

## Rapport de projet :



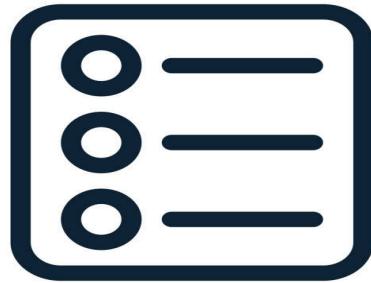
**IUT de Paris - Rives de Seine**  
Université Paris Cité

-BUT1 – SAE 2.01-  
-Développement d'une application-



**VB.NET**

Par NGUYEN Phuong, SUTHARSAN Amsan, MRABET-GHAZI Abdelkarim et  
BEN AKREMI Ali groupe 109



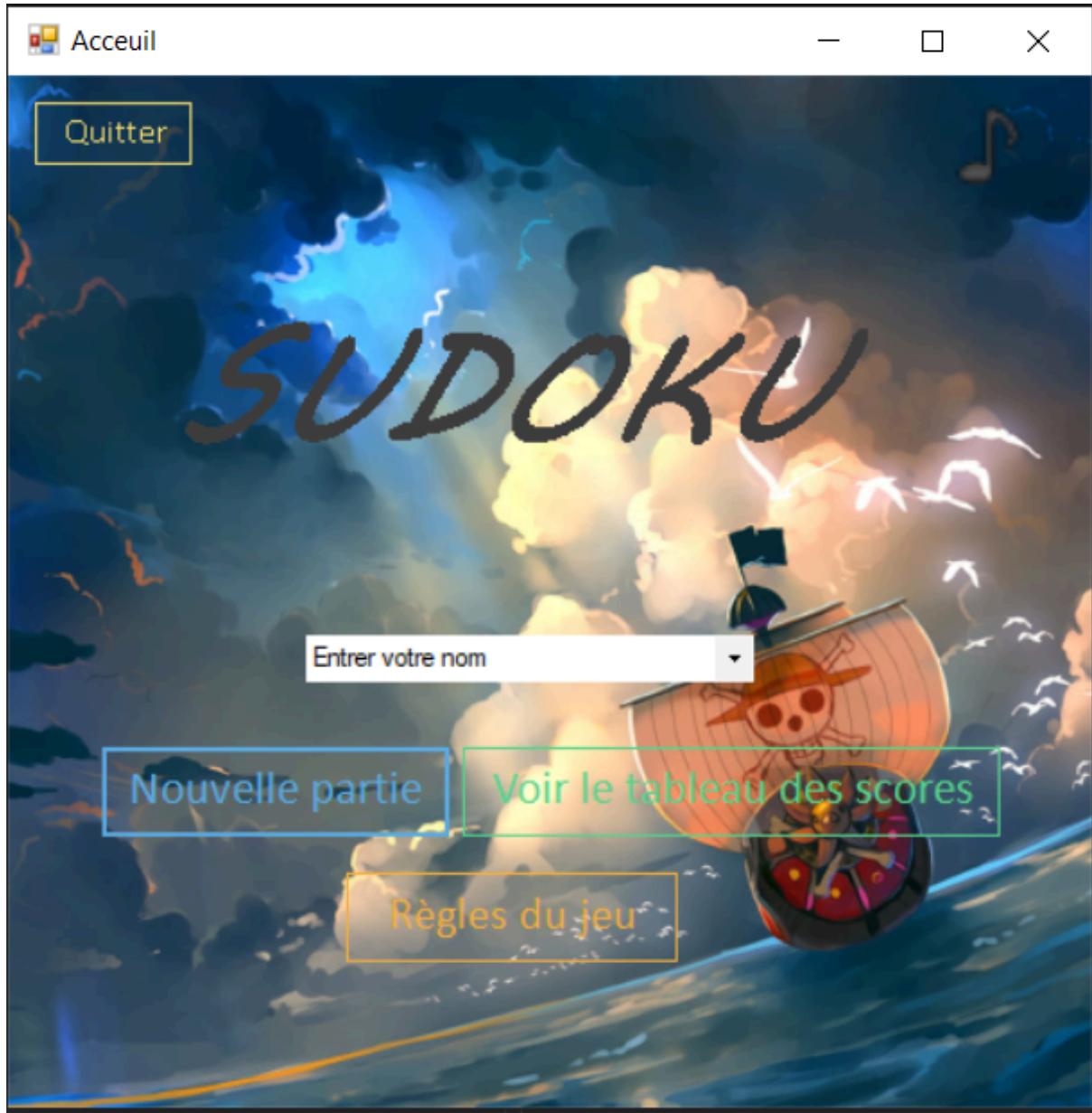
## Table des matières :

- I. Présentation du déroulement l'application, illustrée par des captures d'écran -p3
- II. Description des fonctionnalités implémentées, illustrées par des captures d'écran -p12
- III. Description de ce qui a changé depuis votre "pré-recette" -p21
- IV. Un schéma et une description de la façon dont les données sont structurée dans le module et les éventuels fichiers -p22
- V. Le code source des modules ainsi que des gestionnaires d'événements des formulaires -p23

## I.Présentation :

Voici le Sudoku, le but du jeu est de remplir une grille 9x9 avec une série de chiffres tous différents.

Quand vous lancez le jeu, voici la page qui va apparaître:



Vous avez le bouton “Règles du jeu”, pour afficher les règles du jeu et son fonctionnement pour les nouveaux joueurs qui découvrent le jeu.

Voici la page affichant les règles du jeu:

[Retour](#)

# Règles du jeu

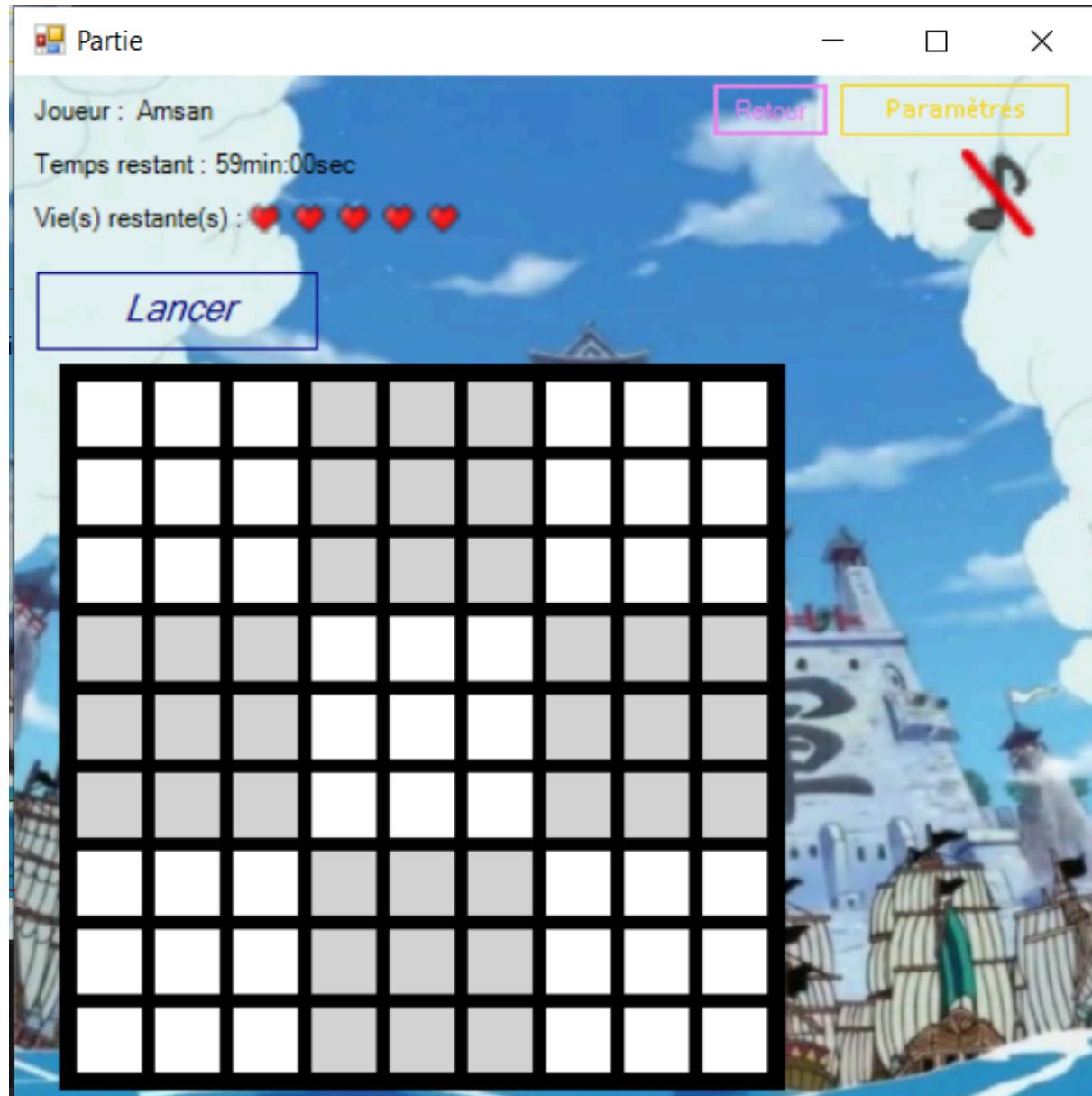
Le but du jeu est de remplir la grille (9x9) avec une série de chiffres tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même région.

Vous disposez d'un temps limité pour le résoudre ainsi que d'un nombre d'erreur limité à 5 fautes.

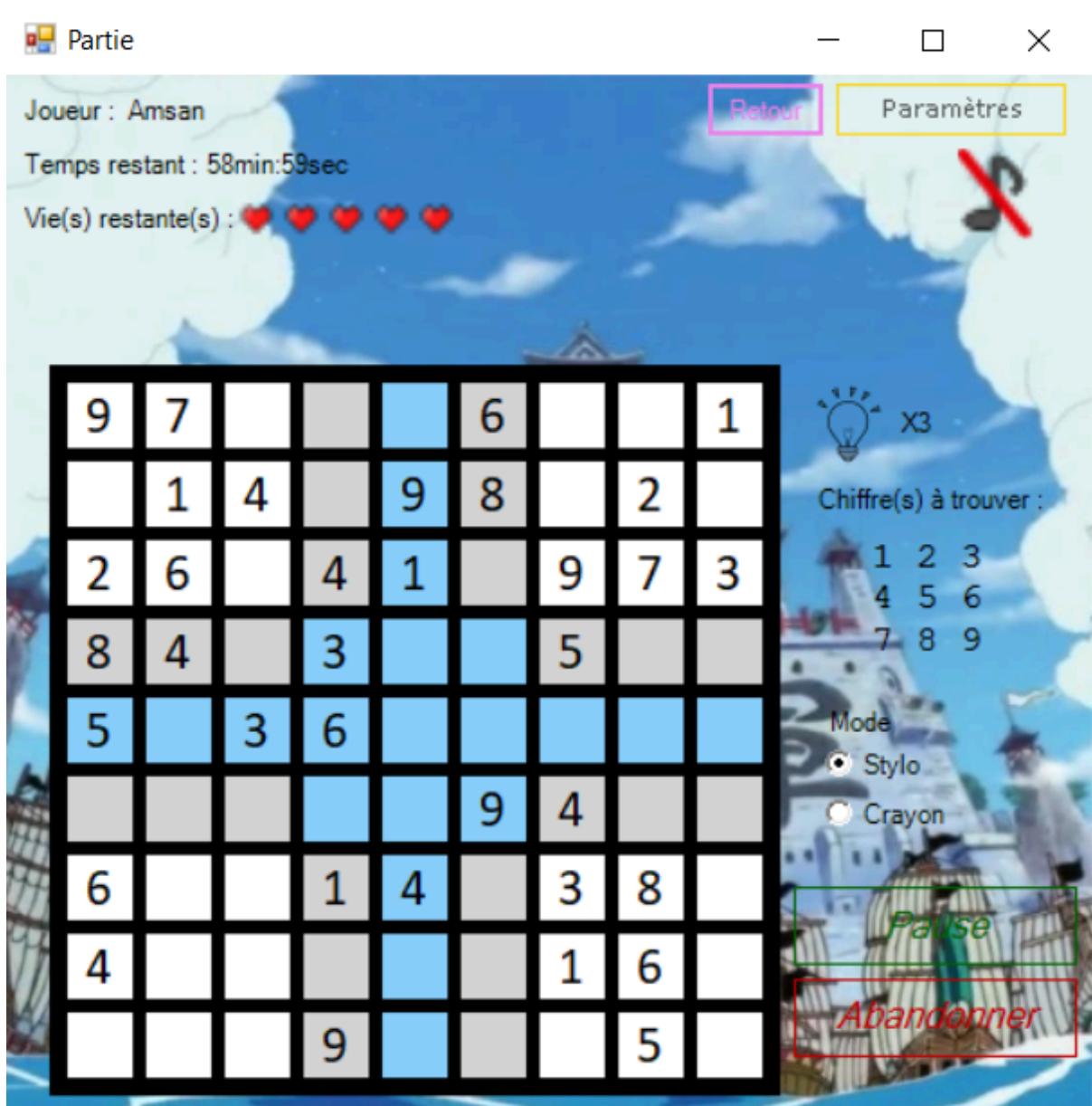
Vous disposez d'outils pour compléter le sudoku, le stylo qui permet de saisir une réponse, le crayon pour placer des notes dans les cases (sans générer d'erreurs) et s'adapte en fonction des réponses. Et les indices (3 par partie) qui révèle aux joueur la case demandée. Vous avez aussi la possibilité de changer la difficulté (le nombres de cases révéler au début), le temps maximum pour résoudre le sudoku, et de jouer avec ou sans les indice.

Grâce à cela, le joueur pourra connaître les règles et débuter le jeu avec connaissance.

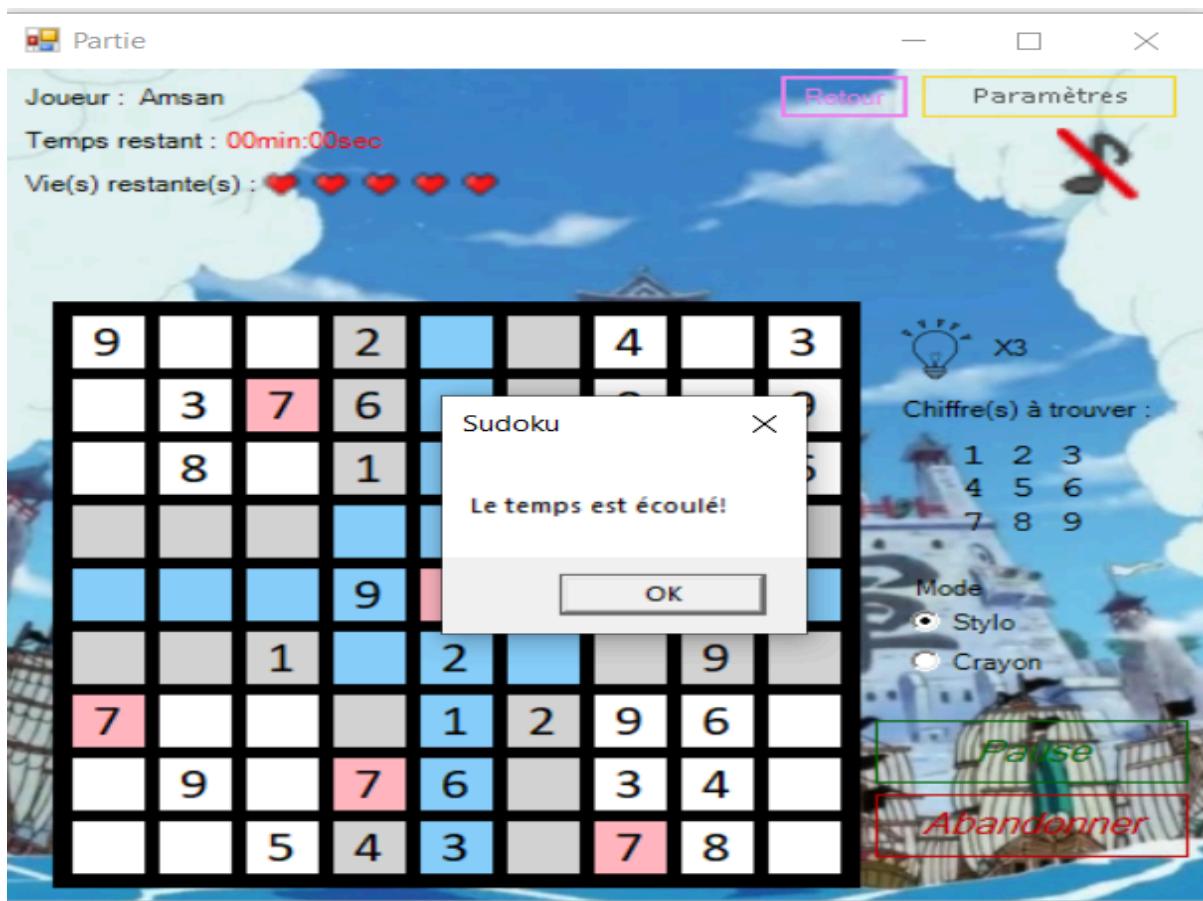
Retournons au menu, pour lancer une partie, vous devez tout d'abord entrer votre nom, ensuite vous arrivez sur la page de la partie.



Et vous appuyez sur lancer pour commencer la partie.



La grille se remplira et vous devez remplir tous les cases manquantes avec les chiffres qui correspondent sans mettre des doublons sur que ça soit sur la ligne, colonne et dans le carré 3x3 avec un temps restant. Vous avez des vies, pour chaque erreur que le joueur fait, il a jusqu'à 5 erreurs possibles sinon le jeu est terminé et aussi si le temps restant tombe à 0.



Joueur : Amsan

Retour

Paramètres

Temps restant : 00min:55sec

Vie(s) restante(s) :



	5		2	7			4	
	2	8	5					
			7	8				
	3							
	1	9						
8	4	2		3			1	5
2		5			9			
	7							2
9					2		8	

Sudoku

Vous n'avez plus de vies

OK



Chiffre(s) à trouver :

1 2 3  
4 5 6  
7 8 9

Mode

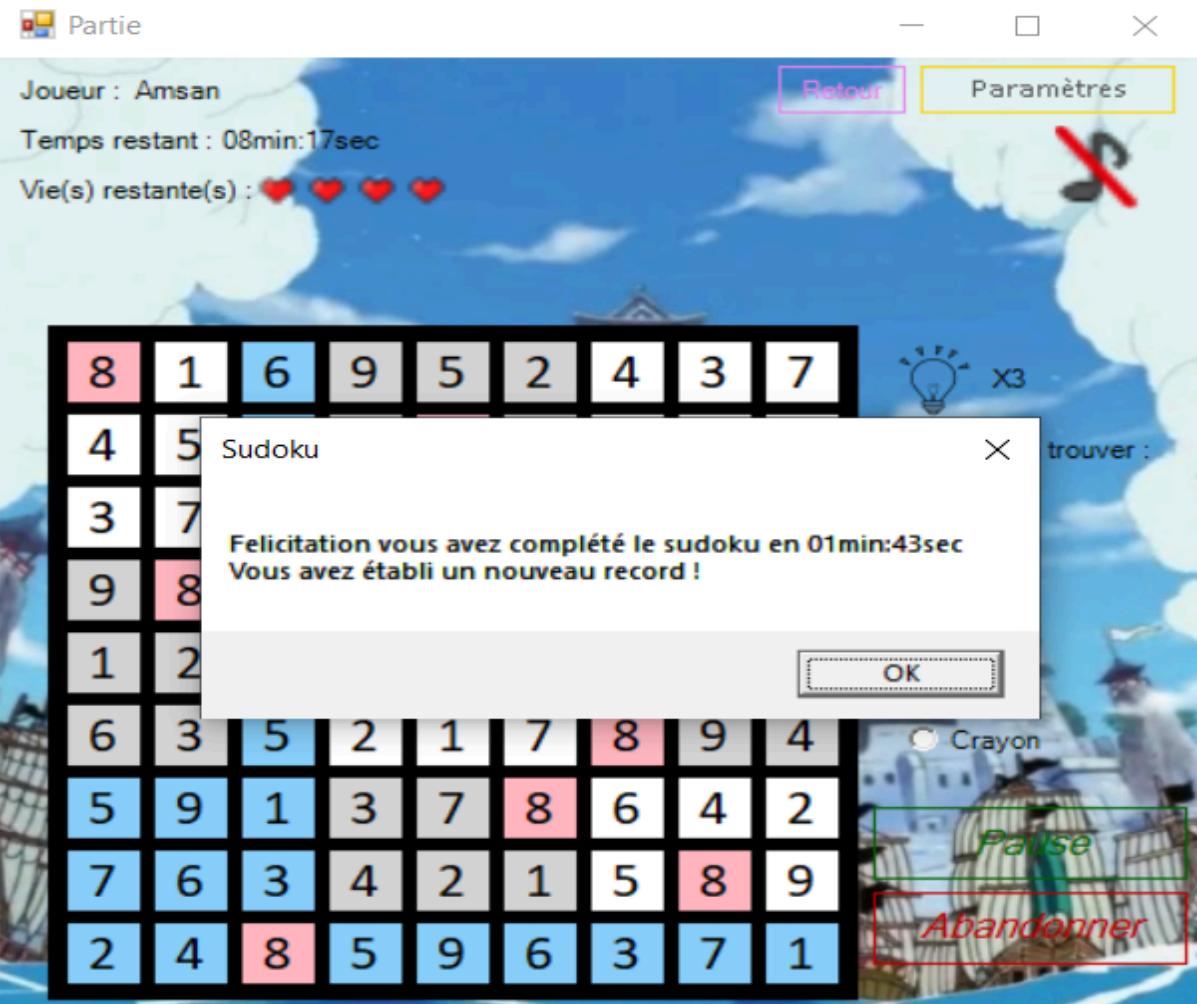
Stylo

Crayon

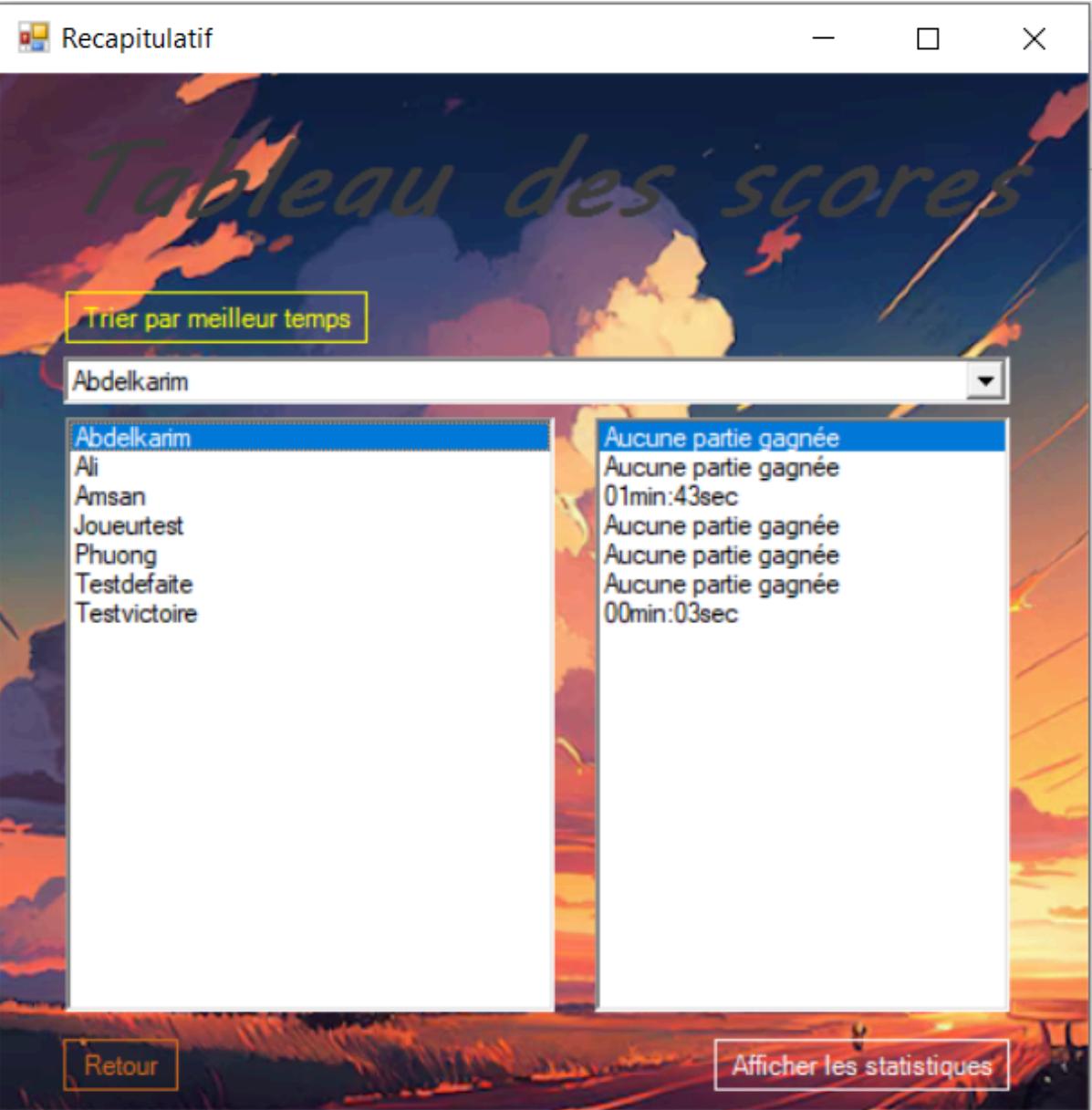
Pause

Abandonner

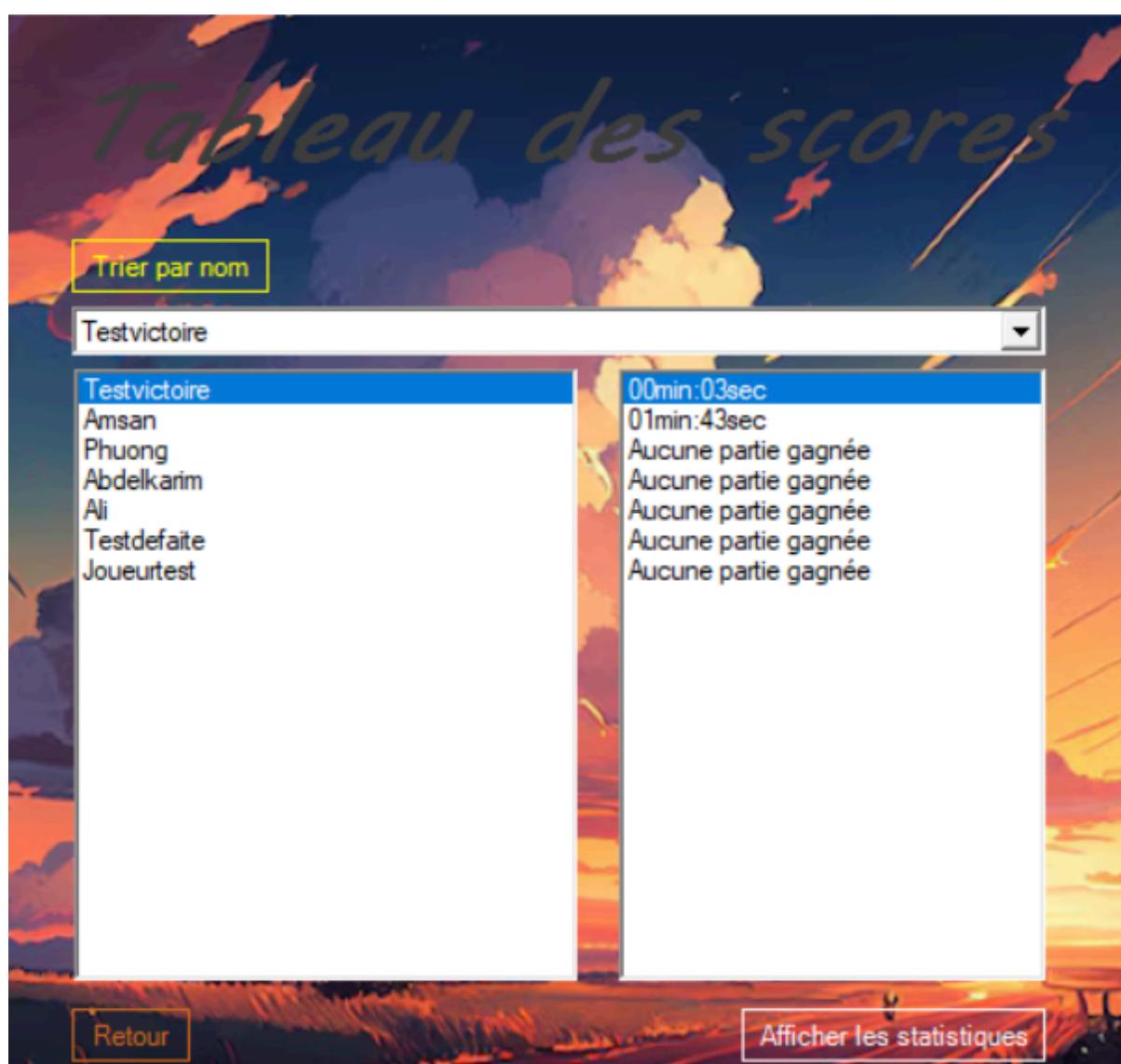
Quand vous complétez le sudoku, vous gagnez la partie et vous recevez un message de félicitation et avec le temps que vous avez passé.



Vous savez maintenant comment jouer et vous pouvez voir votre score dans le menu et tableau des scores qui sont triés par nom et meilleurs temps.

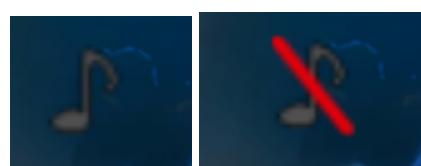


## Recapitulatif

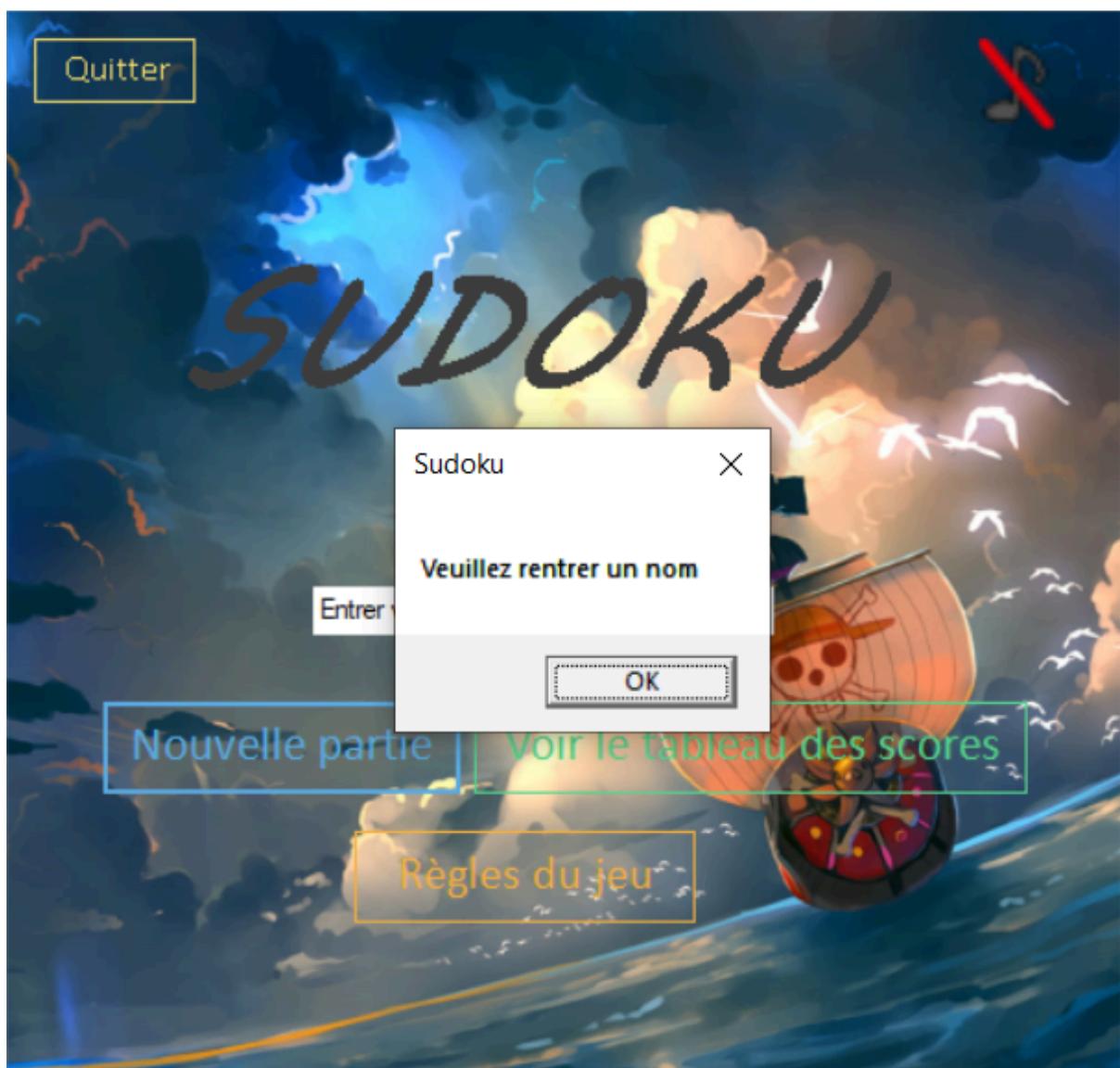


## **II. Description :**

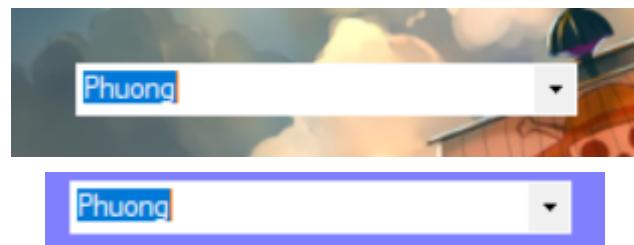
Alors tout d'abord quand vous lancez le jeu, il y aura une musique One piece opening instrumental qui va se lancer automatiquement pour s'ambiancer et se divertir pendant la partie, la musique ne s'arrête pas et se relance à chaque fois mais si le joueur en a marre d'écouter la musique, il peut arrêter la musique en cliquant sur le logo.



On ne peut pas lancer de partie sans que le joueur est d'entrer son nom et un message d'alerte va s'afficher.



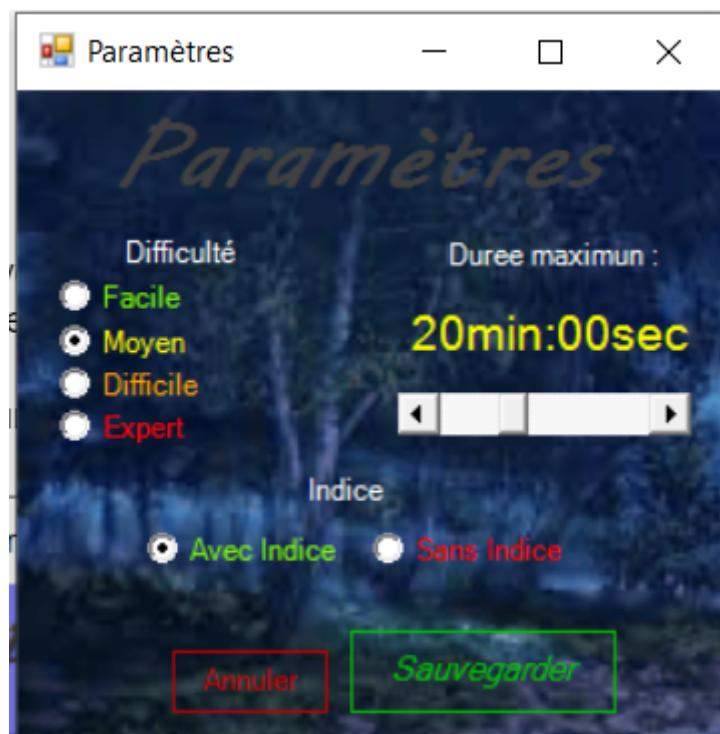
Quand vous avez déjà sélectionné plusieurs noms et que vous voulez ré utiliser un nom précédent vous pouvez regarder dans la liste en cliquant sur la flèche. Vous pouvez faire la roulette pour vous déplacer plus facilement. Et les noms sont triés par ordre alphabétique.



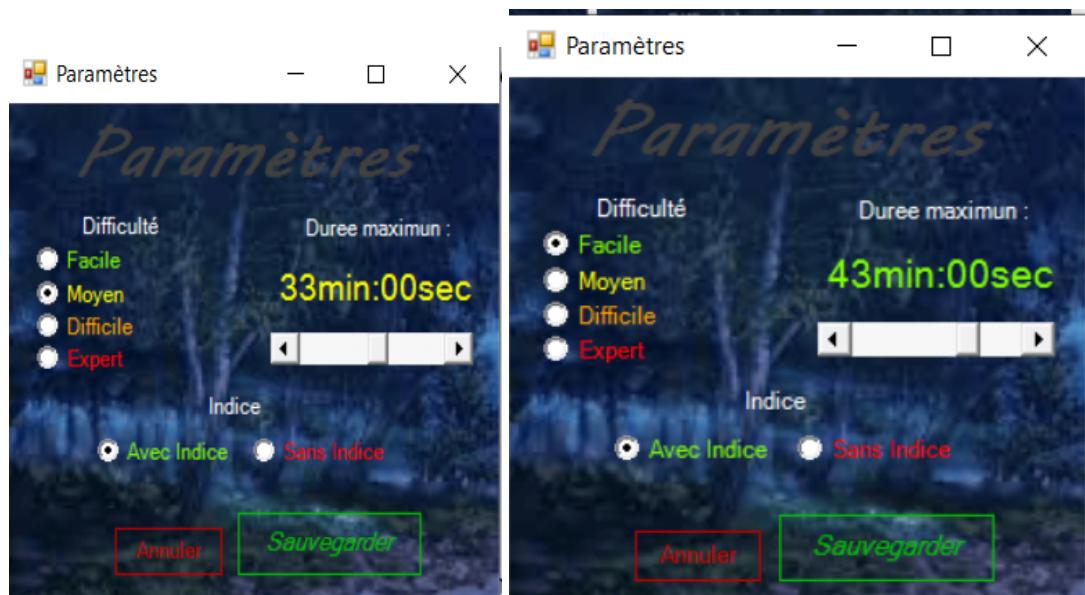
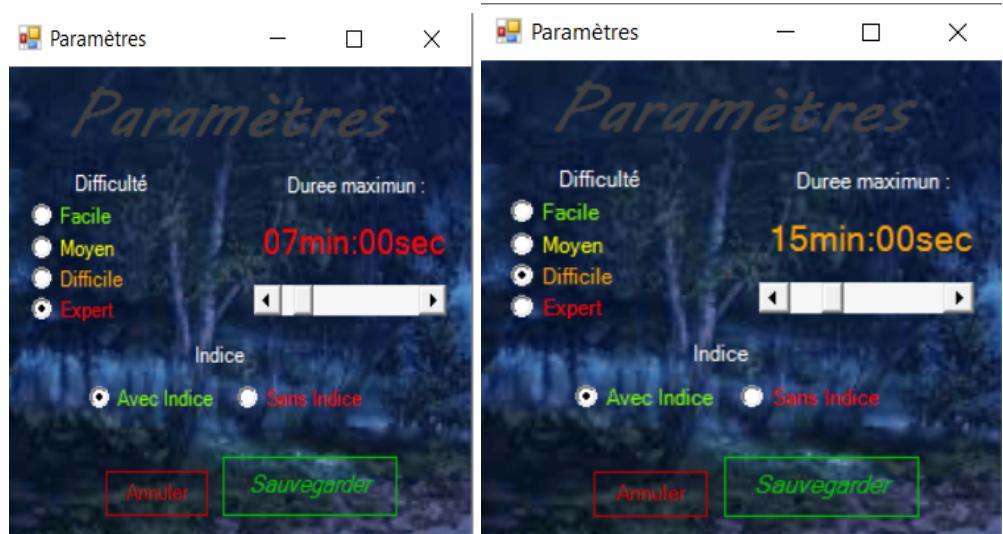
Vous pouvez quitter l'application soit avec le bouton “Quitter” ou sur la croix de la page.



Avant de lancer la partie, vous pouvez aller dans les paramètres pour modifier la difficulté passant par 4 niveaux, la durée de la partie peut être réglée entre 1 min et 59 min en cliquant sur les flèches ou en déplaçant avec la scrollbar et vous pouvez choisir si vous voulez inclure des indices ou non.



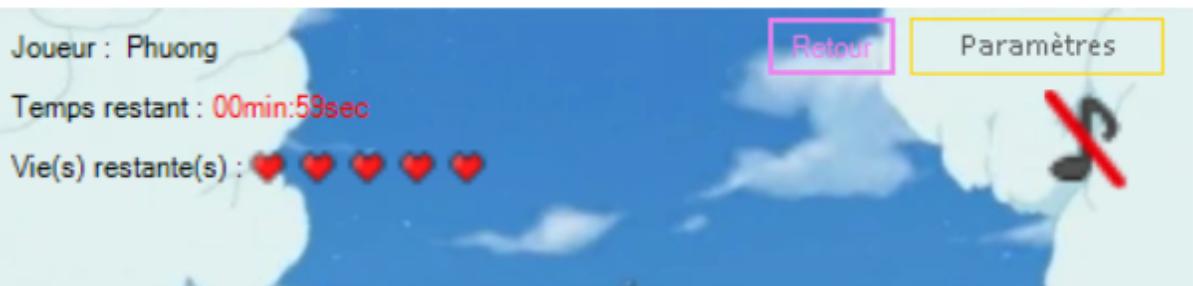
On a fait en sorte que plus le niveau est difficile correspond à une durée basse, de plus la couleur de difficulté est de la même couleur que le temps qui lui correspond.



Quand vous modifiez les paramètres et que vous appuyez sur le bouton “Annuler” cela vous envoie un message disant que vous allez partir sans sauvegarder les nouvelles modifications.



Quand le chronomètre dépasse les 1 minutes, le chronomètre commence à être rouge pour signaler qu'il n'a presque plus de temps.



Vous disposez des indices qui révèlent aux joueurs la réponse à la case demandée et vous avez le droit d'en utiliser jusqu'à 3 maximum. La lampe de départ est éteint pour signifier que vous ne l'utilisez pas et quand vous cliquez dessus pour l'utiliser il s'allume.



	4		9		3	5	8	6
	9	3		2		7	1	
6	5		8		7	2	9	3
	7		4				2	
4		2	3	8				
9	6	8	5			4	3	
1		4	2	6	9	8		7
		9			4	1	6	2
7	2	6	1	5		3	4	9

Tout à droite de la grille, il y a un clavier numérique indiquant tous les chiffres à trouver et vous réussissez à trouver tous les emplacements d'un chiffre, le chiffre dans le clavier numérique disparaît qui permet de se situer plus facilement et ne pas faire d'erreur bête.



8	6		7	2		5	3	1
3		2		1	5	6	4	
5	1		6	3	4	2		8
7				9	6	8	1	2
9	8	1		7			5	
	5	6		8	1	4		
4		5	9			1	8	3
1	3	9		4		7		5
			1		3		2	4

x3

Chiffre(s) à trouver :

2	3	
4	5	6
7	8	9

Mode

Stylo

Crayon

**Pause**

**Abandonner**

Il y 2 modes d'écriture, le stylo qui permet de saisir une réponse et le crayon pour placer des notes dans les cases (sans générer d'erreurs) et s'adapte en fonction des réponses.

Mode

Stylo

Crayon

9	256	678
268	7	
	4	2
1	578	3

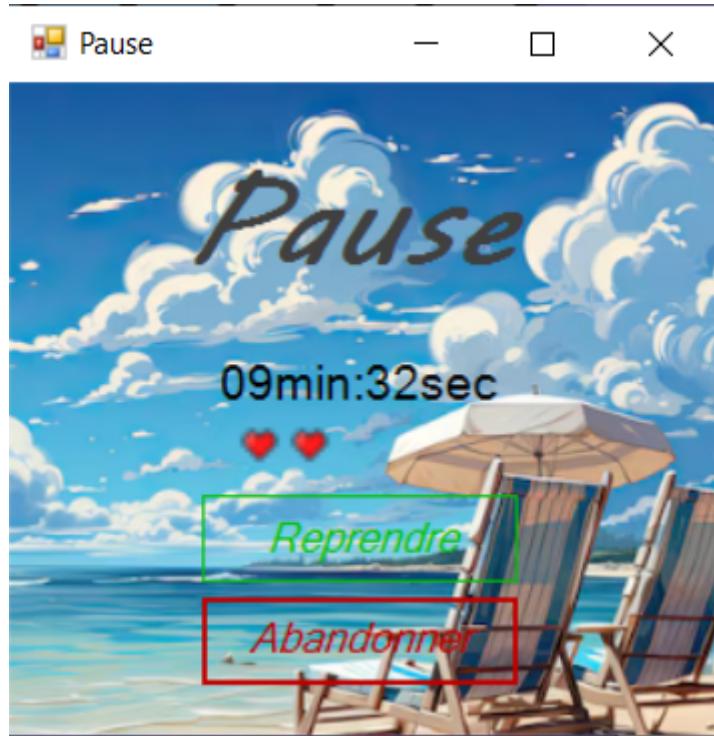
Lorsque vous faites une erreur, les cases contenant déjà ce chiffre sur la ligne, la colonne et dans le carré se colorent en rouge ou sinon ça colorie la case saisie en

rouge. Et à l'endroit de la saisie toutes les cases qui se situent dans le carré, colonne et ligne sont colorées en bleu pour faciliter la lecture.

	8			7		4	3	5
7	5		8	1		6		
	9	3	4	2	5			8
	7		2	6	8			3
	1		7	9	4	5	8	6
	4		5	3	1		2	
4	6	7				3	5	1
1		9	3	5		8	7	
5	3		1	4		2		

	8			7		4	3	5
7	5		8	1		6		
	9	3	4	2	5		1	8
	7		2	6	8			3
	1		7	9	4	5	8	6
	4		5	3	1		2	
4	6	7				3	5	1
1		9	3	5		8	7	
5	3		1	4		2		

On peut mettre en Pause la partie pour stopper la partie et la reprendre quand vous voulez. Vous pouvez voir le temps et le nombre de vie qui vous reste.



Si vous n'arrivez pas à finir le sudoku ou que vous perdez la partie, vous pouvez voir la solution du Sudoku avec la grille remplie avec les cases coloriés en jaune aux cases qui étaient vides.

Solution

- □ ×

Quitter

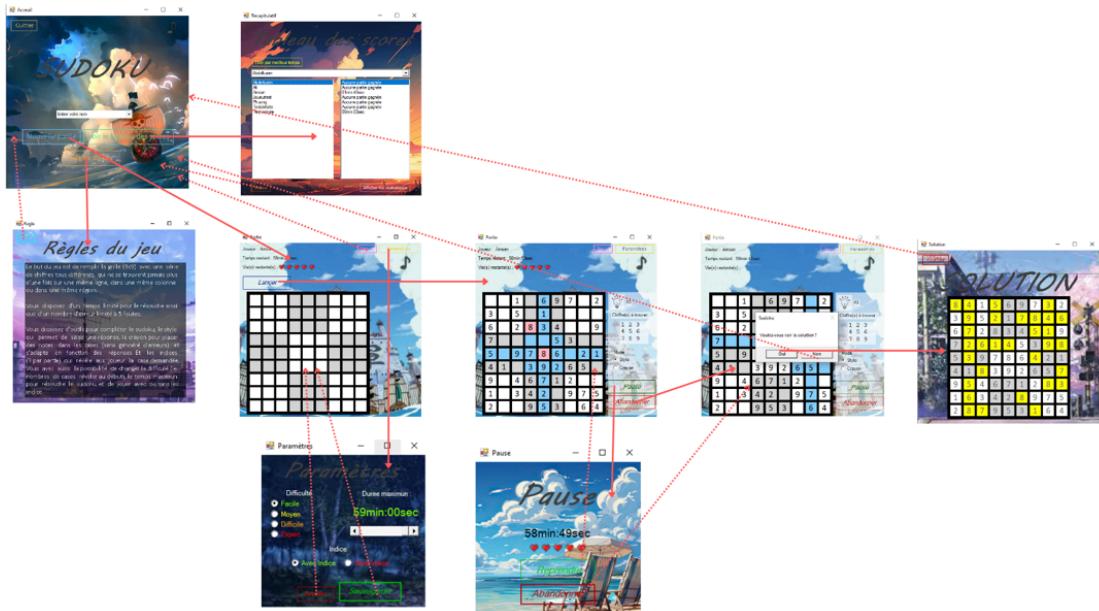
# SOLUTION

2	8	1	6	7	9	4	3	5
7	5	4	8	1	3	6	9	2
6	9	3	4	2	5	7	1	8
9	7	5	2	6	8	1	4	3
3	1	2	7	9	4	5	8	6
8	4	6	5	3	1	9	2	7
4	6	7	9	8	2	3	5	1
1	2	9	3	5	6	8	7	4
5	3	8	1	4	7	2	6	9

### **III. Modification et Correction**

On avait bien fini le projet mais après la soutenance, on a ajouté et modifié notre projet. Comme la page règles du jeu qu'on avait pas mis mais qui était important de mettre pour que les nouveaux joueurs apprennent les règles. Il y a aussi le fait de couper la musique avant ou pendant la partie car le joueur peut en avoir marre d'écouter la musique longtemps. On a bien espacé toutes les commandes dans la partie pour que ça soit plus lisible. On a rajouté des images en arrière plan pour rendre plus joli. On a rajouté un picturebox pour la lampe qui rend plus visible et mieux pour le visuel. On a rajouté des panels transparents pour une meilleure vision des outils et de l'image. Sinon on avait très bien avancer donc on a pas fait de grand changement comparé à l'ancienne version.

## IV.Schéma



## **V.Code Source**

### **ModuleSudoku :**

```
Imports System.Security.Principal
```

```
Module ModuleSudoku
```

```
'Les joueurs:
```

```
Private tabJoueur() As JOUEUR ' Tableau de joueurs  
Private nbJoueurs As Integer = 0 ' Nombre de joueurs enregistré  
Private Const PAS As Integer = 5 ' le pas, de quelle taille on augmente le  
tableau de joueurs
```

```
Public Structure JOUEUR 'Structure du joueur  
    Dim id As Integer  
    Dim nom As String  
    Dim nbPartiesJouees As Integer  
    Dim meilleurTemps As Integer ' Le plus cours temps qu'il a pris pour finir  
    une partie  
    Dim tempsDeJeuTotal As Integer ' Son temps de jeu total (pendant les  
parties)  
    Dim parametresPerso As PARAM ' Sauvegarde ses paramètres (évite  
aux joueurs de toujours rentrer ses paramètres favoris)  
End Structure  
Public Function getNbJoueurs() As Integer ' Obtient le nombre de joueurs  
enregistré  
    Return nbJoueurs  
End Function  
Public Function getJoueurs() As JOUEUR() ' Obtient le tableau des joueurs  
    Return tabJoueur.Take(nbJoueurs).ToArray()  
End Function  
Public Function getJoueur(id As Integer) As JOUEUR ' Obtient le joueur  
correspondant a l'id  
    Return tabJoueur(id)
```

```

End Function
Public Function getJoueur(nom As String) As JOUEUR ' Obtient le joueur
correspondant au nom, le creer si il n'est pas enregistree
    For Each joueurEnregistré As JOUEUR In tabJoueur
        If joueurEnregistré.nom = nom Then
            Return joueurEnregistré
        End If
    Next
    ' Ajoute le joueur et le renvoie :
    Dim newJoueur As JOUEUR = creerJoueur(nom)
    ajouterJoueur(newJoueur)
    Return newJoueur
End Function

```

```

Private Function creerJoueur(nom As String) As JOUEUR 'Return un
nouveau joueur, avec le nom
    Dim NewJoueur As New JOUEUR
    NewJoueur.nom = nom
    NewJoueur.id = nbJoueurs
    NewJoueur.meilleurTemps = Integer.MaxValue
    NewJoueur.parametresPerso = New PARAM With {.nbChiffresRevele =
nbChiffresReveleDefault, .dureeDeLaPartie = dureeDeLaPartieDefault,
.indiceActive = indiceActiveDefault}
    Return NewJoueur
End Function

```

```

Public Sub ajouterJoueur(newJoueur As JOUEUR) ' Ajoute le joueur
    If nbJoueurs >= UBound(tabJoueur) Then ReDim Preserve
tabJoueur(UBound(tabJoueur) + PAS) ' Augmente la taille du tableau
    tabJoueur(nbJoueurs) = newJoueur
    Acceuil.ComboBoxJoueur.Items.Add(tabJoueur(nbJoueurs).nom) ' On
l'ajoute dans la comboBox de l'accueil
    nbJoueurs += 1
End Sub

```

```

Public Sub tempsDeJeuTotalUpdate(id As Integer, temps As Integer) '
Ajoute au joueur (id) le temps qu'il a jouee (temps)
    tabJoueur(id).tempsDeJeuTotal += temps
    tabJoueur(id).nbPartiesJouees += 1 'augmente son nombre de partie
jouee

```

```
End Sub
```

```
Public Sub meilleurTempsUpdate(id As Integer, temps As Integer) '  
Nouveau meilleur temps? (temps) du joueur (id)  
    If temps < tabJoueur(id).meilleurTemps Then 'S'il bat son record, ou qu'il  
en a pas  
        MsgBox("Felicitation vous avez complété le sudoku en " &  
tempstoString(temps) & vbCrLf & "Vous avez établi un nouveau record !")  
        tabJoueur(id).meilleurTemps = temps  
    Else  
        MsgBox("Felicitation vous avez complété le sudoku en " &  
tempstoString(temps))  
    End If
```

```
End Sub
```

```
Private Function toString(joueur As JOUEUR) As String  
    Dim s As String = ""  
  
    s &= "Nom : " & joueur.nom & vbCrLf  
    s &= "Nombre de parties jouées : " & joueur.nbPartiesJouees & vbCrLf  
    If joueur.meilleurTemps = Integer.MaxValue Then  
        s &= "Meilleur Temps : Aucune partie gagnée" & vbCrLf  
    Else  
        s &= "Meilleur temps : " & tempstoString(joueur.meilleurTemps) &  
vbCrLf  
    End If  
  
    If joueur.nbPartiesJouees = 0 Then  
        s &= "Temps de jeu total : Aucune partie jouée" & vbCrLf  
    Else  
        s &= "Temps de jeu total : " & tempstoString(joueur.tempsDeJeuTotal)  
& vbCrLf  
    End If  
  
    Return s  
End Function
```

```
Sub afficherStats(joueur As JOUEUR) ' Affiche les statistiques du joueur
```

```
    MsgBox(toString(joueur))
End Sub
```

'Les paramètres du joueur (Pendant une partie de sudoku):

' (les paramètres par default)

```
Public Const nbChiffresReveleDefault As Integer = 50 ' Difficulté facile
```

```
Public Const dureeDeLaPartieDefault As Integer = 20 * 60 '20 minutes
```

```
Public Const indiceActiveDefault As Boolean = True ' incice active
```

```
Public Structure PARAM 'Structure des parametres
```

```
    Dim nbChiffresRevele As Integer ' Le nombre de chiffre dévoiler au lancement de la partie
```

```
    Dim dureeDeLaPartie As Integer ' La duree de la partie (temps pour résoudre le sudoku)
```

```
    Dim indiceActive As Boolean ' Si les indices sont activé
```

```
End Structure
```

```
Public Function getParametres(id As Integer) As PARAM 'Obtient les parametres perso du joueur grace a son id
```

```
    Return tabJoueur(id).parametresPerso
```

```
End Function
```

```
Public Sub setParametres(id As Integer, newParametres As PARAM)
'Modifie les parametres perso du joueur grace a son id et les nouveaux parametres donné
```

```
    tabJoueur(id).parametresPerso = newParametres
```

```
End Sub
```

'Pendant la partie de sudoku:

```
Public Function chiffrePasDejaDans(Chiffre As String, ListeDeChiffre As String) As Boolean ' Les notes au crayon, permet d'éviter de mettre deux fois le même chiffre
```

```
    Return Not ListeDeChiffre.Contains(Chiffre)
```

```
End Function
```

```
Public Function ordreCroissant(Chiffre As String, ListeDeChiffre As String) As String ' Met dans l'ordre croissant les notes au crayon
```

```
    Dim s As String = ""
```

```
    ListeDeChiffre &= Chiffre
```

```

For i = 1 To 9 ' Les chiffres de 1 à 9
    If ListeDeChiffre.Contains(i) Then
        s &= i
    End If

    If s.Length = 3 Then
        ' Insérer un retour à la ligne après le troisième chiffre
        s = s.Insert(3, vbCrLf)
    End If

    Next
    Return s
End Function

```

'L'affichage du temps:

```

Public Function tempstoString(temp As Integer) As String 'Convertie les
secondes donné en minutes, secondes, et dans le format '59min:59sec'
    Return String.Format("{0:00}min:{1:00}sec", (temp \ 60), (temp Mod
60))
End Function

```

'La music:

Private musicActif As Boolean = True ' Boolean qui indique si la music est actif ou pas :

```

Public Sub changerEtatMusic() ' Inverse l'état du son (coupe le son ou active le son)
    musicActif = Not musicActif
    playOrStopMusic() ' Fonction qui s'occupe du son
    actualiserPictureBoxSon() ' On actualise les images qui indique l'état de la music
End Sub

```

Public Sub playOrStopMusic() ' Démarrer le son ou le coupe selon le boolean MusicActif

If musicActif Then 'Si la music est Actif

```
My.Computer.Audio.Play(My.Resources.One_Piece_We_Are_Instrumental,  
AudioPlayMode.BackgroundLoop)
```

```
Else
```

```
    My.Computer.Audio.Stop()
```

```
End If
```

```
End Sub
```

```
Public Sub actualiserPictureBoxSon() ' Actualise les images
```

```
If musicActif Then 'Si la music est Actif
```

```
    Acceuil.PictureBoxMusicEnJeu.Image = My.Resources.SonActif
```

```
    Partie.PictureBoxMusicEnJeu.Image = My.Resources.SonActif
```

```
Else
```

```
    Acceuil.PictureBoxMusicEnJeu.Image = My.Resources.SonCoupé
```

```
    Partie.PictureBoxMusicEnJeu.Image = My.Resources.SonCoupé
```

```
End If
```

```
End Sub
```

```
'Permet d'avoir des panels transparents
```

```
Public Class FormLisible '
```

```
    Inherits Form
```

```
    Protected Overrides ReadOnly Property CreateParams As  
CreateParams
```

```
        Get
```

```
            Dim crp As CreateParams = MyBase.CreateParams
```

```
            crp.ExStyle = crp.ExStyle Or &H20
```

```
            Return crp
```

```
        End Get
```

```
    End Property
```

```
End Class
```

```
'Le Main:
```

```
Sub Main()
```

```
    ReDim tabJoueur(0)
```

```
    ModuleSauvegarde.chargerJoueurSauvegarder() 'On charge les joueurs  
qu'on a sauvegarder (fichier)
```

```
    Application.Run(Acceuil)
```

```
    ModuleSauvegarde.sauvegarderJoueurs() 'On sauvegarde les joueurs  
(fichier)
```

```
    Application.Exit()
```

```
End Sub  
  
End Module
```

## ModuleSauvegarde :

```
Imports System.IO  
Module ModuleSauvegarde  
    Private filePath As String = Path.Combine(Application.StartupPath,  
        "Joueurs.csv") 'La où est stocker les données
```

```
Public Sub sauvegarderJoueurs() 'Sauvegarde tous les joueurs  
    For Each j As JOUEUR In getJoueurs()  
        sauvegarderJoueur(j)  
    Next  
End Sub
```

```
Public Sub sauvegarderJoueur(j As JOUEUR) 'Sauvegarde le joueur en  
parametres, son id + 1 correspond à la ligne
```

```
    ecrireValeur(j.id + 1, 1, j.id)  
    ecrireValeur(j.id + 1, 2, j.nom)  
    ecrireValeur(j.id + 1, 3, j.nbPartiesJouees)  
    ecrireValeur(j.id + 1, 4, j.meilleurTemps)  
    ecrireValeur(j.id + 1, 5, j.tempsDeJeuTotal)  
    ecrireValeur(j.id + 1, 6, j.parametresPerso.nbChiffresRevele)  
    ecrireValeur(j.id + 1, 7, j.parametresPerso.dureeDeLaPartie)  
    ecrireValeur(j.id + 1, 8, j.parametresPerso.indiceActive)
```

```
End Sub
```

```
Private Function GetJoueurSauvegarde(id As Integer) As JOUEUR 'Obtient  
a partir de l'id la sauvegarde du joueur
```

```
    Dim JoueurSauvegarder As New JOUEUR With {  
        .id = lireValeur(id, 1),  
        .nom = lireValeur(id, 2),  
        .nbPartiesJouees = lireValeur(id, 3),  
        .meilleurTemps = lireValeur(id, 4),  
        .tempsDeJeuTotal = lireValeur(id, 5),
```

```

    .parametresPerso = New PARAM With {.nbChiffresRevele =
    lireValeur(id, 6), .dureeDeLaPartie = lireValeur(id, 7), .indiceActive =
    lireValeur(id, 8)}
}
Return JoueurSauvegarder
End Function

```

```

Public Sub chargerJoueurSauvegarder() ' Charge les joueurs sauvegarder
Dim id As Integer = 1
While (Not lireValeur(id, 1) = "")
    ajouterJoueur(GetJoueurSauvegarde(id))
    id += 1
End While
End Sub

```

```

Public Sub ecrireValeur(ligne As Integer, colonne As Integer, valeur As
String) 'ecrit dans le excel

```

```

    Dim lignes As List(Of String) = File.ReadAllLines(filePath).ToList()

    ' Si la ligne n'existe pas, ajouter des lignes vides
    While lignes.Count < ligne
        lignes.Add(New String(",",c, colonne - 1))
    End While

```

```

    Dim cellules As String() = lignes(ligne - 1).Split(",")

```

```

    ' Si la colonne n'existe pas, ajouter des colonnes vides
    While cellules.Length < colonne
        ReDim Preserve cellules(colonne - 1)
    End While

```

```

    cellules(colonne - 1) = valeur
    lignes(ligne - 1) = String.Join(",", cellules)

```

```

    File.WriteAllLines(filePath, lignes)
End Sub

```

```

' Lire une valeur d'une cellule spécifique

```

```

Public Function lireValeur(ligne As Integer, colonne As Integer) As String 'Lit
dans le excel
    Dim lignes As String() = File.ReadAllLines(filePath)

    If lignes.Length >= ligne Then
        Dim cellules As String() = lignes(ligne - 1).Split(",c")
        If cellules.Length >= colonne Then
            Return cellules(colonne - 1)
        End If
    End If

    Return String.Empty
End Function

End Module

```

## SudokuGenerator:

Module SudokuGenerator 'Ce module n'a pas été fait par nous en raison de son algorithme trop complexe.

'On a quand même modifié certaines choses (les noms des fonctions, des variables et des nombres magiques!!!).

```
Private rand As New Random()
```

```

Public Function genererGrilleSudoku(Dimensions As Integer) As Integer()
    Dim grille(Dimensions - 1, Dimensions - 1) As Integer
    remplirGrille(grille)
    Return grille
End Function

```

```

Private Function estBienPlacer(grille,) As Integer, ligne As Integer, colonne
As Integer, chiffre As Integer) As Boolean
    For posxy As Integer = 0 To grille.GetLength(0) - 1
        If grille(ligne, posxy) = chiffre Or grille(posxy, colonne) = chiffre Then
            Return False
        End If
    Next

```

```

Dim debutLigne As Integer = ligne - ligne Mod 3
Dim debutColonne As Integer = colonne - colonne Mod 3

For i As Integer = 0 To 2
    For j As Integer = 0 To 2
        If grille(i + debutLigne, j + debutColonne) = chiffre Then
            Return False
        End If
    Next
Next

Return True
End Function

```

```

Private Function remplirGrille(ByRef grille() As Integer) As Boolean
    Dim ligne As Integer = -1
    Dim colonne As Integer = -1
    Dim estVide As Boolean = True
    For x As Integer = 0 To grille.GetLength(0) - 1
        For y As Integer = 0 To grille.GetLength(1) - 1
            If grille(x, y) = 0 Then
                ligne = x
                colonne = y
                estVide = False
                Exit For
            End If
        Next
        If Not estVide Then
            Exit For
        End If
    Next

    If estVide Then
        Return True
    End If

    Dim chiffres As List(Of Integer) = Enumerable.Range(1,
grille.GetLength(0)).OrderBy(Function(n) rand.Next()).ToList()

    For Each chiffre As Integer In chiffres

```

```

If estBienPlacer(grille, ligne, colonne, chiffre) Then
    grille(ligne, colonne) = chiffre
    If remplirGrille(grille) Then
        Return True
    End If
    grille(ligne, colonne) = 0
End If
Next
Return False
End Function

End Module

```

## SudokuMasker :

Module SudokuMasker 'Ce module n'a pas été fait par nous en raison de son algorithme trop complexe.

'On a quand même modifié certaines choses (les noms des fonctions, des variables et des nombres magiques!!!).

```

Private rand As New Random()

Public Function masquerGrille(grille() As Integer, nbChiffresRevele As
Integer) As Integer()
    Dim grilleMasquee(grille.GetLength(0) - 1, grille.GetLength(1) - 1) As
Integer
    Array.Copy(grille, grilleMasquee, grille.Length)

    Dim nbChiffresMasquer As Integer = grille.Length - nbChiffresRevele

    While nbChiffresMasquer > 0
        Dim ligne As Integer = rand.Next(0, grille.GetLength(0))
        Dim colonne As Integer = rand.Next(0, grille.GetLength(1))

        If grilleMasquee(ligne, colonne) <> 0 Then
            grilleMasquee(ligne, colonne) = 0
            nbChiffresMasquer -= 1
        End If
    End While
End Function

```

```
    End While

    Return grilleMasquee
End Function

End Module
```

## Acceuil :

```
Imports System.IO.Ports
Public Class Acceuil
    Private fermetureConfirmee As Boolean = False
    Private musicActif As Boolean = True
    Private Sub Acceuil_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        actualiserPictureBoxSon()
        playOrStopMusic()
    End Sub

    Private Sub Partie_FormClosed(sender As Object, e As FormClosingEventArgs) Handles MyBase.FormClosing
        If Not fermetureConfirmee Then
            If MsgBox("Voulez-vous quitter l'application ?", vbYesNo) = vbYes
                Then
                    fermetureConfirmee = True
                    My.Computer.Audio.Stop()
                Else
                    e.Cancel = True
                End If
            End If
        End Sub

        Private Sub ButtonNouvelle_partie_Click(sender As Object, e As EventArgs) Handles ButtonNouvelle_partie.Click ' Nouvelle partie
            If ComboBoxJoueur.Text = "Entrer votre nom" OrElse
                ComboBoxJoueur.Text = "" Then 'Le joueur doit entrer un nom
                    MsgBox("Veuillez rentrer un nom")
                Else
```

```
    getJoueur(ComboBoxJoueur.Text) ' Obtient le joueur, le creer si il  
n'existe pas
```

```
        Me.Hide()  
        Partie.Show()  
    End If  
End Sub
```

```
Private Sub ButtonTableau_des_scores_Click(sender As Object, e As  
EventArgs) Handles ButtonTableau_des_scores.Click ' Tableau des scores  
    Me.Hide()  
    Recapitulatif.Show()  
End Sub
```

```
Private Sub ButtonQuitter_Click(sender As Object, e As EventArgs)  
Handles ButtonQuitter.Click ' Quitter l'application  
    Me.Close()  
End Sub
```

```
Private Sub ComboBoxJoueur_Click(sender As Object, e As EventArgs)  
Handles ComboBoxJoueur.Click ' On supprime le message " Entrer votre nom  
" , Ce Sub ne fonctionne que la premiere fois  
    ComboBoxJoueur.Text = ""  
    RemoveHandler ComboBoxJoueur.Click, AddressOf  
    ComboBoxJoueur_Click  
End Sub
```

```
Private Sub ComboBoxJoueur_KeyPress(sender As Object, e As  
KeyPressEventArgs) Handles ComboBoxJoueur.KeyPress ' Le joueur ne peut  
pas mettre de chiffre dans son nom  
    If e.KeyChar <> vbBack AndAlso Not Char.IsLetter(e.KeyChar) Then  
        e.Handled = True  
    End If  
End Sub
```

```
Private Sub ComboBoxJoueur_TextUpdate(sender As Object, e As  
EventArgs) Handles ComboBoxJoueur.TextUpdate ' Met le nom du joueur en  
ProperCase  
    sender.Text = StrConv(sender.Text, vbProperCase)  
    sender.SelectionStart = sender.Text.Length
```

```
End Sub
```

```
Private Sub ButtonRègle_Click(sender As Object, e As EventArgs) Handles  
ButtonRègle.Click 'Le form des règles  
    Me.Hide()  
    Regle.Show()  
End Sub
```

```
Private Sub PictureBoxMusicEnJeu_Click(sender As Object, e As  
EventArgs) Handles PictureBoxMusicEnJeu.Click 'Coupé ou allumer le son en  
jeu  
    changerEtatMusic()  
End Sub
```

```
End Class
```

## Parametres :

```
Public Class Parametres
```

```
    Private joueurActuelle As JOUEUR 'Joueur qui lance la partie
```

```
    Private newParametres As PARAM
```

```
    Private Sub Parametres_Load(sender As Object, e As EventArgs) Handles  
MyBase.Load
```

```
        'On attribue a nos radio button des tag qui nous permettra de mieux  
gerer le temps
```

```
        HScrollBarDureeMax.Minimum = 1
```

```
        HScrollBarDureeMax.Maximum = 59
```

```
        HScrollBarDureeMax.Maximum += HScrollBarDureeMax.LargeChange -
```

```
        1
```

```
        RadioButtonAvecIndice.Tag = True
```

```
        RadioButtonSansIndice.Tag = False
```

```
        'On prefere limité le choix du joueur pour les chiffres a révéler, donc on a  
que 4 possibilité
```

```
        RadioButtonFacile.Tag = 50
```

```
        RadioButtonMoyen.Tag = 40
```

```
        RadioButtonDifficile.Tag = 35
```

```
RadioButtonExpert.Tag = 24
```

```
joueurActuelle = getJoueur(Acceuil.ComboBoxJoueur.Text) 'On prend le joueur de la comboBox
```

```
newParametres = getParametres(joueurActuelle.id) 'On prend ses parametres
```

```
'On fait des perform click qui correspond a ses parametres
```

```
For Each difficulte As RadioButton In PanelDifficulte.Controls
```

```
If difficulte.Tag = newParametres.nbChiffresRevele Then
```

```
difficulte.PerformClick()
```

```
End If
```

```
Next
```

```
If newParametres.indiceActive Then
```

```
RadioButtonAvecIndice.PerformClick()
```

```
Else
```

```
RadioButtonSansIndice.PerformClick()
```

```
End If
```

```
'la scrollbar a la meme valeur que les parametre
```

```
HScrollBarDureeMax.Value = newParametres.dureeDeLaPartie \ 60
```

```
If newParametres.dureeDeLaPartie \ 60 = HScrollBarDureeMax.Minimum
```

```
Then 'Lorsque la valeur est égale au min (n'appelle pas le value change)
```

```
LabelDureeMax.Text =
```

```
tempstoString(newParametres.dureeDeLaPartie)
```

```
LabelDureeMax.ForeColor = Color.Red
```

```
End If
```

```
End Sub
```

```
Private Sub ButtonAnnuler_Click(sender As Object, e As EventArgs)  
Handles ButtonAnnuler.Click
```

```
Me.Close()
```

```
End Sub
```

```

Private Sub ButtonSauvegarder_Click(sender As Object, e As EventArgs)
Handles ButtonSauvegarder.Click
    setParametres(joueurActuelle.id, newParametres)
    Partie.actualiserParametres()
    Me.Close()
End Sub

Private Sub Parametres_FormClosing(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing
    If newParametres.Equals(getParametres(joueurActuelle.id)) Then 'Si les
parametre n'ont pas ete modifie
        Partie.Show()
    ElseIf MsgBox("Voulez-vous partir sans sauvegarder les nouveaux
paramètres", vbYesNo) = vbNo Then ' Si il veut partir sans sauvegarder les
changement
        e.Cancel = True 'Ne ferme pas la page
    Else
        Partie.Show()
    End If

End Sub

'Difficulté change
Private Sub RadioButtonDifficile_CheckedChanged(sender As Object, e
As EventArgs) Handles RadioButtonFacile.CheckedChanged,
RadioButtonMoyen.CheckedChanged, RadioButtonDifficile.CheckedChanged,
RadioButtonExpert.CheckedChanged
    newParametres.nbChiffresRevele = sender.Tag
End Sub

'Duree de la partie change
Private Sub HScrollBarTempsMaximun_ValueChanged(sender As Object, e
As EventArgs) Handles HScrollBarDureeMax.ValueChanged
    newParametres.dureeDeLaPartie = sender.value * 60
    LabelDureeMax.Text = tempstoString(newParametres.dureeDeLaPartie)
    'Modifie la couleur du label selon le temps
    Select Case newParametres.dureeDeLaPartie \ 60
        Case 0 To 9 'expert
            LabelDureeMax.ForeColor = Color.Red
        Case 10 To 19 'Difficile
            LabelDureeMax.ForeColor = Color.Orange
    End Select
End Sub

```

```

Case 20 To 39 'Moyen
    LabelDureeMax.ForeColor = Color.Yellow
Case 40 To 59 'Facile
    LabelDureeMax.ForeColor = Color.Chartreuse 'Vert claire
End Select

End Sub

'Indice change
Private Sub RadioButtonIndice_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButtonAvecIndice.CheckedChanged,
RadioButtonSansIndice.CheckedChanged
    newParametres.indiceActive = sender.Tag
End Sub

End Class

```

## Partie :

```

Imports System.ComponentModel
Imports System.Xml

Public Class Partie
    Private joueurActuelle As JOUEUR 'Joueur qui lance la partie
    Private textBoxSelectionner As TextBox 'Dernier textBox selectionner par le
joueur

    Private partieCommence As Boolean = False 'Indique si la partie a
commence
    Private messageAlerte As Boolean = True 'Permet de vérifier si un
message d'alerte (MsgBox qui demande une confirmation) doit etre afficher,
pour quand on ferme la page
    Private indiceActif As Boolean = False 'Variable pour indiquer si l'indice est
actif
    Private musicActif As Boolean 'Indique si le son est actif

    Private tempsRestant As Integer 'Le temps restant avant la fin de la partie
pour le joueur

```

```
Private parametresPartie As PARAM 'Les parametre de la partie
```

```
Private grille As Integer(,) ' La grille de 9x9 qui stocke le sudoku
```

```
Private grilleChiffresTrouvee As Integer(,) ' La grille de 9x9 qui stocke les chiffres saisie par le joueur
```

```
Private Const Dimensions As Integer = 9 'les dimensions du sudoku sont de 9x9
```

```
Private nbErreursRestantes As Integer 'Le nombre d'erreurs qu'il lui reste
```

```
Private Const nbErreursMax As Integer = 5 'Le nombre erreurs max qu'il peut faire
```

```
Private listeChiffresTrouvee As String 'Liste des chiffres trouvés (quand tout les il y'a 9 "1" alors tous les "1" sont placés
```

```
Private Const nbChiffres As Integer = 9
```

```
Private nbChiffresTrouves(nbChiffres - 1) As Integer ' Le 0 est exclu ( -1 ), Compte le nombre de chiffre placé
```

```
Private formatStylo As Font = New Font("Calibri", 17) 'Le stylo
```

```
Private formatCrayon As Font = New Font("Calibri", 8) 'Le crayon
```

```
Private Sub Partie_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
joueurActuelle = getJoueur(Acceuil.ComboBoxJoueur.Text)
```

```
Dim CouleurCase As Color
```

```
'Ajoute les textbox dans TableLayoutPanel
```

```
For ligne As Integer = 0 To Dimensions - 1
```

```
    For colonne As Integer = 0 To Dimensions - 1
```

```
        Dim textbox As New TextBox()
```

```
        If (ligne \ 3 + colonne \ 3) Mod 2 = 0 Then ' Permet de séparer les régions par couleur (Les régions n'ont pas la même couleur que leur voisine verticale, horizontale)
```

```
            CouleurCase = Color.White
```

```
        Else
```

```
            CouleurCase = Color.LightGray
```

```
        End If
```

```
' Defini le style :
```

```
        textbox.BackColor = CouleurCase
```

```
        textbox.Tag = CouleurCase
```

```
        textbox.Dock = DockStyle.Fill
```

```
        textbox.BorderStyle = BorderStyle.None
```

```
' Defini le format du text :
```

```

    textbox.Multiline = True
    textbox.TextAlign = HorizontalAlignment.Center
    textbox.MaxLength = 1
    textbox.Font = formatStylo
    ' Ajoute la textbox dans le formulaire :
    TableLayoutPanelQuadrillage.Controls.Add(textbox, colonne, ligne)
    ' On ajoute des evenements :
    AddHandler textbox.KeyPress, AddressOf Saisie
    AddHandler textbox.MouseClick, AddressOf Click
    Next
Next

' Paramètres du joueurs
parametresPartie = getParametres(joueurActuelle.id)
PictureBoxIndice.Tag = 3 'Le joueur possède 3 indices
LabelIndice.Text = "X" & PictureBoxIndice.Tag 'On affiche
actualiserParametres() 'L'affichage dans la partie

' Action qu'on ne peut pas faire avant le lancement de la partie:
PanelPartieEnCours.Hide()
PanelPartieEnCours.Enabled = False

' On affecte nos variables, objets
LabelNomJoueur.Text = joueurActuelle.nom
textBoxSelectionner = getTextBox(4, 4)
nbErreursRestantes = nbErreursMax
listeChiffresTrouvee = "0" ' On considère qu'on a trouvé tous les 0
LabelChronometre.Text = tempstoString(tempsRestant)
LabelChiffresPasTrouve.Font = New Font("Courier New", 12)

' On genere le Sudoku
grille = SudokuGenerator.gererGrilleSudoku(Dimensions)
ReDim grilleChiffresTrouvee(Dimensions - 1, Dimensions - 1)

' On charge les images (vies et son en jeu)
Dim Id As Integer = 5
For Each Vie As PictureBox In PanelViesRestantes.Controls
    Vie.Image = My.Resources.Vie
    Vie.Tag = Id 'On met des tag different pour chaque vie, ceux qui nous
permettra ensuite de les enlever 1 par 1 en cas d'erreur

```

```

Id -= 1
Next
PictureBoxIndice.Image = My.Resources.lampecoupé 'On initialise
l'image de l'indice
    actualiserPictureBoxSon() ' On affiche la bonne image
    actualiserChiffreATrouver() ' On actualise les chiffres a trouver
End Sub

'Fermeture du formulaire
Private Sub Partie_FormClosed(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing
    If Not partieCommence Then ' Si la partie n'a pas commencé
        Acceuil.Show() 'retour a l'accueil
    ElseIf messageAlerte Then ' Si le joueur ferme la page manuellement
(appuie sur la croix)
        If MsgBox("Si vous fermer la fenetre la partie sera terminée,
voulez-vous retourner à l'accueil ?", vbYesNo) = vbYes Then
            proposerSolution()
            e.Cancel = True ' La page ne se ferme pas
        Else
            e.Cancel = True ' La page ne se ferme pas
        End If
    Else
        Acceuil.Show()
    End If
End Sub

Private Sub proposerSolution() 'On propose au joueur d'afficher la solution
    messageAlerte = False
    If MsgBox("Voulez-vous voir la solution ?", vbYesNo) = vbYes Then
        Me.Hide()
        Solution.Show()
    Else
        Me.Close()
    End If
End Sub

Public Sub arretForceDeLaPartie() ' Lorsque que le joueur est en pleine
partie et qu'il quitte veut quitter

```

```
    Dim TempsDeJeu As Integer = parametresPartie.dureeDeLaPartie -  
    tempsRestant
```

```
        messageAlerte = False
```

```
        Chronomètre.Stop()
```

```
        tempsDeJeuTotalUpdate(joueurActuelle.id, TempsDeJeu)
```

```
        proposerSolution()
```

```
    End Sub
```

```
Private Sub ButtonRetour_Click(sender As Object, e As EventArgs)  
Handles ButtonRetour.Click
```

```
    If Not partieCommence Then ' Si la partie n'a pas commencé
```

```
        Me.Close()
```

```
    ElseIf MsgBox("Si vous retournez à l'accueil la partie sera terminée,  
voulez-vous retourner à l'accueil ?", vbYesNo) = vbYes Then
```

```
        arretForceDeLaPartie()
```

```
    End If
```

```
End Sub
```

```
Private Sub ButtonAbandonner_Click(sender As Object, e As EventArgs)  
Handles ButtonAbandonner.Click
```

```
    If MsgBox("Êtes-vous sûr de vouloir abandonner ?", vbYesNo) = vbYes  
Then
```

```
        arretForceDeLaPartie()
```

```
    End If
```

```
End Sub
```

```
Private Sub PictureBoxMusicEnJeu_Click(sender As Object, e As  
EventArgs) Handles PictureBoxMusicEnJeu.Click 'Coupé ou allumer le son en  
jeu
```

```
    changerEtatMusic()
```

```
End Sub
```

```
Private Function getTextBox(ligne As Integer, colonne As Integer) As  
TextBox ' Obtient la TextBox a la position donné
```

```
    Return
```

```
    TryCast(TableLayoutPanelQuadrillage.GetControlFromPosition(colonne,  
ligne), TextBox)
```

```
End Function
```

```
Private Function getPosition(texbox As TextBox) As  
TableLayoutPanelCellPosition ' Obtient la position de la TextBox dans le  
sudoku  
    Return  
    TableLayoutPanelQuadrillage.GetPositionFromControl(textBoxSelectionner)  
End Function  
  
Public Function getNbErreursRestantes() As Integer 'getter du nombre  
erreurs restant du joueur (Pour le forms Pause)  
    Return nbErreursRestantes  
End Function
```

```
Public Function getTempsRestant() As Integer 'getter du temps qu'il lui  
reste pour finir (Pour le forms Pause)  
    Return tempsRestant  
End Function
```

```
Public Function getGrilleChiffresTrouvee() As Integer()  
    Return grilleChiffresTrouvee  
End Function
```

```
Public Function getGrille() As Integer()  
    Return grille  
End Function
```

```
'Parametres de la Partie  
Private Sub ButtonParamètres_Click(sender As Object, e As EventArgs)  
Handles ButtonParamètres.Click 'Accede au form parametres  
    Me.Hide()  
    Parametres.Show()  
End Sub  
  
Public Sub actualiserParametres() ' On actualise les labels, selon les  
parametres du joueur  
    parametresPartie = getParametres(joueurActuelle.id)  
    tempsRestant = parametresPartie.dureeDeLaPartie  
    LabelChronometre.Text = tempstoString(tempsRestant)  
End Sub
```

'Pendant le jeu : (Les fonction principaux au début)

```

Private Sub ButtonLancer_Click(sender As Object, e As EventArgs)
Handles ButtonLancer.Click
    ' Affiche au joueur le sudoku
    afficherSudoku(parametresPartie.nbChiffresRevele)
    ' Affiche le timer :
    LabelChronometre.Text = tempstoString(tempsRestant)
    Chronometre.Start()
    ' Option qu'il ne peut plus faire
    ButtonParamètres.Enabled = False
    ButtonLancer.Enabled = False
    ButtonLancer.Hide()
    partieCommence = True 'La partie a commence
    ' Option qu'il peut désormais faire
    TableLayoutPanelQuadrillage.Enabled = True 'Le sudoku est visible
    PanelPartieEnCours.Show()
    PanelPartieEnCours.Enabled = True
    Dim nouvelleOpacite As Double = 0.6
    PanelPartieEnCours.BackColor = Color.White
    PanelPartieEnCours.BackColor = Color.FromArgb(CInt(nouvelleOpacite * 255), PanelPartieEnCours.BackColor.R, PanelPartieEnCours.BackColor.G, PanelPartieEnCours.BackColor.B)

    If Not parametresPartie.indiceActive Then
        PictureBoxIndice.Hide()
        PictureBoxIndice.Enabled = False
    End If

    ' Simule un click sur la textbox
    Click(textBoxSelectionner, EventArgs.Empty)
    'Le joueur a le stylo en main
    RadioButtonStylo.PerformClick()
End Sub

Private Sub ButtonPause_Click(sender As Object, e As EventArgs)
Handles ButtonPause.Click ' Met en pause le jeu
    Me.Hide()
    Pause.Show()
    Chronometre.Stop()
End Sub

```

```

Private Sub Click(sender As Object, e As EventArgs) ' Met en couleur la
ligne, la colonne, et la région où le joueur a cliquer
    decolorier()
    textBoxSelectionner = DirectCast(sender, TextBox)
    Dim position = getPosition(textBoxSelectionner)
    Dim DebutLigneRegion As Integer = (position.Row \ 3) * 3
    Dim DebutColonneRegion As Integer = (position.Column \ 3) * 3

    If textBoxSelectionner.Text = "" Then ' Si la texbox est vide alors on
modifie son format (permet au curseur de s'adapter au mode)
        If RadioButtonStylo.Checked Then
            textBoxSelectionner.Font = formatStylo
        Else
            textBoxSelectionner.Font = formatCrayon
        End If
    End If

'Ligne et colonne
For posxy As Integer = 0 To Dimensions - 1
    getTextBox(posxy, position.Column).BackColor = Color.LightSkyBlue
    getTextBox(position.Row, posxy).BackColor = Color.LightSkyBlue

Next
'Region
For ligne As Integer = DebutLigneRegion To DebutLigneRegion + 2
    For colonne As Integer = DebutColonneRegion To
        DebutColonneRegion + 2
            getTextBox(ligne, colonne).BackColor = Color.LightSkyBlue
    Next
Next

If grilleChiffresTrouvee(position.Row, position.Column) <> 0 Then 'Si il
contient un chiffre valide, évite de colorier les notes
    colorier(textBoxSelectionner.Text)
End If

If indiceActif AndAlso grilleChiffresTrouvee(position.Row,
position.Column) = 0 Then 'Revele la case si l'indice est activé et que la case
ne contient pas déjà un chiffre valide
    Dim reponse As Integer = grille(position.Row, position.Column)

```

```

grilleChiffresTrouvee(position.Row, position.Column) = reponse
textBoxSelectionner.Text = reponse.ToString()
textBoxSelectionner.Font = formatStylo
textBoxSelectionner.ReadOnly = True ' La TextBox ne peut plus être
modifiée
nbChiffresTrouves(reponse - 1) += 1 ' Augmente le compteur
actualiserChiffreATrouver()
retirerNoteCrayon(reponse.ToString())
colorier(reponse)
textBoxSelectionner.BackColor = Color.Yellow
PictureBoxIndice.Image = My.Resources.lampecoupé
PictureBoxIndice.Tag -= 1
LabelIndice.Text = "X" & PictureBoxIndice.Tag

```

```

If sudokuRempli() Then ' Si le joueur a complété le Sudoku
    victoire()
End If
indiceActif = False ' Désactive l'indice après utilisation
End If
End Sub
Private Sub Saisie(sender As Object, e As KeyPressEventArgs) 'Lorsque le
joueur saisie quelque chose dans une casse
    Click(sender, EventArgs.Empty) 'Simule un click ( lorsque le joueur reste
    sur la textbox après une erreur)
    Dim textBox As TextBox = DirectCast(sender, TextBox)
    Dim position = getPosition(textBox)
    Dim chiffreSaisie As String = e.KeyChar.ToString()

    If chiffreSaisie <> vbBack AndAlso (Not Char.IsDigit(chiffreSaisie) OrElse
    listeChiffresTrouvee.Contains(chiffreSaisie)) Then ' Autres que les chiffres (0
    exclu ainsi que les chiffres trouvés, 7 est trouvé quand y'en a 9 sur le sudoku)
        e.Handled = True
        Exit Sub
    End If
    If e.KeyChar = vbBack AndAlso grilleChiffresTrouvee(position.Row,
    position.Column) = 0 Then 'Peut supprimer seulement si il y'a des notes au
    crayon (grilleChiffresTrouvee = 0, veut dire qu'il n'a pas trouverContient un
    chiffre bien placé)
        Exit Sub

```

```

End If
' Mode Stylo
If RadioButtonStylo.Checked AndAlso
grilleChiffresTrouvee(position.Row, position.Column) = 0 Then 'Si le joueur
n'as pas trouve de chiffre
    If Not chiffreValide(e.KeyChar) Then ' Si le joueur se trompe
        e.Handled = True
        perdUneVie()
        Exit Sub
    End If
    ' Tout est correct, Permet d'écrire par-dessus les notes au crayon
    textBox.Text = chiffreSaisie
    textBox.Font = formatStylo
    textBox.MaxLength = 1
    textBox.SelectionStart = chiffreSaisie.Length
    textBox.ReadOnly = True ' La textBox ne peut plus être modifié
    grilleChiffresTrouvee(position.Row, position.Column) =
    grille(position.Row, position.Column)
    colorier(chiffreSaisie)
    If sudokuRempli() Then 'Vérifie si le joueur a remplie le sudoku
        victoire()
    End If
    e.Handled = True ' La saisie a déjà été ajouté dans la fonction (Ajoute
2 fois sinon)
    Exit Sub
End If
'Mode Crayon
If grilleChiffresTrouvee(position.Row, position.Column) = 0 AndAlso
chiffrePasDejaDans(chiffreSaisie, textBox.Text) Then
    textBox.MaxLength = 7
    textBox.Font = formatCrayon
    textBox.Text = ordreCroissant(chiffreSaisie, textBox.Text)
    textBox.SelectionStart = textBox.Text.Length
    e.Handled = True ' La saisie a été ajouté dans la fonction (Ajoute 2 fois
sinon)
Else
    e.Handled = True
End If
End Sub

```

```

Private Sub Chronomètre_Tick(sender As Object, e As EventArgs) Handles
Chronomètre.Tick ' Le Timer de la partie
    If tempsRestant > 0 Then
        If tempsRestant = 60 Then 'Quand il reste 1 minute le temps est
affiché en rouge!!!!
            LabelChronometre.ForeColor = Color.Red
        End If
        tempsRestant -= 1
        LabelChronometre.Text = tempstoString(tempsRestant) ' Affiche le
temps restant
    Else
        Chronomètre.Stop()
        defaite("Le temps est écoulé!") 'Quand le joueur perd !!! :(
    End If
End Sub

```

```

Private Sub afficherSudoku(nbChiffreRevele As Integer) 'Dévoile le sudoku
    ' Masquer certaines cellules pour créer une grille partiellement remplie
    Dim grilleMasquee As Integer(,) = SudokuMasker.masquerGrille(grille,
nbChiffreRevele)
    ' Afficher la grille de Sudoku dans les TextBox appropriés
    For ligne As Integer = 0 To Dimensions - 1
        For colonne As Integer = 0 To Dimensions - 1
            Dim textBox As TextBox = getTextBox(ligne, colonne)
            Dim Chiffre As String = grilleMasquee(ligne, colonne)
            If Chiffre <> 0 Then
                textBox.Text = grilleMasquee(ligne, colonne).ToString()
                nbChiffresTrouves(Chiffre - 1) += 1 ' Augmente le compteur
                actualiserChiffreATrouver()
                textBox.ReadOnly = True 'Le joueur ne peut plus écrire mais peut
toujours cliquer
            End If
        Next
    Next
    grilleChiffresTrouvee = grilleMasquee
End Sub
Private Function chiffreValide(chiffre As String) As Boolean 'Le chiffre est à
la bonne position = True , sinon colorie les erreurs

```

```
Dim position = getPosition(textBoxSelectionner) 'On obtient la position de la textbox
```

```
If grille(position.Row, position.Column) <> chiffre Then 'On vérifie avec la grille si le chiffre est bien placé
    colorierErreur(chiffre)
    Return False
End If
nbChiffresTrouves(chiffre - 1) += 1 ' Augmente le compteur
actualiserChiffreATrouver()
retirerNoteCrayon(chiffre) 'On retire les notes qui ne sert à rien
Return True
End Function
```

```
Private Sub colorierErreur(chiffre As String) ' Met en évidence les erreurs du joueur, celle qui sont visibles
```

```
Dim position = getPosition(textBoxSelectionner)
' Position de la région de la Textbox
Dim debutLigneRegion As Integer = (position.Row \ 3) * 3
Dim debutColonneRegion As Integer = (position.Column \ 3) * 3
Dim nbTextBoxColorie As Integer = 0
For posxy As Integer = 0 To Dimensions - 1
    Dim textboxMemeLigne As TextBox = getTextBox(posxy,
position.Column)
    Dim textboxMemeColonne As TextBox = getTextBox(position.Row,
posxy)
```

```
' Si la grille des chiffres trouvé est différente de 0, (Ne prend pas en compte les notes au crayon)
```

```
If grilleChiffresTrouvee(posxy, position.Column) <> 0 AndAlso
    textboxMemeLigne.Text = chiffre Then 'Si il y'a un chiffre dans la même ligne
        textboxMemeLigne.BackColor = Color.OrangeRed
        nbTextBoxColorie += 1
    End If
    If grilleChiffresTrouvee(position.Row, posxy) <> 0 AndAlso
        textboxMemeColonne.Text = chiffre Then 'Si il y'a un chiffre dans la même colonne
            textboxMemeColonne.BackColor = Color.OrangeRed
            nbTextBoxColorie += 1
        End If
```

```

    Next
    For ligne As Integer = debutLigneRegion To debutLigneRegion + 2
        For colonne As Integer = debutColonneRegion To
debutColonneRegion + 2
            Dim textboxMemeRegion As TextBox = getTextBox(ligne, colonne)

                If grilleChiffresTrouvee(ligne, colonne) <> 0 AndAlso
textboxMemeRegion.Text = chiffre Then 'Si il y'a un chiffre dans la meme
region
                    textboxMemeRegion.BackColor = Color.OrangeRed
                    nbTextBoxColorie += 1
                End If
            Next
        Next
        If nbTextBoxColorie = 0 Then ' Si l'erreur n'est pas visible on colorie la
textbox où il se trouve
            textBoxSelectionner.BackColor = Color.OrangeRed
        End If
    End Sub

```

```

Private Sub colorier(chiffre As String) 'Colorie à Rose dans le sudoku les
chiffres qui sont égaux au chiffre en paramètre
    For ligne As Integer = 0 To Dimensions - 1
        For colonne As Integer = 0 To Dimensions - 1
            Dim textbox As TextBox = getTextBox(ligne, colonne)
            'de ne pas prendre en compte les notes au crayon
            If grilleChiffresTrouvee(ligne, colonne) <> 0 AndAlso textbox.Text =
chiffre Then
                textbox.BackColor = Color.LightPink
            End If
        Next
    Next
End Sub

Private Sub decolorier() 'Remet les couleurs par defaut (Inverse de colorier)
    For ligne As Integer = 0 To Dimensions - 1
        For colonne As Integer = 0 To Dimensions - 1
            Dim textbox As TextBox = getTextBox(ligne, colonne)
            textbox.BackColor = textbox.Tag
        Next
    Next

```

```
    Next  
End Sub
```

```
Private Sub retirerNoteCrayon(Chiffre As String) ' Retire les notes au  
crayon qui sont égaux au chiffre saisie dans la même ligne,colonne et région
```

```
    Dim position = getPosition(textBoxSelectionner)  
    Dim DebutLigneRegion As Integer = (position.Row \ 3) * 3  
    Dim DebutColonneRegion As Integer = (position.Column \ 3) * 3
```

```
    For posxy As Integer = 0 To Dimensions - 1
```

```
        Dim textboxMemeLigne As TextBox = getTextBox(posxy,  
position.Column)
```

```
        Dim textboxMemeColonne As TextBox = getTextBox(position.Row,  
posxy)
```

```
        ' On regarde dans la grille si le joueur a trouvé un chiffre (prend que  
les notes au crayon)
```

```
        If grilleChiffresTrouvee(posxy, position.Column) = 0 AndAlso  
textboxMemeLigne.Text.Contains(Chiffre) Then 'Si il y'a un chiffre dans la  
même ligne
```

```
            textboxMemeLigne.Text = textboxMemeLigne.Text.Replace(Chiffre,  
"") 'On l'enlève
```

```
        End If
```

```
        If grilleChiffresTrouvee(position.Row, posxy) = 0 AndAlso  
textboxMemeColonne.Text.Contains(Chiffre) Then 'Si il y'a un chiffre dans la  
même colonne
```

```
            textboxMemeColonne.Text =  
textboxMemeColonne.Text.Replace(Chiffre, "")
```

```
        End If
```

```
    Next
```

```
    For ligne As Integer = DebutLigneRegion To DebutLigneRegion + 2
```

```
        For colonne As Integer = DebutColonneRegion To  
DebutColonneRegion + 2
```

```
            Dim textboxMemeRegion As TextBox = getTextBox(ligne, colonne)
```

```
            If grilleChiffresTrouvee(ligne, colonne) = 0 AndAlso  
textboxMemeRegion.Text.Contains(Chiffre) Then 'Si il y'a un chiffre dans la  
même région
```

```

    textboxMemeRegion.Text =
textboxMemeRegion.Text.Replace(Chiffre, "")
End If
Next
Next

End Sub

```

Private Sub actualiserChiffreATrouver() 'Actualise le Label qui permet de voir quel chiffre n'a pas été placés 9 fois, permet au joueur de voir quel chiffre il lui reste placé

```

LabelChiffresPasTrouvé.Text = ""
For i = 1 To 9
If nbChiffresTrouves(i - 1) = 9 Then
    listeChiffresTrouvee &= i ' Ajoute si il y'en a 9
End If

If Not listeChiffresTrouvee.Contains(i) Then
    LabelChiffresPasTrouvé.Text &= i & Space(1)
Else
    LabelChiffresPasTrouvé.Text &= Space(2)

End If
If i Mod 3 = 0 Then 'On saute a la ligne tous les 3 chiffre
    LabelChiffresPasTrouvé.Text &= vbCrLf
End If
Next
End Sub

```

Private Sub victoire() ' Lorsque le joueur gagne la partie

```

Chronomètre.Stop()
Dim tempsDeJeu As Integer = parametresPartie.dureeDeLaPartie -
tempsRestant
Dim tempsDeJeuToString As String =
tempstoString(tempsDeJeu.ToString)
messageAlerte = False
tempsDeJeuTotalUpdate(joueurActuelle.id, tempsDeJeu)
meilleurTempsUpdate(joueurActuelle.id, tempsDeJeu)
Me.Close()

```

```

        Acceuil.Show()
End Sub

Private Sub defaite(MessageDeDefaite As String) ' Lorsque le joueur perd
la partie
    Chronometre.Stop()
    messageAlerte = False
    Dim tempsDeJeu As Integer = parametresPartie.dureeDeLaPartie -
tempsRestant
    MsgBox(MessageDeDefaite)
    tempsDeJeuTotalUpdate(joueurActuelle.id, tempsDeJeu)
    If MsgBox("Voulez-vous voir la solution ?", vbYesNo) = vbYes Then
        Me.Hide()
        Solution.Show()
    Else
        Me.Close()
    End If
End Sub

Private Sub perdUneVie() ' Enleve un vie au joueur
    For Each vie As PictureBox In PanelViesRestantes.Controls
        If vie.Tag = nbErreursRestantes Then
            vie.Hide() 'Enleve une vie de l'ecran
        End If
    Next
    nbErreursRestantes -= 1 ' Enlever une vie

    If nbErreursRestantes = 0 Then ' Si le joueur n'a plus de vie
        defaite("Vous n'avez plus de vies")
    End If
End Sub

Private Function sudokuRempli() As Boolean ' Fonction qui verifie si toutes
les textBox sont remplie
    For ligne As Integer = 0 To Dimensions - 1
        For colonne As Integer = 0 To Dimensions - 1
            If grilleChiffresTrouvee(ligne, colonne) = 0 Then
                Return False
            End If
        Next
    Next

```

```
    Return True  
End Function
```

```
Private Sub PictureBoxIndice_Click(sender As Object, e As EventArgs)  
Handles PictureBoxIndice.Click  
    indiceActif = Not indiceActif 'Inverse la valeur du boolean, permet au  
joueur d'annuler l'indice  
    If indiceActif AndAlso sender.Tag > 0 Then  
        PictureBoxIndice.Image = My.Resources.lampe  
    ElseIf sender.Tag > 0 Then  
        PictureBoxIndice.Image = My.Resources.lampecoupé  
    Else  
        MsgBox("Vous n'avez plus d'indice")  
        indiceActif = False  
    End If  
End Sub
```

```
End Class
```

Pause :

```
Public Class Pause  
    Private Sub Pause_Load(sender As Object, e As EventArgs) Handles  
 MyBase.Load  
        ' On charge les images (vies et son en jeu)  
        Dim Id As Integer = 5  
        For Each Vie As PictureBox In PanelViesRestantes.Controls  
            Vie.Image = My.Resources.Vie  
            Vie.Tag = Id 'On met des tag different pour chaque vie, ceux qui nous  
permettra ensuite de les enlever 1 par 1 en cas d'erreur  
            Id -= 1  
            If Vie.Tag >= Partie.getNbErreursRestantes() + 1 Then Vie.Hide()  
        Next  
        LabelChronometre.Text = tempstoString(Partie.getTempsRestant)  
    End Sub
```

```

Private Sub ButtonAbandonner_Click(sender As Object, e As EventArgs)
Handles ButtonAbandonner.Click 'Le joueur peut abandonner depuis le form
pause
    If MsgBox("Êtes-vous sûr de vouloir abandonner ?", vbYesNo) = vbYes
Then
    ArretForceDeLaPartie()
End If
End Sub
Private Sub ArretForceDeLaPartie() ' Lorsque que le joueur est en pleine
partie et qu'il quitte veut quitter
    Me.Close()
    Partie.arretForceDeLaPartie()
End Sub
Private Sub ButtonReprendre_Click(sender As Object, e As EventArgs)
Handles ButtonReprendre.Click
    Me.Close()
End Sub
Private Sub Pause_FormClosed(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing ' reprendre le jeu
    Partie.Show()
    Partie.Cronomètre.Start() 'On redémarre le chronomètre
End Sub

End Class

```

## Recapitulatif :

```

Imports System.Windows.Forms.VisualStyles.VisualStyleElement

Public Class Recapitulatif
    Private JoueurSelectioner As JOUEUR
    Private Sub Recapitulatif_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        If getNbJoueurs() > 0 Then ' On rempli la comboBox, et les deux liste
box
            For i = 0 To (getNbJoueurs() - 1)
                ComboBoxNomJoueur.Items.Add(getJoueur(i).nom)
            Next
        End If
    End Sub

```

```

    ListBoxMeilleursTemps.Items.Add(getJoueur(i).meilleurTemps)
    ListBoxNoms_des_joueurs.Items.Add(getJoueur(i).nom)
    Next
    trierOrdreAlphabetique() 'On trie par nom au debut
    ComboBoxNomJoueur.Text = ComboBoxNomJoueur.GetItemText(0)
    JoueurSelectioner = getJoueur(ComboBoxNomJoueur.Text)
End If
End Sub

```

```

Private Sub Recapitulatif_FormClosed(sender As Object, e As EventArgs)
Handles MyBase.FormClosed
    Acceuil.Show()
End Sub

```

```

Private Sub ButtonRetour_Click(sender As Object, e As EventArgs)
Handles ButtonRetour.Click
    Me.Close()
    Acceuil.Show()
End Sub

```

```

Private Sub trierOrdreAlphabetique()
    Dim joueursTries() As JOUEUR = getJoueurs.OrderBy(Function(j)
j.nom).ToArray() 'stocke dans une nouvelle liste, la liste trier par nom
    mettreAJourListes(joueursTries)
End Sub

```

```

Private Sub trierParMeilleurTemps()
    Dim joueursTries() As JOUEUR = getJoueurs.OrderBy(Function(j)
j.meilleurTemps).ToArray() 'stocke dans une nouvelle liste, la liste trier par
meilleur temps
    mettreAJourListes(joueursTries)
End Sub

```

```

Private Sub mettreAJourListes(joueurs As JOUEUR())
    'On vide tout
    ComboBoxNomJoueur.Items.Clear()
    ListBoxMeilleursTemps.Items.Clear()
    ListBoxNoms_des_joueurs.Items.Clear()

```

```
For Each joueur As JOUEUR In joueurs 'Et en rerempli dans l'ordre
    ComboBoxNomJoueur.Items.Add(joueur.nom)
    ListBoxMeilleursTemps.Items.Add>If(joueur.meilleurTemps =
Integer.MaxValue, "Aucune partie gagnée",
tempToString(joueur.meilleurTemps)))
    ListBoxNoms_des_joueurs.Items.Add(joueur.nom)
Next
```

```
If ComboBoxNomJoueur.Items.Count > 0 Then 'On selectionne le
premier de la liste
    ComboBoxNomJoueur.SelectedIndex = 0
    JoueurSelectioner = getJoueur(ComboBoxNomJoueur.Text)
End If
End Sub
```

```
Private Sub ButtonTrierParNom_Click(sender As Object, e As EventArgs)
Handles ButtonTrierParNom.Click
    trierOrdreAlphabetique()
    'On affiche le button trier par meilleur temps, et on cache celui la
(Les button sont au meme endroit)
    sender.Visible = False
    sender.Enabled = False
    ButtonTrierParMeilleurTemps.Visible = True
    ButtonTrierParMeilleurTemps.Enabled = True
End Sub
```

```
Private Sub ButtonTrierParMeilleurTemps_Click(sender As Object, e As
EventArgs) Handles ButtonTrierParMeilleurTemps.Click
    trierParMeilleurTemps()
    sender.Visible = False
    sender.Enabled = False
    ButtonTrierParNom.Visible = True
    ButtonTrierParNom.Enabled = True
End Sub
```

```
Private Sub ButtonAfficher_statistiques_Click(sender As Object, e As
EventArgs) Handles ButtonAfficher_statistiques.Click
    If getNbJoueurs() > 0 Then
        afficherStats(JoueurSelectioner)
    Else
```

```

    MsgBox("Aucun joueur n'est inscrit")
End If
End Sub

'La combobox et les deux liste box sont lier
Private Sub Autre_Joueur_Selection(sender As Object, e As EventArgs)
Handles ComboBoxNomJoueur.SelectedIndexChanged,
ListBoxNoms_des_joueurs.SelectedIndexChanged,
ListBoxMeilleursTemps.SelectedIndexChanged
    ComboBoxNomJoueur.SelectedIndex = sender.SelectedIndex
    ListBoxMeilleursTemps.SelectedIndex = sender.SelectedIndex
    ListBoxNoms_des_joueurs.SelectedIndex = sender.SelectedIndex
    JoueurSelectioner = getJoueur(ComboBoxNomJoueur.Text)
End Sub

End Class

```

## Regle :

```

Public Class Regle

    Private Sub ButtonRetour_Click(sender As Object, e As EventArgs)
Handles ButtonRetour.Click
    Me.Close()
End Sub

    Private Sub Regle_FormClosing(sender As Object, e As FormClosingEventArgs) Handles MyBase.FormClosing
        Acceuil.Show()
        Me.Hide()
        e.Cancel = True
    End Sub

    Private Sub Regle_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim nouvelleOpacite As Double = 0.6
        LabelRegleText.BackColor = Color.Black
    End Sub

```

```
    LabelRegleText.BackColor = Color.FromArgb(CInt(nouvelleOpacite *  
255), LabelRegleText.BackColor.R, LabelRegleText.BackColor.G,  
LabelRegleText.BackColor.B)
```

```
End Sub
```

```
End Class
```

## Solution :

```
Public Class Solution  
    Private Const Dimensions As Integer = 9 'les dimensions du sudoku sont de  
9x9  
    Private formatStylo As Font = New Font("Calibri", 17) 'Le stylo  
  
    Private Sub Solution_Load(sender As Object, e As EventArgs) Handles  
 MyBase.Load  
        Dim CouleurCase As Color  
  
        For ligne As Integer = 0 To Dimensions - 1  
            For colonne As Integer = 0 To Dimensions - 1  
                Dim textbox As New TextBox()  
                If (ligne \ 3 + colonne \ 3) Mod 2 = 0 Then ' Permet de séparer les  
régions par couleur (Les régions n'ont pas la même couleur que leur voisine  
verticale, horizontale)  
                    CouleurCase = Color.White  
                Else  
                    CouleurCase = Color.LightGray  
                End If  
                ' Defini le style :  
                textbox.BackColor = CouleurCase  
                textbox.Dock = DockStyle.Fill  
                textbox.BorderStyle = BorderStyle.None  
                textbox.Multiline = True  
                textbox.TextAlign = HorizontalAlignment.Center  
                textbox.MaxLength = 1  
                textbox.Font = formatStylo  
                ' Ajoute la textbox dans le formulaire :  
                TableLayoutPanelQuadrillage.Controls.Add(textbox, colonne, ligne)
```

```

    Next
    Next

    afficherSolution()

End Sub

Private Function getTextBox(ligne As Integer, colonne As Integer) As
TextBox ' Obtient la TextBox a la position donné
    Return
TryCast(TableLayoutPanelQuadrillage.GetControlFromPosition(colonne,
ligne), TextBox)
End Function

Private Sub afficherSolution()
    Dim grille As Integer(,) = Partie.getGrille()
    Dim grilleChiffresTrouvee As Integer(,) = Partie.getGrilleChiffresTrouvee()
    For ligne As Integer = 0 To Dimensions - 1
        For colonne As Integer = 0 To Dimensions - 1
            Dim textbox = getTextBox(ligne, colonne)
            textbox.Text = grille(ligne, colonne)
            If grilleChiffresTrouvee(ligne, colonne) = 0 Then
                textbox.BackColor = Color.Yellow
            End If
        Next
    Next
End Sub

```

```

Private Sub ButtonQuitter_Click(sender As Object, e As EventArgs)
Handles ButtonQuitter.Click

```

```
    Me.Close()
```

```
End Sub
```

```
'Fermeture du formulaire
```

```
Private Sub Solution_FormClosed(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing
```

```
    Partie.Close()
```

```
    Acceuil.Show()
```

```
End Sub
```

End Class

## Pause :

```
Public Class Pause
    Private Sub Pause_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' On charge les images (vies et son en jeu)
        Dim Id As Integer = 5
        For Each Vie As PictureBox In PanelViesRestantes.Controls
            Vie.Image = My.Resources.Vie
            Vie.Tag = Id 'On met des tag different pour chaque vie, ceux qui nous
permettra ensuite de les enlever 1 par 1 en cas d'erreur
            Id -= 1
            If Vie.Tag >= Partie.getNbErreursRestantes() + 1 Then Vie.Hide()
        Next
        LabelChronometre.Text = tempToString(Partie.getTempsRestant)
    End Sub
    Private Sub ButtonAbandonner_Click(sender As Object, e As EventArgs)
Handles ButtonAbandonner.Click 'Le joueur peut abandonner depuis le form
pause
        If MsgBox("Êtes-vous sûr de vouloir abandonner ?", vbYesNo) = vbYes
Then
            ArretForceDeLaPartie()
        End If
    End Sub
    Private Sub ArretForceDeLaPartie() ' Lorsque que le joueur est en pleine
partie et qu'il quitte veut quitter
        Me.Close()
        Partie.arretForceDeLaPartie()
    End Sub
    Private Sub ButtonReprendre_Click(sender As Object, e As EventArgs)
Handles ButtonReprendre.Click
        Me.Close()
    End Sub
    Private Sub Pause_FormClosed(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing ' reprendre le jeu
        Partie.Show()
    End Sub
```

```
    Partie.Cronomètre.Start() 'On redemarre le chronometre  
End Sub
```

```
End Class
```