

Projet IHM : Memory



VB

Rapport de projet

Tann Logan

Groupe 105

Terki Sofiane

Groupe 105

DUT 1^{ère} Année

Table des matières

<i>Introduction.....</i>	<i>3</i>
<i>Problématique du projet :.....</i>	<i>3</i>
<i>Rôle fonctionnel du projet :.....</i>	<i>3</i>
<i>Fonctionnalités.....</i>	<i>3</i>
<i>Schéma d'ordonnancement des formulaires</i>	<i>11</i>
<i>Fichier de sauvegarde.....</i>	<i>12</i>
<i>Conclusion.....</i>	<i>13</i>

Introduction

Le jeu de memory est un jeu de société solo, qui demande de la réflexion et de la mémorisation de la part du joueur dont le but est de trouver tous les carrés cachés. Au début du jeu toutes les cartes sont mélangées et retournées afin que le joueur ne puisse les voir, l'objectif, dans un temps défini, est de trouver le plus de carré (4 cartes similaires). La partie se termine si le joueur trouve tous les carrés ou si le temps est écoulé.

Problématique du projet :

Programmer un code qui pourra permettre de représenter une partie du jeu « Memory » à partir d'un langage événementiel, le Visual basic.

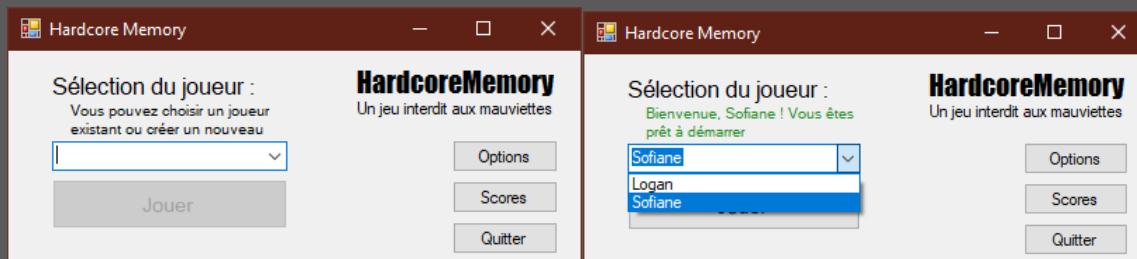
Rôle fonctionnel du projet :

L'objectif de ce projet est de programmer de créer des formulaires, des modules et des structures ayant une certaine logique car nous programmons un langage événementiel, Ainsi les formulaires se chargeront de l'affichage et les modules se chargeront de stocker les informations importantes.

Dans notre version, il est possible d'abandonner la partie en cours, il est aussi possible de modifier les paramètres avant de débiter une partie afin de configurer sa propre difficulté. Nous allons donc voir toutes les fonctionnalités qu'offre notre application

Fonctionnalités

Le formulaire principal :

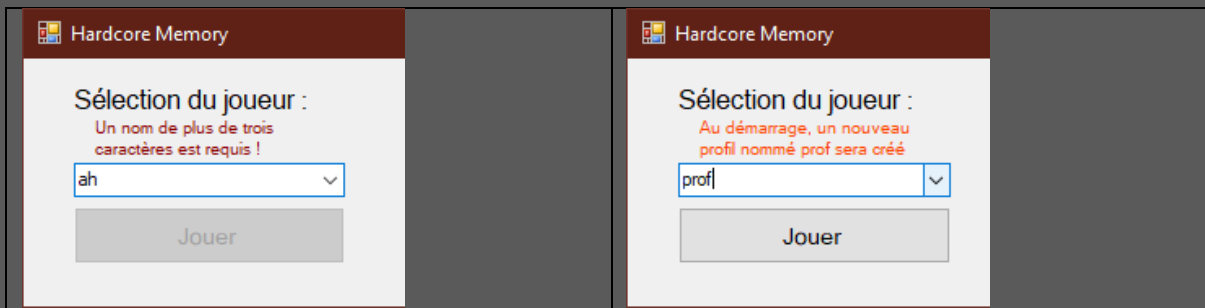


Le formulaire récupère la liste des profils depuis le module GameStorage. Celui-ci va obtenir les noms depuis le fichier de sauvegarde, ou s'il s'agit du premier lancement, charger deux profils vides par défaut.

Une fonction interne vérifie si le nom est valide à chaque mise à jour de la comboBox. Si le nom est invalide, le message deviendra rouge et empêchera le début de partie. Si le nom ne fait pas partie de la liste des noms référencés, GameStorage s'occupera de créer le profil au besoin lors de l'étape d'enregistrement.

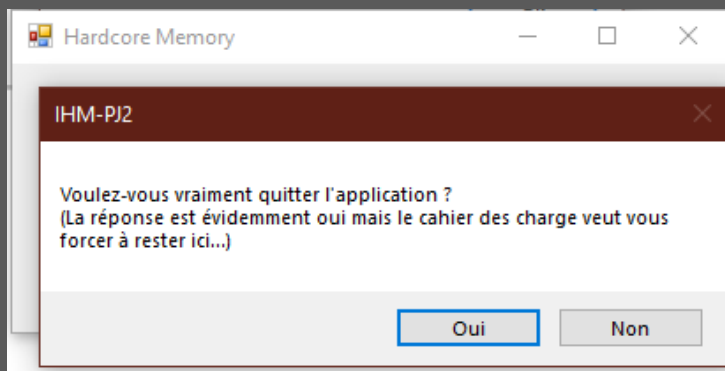
Profil invalide : rouge

Profil inexistant : orange



Au clic sur le bouton Play, le formulaire envoie à GameStorage le nom du joueur qui sera réutilisé ultérieurement

Lors de la fermeture, l'application demande confirmation au moment de quitter.



Le formulaire des options :

Le formulaire récupère les options depuis GameStorage, et formate l'affichage du temps à l'aide d'une fonction de GameUtils.

Suite à un manque de temps, nous n'avons implémenté que des options de difficulté.

Ainsi, quatre options sont réglables par le joueur :

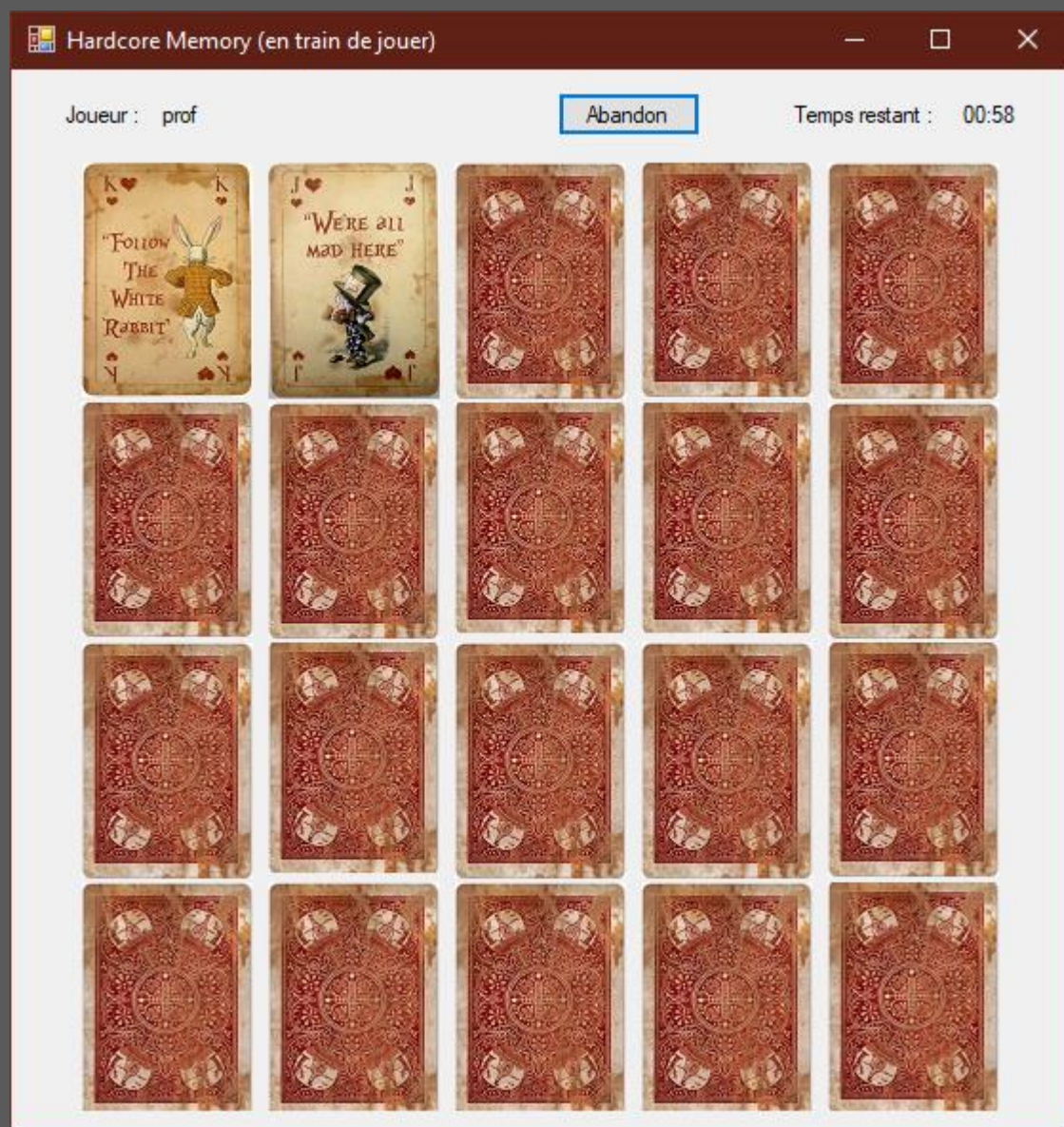
- Afficher le bouton pause
- ne pas trier les cartes (désactive le random // son seul but est de faciliter le debug)
- Temps modifiable entre 0 et 3 minutes
- Si le temps est mis à zéro, alors la fonctionnalité de timer est désactivée

Les boutons à bascules changent selon leur état (tant au texte qu'à la couleur)

Au clic sur le bouton enregistrer, le formulaire « set » les paramètres modifiés et GameStorage se charge ensuite de sauvegarder les nouvelles options dans le fichier de sauvegarde.

Configuration par défaut	Configuration de test (jeu très facile)
<div><p>Options du jeu</p><div><p>Règlage de la difficulté</p><p>La couleur verte indique que la partie sera très facile</p><div><div>Pause interdite</div><div>cartes mélangées</div></div><p>Temps alloué :</p><div><div>-</div><div>01:30</div><div>+</div></div></div><div><div>Enregistrer</div><div>Annuler</div></div></div>	<div><p>Options du jeu</p><div><p>Règlage de la difficulté</p><p>La couleur verte indique que la partie sera très facile</p><div><div>Pause autorisée</div><div>cartes non mélangées</div></div><p>Temps alloué :</p><div><div>-</div><div>Aucune limite de temps</div><div>+</div></div></div><div><div>Enregistrer</div><div>Annuler</div></div></div>

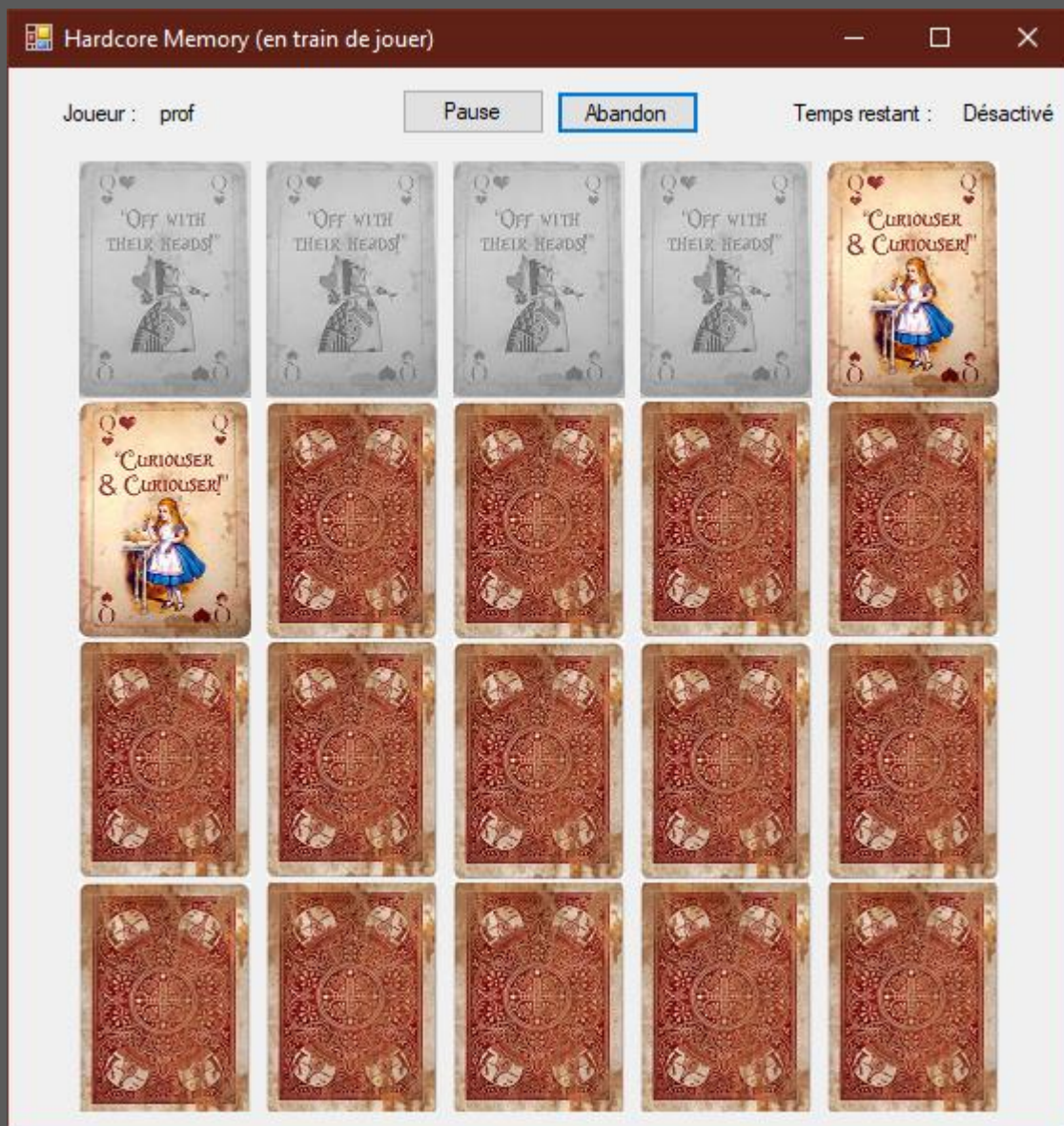
La fenêtre de jeu



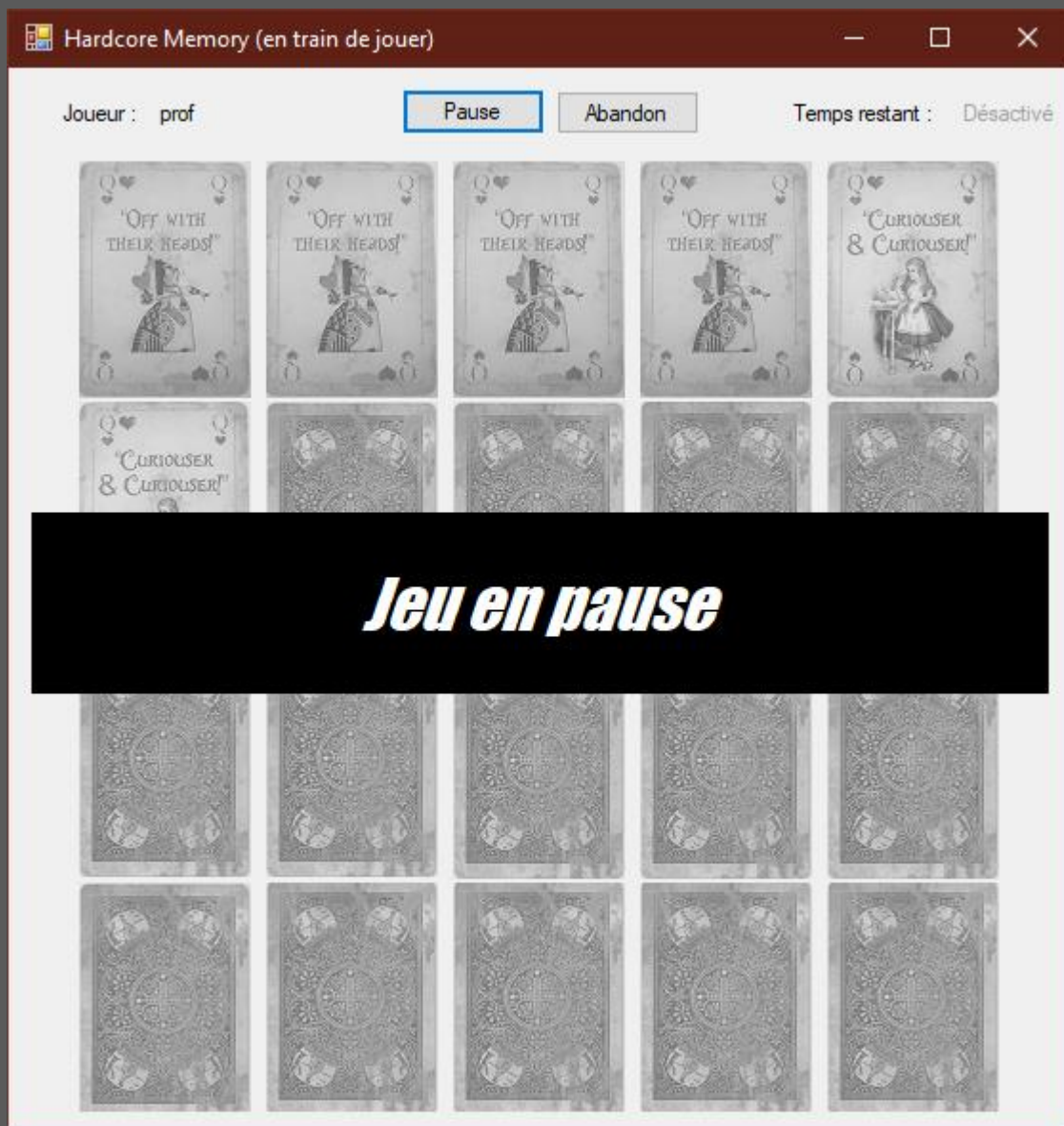
Fenêtre de jeu : tout ce qu'il y a de plus normal. Les cartes sont triées, pas de pause et 1 minute. Le temps défile au moyen d'un timer.

Le formulaire récupère depuis GameStorage les options du jeu ainsi que le nom du joueur courant.

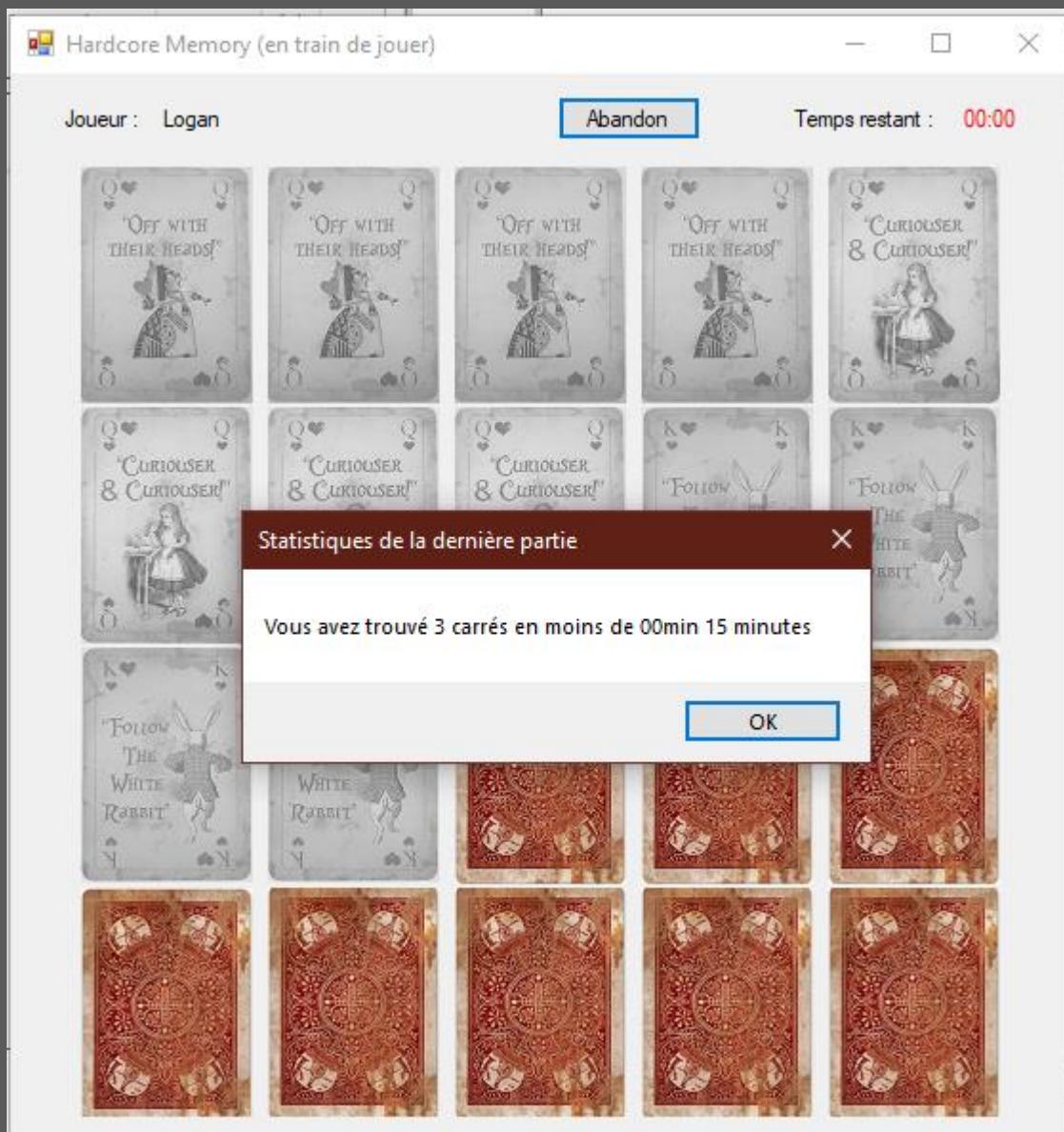
Une fonction très utile au sein du module GameUtils permet de convertir des secondes en une chaîne de caractère formatée. Fournir « mm minutes et ss secondes » avec un temps de 61 donnera « 01 minutes et 01 secondes ».



Cette partie est en niveau de difficulté nulle : carte pas triées, bouton pause et pas de limite de temps



Ecran de pause : Le timer et le panel contenant les cartes sont désactivés.



Les informations de fin de partie sont affichées en fin de jeu. Le formulaire se chargera ensuite d'envoyer le score à GameStorage pour faire si nécessaire une modification de statistiques du joueur actuel et une écriture dans le fichier de sauvegarde.

Tableau des scores

prof Statistiques ▼ Ordre de tri

Joueurs	Cumul tps jeu	Nbre parties	Score Max	Temps Lié
prof	00min 10	2	5	00min 09
Logan	00min 41	4	5	00min 12
Sofiane	00min 15	1	3	00min 06

Quitter

Le tableau des scores récupère les scores depuis GameStorage et formate les minutes à partir de la fonction de GameUtils. ComboBox et ListBox sont synchronisés.

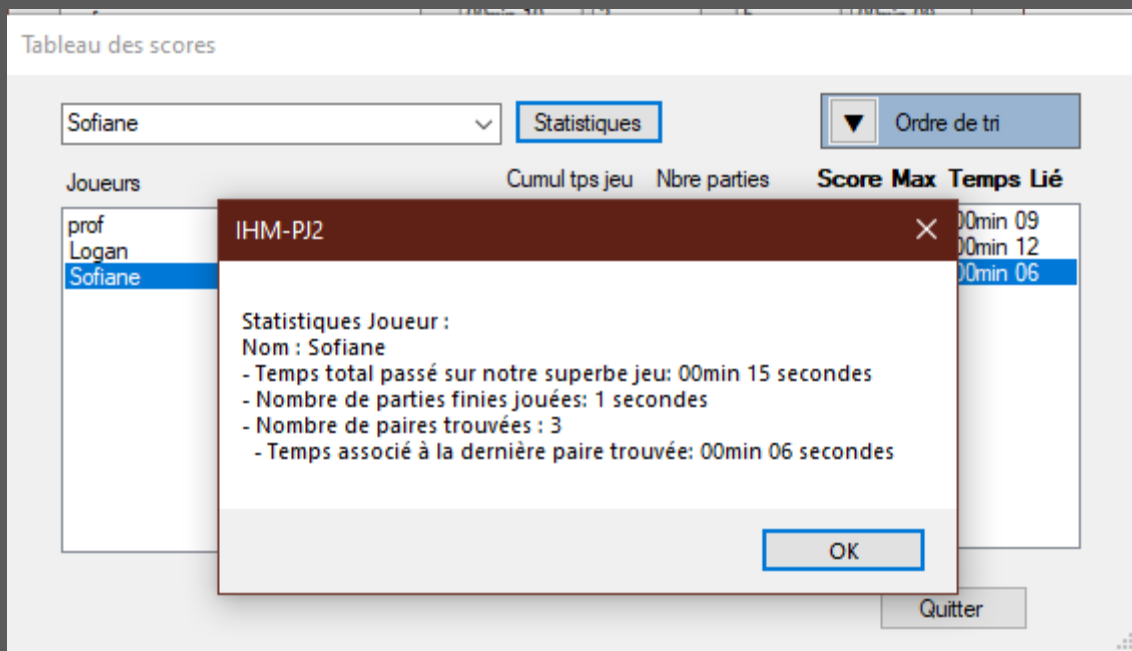
Tableau des scores

Logan Statistiques ▲ Ordre de tri

Joueurs	Cumul tps jeu	Nbre parties	Score Max	Temps Lié
Sofiane	00min 15	1	3	00min 06
Logan	00min 41	4	5	00min 12
prof	00min 10	2	5	00min 09

Quitter

Il est possible d'inverser l'ordre de tri en un clic. Nous avons utilisé la fonction `listeJoueurs.sort(Comparaison(Of Joueur))` en créant au préalable une fonction de comparaison entre deux joueurs. Plus d'informations et référence de Doc dans notre code.



Au clic d'une bouton, l'application récupère et affiche les stats du joueur sélectionné.

Schéma d'ordonnancement des formulaires

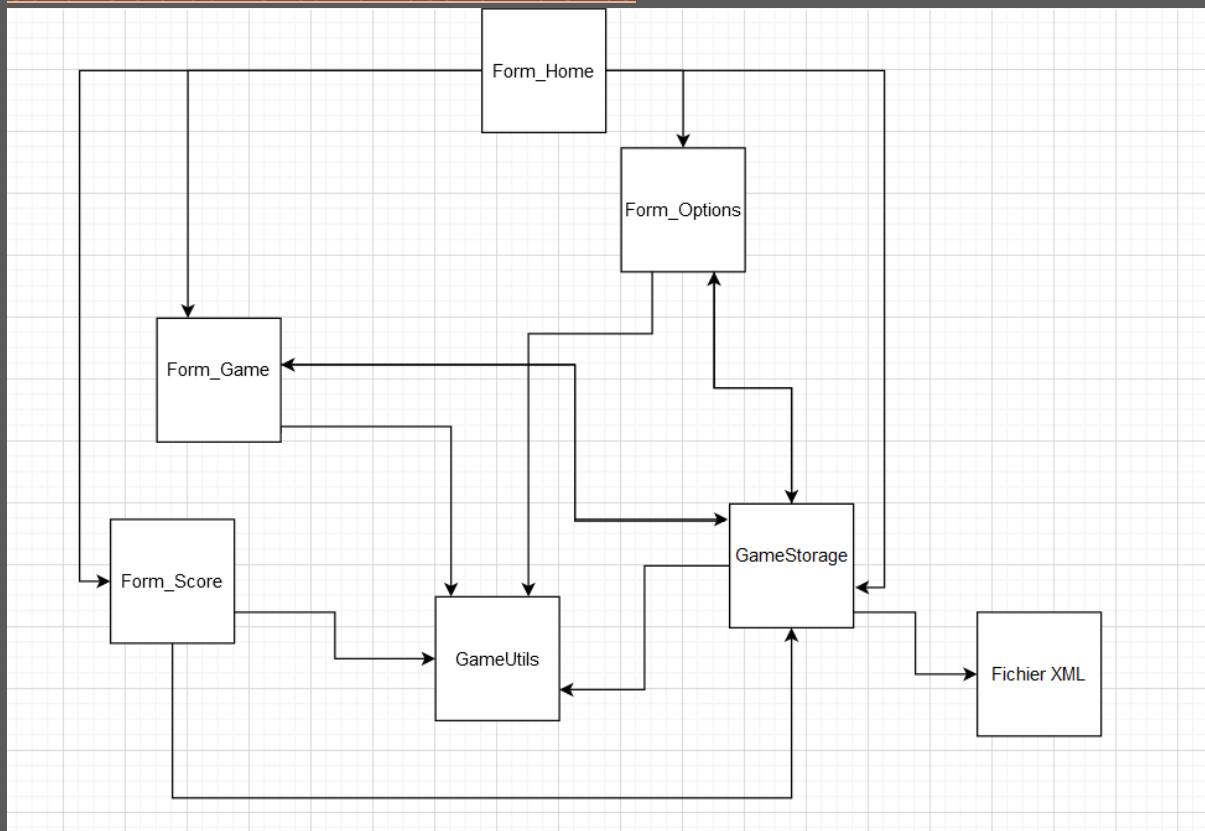
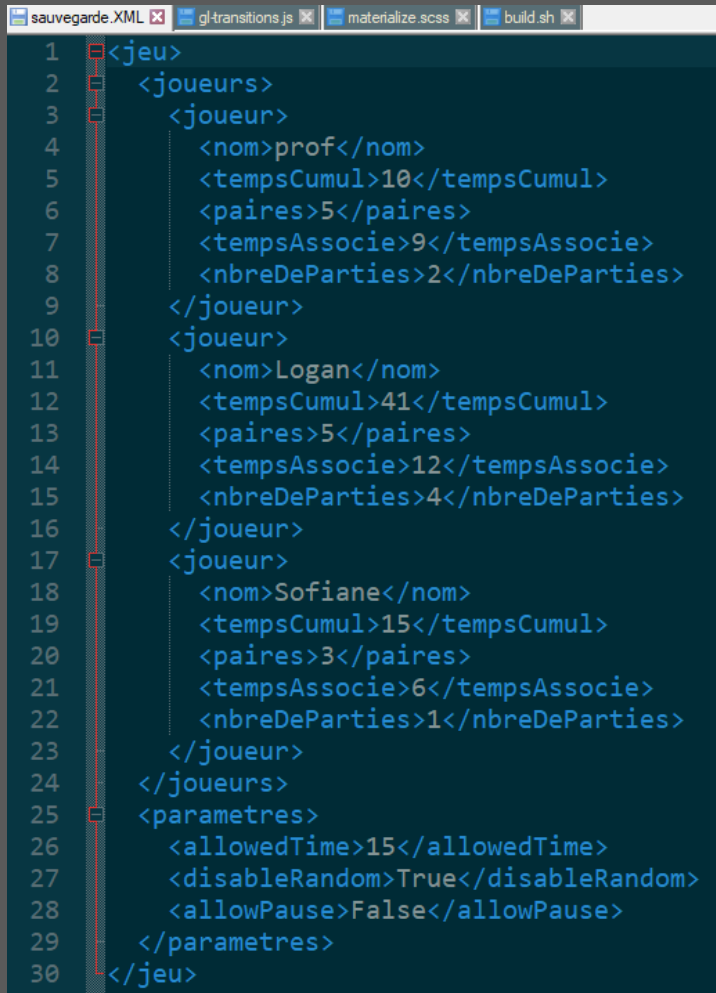


Diagramme de cas d'utilisation.

Fichier de sauvegarde



```
1 <jeu>
2   <joueurs>
3     <joueur>
4       <nom>prof</nom>
5       <tempsCumul>10</tempsCumul>
6       <paires>5</paires>
7       <tempsAssocie>9</tempsAssocie>
8       <nbreDeParties>2</nbreDeParties>
9     </joueur>
10    <joueur>
11      <nom>Logan</nom>
12      <tempsCumul>41</tempsCumul>
13      <paires>5</paires>
14      <tempsAssocie>12</tempsAssocie>
15      <nbreDeParties>4</nbreDeParties>
16    </joueur>
17    <joueur>
18      <nom>Sofiane</nom>
19      <tempsCumul>15</tempsCumul>
20      <paires>3</paires>
21      <tempsAssocie>6</tempsAssocie>
22      <nbreDeParties>1</nbreDeParties>
23    </joueur>
24  </joueurs>
25  <parametres>
26    <allowedTime>15</allowedTime>
27    <disableRandom>True</disableRandom>
28    <allowPause>False</allowPause>
29  </parametres>
30 </jeu>
```

Fonctionnement du système de sauvegarde Le module GameStorage s'occupe de stocker les statistiques des joueurs et les options du jeu, représentés respectivement par deux structures. Le module s'utilisant comme une classe, nous avons défini un attribut privé stockant la structure de paramètres, et une List (Of Joueur) stockant les joueurs. Nous avons ensuite défini des méthodes d'entrées et de sortie afin que les formulaires récupèrent la liste des joueurs, modifient leurs statistiques, obtiennent la liste d'options etc. Nous avons enfin effectué une fonction Sauvegarder () ou Charger () qui permet respectivement d'écrire ou de lire le fichier de sauvegarde. Nous avons choisi de stocker les statistiques de chaque joueurs et les options du jeu dans un fichier XML pour plusieurs raisons : - Le fichier de sauvegarde était rendu plus élégant - Le XML était intégré nativement au sein de .NET - C'était un format que nous ne connaissions pas bien et nous avons voulu essayer (screen du fichier sauvegarde.XML) Même si cela reste plus élégant en comparaison à un fichier texte, la création d'un document XML est très verbeuse donc assez longue. Il faut compter environ 4 lignes par balises, représentant chacune une structure ou leurs propriétés. Il est également facilement possible de se perdre entre elles. Une autre possibilité d'implémentation aurait été d'utiliser le format JSON. [18 :37]

Conclusion

Conclusion Pour conclure, ce projet a été plutôt intéressant. L'idée de créer un memory avec des fonctionnalités précises était très adapté au VB, ce qui n'était pas réellement le cas avec le projet GPI dont le type de projet aurait été bien plus facile à faire en tant qu'application en ligne plutôt que depuis les formulaires Access. Malgré une syntaxe peu usuelle, le VB.NET tire beaucoup de sa force dans la librairie standard et permet la réalisation d'applications graphiques efficace (au prix d'un design peu moderne) très rapidement. En observant les questions d'autres camarades, on remarque aussi que le projet aurait pu être fait de multiples manières pour un même résultat, et nous avons trouvé ce dernier concept tout aussi intéressant. Nous n'avons pas eu de difficultés particulières, si ce n'est que le temps. Nous avons pu finir le projet mais nous aurions bien voulu créer de nouvelles fonctionnalités ou réécrire certaines fonctions. Si nous devrions citer le point qui nous a le moins plu, il s'agit de la création d'une boîte de confirmation pour quitter l'application. En effet, il existe un event permettant de détecter lorsqu'une application se ferme (ce qui est le seul moyen de détecter le clic du bouton X rouge de fermeture), mais celui-ci se déclenche également lorsque nous faisons Me.Close(). Résultat : nous devons nous retrancher sur Me.Hide() qui ne s'occupe que de cacher le formulaire sans le fermer complètement, ce qui peut occasionner certains bugs. De plus, confirmer la fermeture est souvent une pratique contre-intuitive et agressive, à moins de confirmer l'enregistrement d'un fichier. Le code source est disponible sur GitHub à l'adresse suivante : <https://github.com/LoganTann/IHM-PJ2>