

- Chaque fonctionalité de la page de Simulation

- Aussi le carrefour

Suffisemment d'explication

# Réalisation

## Spécifications détaillées

## Design du système

## Configuration xyz

Conseil: D’autres chapitres peuvent s’ajouter à cette partie en fonction du projet.

# Différence entre la conception et la réalisation

Trouver des choses à dire…

# Test

## Procédure de test

Représentation du processus de test..Qui teste quoi à l’aide de quels moyens ?

## Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

Protocole de test détaillé avec Visa.

## Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Conclusion

Conclusion sur le projet en général. Les objectifs sont-ils atteints ? Si non, pourquoi ?

## Améliorations possibles

Y a-t-il des améliorations possibles (nouvelles fonctionnalités, amélioration et/ou modification du design, modification du support de données…) ?

## Auto-évaluation

Quels sont mes sentiments vis-à-vis du travail effectué. Enoncé des facteurs qui expliquent le succès ou l’échec de la réalisation des objectifs du projet.

# Bibliographie: liste des sources et références

Conseil: Cette partie peut contenir une liste des sources et éléments littéraires utilisés.

Aussi notions en modules etc

# Glossaire trier par ordre alphabetique

|  |  |
| --- | --- |
| Terme | Signification |
|  |  |
|  |  |
|  |  |

# Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le XX.XX.20XX dans le cadre de la procédure de qualification de mon CFC d’informaticien/-ne, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

L’apprenti/-e doit signer la documentation avant la remise afin de témoigner de la justesse des informations qui y figurent.

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Annexes

Code imprimé, protocole de séances, etc.: spécifier complètement et annexer à la documentation.

Listings de scripts et programmes. Le travail fourni par la candidate ou le candidat doit être entièrement documenté. Omettre le code généré automatiquement s’il n’est pas nécessaire à la compréhension.

Dans le cas où des manuels ont été créés, vous pouvez les ajouter ici en tant qu’annexe.