

SAE 2.01 - Développement d'une application**Une variante du jeu de Memory**

Introduction.....	2
Description, Formulaire et codes.....	3-6
-Formulaire 1 (Accueil).....	3
-Formulaire 2 (Jeu).....	4
-Formulaire 3 (Score).....	5
-Formulaire 4 (FinJeu).....	6
Module.....	7
Codes, méthodes utilisées.....	8

Introduction

Notre projet de jeu Memory est fait par une personne : Nguyen Quoc Dat.

Le projet consiste à développer une application similaire à Memory avec un temps limité et quelque changement des règles du jeu qui est :

- il y a 25 cartes à révéler
- le nombre de cartes à révéler est augmenté jusqu'à 4 cartes au lieu de 2
- une durée limitée au choix

L'application est programmée sur Visual Studio 2022 avec le modèle « Application Windows Forms (.NET Framework).

L'application contient 4 formulaires, 1 module, 1 dossier d'images, 1 dossier pour l'enregistrement des joueurs dans un fichier txt.

Les 4 formulaires sont :

- L'accueil
- Le jeu
- Les scores
- Fin jeu

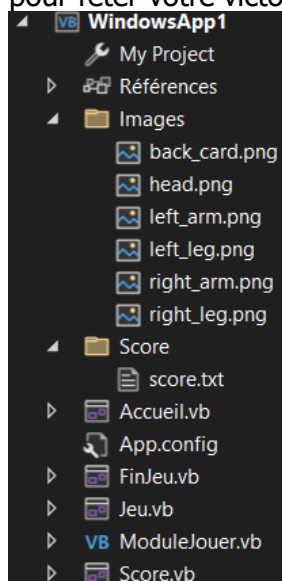
Le module (ModuleJoueur) sert à concevoir les informations des joueurs, les enregistrer et les afficher.

Le dossier d'images sert à stocker les images afin de les réafficher sur les cartes.

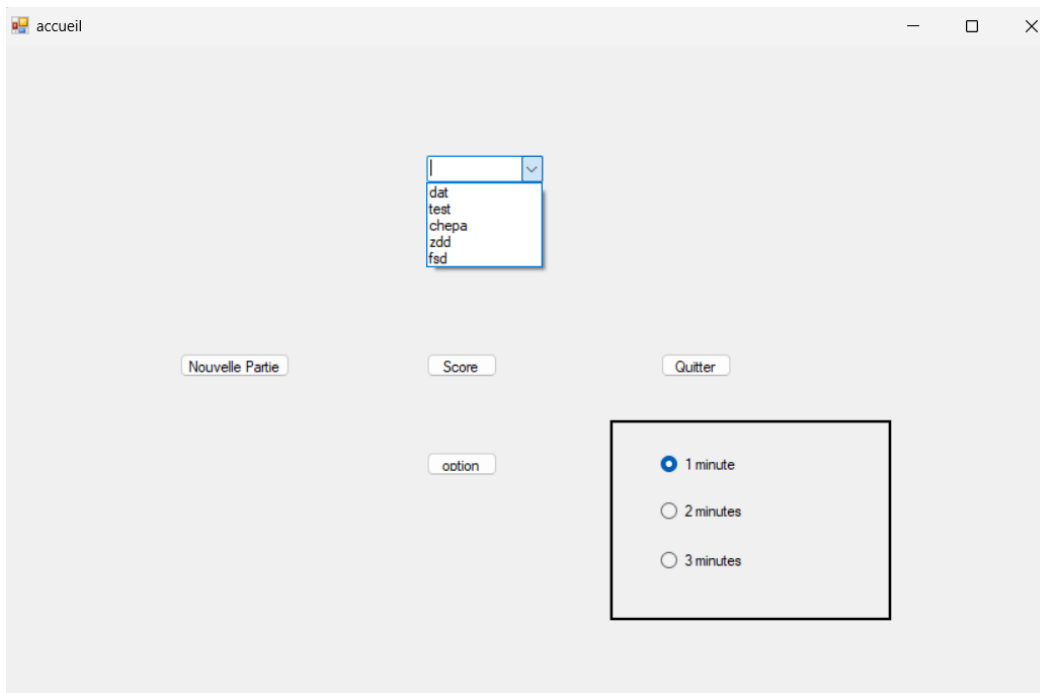
Le dossier dotant du fichier txt sert à stocker des informations des joueurs à la fermeture de l'application

A l'exécution de l'application nous retrouverons sur le formulaire principal Accueil qui peut nous diriger sur d'autres formulaires, le Jeu et les Scores.

Pour aller au formulaire score il n'y a pas de contrainte par contre pour se diriger vers le formulaire Jeu il y a des contraintes (page 3), si vous avez fini le jeu avant que le temps s'est écoulé alors vous avez gagné et dirigera automatiquement vers le formulaire (FinJeu) pour fêter votre victoire, si vous n'avez pas réussi vous retournerez vers la page d'accueil.



Formulaire 1 : Accueil



Outils utilisés :

- 1 combobox
- 4 Buttons
- 1 panel
- 3 Radio-Buttons

Au chargement du formulaire le panel sera caché il est accessible seulement via le Button « option », le fichier txt des joueurs sera chargé et ses informations seront utilisées tout au long de l'application, la méthode `Private Sub Accueil_Load()` fait tout cela.

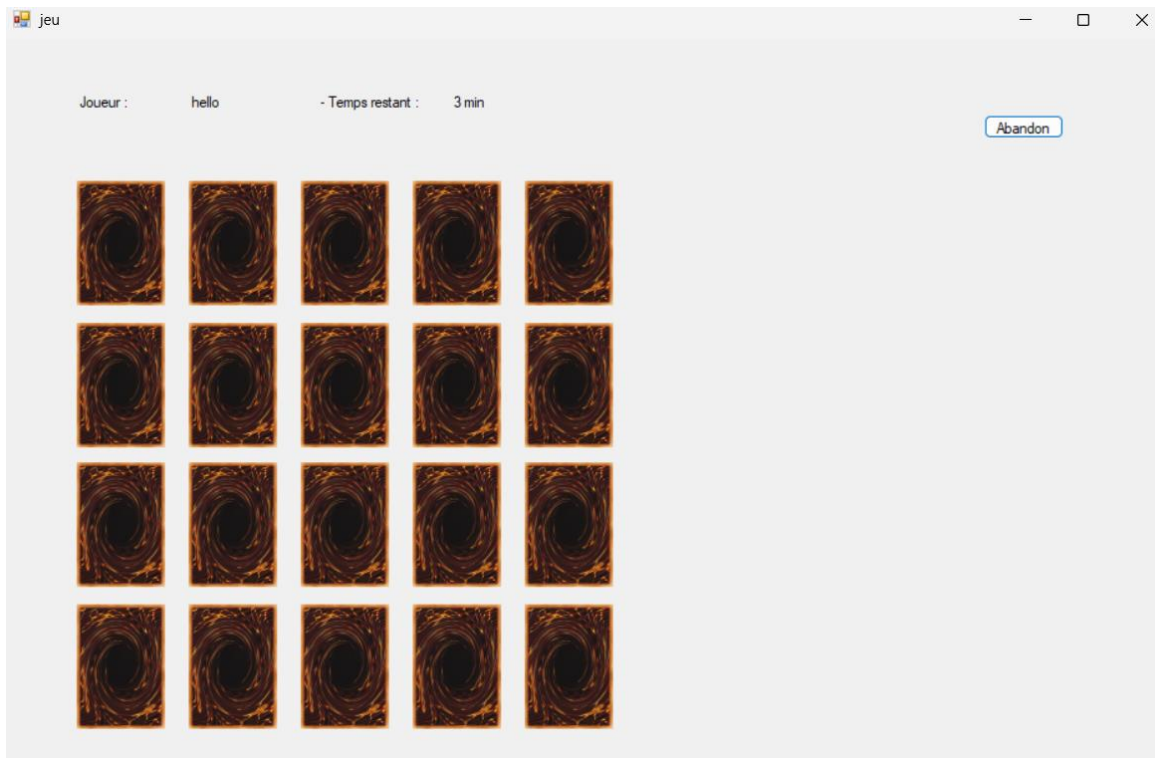
Pour lancer une nouvelle partie il nous mènera jusqu'à le formulaire « Jeu » il suffit de remplir votre pseudo dans le combobox mais il y a des contraintes à respecter, au moins 3 caractères à l'exception du caractère « ; » qui est utilisé dans la façon dont enregistrer les joueurs, un msgbox sera affiché si vous ne respectez pas cela, la méthode `Private Sub NouvellePartie_Click()` fait tout cela.

Le Button score nous dirigera vers le formulaire « score » où il aura tous les scores de tous les joueurs ayant jouer au moins une fois la méthode `Private Sub Score_Click()` fait tout cela.

Le button « quitter » nous permet de quitter le jeu avec confirmation via msgbox, la méthode `Private Sub Quitter_Click()` fait cela.

le button « option » nous affichera un panel contenant 3 radio button afin d'ajuster le temps de jeu par partie selon le choix du joueur, la variable « `tmp_add` » enregistrera le choix du joueur via ses button radio et enverra l'information vers le formulaire « jeu » lorsque le joueur lance une nouvelle partie, tous les méthodes liées avec le button « option », le panel, les radio button sont concernés.

Formulaire 2 : Jeu



Outils utilisés :

-24 labels (20 cartes)

-1 Button

-1 Timer clock

Au chargement du formulaire, nous importerons les informations venant de la page précédente (accueil), le nom du joueur (`NomJoueurF2()`) pour les afficher et le temps jeu qu'il a choisi (`tmpAdd()`) et le chemin du fichier txt (`CheminFichierF2()`) afin qu'il enregistre les données à la fin de la partie ou lors de la fermeture de cette formulaire, résumer avec la méthode `initInfo()`.

Les 20 labels en dessous deviendront des images qu'on a importé depuis le dossier « Images », il y a 6 images en tout : 1 pour le dos de la carte (`back_card`) et les 5 images différentes (`head`, `right_arm`, `left_arm`, `right_leg`, `left_leg`).

Tout d'abord nous créons un objet « Card » qui prendra son image (`Image`), son statut révéler ou pas en booléen (`Devoile`) et le nom de l'image (`Id`) car nous ne pouvons pas comparer image avec image mais on peut comparer leur nom.

Puis on crée une liste de « Card » (`cards`) et on ajoute respectivement 4 fois de la même carte pour 5 cartes différents et puis on mélange la liste aléatoirement tout dans la méthode `initCard()`.

La première carte jouée débutera le jeu en écoulant le temps, `Timer1_Tick()`

Ensuite chaque fois qu'on click sur une carte cela révélera l'image selon son indice dans la liste ex : `label1` -> indice « 0 » donc dans la liste on prendra la carte à l'indice « 0 », pourquoi « 0 » ? car la position du premier objet est à l'indice « 0 », `Card_Click()`

Toutefois quand une carte est révélée nous ne pouvons plus la révéler à nouveau.

Lorsque le jeu est terminer les données seront enregistrées dans une liste voir (la module)

Il existe plusieurs façon de finir le jeu soit par abandon avec le Button soit le temps s'écoule soit par fermeture du formulaire, ou après avoir réalisé 5 carré.

Formulaire 3 : Score

nom	score	records	parties	temps jouer
chepa	5	84	2	85
zdd	1	20	1	21
test	1	13	1	15
dat	0	0	1	27
fsd	0	0	1	0
hello	0	0	2	0
sdf	0	0	1	3

chepa find

Croissant

Joueur : chepa
Score : 5
Records : 84sec
Nombre de parties : 2
Temps jouer : 85sec

OK

retour

Outils utilisés :

- 5 listbox
- 5 labels
- 3 button
- 1 combobox

Ce formulaire affiche les meilleurs scores des joueurs ayant jouer au moins 1 fois, au chargement du formulaire la liste des joueurs sera triée par ordre décroissant par score et son record `Private Sub Score_Load()`, les informations seront affichées dans les différents listbox selon son type d'information `Private Sub initListBox()`, le button « Croissant » permettra de trier les liste box en ordre croissant `Private Sub trieCroissant()`, `Private Sub trieDecroissant()`.

On peut choisir dans la première listbox des noms des joueurs `Private Sub ListBox1_SelectedIndexChanged()` et fait apparaître dans le combobox ou faire une recherche directement dans le combobox et lorsqu'on appui sur le button « find » `Private Sub Button2_Click()` il affichera les données du joueur qu'on cherche si il existe ou pas on informe le joueur via une msgbox, `Private Sub GetStringStatJoueur()`.

Le button retour permet de retourner dans le formulaire principal `Private Sub Button_Click1()`.

Formulaire 4 : Fin jeu

The screenshot shows a Windows-style window titled 'FinJeu'. Inside, there are two labels: 'Joueur : Label5' and 'temps : Label7'. Below these is a large dashed rectangular panel. Inside this panel, there are several labels and images: 'Label3' is above the text 'You La'; 'Label10' is below 'You La'; 'Label8' is below the text 'human the'; 'Label9' is above a large 'A' character. There are also some small, partially visible characters and symbols within the panel.

Il n'y a pas grande chose à dire sur ce formulaire, c'est un bonus pour le joueur quand il a réussi à trouver 5 carrés.

Outils utilisés :

- 10 labels
- 1 panel

Au chargement du formulaire le nom du joueur et son temps seront importés et affichés, un panel pour féliciter le joueur, puis le panel sera caché après un temps déterminé et il aura une animation pour afficher les cartes que le joueur avait trouvées, après un certain temps donné le joueur se dirigera automatiquement vers l'accueil afin qu'il puisse jouer une autre partie ou voir son score, `Private Async Sub FinJeu_Load()`.

Le Module

Le concept d'un module est de stocker les informations des joueurs, les modifier et les réutiliser.

Dans notre cas, nous avons créé un objet « **Joueur** » dedans on retrouve son nom, son score, son temps réaliser, son nombre de parties et son temps de jeu en total.

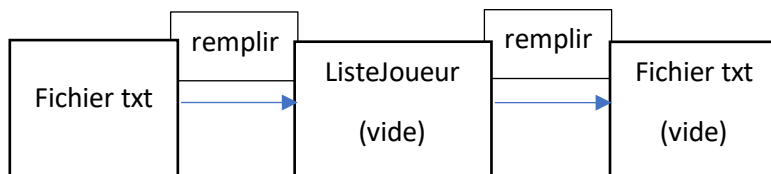
Puis nous créé une liste de Joueur (**listeJoueur**) dans la liste il y aura tous les joueurs.

Le module contient les méthodes pour initialiser la liste, le mettre à jour, enregistrer dans le fichier txt.

Lors du chargement du formulaire « **Accueil** » la liste des joueurs est vidée avant d'être initialisé **Public Sub CreerJoueur()**, en important tous les données du fichier txt, chaque ligne de texte représente un joueur, le syntaxe sera

« **nom;score;temps_réaliser;nbr_parties;temps_jouer** », entre chaque information du joueur est séparé par un caractère « ; » qui va permet de distinguer individuellement chaque information et ce choix de caractère au lieu d'un simple espace est qu'on veut que le joueur aille un nom qui peut contenir des espaces.

Quand le joueur lance une nouvelle partie, un nouveau joueur sera créé à part si on n'a pas trouver un nom similaire déjà existant **Public Function TrouverOuCreerJoueur**, ce joueur sera ajouté directement dans la liste avant que le joueur fait son premier action de révéler une carte, à la fin de la partie quoique soit la façon dont il termine par fermeture du jeu, abandon, gagné, le joueur dans la liste sera mise à jour **Public Sub MAJ_Jouer()**, et on va vider le fichier txt et commencer à le remplir à partir de la liste des joueurs **Public Sub EnregistrerFichier()**.



Codes, méthodes utilisées

Formulaire 1 (Accueil) :

```
Private Sub Accueil_Load()  
Private Sub NouvellePartie_Click()  
Private Sub Score_Click()  
Private Sub Quitter_Click()  
Private Sub Option_Click()  
Private Sub PanelOption_Paint()  
Private Sub RadioButton1Min_CheckedChanged()  
Private Sub RadioButton2Min_CheckedChanged()  
Private Sub RadioButton3Min_CheckedChanged()
```

Formulaire 2 (Jeu) :

```
Public Sub NomJoueurF2()  
Public Sub initChemin()  
Public Sub tmpAdd()  
Private Sub Form2_END()  
Private Class Card  
Private Sub Jeu_Load()  
Private Sub initInfo()  
Private Sub initCard()  
Private Async Sub Card_Click()  
Private Sub Abandon_Click()  
Private Sub finJeu()
```

Formulaire 3 (Score) :

```
Private Sub Score_Load()  
Private Sub Button1_Click()  
Private Sub ListBox1_SelectedIndexChanged()  
Private Sub initListBox()  
Private Sub trieCroissant()  
Private Sub trieDecroissant()  
Private Sub GetStringStatJoueur()  
Private Sub Button3_Click()  
Private Sub Find_Click()
```

Formulaire 4 (FinJeu) :

```
Public Sub initImages()  
Private Async Sub FinJeu_Load()  
Private Sub Panel1_Paint()
```

Module :

```
Public Class Joueur  
Public Sub MettreAJourStats()  
Public Function TrouverOuCreerJoueur()  
Public Sub CreerJoueur()  
Public Sub MAJ_Jouer()  
Public Sub EnregistrerFichier()
```