# Langage C# « La planche d’équilibre »

| Auteur | Dwenn Kaufmann |
|---|---|
| Date de début de projet | 24/04/2013 |
| Date de fin de projet | 28/05/2013 |
| Date de reddition du rapport | 27/05/2013 |
| Version | 1.0 |

# Table des matières


1 Introduction ..... 4
2 Documentation de développement ..... 4
2.1 Explications détaillées du projet ..... 4
2.2 Diagramme des cas d'utilisation ..... 5
2.3 Définition des conventions applicables ..... 5
2.4 Planning de livraison global ..... 6
3 Réalisation des cas d'utilisation ..... 6
3.1 Cas d'utilisation «Login » ..... 6
3.1.1 Scénario ..... 6
3.1.2 Maquettes ..... 6
3.1.3 Analyse du scénario ..... 6
3.1.3.1 Algorithme ou Structogramme ..... 6
3.1.3.2 Explications détaillées ..... 7
3.1.4 La phase de programmation ..... 7
3.1.5 La phase de tests ..... 7
3.2 Cas d'utilisation « Paramètres » ..... 8
3.2.1 Scénario ..... 8
3.2.2 Maquettes ..... 8
3.2.3 Analyse du scénario ..... 9
3.2.3.1 Algorithme ou Structogramme ..... 9
3.2.3.2 Explications détaillées ..... 9
3.2.4 La phase de programmation ..... 9
3.2.5 La phase de tests ..... 10
3.3 Cas d'utilisation « Connecter la carte » ..... 10
3.3.1 Scénario ..... 10
3.3.2 Maquettes ..... 11
3.3.3 Analyse du scénario ..... 11
3.3.3.1 Algorithme ou Structogramme ..... 11
3.3.3.2 Explications détaillées ..... 11
3.3.4 La phase de programmation ..... 11
3.3.5 La phase de tests ..... 11
3.4 Cas d'utilisation « Ejecter la carte » ..... 12
3.4.1 Scénario ..... 12
3.4.2 Maquettes ..... 12
3.4.3 Analyse du scénario ..... 12
3.4.3.1 Algorithme ou Structogramme ..... 12
3.4.3.2 Explications détaillées ..... 12
3.4.4 La phase de programmation ..... 13
3.4.5 La phase de tests ..... 13
3.5 Cas d'utilisation « Changer d'utilisateur » ..... 13
3.5.1 Scénario ..... 13
3.5.2 Maquettes ..... 14
3.5.3 Analyse du scénario ..... 14
3.5.3.1 Algorithme ou Structogramme ..... 14
3.5.3.2 Explications détaillées ..... 14
3.5.4 La phase de programmation ..... 14

3.5.5 La phase de tests ..... 14
3.6 Cas d'utilisation « Tableau des scores » ..... 14
3.6.1 Scénario ..... 15
3.6.2 Maquettes ..... 15
3.6.3 Analyse du scénario ..... 15
3.6.3.1 Algorithme ou Structogramme ..... 15
3.6.3.2 Explications détaillées ..... 15
3.6.4 La phase de programmation ..... 16
3.6.5 La phase de tests ..... 16
4 Mode d'emplois utilisateur ..... 16
4.1 Login ..... 16
4.2 Connecter / Déconnecter la carte ..... 17
4.3 Paramétrer l'application ..... 17
4.4 Changer d'utilisateur ..... 18
4.5 Affichage des scores ..... 18
4.5.1 Scores par utilisateur ..... 19
4.5.2 Scores par temps ..... 19
4.5.3 Scores par difficulté ..... 19
4.6 Vider les scores ..... 19
4.6.1 Vider les scores de l'utilisateur en cours uniquement ..... 19
4.6.2 Vider les scores de tous les utilisateurs ..... 20
4.7 Imprimer les scores ..... 20
5 Conclusions ..... 21
6 Annexes ..... 21
6.1 Journal de bord ..... 21
6.2 Schéma Synoptique ..... 22
6.3 Cahier des charges ..... 23
6.4 Code source ..... 23
6.5 Références ..... 23

# 1 Introduction

Le but de ce travail est de proposer à l'utilisateur de tester son habileté à rester en équilibre sur une planche en bois rendue instable par la pose d'un demi-cylindre en son centre.

Cette planche en bois est elle-même reliée à une carte électronique, de modèle Velleman K8055, qui sera reliée à l'ordinateur via USB.

Des capteurs, de type contact « ouverts », sont disposés de chaque côté de la planche, afin de déterminer lorsque l'utilisateur a perdu l'équilibre, et donc touche le sol avec un des deux capteurs.

Lorsque l'utilisateur décide de lancer une partie en cliquant sur le bouton « Jouer », il dispose alors d'un temps imparti où il devra faire le meilleur score, en gardant son équilibre le mieux possible.

# 2 Documentation de développement

## 2.1 Explications détaillées du projet

Le programme doit en premier afficher une fenêtre de connexion, obligeant l'utilisateur à entrer ses informations, qui sont : son Nom, son Prénom, ainsi que sa date d'anniversaire. Si l'utilisateur a déjà, par le passé, entré ses informations, il ne sera pas enregistré dans la base de données à nouveau.

Une fois la connexion effectuée, l'utilisateur arrive sur la fenêtre de jeu. Avant de pouvoir jouer, il va d'abord devoir cliquer sur le bouton permettant d'effectuer la connexion avec la carte Velleman reliée par USB à l'ordinateur.

Si la carte est connectée à l'ordinateur, la partie va se lancer suivant le temps, la difficulté et les temps d'attente entre deux touchés, paramétrés depuis la fenêtre accessible en cliquant dans le menu sur le bouton « Paramètres ».

A tout moment, l'utilisateur peut changer d'utilisateur, ce qui le ramènera au même stade qu'à l'ouverture du programme, le forçant à s'authentifier à nouveau.

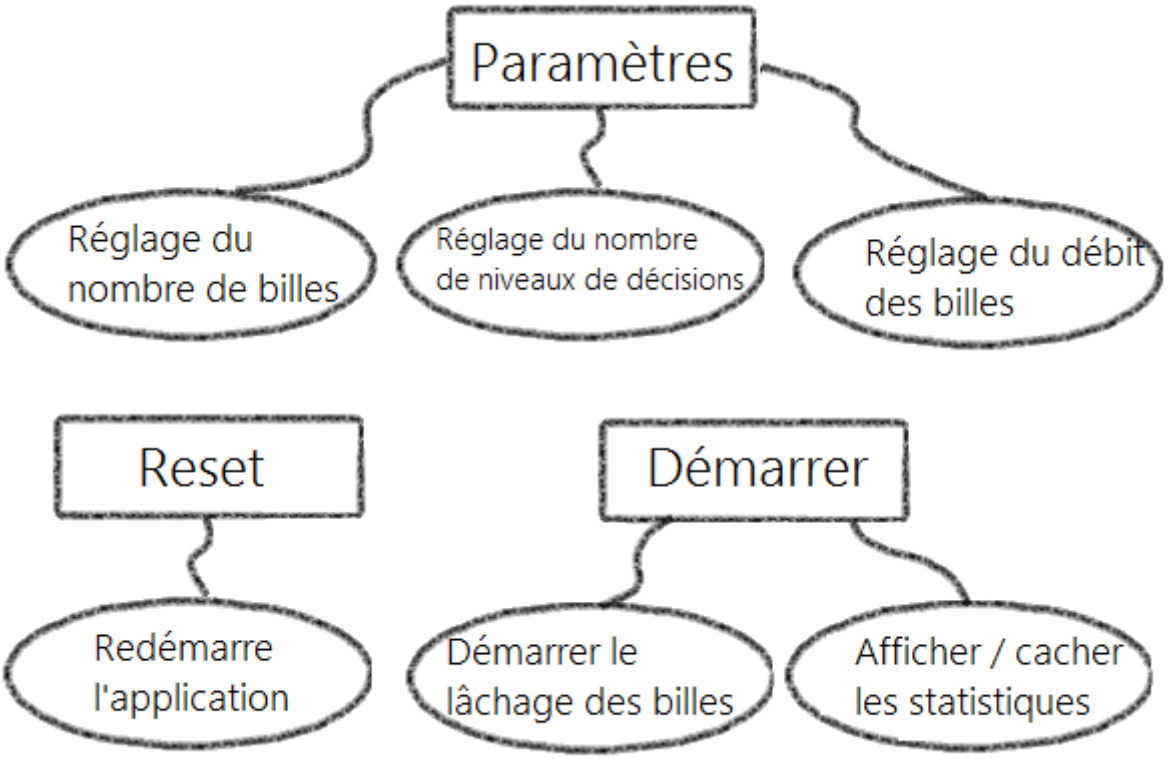
Un bouton permettant de déconnecter la carte Velleman est présent, et s'il est cliqué lorsqu'une partie est en cours, elle sera considérée comme terminée.

Le tableau des scores est accessible en cliquant sur le bouton du même nom, affichant un formulaire présentant un classement des scores triées par temps de partie, difficulté, et par nom d'utilisateur (Soit l'utilisateur en cours, soit tout le monde). L'impression des résultats est possible en cliquant sur le bouton « Imprimer » présent dans ce formulaire. Il est également possible de vider les scores de l'utilisateur en cours, ou de tous les utilisateurs.

Pendant la partie, une barre de progression indique le temps écoulé sur le temps total. Le nombre de touchés gauches et droits que l'utilisateur a fait sont affichés sous forme de texte, et des images fléchées se colorent en bleu pour signaler de quel côté le connecteur fait contact lorsque l'utilisateur perds son équilibre sur la planche.

Lorsque le temps d'une partie est écoulé, un formulaire affichant les résultats de la partie s'affiche, incluant : Le nombre de touchés totaux, le nombre de touchers gauches et droits, le temps de la partie, la difficulté. L'utilisateur est invité à cliquer sur le bouton fermant ce formulaire, et ainsi à revenir sur le formulaire de jeu.

## 2.2 Diagramme des cas d'utilisation




## *2.3* Définition *des conventions applicables*

Chaque variable typée aura un préfixe permettant de reconnaître son type plus simplement, comme appris ici au CPLN.

- Les variables de type Entier auront le préfixe « i »
- Les variables de type Double auront le préfixe « dbl »
- Les variables de type Texte auront le préfixe « str »
- Les variables de type Booléen auront le préfixe « b »

Pour ce qui est du nom après le préfixe, un nom le plus représentatif de l'utilité de la variable sera choisi, afin de faciliter la compréhension.

Chaque nom de variable utilise une convention appelée « camelCase », qui va consister à séparer chaque « partie » du mot par des majuscules, tout en gardant la première lettre de la variable en minuscule, pour améliorer la lisibilité et la compréhension.

Ainsi, une méthode servant à créer des camions s'appellerait donc « creationCamion ».

## 2.4 Planning de livraison global

La planification est en annexe.

# 3 Réalisation des cas d'utilisation

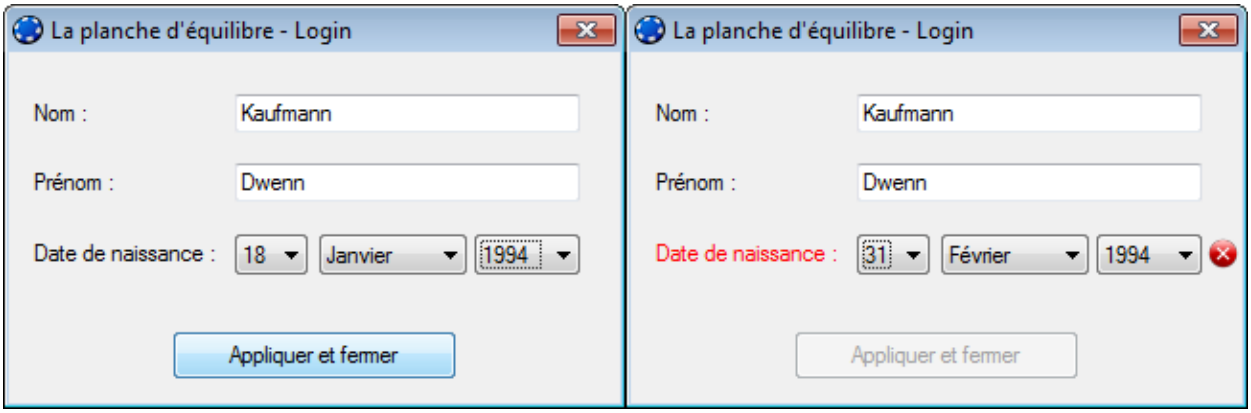
## 3.1 Cas d'utilisation «Login »

La première action que va avoir à faire l'utilisateur lors de l'ouverture de l'application sera s'authentifier. Si la combinaison du nom de famille, du prénom et de la date de naissance n'a jamais été saisie, l'utilisateur sera stocké dans la base de données liée au programme.

### 3.1.1 Scénario

1. L'utilisateur rentre son nom de famille.
2. L'utilisateur rentre son prénom.
3. L'utilisateur rentre sa date de naissance
4. L'utilisateur appuie sur le bouton « Appliquer et fermer »
5. Les données sont testées à la recherche d'éventuelles erreurs ou de champs vides.
6. Si la combinaison de toutes les données entrées n'est pas déjà présente dans la base de données, l'utilisateur est sauvegardé, un MessageBox l'en informe.
7. Les données de l'utilisateur sont également sauvegardées dans le fichier Params.settings pour une utilisation ultérieure
8. Le formulaire Login.cs est caché
9. Le formulaire Jeu.cs est affiché.

### 3.1.2 Maquettes




### 3.1.3 Analyse du scénario

#### 3.1.3.1 Algorithme ou Structogramme

#### 3.1.3.2 Explications détaillées

A l'ouverture du formulaire Login.cs, la ComboBox « cbxAnnee » est peuplée via une boucle, afin que les valeurs contenues dans la liste restent à jour, peu importe l'année d'ouverture du programme. Les années affichées commencent depuis 1940 jusqu'à l'année actuelle stockée sur l'ordinateur.

Dans les TextBox servant à renseigner le Nom ainsi que le prénom, seuls les lettres, les espaces et les tirets sont acceptés. On ne peut pas copier/coller du texte à l'intérieur.

Une fois le jour, le mois et la date de naissance renseignés, l'utilisateur pourra constater, à chaque fois qu'il modifie une de ces 3 informations, qu'elles sont contrôlées afin de déterminer si elles sont valides ou non. Si elles ne le sont pas, par exemple lors d'un 31 février, le bouton permettant d'appliquer et fermer la fenêtre se désactive, une icône d'erreur rouge surgit à côté des ComboBox, et le label « Date de naissance » devient de couleur rouge. Tout ceci disparaît une fois une date d'anniversaire valide saisie.

Si l'utilisateur laisse un champ vide, par exemple son nom, et clique sur le bouton Appliquer et fermer, une fenêtre d'erreur surgit et l'en informe.

Si toutes les données saisies sont correctes, l'utilisateur va être enregistré dans la base de données s'il n'avait jamais entré ses informations auparavant, puis les données seront stockées en tant que paramètres dans le fichier Params.settings pour être utilisées plus tard.

La fenêtre va ensuite se cacher, et le formulaire Jeu.cs va être affiché à l'écran.

### 3.1.4 La phase de programmation

Le code source est en annexe.

### 3.1.5 La phase de tests

| **Test à effectuer « Login.cs »** | | **Résultat escompté** | **Résultat obtenu** | **Constatation** |
|---|---|---|---|---|
| Généralités | | | | |
| | Un des champs est vide. | Indique qu'aucun champ ne doit être vide. | Ok | Une fenêtre d'erreur surgit. Il faudra remplir les champs afin de continuer. |

| | | | | |
|---|---|---|---|---|
| | Des caractères spéciaux sont entrés dans un des champs. | Il est impossible d'écrire des caractères spéciaux. | Ok | Si l'on tente d'écrire un caractère spécial, hormis « - », « « ' » ou un espace, il n'est pas écrit |
| | L'utilisateur tente de copier/coller via un raccourcis clavier ou le menu par le clic droit. | L'action n'est pas possible. | Ok | Rien ne se passe. |
| | La date d'anniversaire n'est pas correcte. | Impossible de continuer. | Ok | Une icône d'erreur s'affiche, le bouton Appliquer se désactive, le label correspondant devient rouge. |
| | Tous les champs sont renseignés, et la date d'anniversaire est correcte. | Cache la fenêtre, sauvegarde les valeurs et passe à la fenêtre principale. | Ok | Si l'utilisateur n'était pas déjà enregistré dans la base, il l'est maintenant. |

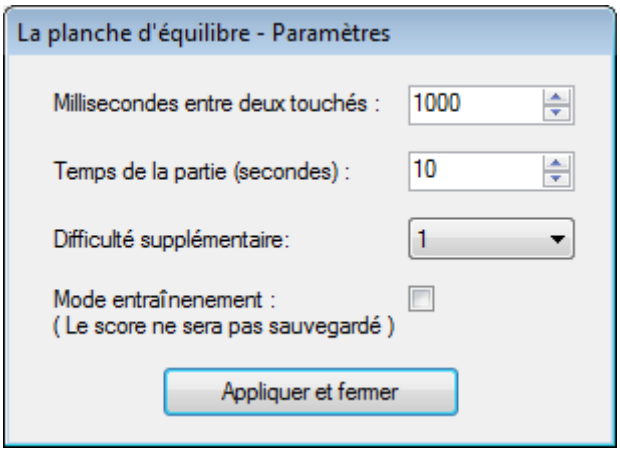
## 3.2 Cas d'utilisation « Paramètres »

Depuis le formulaire Jeu.cs, un clic sur le bouton tsbParametres va afficher le formulaire Parametres.cs, permettant de modifier la configuration, tel que le temps à attendre avant de comptabiliser un deuxième toucher, le temps total de la partie, la difficulté, ainsi que de déterminer si le mode d'entrainement est activé ou non.

### 3.2.1 Scénario

1. La méthode Dispose du timer tmrJeu est appelée pour stopper la partie en cours si elle avait lieu.
2. Le formulaire Jeu.cs est caché.
3. Le formulaire Parametres.cs est affiché.
4. Le ComboBox régissant la difficulté, la checkBox qui permet de choisir le mode entraînement, les numericUpDown permettant de saisir le nombre de millisecondes entre deux touchés ainsi que celui permettant de définir le temps de la partie sont remplis avec les valeurs stockées dans le fichier Params.settings
5. Si l'utilisateur saisit des paramètres jugés correct par l'application et clique sur le bouton btnAppliquer, ils sont sauvegardés dans le fichier Params.settings
6. Le formulaire Parametres.cs se cache
7. Le formulaire Jeu.cs s'affiche, prenant en compte les nouveaux paramètres

### 3.2.2 Maquettes




### 3.2.3 Analyse du scénario

#### 3.2.3.1 Algorithme ou Structogramme

#### 3.2.3.2 Explications détaillées

Le timer tmrJeu subit la méthode « Dispose », ce qui va libérer toute ressource utilisée par celui-ci et le stopper immédiatement.

La propriété « Visible » du formulaire Jeu.cs passe à false, le formulaire Parametres.cs est déclaré et affiché.

La propriété « Value » du NumericUpDown « tbxMillisecondes » est récupérée depuis la variable iMillisecondes du fichier Params.settings.

La propriété « Value » du NumericUpDown « tbxTempsPartie » est récupérée depuis la variable iTempspartie du fichier Params.settings.

La propriété « Checked » de la CheckBox« checkEntrainement » est récupérée depuis la variable bEntrainement du fichier Params.settings.

Après un clic sur le bouton btnAppliquer, si les propriétés « Text » de tbxMillisecondes, tbxTempsPartie et cbxDifficulte ne sont pas vides, et si leur valeur sont positives et inférieures ou égales à leur propriété « Maximum », les paramètres seront sauvegardés dans le fichier Params.cs. Autrement, une erreur surgit et avertit l'utilisateur de son erreur de saisie.

La fenêtre se cache, et le formulaire Jeu.cs est affiché à nouveau.

### 3.2.4 La phase de programmation

Le code source est en annexe.

### 3.2.5 La phase de tests

| Test à effectuer Params.cs | | Résultat escompté | Résultat obtenu | Constatation |
|---|---|---|---|---|
| Généralités | | | | |
| | Un ou plusieurs champs sont vides. | Indique qu'aucun champ ne doit être vide. | Ok | Il faudra remplir les champs afin de continuer. |
| | Des caractères spéciaux sont entrés dans un des champs | Impossible d'écrire autre chose que des chiffres. | Ok | Les entrées claviers sont contrôlées, et le copier/coller est désactivé. |
| | Un nombre négatif, ou inférieur/supérieur aux valeurs limites est entré dans un des champs | Il est impossible d'entrer un nombre négatif, et pour un nombre supérieur, la valeur maximal est prise | Ok | |
| | L'utilisateur entre des valeurs correctes dans les deux champs | Ferme la fenêtre, sauvegarde les valeurs et repasse à la fenêtre principale | Ok | |

## 3.3 Cas d'utilisation « Connecter la carte »

Connecter la carte à l'ordinateur via le câble USB puis presser le bouton « Connecter la carte » dans le formulaire principal est une étape nécessaire avant de pouvoir jouer.

Si l'utilisateur clique sur le bouton alors que la carte n'est pas connectée à l'ordinateur, un message d'erreur apparaît et l'en informe.

Une fois la carte connectée, le bouton « Connecter la carte » sera grisé et désactivé, et le bouton « Ejecter la carte » auparavant grisé et désactivé deviendra actif et d'une couleur normale, ceci afin d'éviter à l'utilisateur de connecter/éjecter la carte alors qu'elle l'est déjà.

Lors de changement de formulaire ou si le formulaire a à se réafficher, les boutons resteront dans l'état où ils étaient précédemment grâce au code inséré dans l'évènement « VisibleChanged ».

### 3.3.1 Scénario

1. L'utilisateur clique sur le bouton « Connecter la carte ».
2. Si la carte n'est pas connectée à l'ordinateur, une fenêtre d'erreur surgit et l'action s'arrête là.
3. Si la carte est connectée et que la connexion a pu être effectuée, la propriété « Enabled » des boutons tsbConnecter, tsbEjecter et btnJouer est inversée.

4. La méthode inverserCouleurMenu est appelée, qui va inverser la couleur du bouton tsbConnecter avec le bouton tsbEjecter.
5. La variable bIsAlreadyConnected passe à true dans le fichier de paramètres, Params.settings, afin de savoir plus tard si le bouton doit être réaffiché en tant que déjà pressé ou non.

### 3.3.2 Maquettes


### 3.3.3 Analyse du scénario

#### 3.3.3.1 Algorithme ou Structogramme

#### 3.3.3.2 Explications détaillées

Une variable de type int est déclarée afin de déterminer l'état futur de la carte, car la méthode de connexion fournie avec la .DLL récupérée sur le CD fourni avec la carte par le maître d'atelier renvoie un code après la tentative, indiquant si l'action est réussie ou non. (0 = carte présente, -1 = carte non connectée à l'ordinateur)

Je me sers donc de cette variable à l'aide d'un test pour vérifier si la connexion est réussie ou non. Si c'est un échec, un message apparaîtra à l'utilisateur, lui disant « Veuillez d'abord connecter la carte à l'ordinateur via le cable USB».

Si l'état de la variable est 0, la méthode inverserBoutons est appelée, servant à inverser la propriété « Enabled » entre le bouton tsbConnecter et tsbEjecter.

La couleur de ces deux boutons est ensuite inversée, permettant une meilleure visibilité de l'état de connexion de la carte pour l'utilisateur.

Dans le fichier de paramètres « Params.settings », la variable bIsAlreadyConnected passe à l'état « true » si la connexion a réussi. Ceci afin de savoir lors de réaffichages futurs de la fenêtre de jeu, l'état que doivent avoir les boutons btnConnecter et btnEjecter.

### 3.3.4 La phase de programmation

Le code source est en annexe.

### 3.3.5 La phase de tests

| **Test à effectuer bouton « tsbConnecter »** | **Résultat escompté** | **Résultat obtenu** | **Constatation** |
|---|---|---|---|
| Généralités | | | |

| | | | | |
|---|---|---|---|---|
| | La carte n'est pas connectée via USB à l'ordinateur. | Le bouton tsbConnecter ne se désactive pas pas, il reste impossible de jouer car le bouton de Jeu reste désactivé | Ok | Un message d'erreur surgit indiquant à l'utilisateur qu'il faut d'abord connecter la carte par USB avant de presser ce bouton. |
| | La carte est connectée via USB à l'ordinateur | La propriété Enabled du bouton ainsi que sa couleur s'inversent avec le bouton tsbEjecter | Ok | |

## 3.4 Cas d'utilisation « Ejecter la carte »

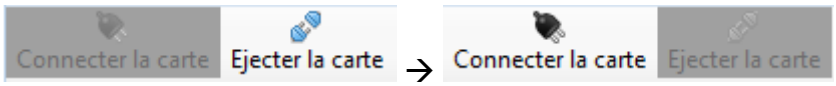
Le bouton tsbEjecter sert à libérer la carte, elle en sera ainsi plus utilisable avec le programme tant qu'elle ne sera pas reconnectée.

Le bouton n'est pas cliquable tant que la carte n'est pas connectée, et un clic sur le bouton met immédiatement fin à la partie en cours si elle avait lieu.

### 3.4.1 Scénario

1. L'utilisateur clique sur le bouton « Reset ».
2. La propriété « Enabled » des boutons tsbConnecter, tsbEjecter et btnJouer est inversée.
3. La méthode inverserCouleurMenu est appelée, qui va inverser la couleur du bouton tsbConnecter avec le bouton tsbEjecter.
4. La variable bIsAlreadyConnected passe à false dans le fichier de paramètres, Params.settings, afin de savoir plus tard si le bouton doit être réaffiché en tant que déjà pressé ou non.
5. Si le timer permettant de déplacer les billes est en cours, il sera stoppé.
6. La carte est éjectée, et ne peux plus être utilisée avec le programme tant qu'elle ne sera pas reconnectée.

### 3.4.2 Maquettes




### 3.4.3 Analyse du scénario

#### 3.4.3.1 Algorithme ou Structogramme

#### 3.4.3.2 Explications détaillées

La méthode « CloseDevice » fournie dans la .DLL présente sur le CD obtenu avec la carte est appelée. Elle permet de déconnecter la carte avec le programme en cours, et libère le driver. Il sera impossible d'effectuer des actions liées à la carte par la suite, à moins de la reconnecter.

La méthode inverserBoutons est appelée, servant à inverser la propriété « Enabled » entre le bouton tsbConnecter et tsbEjecter.

La couleur de ces deux boutons est ensuite inversée, permettant une meilleure visibilité de l'état de connexion de la carte pour l'utilisateur.

Dans le fichier de paramètres « Params.settings », la variable bIsAlreadyConnected passe à l'état « false ». Ceci afin de savoir lors de réaffichages futurs de la fenêtre de jeu, l'état que doivent avoir les boutons btnConnecter et btnEjecter.

Un test est effectué afin de savoir si une partie est en cours. Si c'est vrai, le timer tmrJeu sera stoppé, et la partie sera stoppée, affichant néanmoins les résultats à l'utilisateur.

### 3.4.4 La phase de programmation

Le code source est en annexe.

### 3.4.5 La phase de tests

| **Test à effectuer bouton « tsbEjecter »** | | **Résultat escompté** | **Résultat obtenu** | **Constatation** |
|---|---|---|---|---|
| Généralités | | | | |
| | Le bouton est cliqué lorsque rien ne se passe. | La carte est déconnectée | Ok | |
| | Le bouton est cliqué lorsqu'une partie est en cours. | La partie se termine, la carte est déconnectée. | Ok | |
| | Le bouton est cliqué lorsque la carte n'est pas connectée. | Le bouton est désactivé et non cliquable. | Ok | |

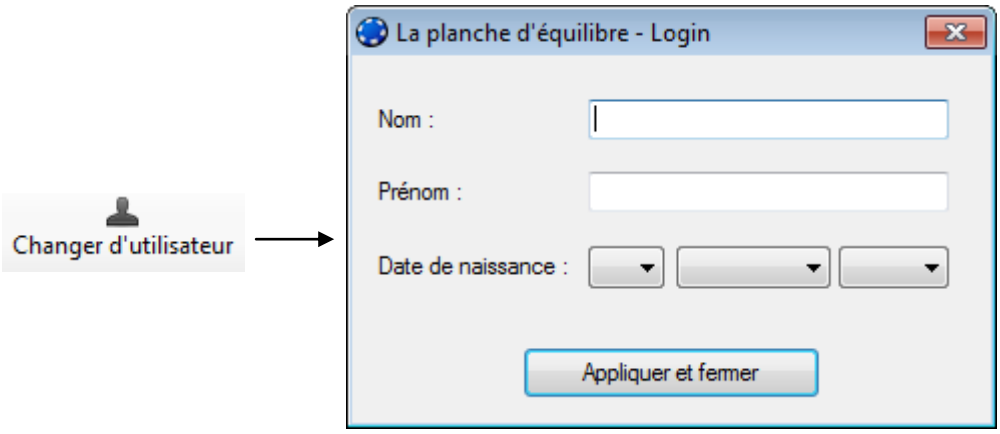
## 3.5 Cas d'utilisation « Changer d'utilisateur »

Depuis le formulaire principal, il est à tout moment possible de changer d'utilisateur si une autre personne désire jouer sans devoir fermer le programme. Le bouton se trouve dans le menu du formulaire Jeu.cs.

### 3.5.1 Scénario

1. Cliquer sur le bouton « tsbChangerUtilisateur ».
2. Le formulaire Jeu.cs est caché
3. Le formulaire Login.cs est affiché comme lors de l'ouverture du programme, forçant l'utilisateur à s'authentifier à nouveau.

### 3.5.2 Maquettes




### 3.5.3 Analyse du scénario

#### 3.5.3.1 Algorithme ou Structogramme

#### 3.5.3.2 Explications détaillées

Le timer tmrJeu subit la méthode « Dispose », ce qui va libérer toute ressource utilisée par celui-ci et le stopper immédiatement.

La propriété « Visible » du formulaire Jeu.cs passe à false.
La méthode « Hide » du formulaire Jeu.cs est appelée afin de le cacher.

Le formulaire Login.cs est déclaré et affiché.

### 3.5.4 La phase de programmation

Le code source est en annexe.

### 3.5.5 La phase de tests

| **Test à effectuer « tsbChangerUtilisateur »** | | **Résultat escompté** | **Résultat obtenu** | **Constatation** |
|---|---|---|---|---|
| Généralités | | | | |
| | L'utilisateur clique sur le bouton lorsque rien ne se passe. | Le formulaire Login s'affiche. | Ok | |
| | L'utilisateur clique sur le bouton lorsqu'une partie est en cours. | Le formulaire Login s'affiche | Ok | La partie est stoppée nette, sans affichage des résultats. |

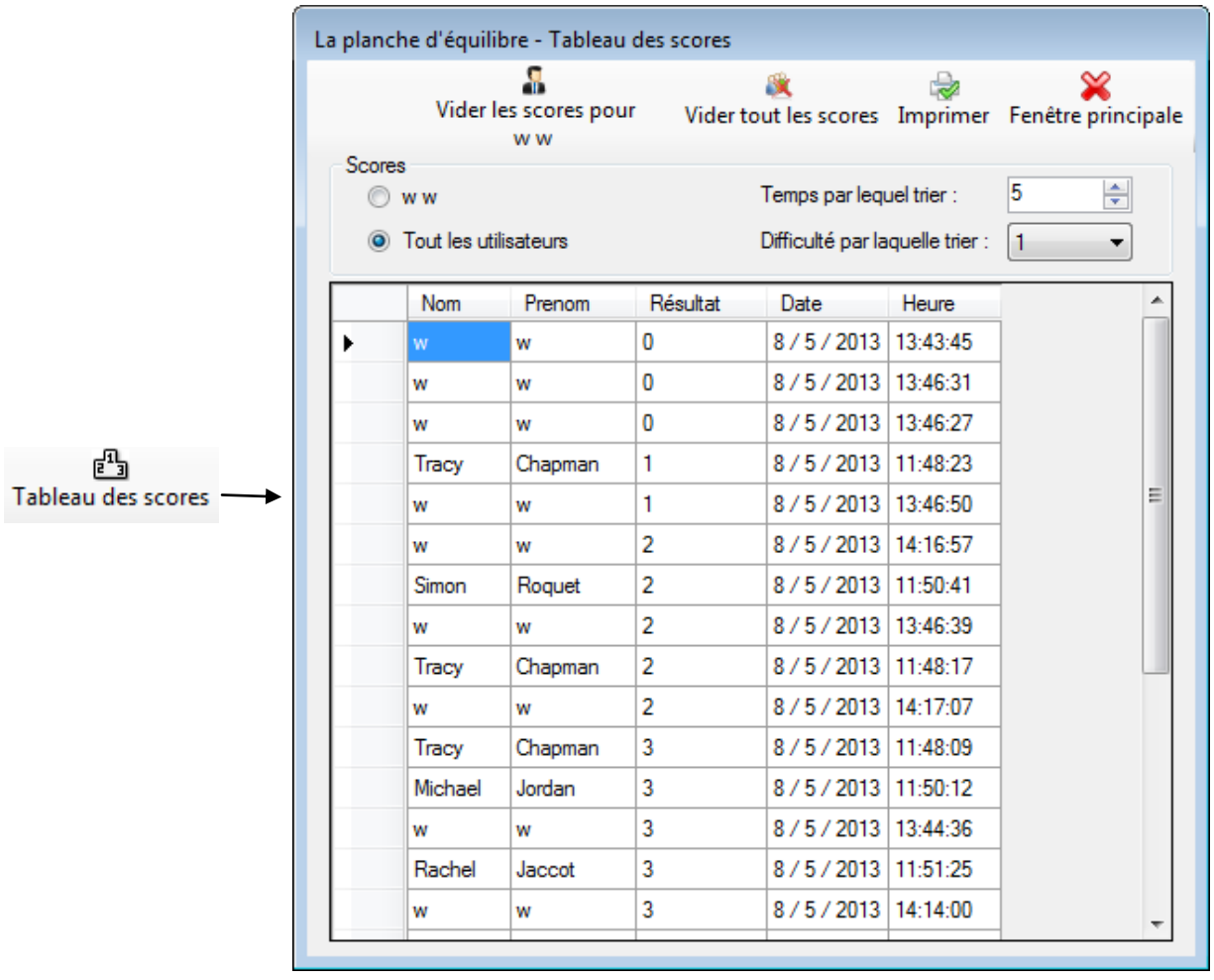
## 3.6 Cas d'utilisation « Tableau des scores »

Depuis le formulaire Jeu.cs, un clic sur le bouton tsbScores va afficher le formulaire Scores.cs, permettant de consulter la liste des résultats triés par utilisateur, temps et difficulté.

### 3.6.1 Scénario

1. Cliquer sur le bouton tsbScores depuis le formulaire Jeu.cs, le formulaire Scores.cs va s'afficher.
2. Par défaut, les scores affichés lors de l'ouverture de ce formulaire sont les mêmes paramètres que l'utilisateur a paramétrés pour sa partie.
3. L'utilisateur peut sélectionner soit uniquement ses propres résultats, soit ceux de tous les utilisateurs présents dans la base de données.
4. Les résultats peuvent être triés par temps en modifiant la valeur en seconde présente dans le TextBox « tbx_Temps »
5. Les résultats sont également triables par difficulté, allant de Aucune à 3.

### 3.6.2 Maquettes




### 3.6.3 Analyse du scénario

#### 3.6.3.1 Algorithme ou Structogramme

#### 3.6.3.2 Explications détaillées

Le timer tmrJeu subit la méthode « Dispose », ce qui va libérer toute ressource utilisée par celui-ci et le stopper immédiatement.

La propriété « Visible » du formulaire Jeu.cs passe à false.
La méthode « Hide » du formulaire Jeu.cs est appelée afin de le cacher.

Le formulaire Scores.cs est déclaré et affiché.

Dans l'évènement Load du formulaire Scores.cs, le nom et le prénom de l'utilisateur est inséré dans le bouton tsbSupressionScoresUtilisateur, le niveau de difficulté du ComboBox cbxDifficulte est rempli en fonction de la variable iDifficulte dans le fichier Params.cs, le texte présent dans le RadioButton rbtn_Personnel est également rempli à partir d'une concaténation entre les variables strNom et strPrenom du fichier Params.cs, et la valeur du NumericUpDown tbx_Temps est remplie en fonction de la variable iTempsPartie, toujours depuis le fichier Params.cs.

### 3.6.4 La phase de programmation

Le code source est en annexe.

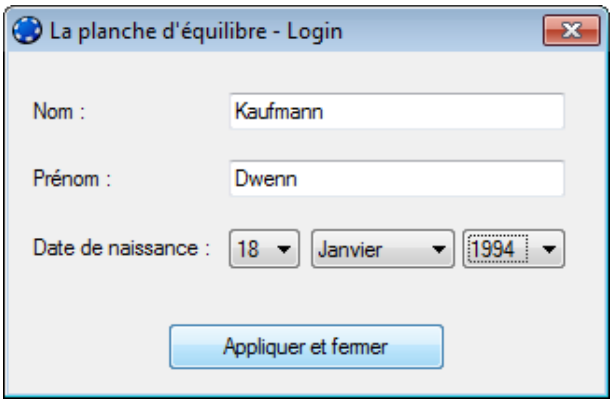
### 3.6.5 La phase de tests

| **Test à effectuer « Scores.cs »** | **Résultat escompté** | **Résultat obtenu** | **Constatation** |
|---|---|---|---|
| Généralités | | | |
| L'utilisateur clique sur le bouton "Afficher/cacher les stats" avant d'avoir cliqué sur le bouton démarrer. | Le bouton n'est pas cliquable car il a été désactivé | Ok | |
| L'utilisateur clique sur le bouton "Afficher/cacher les stats" après avoir cliqué sur le bouton démarrer. | Le bouton est cliquable, et la propriété "Visible" des contrôles concernés est inversée | Ok | |

# 4 Mode d'emplois utilisateur

## *4.1 Login*

A l'ouverture de l'application, une fenêtre invitant l'utilisateur à s'authentifier surgit.
Il suffit de rentrer son Nom, Prénom, et sa date de naissance puis de cliquer sur « Appliquer et fermer ».



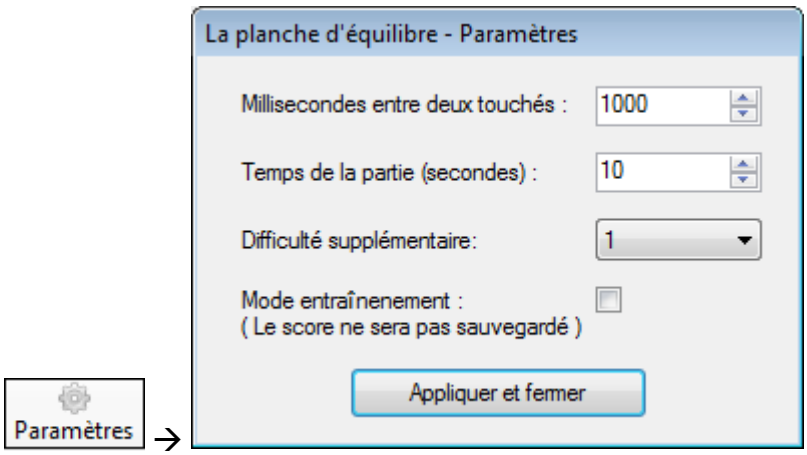

## 4.2 Connecter / Déconnecter la carte

Une fois authentifié, la fenêtre de jeu apparaît. Mais avant de pouvoir jouer, il faut au préalable connecter la carte Velleman par USB à l'ordinateur, et la planche d'équilibre à celle-ci. Une fois fait, un simple clic sur le bouton « Connecter la carte » activera les boutons permettant à l'utilisateur de pouvoir jouer.


La déconnexion de la carte est possible, et stoppe la partie en cours si elle avait lieu.

## 4.3 Paramétrer l'application

Dans la fenêtre de jeu, un clic sur le bouton Paramètres permet d'afficher une fenêtre dans laquelle figure toutes les options de configuration de l'application.




Millisecondes entre deux touchés : Correspond au nombre de millisecondes à attendre avant de compter à nouveau un touché lorsqu'il y a contact.

Temps de la partie : Définit le nombre, en secondes, de la partie.

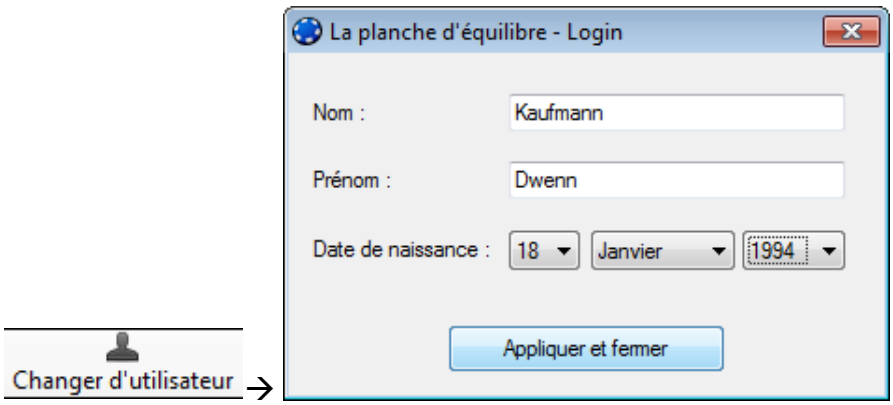
Difficulté supplémentaire : Aucune, 1, 2, ou 3. Des accessoires sont fournis avec la planche afin de corser la difficulté. L'utilisateur doit, en plus de tenir en équilibre sur la planche, porter des objets.

Mode entraînement : Si cette case est cochée, les scores ne sont pas sauvegardés à la fin d'une partie.

Un clic sur le bouton « Appliquer et fermer » applique les paramètres, ainsi lors de la prochaine partie, les modifications faites seront prises en compte.

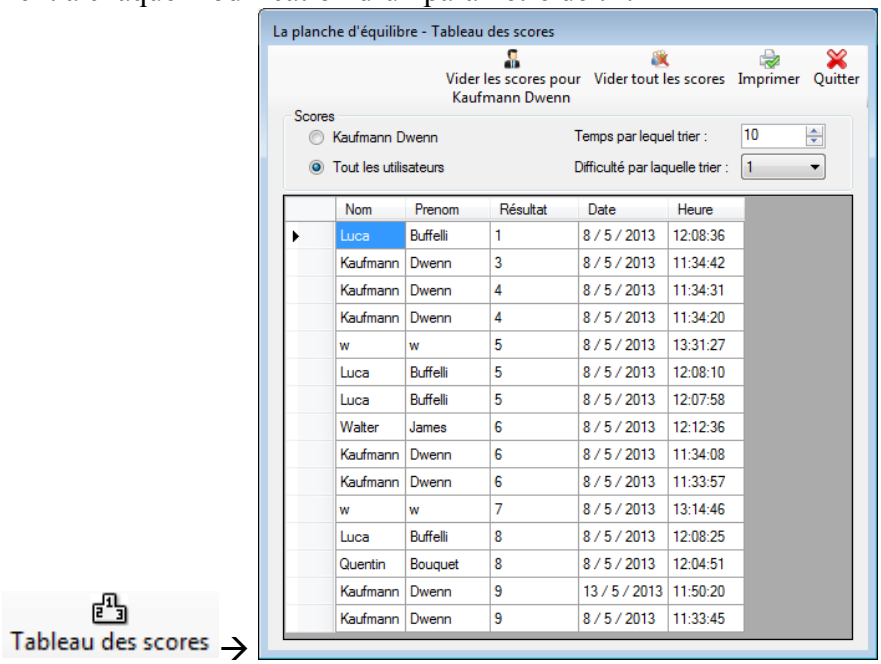
## 4.4 Changer d'utilisateur

Un clic sur ce bouton renvoie simplement à la première fenêtre, obligeant l'utilisateur à s'authentifier à nouveau.



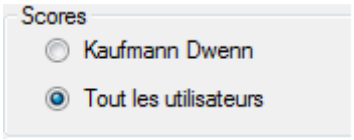

## 4.5 Affichage des scores

Un clic sur le bouton « Tableau des scores » depuis le formulaire de jeu permet de faire apparaître une nouvelle fenêtre, dans laquelle figure un tableau regroupant les scores triés par temps et difficulté. Aucun bouton permettant d'actualiser n'est présent car cela se fait automatiquement à chaque modification d'un paramètre de tri.



### 4.5.1 Scores par utilisateur

Dépendamment du choix de l'utilisateur, il est possible de trier les résultats affichés afin de ne montrer que les siens, ou ceux de tout le monde via un bouton radio.




### 4.5.2 Scores par temps

Les scores sont triés également par le temps de la partie. L'utilisateur doit juste entrer le nombre, en secondes, du temps des parties qu'il veut consulter.


### 4.5.3 Scores par difficulté

La difficulté est aussi un paramètre de tri dans l'affichage des scores. Allant de Aucune à 3, l'utilisateur doit sélectionner le niveau choisi afin de voir les scores correspondants.


## *4.6 Vider les scores*

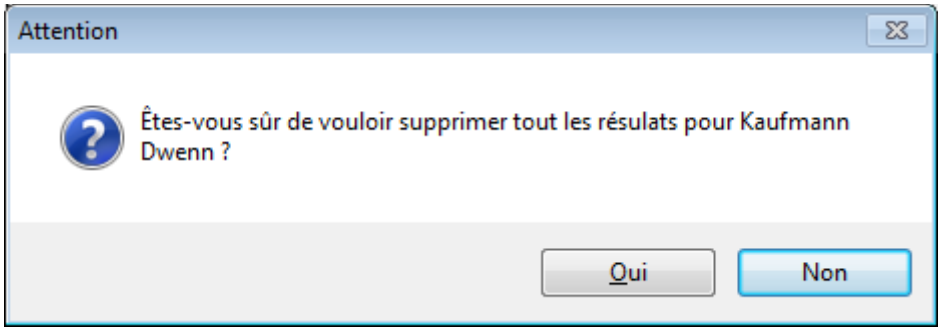
Une fois arrivé dans la fenêtre « Tableau des scores », il est possible de supprimer ceux-ci.

### 4.6.1 Vider les scores de l'utilisateur en cours uniquement

Un bouton permet de faire cette action, le voici :


L'utilisateur en cours s'appelant Kaufmann Dwenn, son nom est inscrit dans le bouton afin que l'utilisateur s'aperçoive que seuls ses résultats personnels seront supprimés.

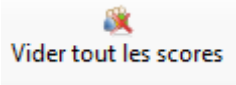
Une fenêtre d'avertissement surgit, pour éviter les erreurs, afin d'avertir l'utilisateur de l'action qu'il tente d'effectuer.



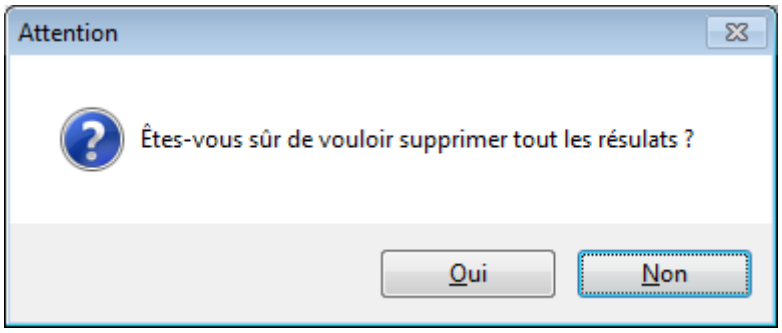

S'il clique sur « Oui », tous ses résultats personnels seront effacés définitivement et la liste de ses résultats (vides) sera affichée.

### 4.6.2 Vider les scores de tous les utilisateurs

Un bouton permet de faire cette action, le voici :



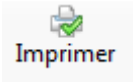

Ce bouton permet, comme son nom l'indique, à supprimer définitivement tous les résultats de tous les utilisateurs. Cette action étant irréversible, un avertissement apparaît :



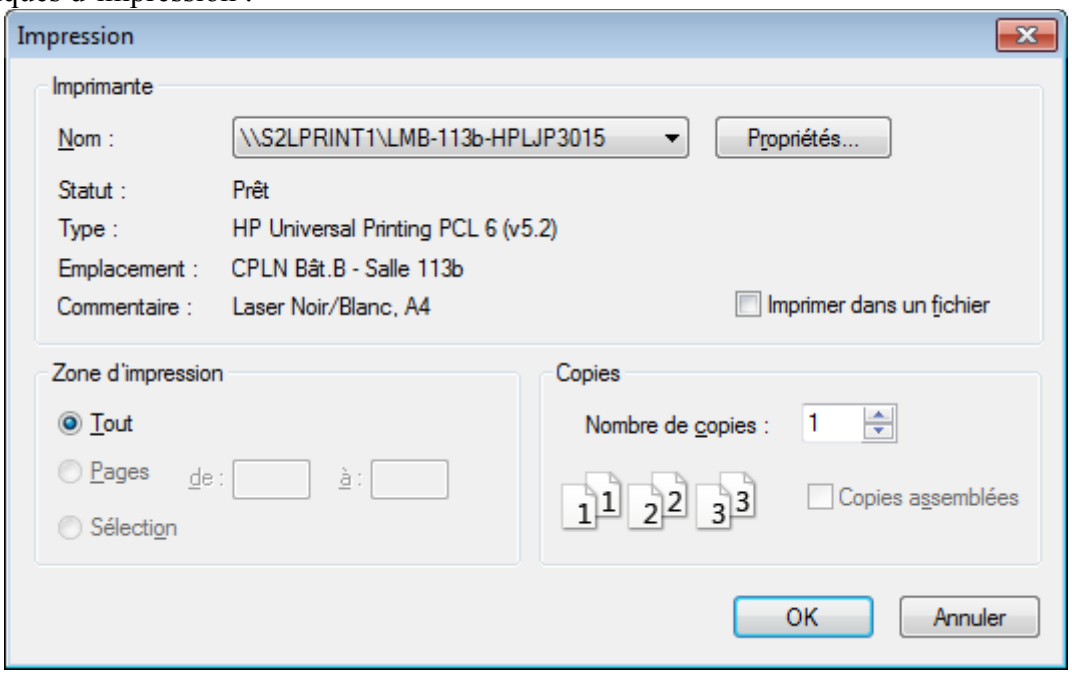

Si la réponse est oui, absolument tous les résultats seront effacés définitivement et l'affichage de la liste des résultats sera actualisé.

## *4.7 Imprimer les scores*

Si l'on souhaite imprimer les résultats affiché dans la fenêtre « Tableau des scores », il suffit de cliquer sur le bouton « Imprimer » :




Une fenêtre « normale » d'impression apparaît alors, permettant de modifier les paramètres classiques d'impression :




Lors d'un clic sur le bouton « Ok », l'impression des résultats se fera immédiatement.

# 5 Conclusions

Lors du développement de mon projet, la prise d'information et la compréhension quant à la gestion et l'affichage de multiples objets m'a causé de gros problèmes. Ne maîtrisant pas ces points, il m'a été difficile de trouver des informations, que ce soit sur internet ou dans des livres, car sur la majorité des documents consultés, les explications se résumaient à donner les bases, et rien de plus. De ce fait il m'a été difficile de comprendre rapidement le fonctionnement et la marche à suivre, cela m'a donc pris beaucoup de temps à faire dès le début du projet.

J'ai eu du plaisir durant mon travail car j'apprécie la programmation en langage C#, c'est la chose qui m'a toujours intéressée le plus ici au CPLN.

Un point important a été de comprendre comment utiliser la .DLL fournie sur le CD afin de récupérer les informations sur les contacts de la carte électronique.

Je suis maintenant plus à l'aise pour ce qui est de l'utilisation d'une DLL annexe, de plus celle fournie n'était pas prévue à l'origine pour le C#, mais le C++.

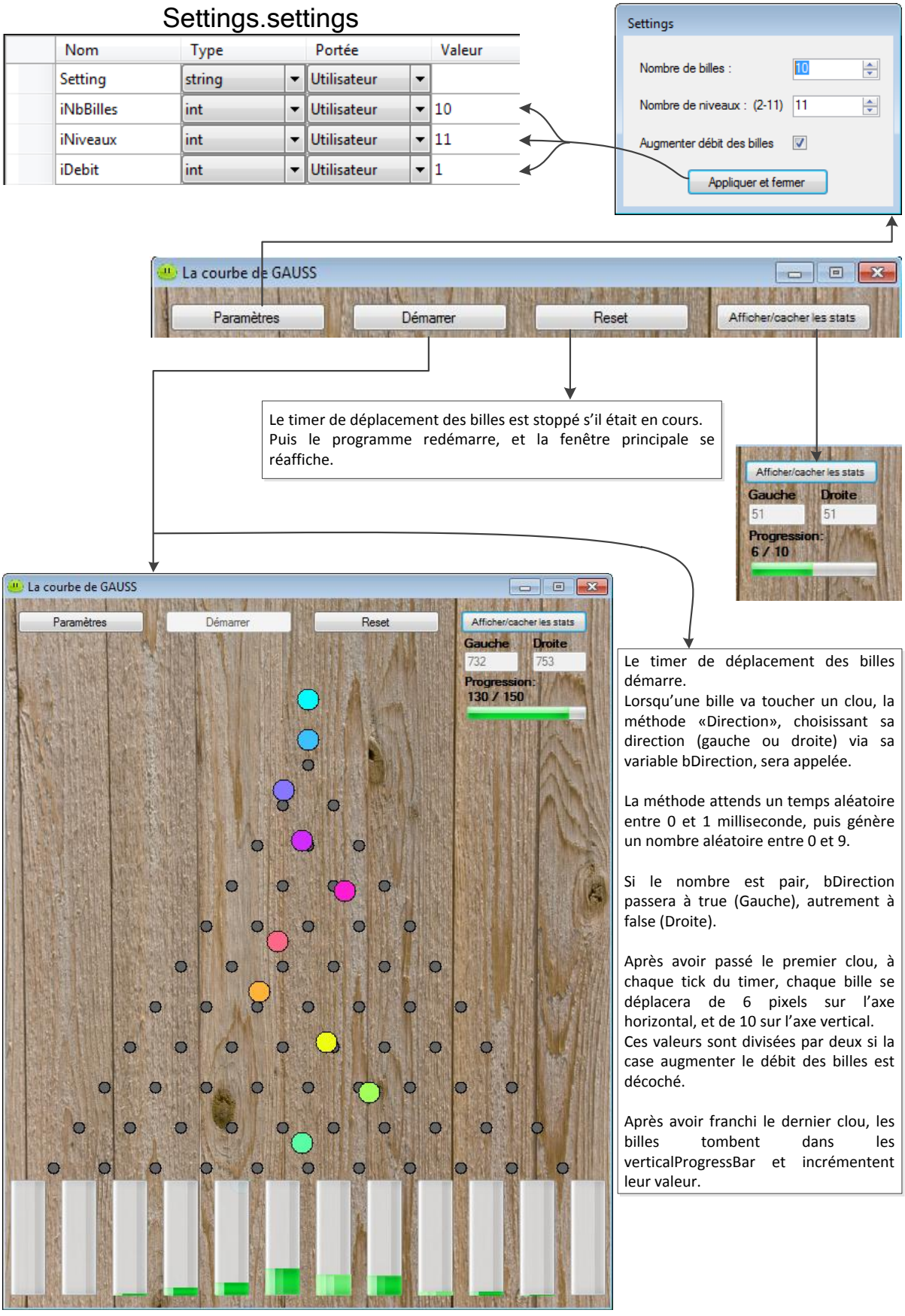
L'organisation quant aux tâches à effectuer, et l'attribution d'un temps par tâche est important, c'est pourquoi j'ai fait attention à y réfléchir correctement dès le début.

# 6 Annexes

## *6.1 Journal de bord*

Le journal de bord est en annexe

## 6.2 Schéma Synoptique



## 6.3 Cahier des charges

Le cahier des charges est en annexe.

## 6.4 Code source

Le code source est en annexe.

## 6.5 Références


Classe C# regroupant les fonctions liées à la carte 8055 :
http://www.protung.ro/2009/08/velleman-k8055-c-test-application/

Mettre du texte dans une progressBar :
http://www.codeproject.com/Articles/31406/Add-the-Percent-or-Any-Text-into-a-Standard-Progre

Intercepter les touches dans un contrôle et les filtrer : (BFree)
http://stackoverflow.com/questions/463299/how-do-i-make-a-textbox-that-only-accepts-numbers

Vérifier la validité d'une date :
http://msdn.microsoft.com/fr-fr/library/vstudio/ch92fbc1.aspx

Empêcher le copier/coller dans les contrôles (Matthew Gertz):
http://social.msdn.microsoft.com/Forums/en-US/vbide/thread/1bf46245-4696-44f0-ab36-5fb51cb546cd/

Icônes des menus :
http://findicons.com/


Dwenn Kaufmann, le lundi 27 mai 2013.