Projet de Programmation

Jeu du 2048

Groupe - David Navarre
WANG Yiyang
GUO Yanrui

Sommaire

Sommaire ..... - 1 -

I. Introduction ..... - 2 -

1) Objectif : ..... - 2 -

2) Exigences opérationnelles : ..... - 2 -

II. Conception du jeu ..... - 2 -

1) Algorithme de jeu: ..... - 2 -

2) Les consignes du coté module: ..... - 3 -

III. L'interface ..... - 4 -

1) Version classique ..... - 5 -

2) Version DRAGON BALL ..... - 9 -

IV. Le code de la Form ..... - 13 -

1) Menu ..... - 13 -

2) Version classique ..... - 14 -

3) Version DRAGON BALL ..... - 21 -

V. Noyau fonctionnel ..... - 29 -

VI. Conclusion ..... - 37 -

Introduction

1) Objectif :

L'objectif de ce projet est de développer une application Visual Basic qui implémente la version classique du jeu du 2048.

2) Exigences opérationnelles :

1- Les déplacements se font par des clics sur les boutons Left, Right, Up et Down (ou les touches du clavier Q, D, Z, S, ←, →, ↑, ↓).

2- L'annulation du dernier coup par le bouton Undo.

3- Le fait de relancer une partie par le bouton Restart.

Environnement de développement: Microsoft Visual Studio 2010
Langue d'écriture: VB.NET

I. Conception du jeu

1) Algorithme de jeu:

1. Au début du jeu, deux carrés d'une valeur de 2 ou 4 sont générés aléatoirement. Ces deux carrés apparaissent à la position aléatoire de la table, et les carrés restants étant à 0.
2. Chaque fois que vous déplacez la souris ou appuyez sur la touche, la valeur de grille calculée est calculée ligne par ligne. L'algorithme pour déplacer chaque ligne est d'abord déplacer tous les nombres autres que 0 au début de la ligne. Après avoir pu comparer un par un avec le précédent depuis le début de la ligne, les deux grilles sont fusionnées si elles sont égales.
3. Chaque fois qu'une grille est fusionnée, sa valeur est accumulée dans le score total du jeu. Lorsque ce score total est supérieur au score le plus élevé historique, le score le plus élevé sera mis à jour.
4. Après un déplacement, une valeur de 2 ou 4 est assignée au hasard à tous les carrés avec une valeur de 0.
5. Si tous les carrés ne sont pas 0 et que les cases adjacentes ne sont pas égales, le jeu se termine.

6. Lorsque le nombre dans le table est 2048, le jeu gagne.
7. La probabilité de 2 apparaît est de 90%, et la probabilité de 4 apparaît est de 10%.

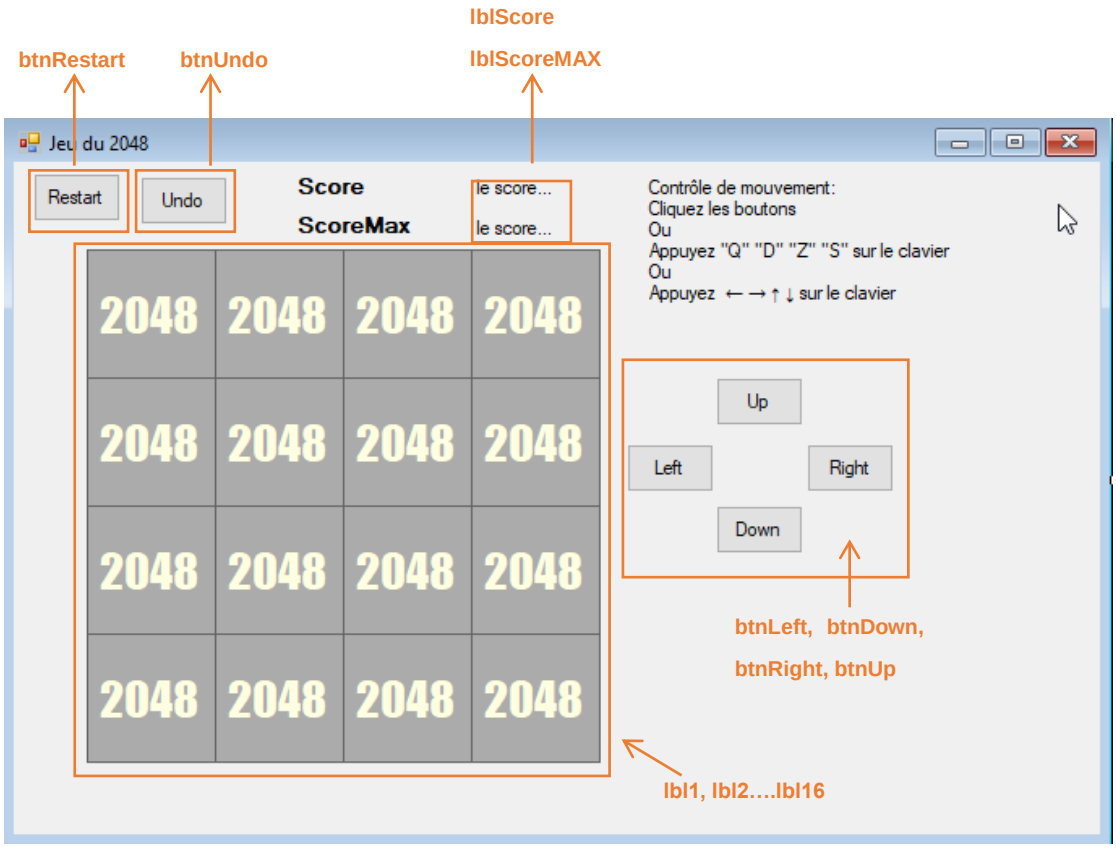
2) Les consignes du coté module:

- Le jeu est représenté par la variable Jeu qui est une variable tableau de 4 lignes et 4 colonnes contenant des valeurs de type short (ou plus grande si vous êtes excellent joueur).
- Le score est représenté par la variable Score de type short. Et Le score maximal est représenté par la variable ScoreMax de type short.
- Les variables ScoreAnnuler, ScoreMaxAnnuler, scoretem, scoreMaxTem sont les paramètres qui gardent temporairement les scores pour réaliser l'annulation du dernier coup et vérifier les conditions pour terminer le jeu.
- Les valeurs et les scores de jeu sont enregistrés temporairement par la variable tabletem qui est une variable tableau de 4 lignes et 4 colonnes contenant des valeurs de type short.
- Lorsque l'implémentation du jeu revient à l'étape précédente, la valeur du jeu à afficher est stockée dans la variable JeuAnnuler qui est une variable tableau de 4 lignes et 4 colonnes contenant des valeurs de type short.
- La variable PointGagne de type Integer qui est égale à 2048 signifie la condition gagnante.

- La procédure **Initialiser** initialise le jeu en positionnant dans deux cases du jeu une tuile aléatoire de valeur 2 ou 4 ; les autres cases du jeu sont vides.
- La procédure **TirerAleatoirement** permet de tirer aléatoirement une valeur (un 2 ou un 4) et de la positionner dans une case vide du jeu. Dans le bloc généré, une méthode aléatoire est appelée, et tout nombre compris entre 1 et 100 est généré aléatoirement: le nombre est inférieur à 90, le carré indique le nombre 2 et, dans le cas contraire, il est égal à 4.
- La fonction **tem** enregistre la table de jeu et les scores avant de déménager.
- La procédure **SubComparer** vérifie qu'après chaque mouvement la valeur avant et après de jeu a changé.
- Avant avoir demandé un retour au jeu, la fonction **TableAnnuler** est utilisée pour obtenir toutes les valeurs qui devraient apparaître dans l'interface du jeu.
- La fonction **Annuler** est utilisée pour charger la valeur de l'étape précédente, appelez cette méthode lors d'une opération d'annulation
- La fonction **JeuGagner** est utilisée pour déterminer s'il y a une valeur est égale à 2048, c'est-à-dire si le jeu gagne.
- La fonction **JeuTermine** indique si le jeu est terminé.
- La fonction **peutcontinue** vérifie que chaque valeur de la table est non nulle
- La fonction **peutligne** indique si le jeu peut se déplacer dans les deux directions : gauche et droite.

- La fonction **peutcolonne** indique si le jeu peut se déplacer dans les deux directions : haut et bas.
- La procédure **Deplacer** qui permet de déplacer les tuiles du jeu en accord avec le choix de l'utilisateur (gauche, droite, haut, bas).

II. L'interface



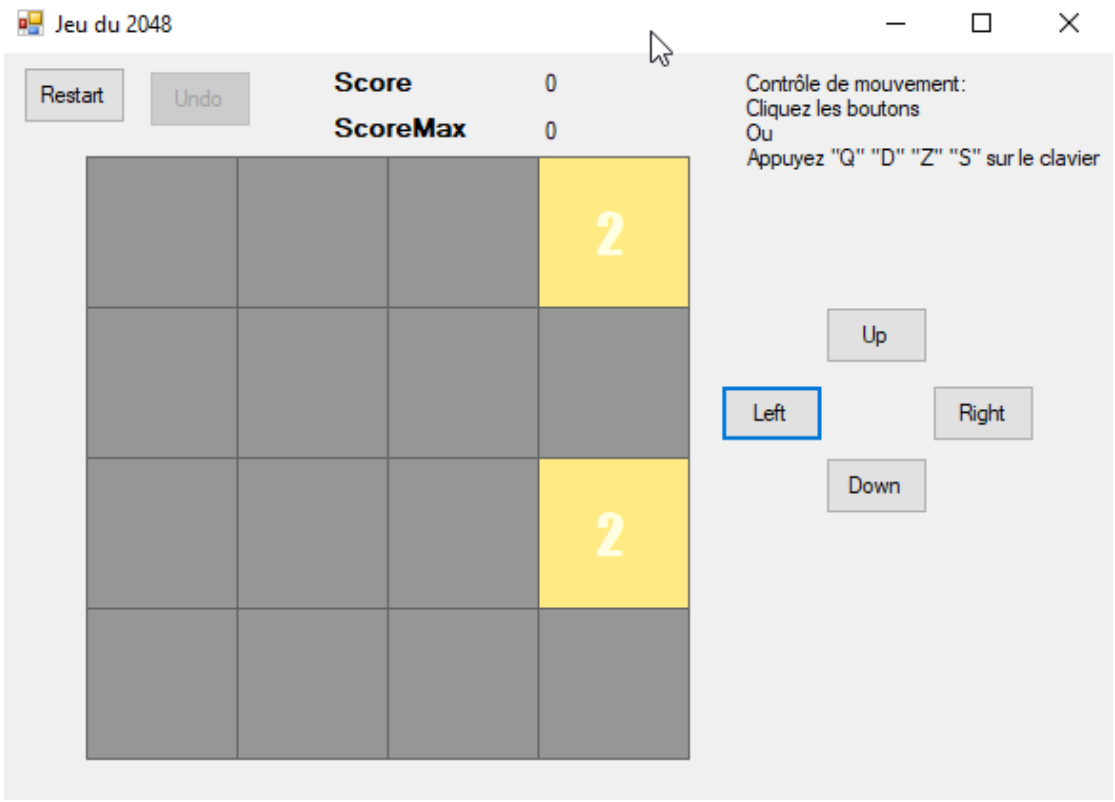
Le bouton "Restart" est cliqué pour initialiser le jeu.
Le bouton "Undo" est cliqué pour annuler un dernier coup.
Les labels Score et ScoreMax affichent le score en temps réel et le score maximal historique.
Les 4 boutons de direction (btnLeft, btnDown, btnRight, btnUp) réalisent les déplacements de jeu.
Le jeu est représenté sous les labels lbl1, …, lbl16, chacun correspondant à une case du jeu.

Interface menu :
Un msgBox est présente pour demander le choix de version du jeu quand l'application est démarrée. La seule différence entre les deux versions est la beauté de l'interface. On ajoute le lecture de sons en arrière-plan et un bouton pour arrêter et un bouton pour

rejouer.

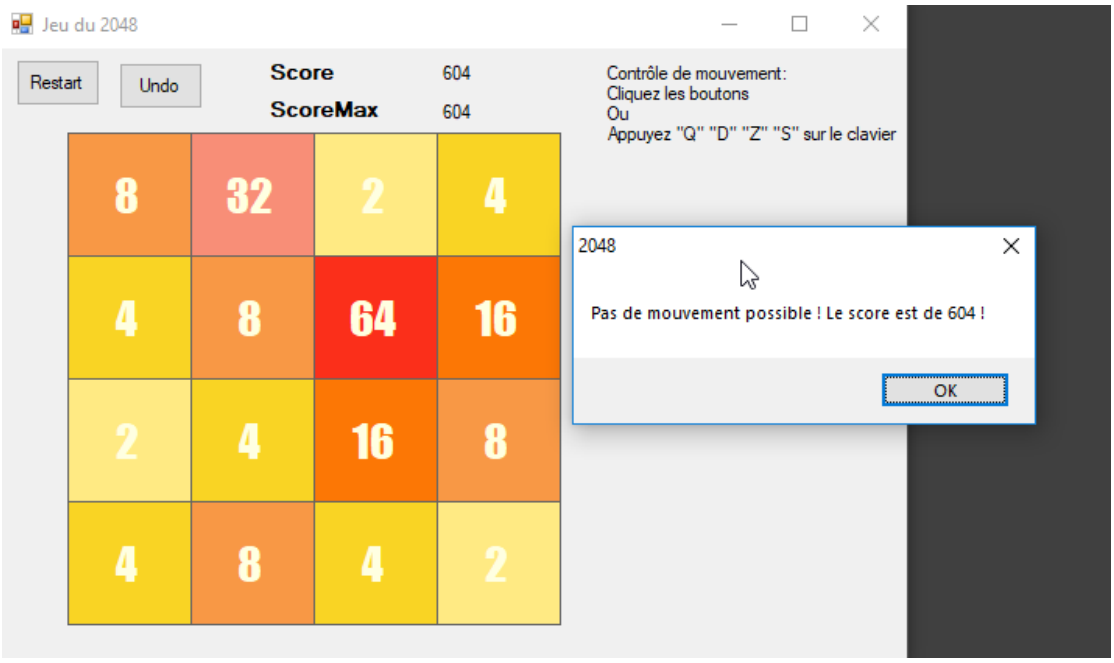
1) Version classique

Interface initiale :



Interface en cours de jeu :

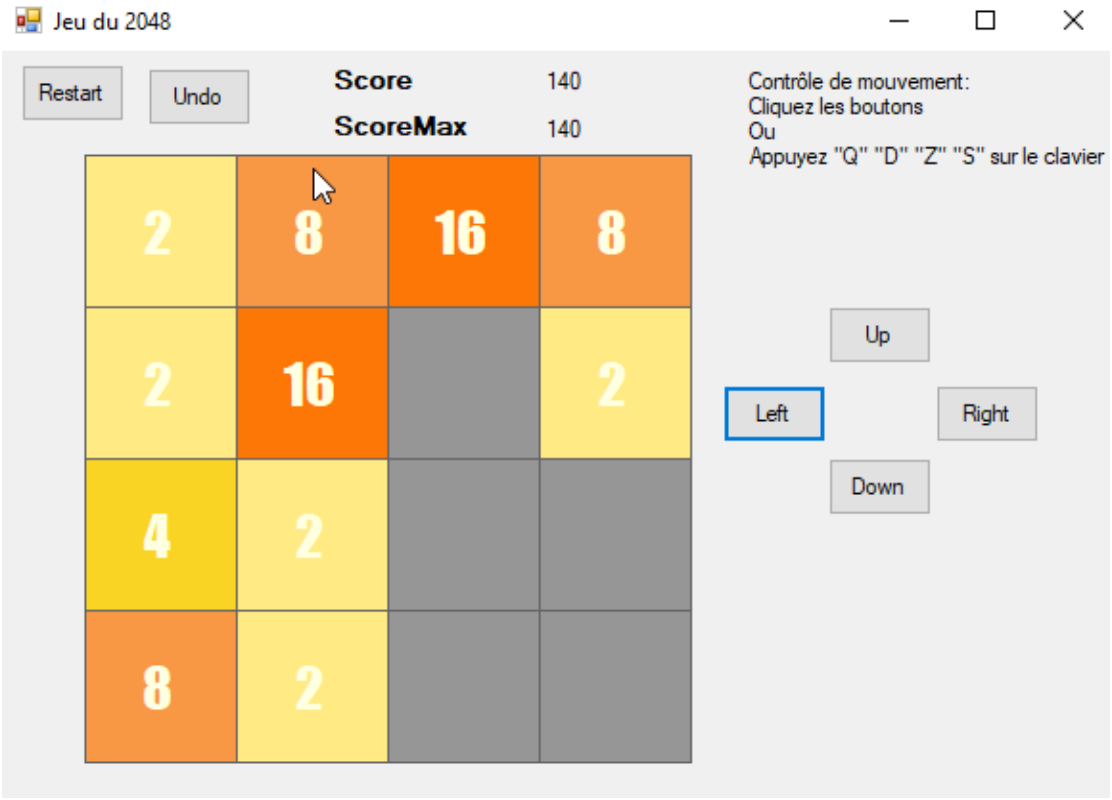
Jeu termine :



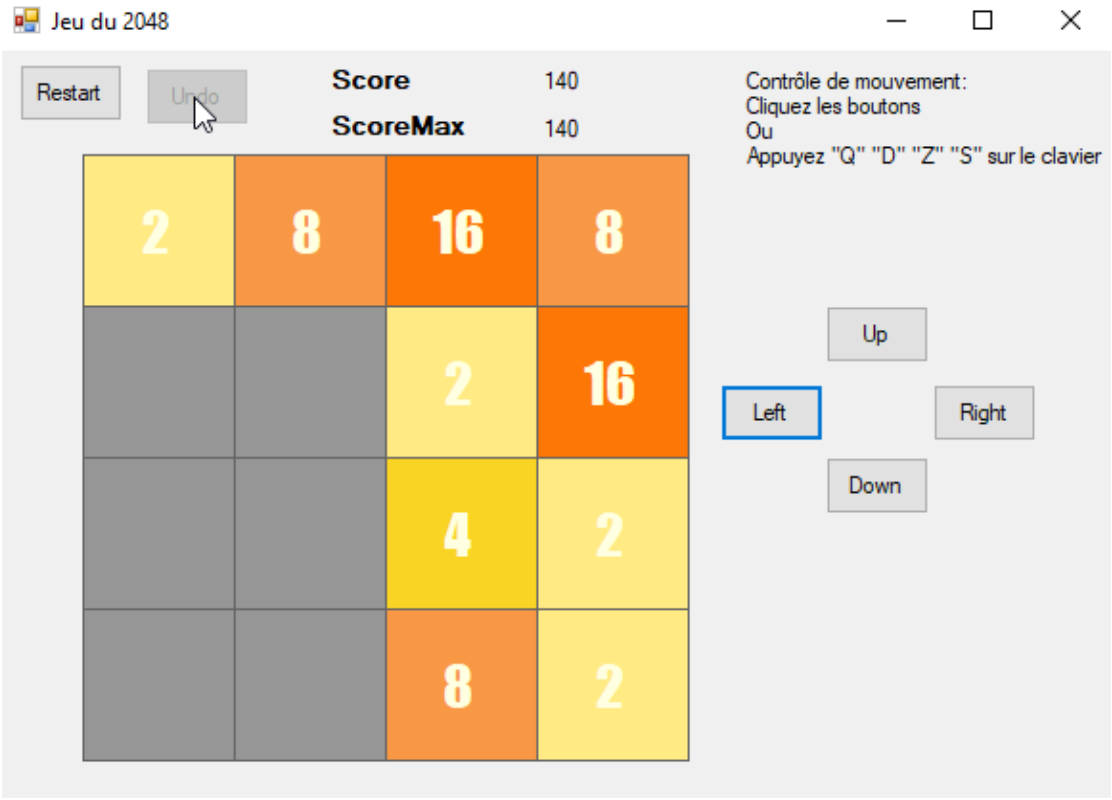
UNDO

Le bouton Undo est disponible après chaque déplacement. Cliquez ce bouton afin de afficher l'interface du dernier coup et alors bouton est indisponible.

Avant cliquez Undo :

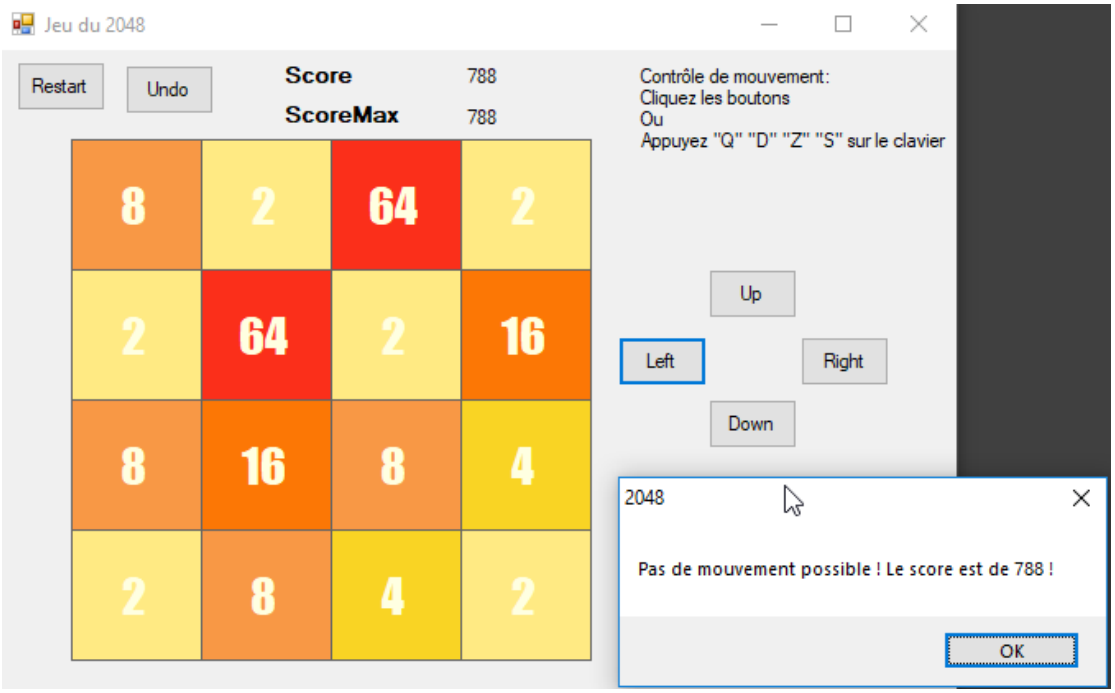


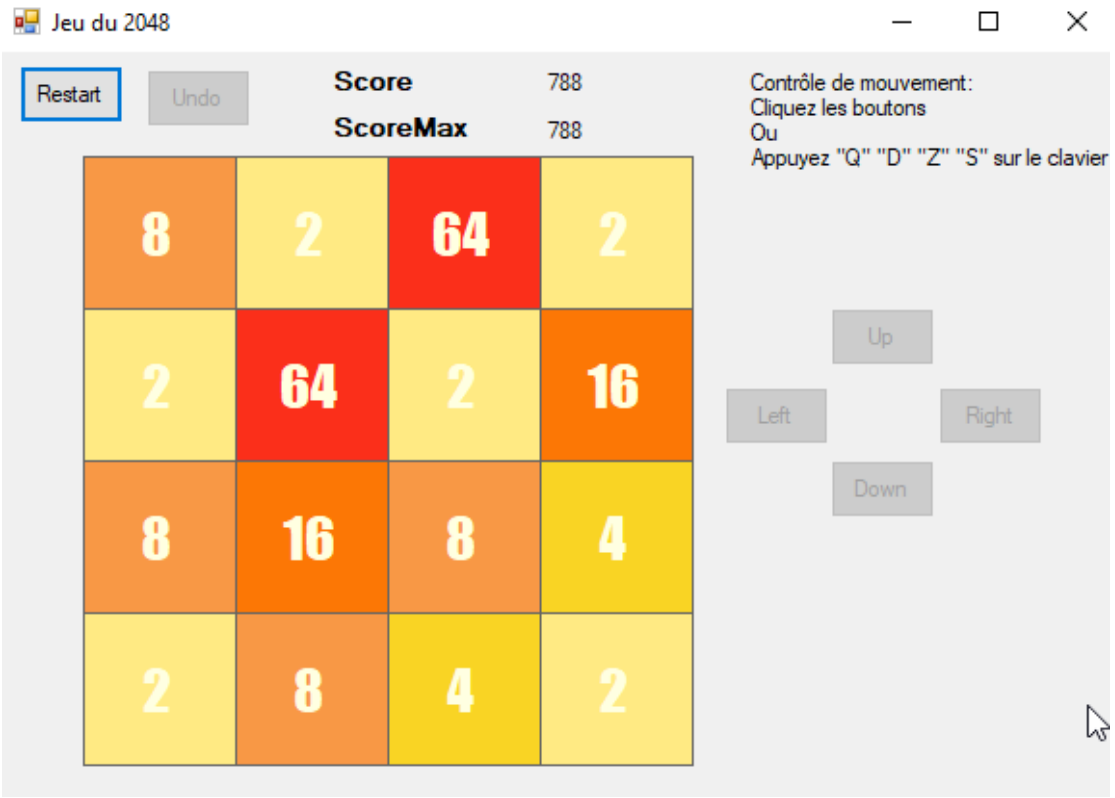
Après cliquez Undo :

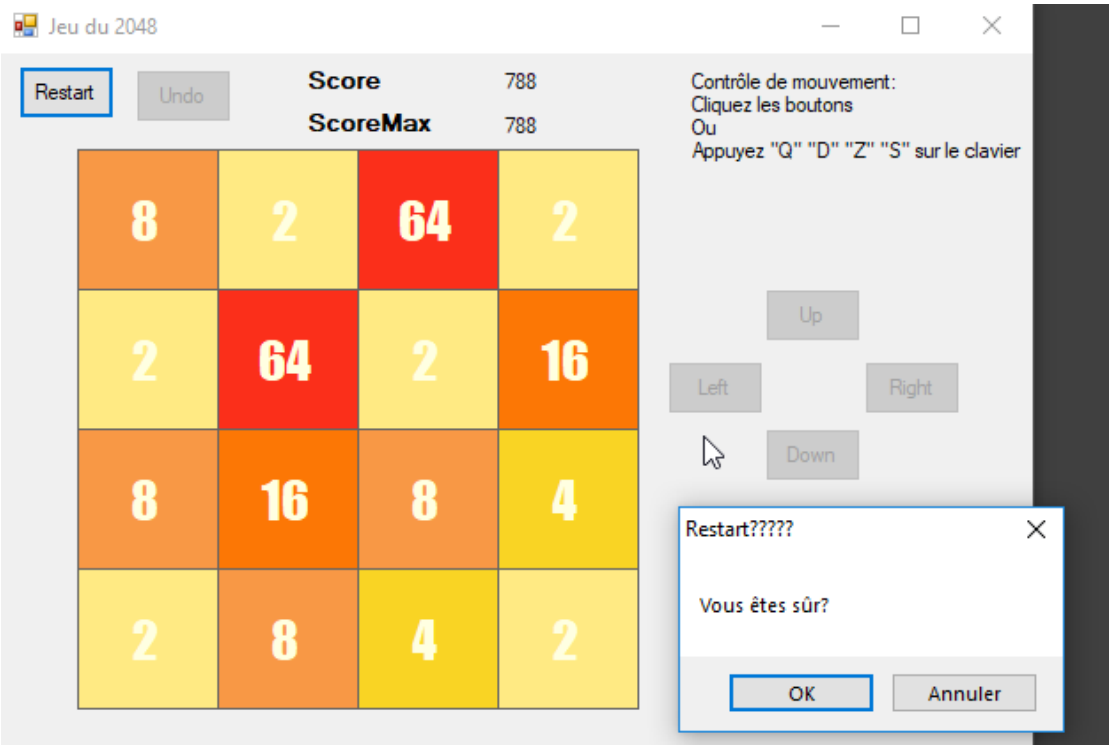


Un cas particulier est que l'interface du jeu (les valeurs et les scores) ne change pas même si on clique plusieurs fois les boutons de direction. Dans ce cas-là, le bouton Undo est cliqué pour afficher l'interface du jeu qui formait avant l'interface constante.

RESTART



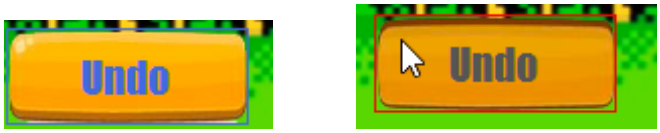




Quand le jeu est terminé, un msgBox est présent pour afficher le score de ce jeu. Seulement le bouton Restart est disponible après cliquer OK. Pour éviter les opérations inattendues, un msgBox pour confirmer la demande de rejouer 2048. Si ok, le jeu est initialisé, sinon, l'interface reste à l'interface du jeu termine.

2) Version DRAGON BALL

On utilise les images pour afficher l'interface plus jolie.



L'image à gauche est le bouton Undo disponible, et celle à droit est le bouton Undo indisponible.

Interface initiale :

Interface en cours de jeu :

Jeu termine :


Jeu du 2048
DRAGON BALL
2048
2048
Pas de mouvement possible ! Le score est de 672 !
OK
Score 672
Best 1420
Restart
Undo

III. Le code de la Form

1) Menu

```
1.  Public Class main
2.
3.      Private Sub playmusic()
4.          'joue en boucle

        My.Computer.Audio.Play("..\..\images\dragonBall.wav",
AudioPlayMode.BackgroundLoop)

5.      End Sub
6.  _____________________________________________________________
7.      Private Sub stopmusic()
8.          'Arrêt de la lecture de sons
9.          My.Computer.Audio.Stop()
10.     End Sub
11. _____________________________________________________________
12.     Private Sub Button1_Click(sender As System.Object, e As
    System.EventArgs) Handles Button1.Click
13.         'Cliquer ce boutons pour choisir la version classique
14.         Form2048_1.Show()
15.     End Sub
16. _____________________________________________________________
17.     Private Sub Button2_Click(sender As System.Object, e As
    System.EventArgs) Handles Button2.Click
18.         'Cliquer ce boutons pour choisir la version classique Dragon Ball
19.         Form2048.Show()
20.     End Sub
21. _____________________________________________________________
22.     Private Sub Button3_Click(sender As System.Object, e As
    System.EventArgs) Handles Button3.Click
23.         'joue en boucle par cliquer le bouton
24.         Call playmusic()
25.     End Sub
26. _____________________________________________________________
27.     Private Sub Button4_Click(sender As System.Object, e As
    System.EventArgs) Handles Button4.Click
28.         Call stopmusic()
29.     End Sub
30. _____________________________________________________________
```

```
31.     Private Sub main_Load(sender As System.Object, e As System.EventArgs)
    Handles MyBase.Load
32.         'joue en boucle en arrière-plan
33.         Call playmusic()
34.     End Sub
35. End Class
```

2) Version classique

```
1.  Public Class Form2048_1
2.
3.     Public lbl(16) As Label
4.  ____________________________________________________________________
5.     Public Sub controlelabel()
6.  ' Connecter les labels dans l'interface et les labels dans Form2048
7.         lbl(1) = lbl1
8.         lbl(2) = lbl2
9.         lbl(3) = lbl3
10.        lbl(4) = lbl4
11.        lbl(5) = lbl5
12.        lbl(6) = lbl6
13.        lbl(7) = lbl7
14.        lbl(8) = lbl8
15.        lbl(9) = lbl9
16.        lbl(10) = lbl10
17.        lbl(11) = lbl11
18.        lbl(12) = lbl12
19.        lbl(13) = lbl13
20.        lbl(14) = lbl14
21.        lbl(15) = lbl15
22.        lbl(16) = lbl16
23.
24.    End Sub
25. ____________________________________________________________________
26. Protected Overrides Function processdialogkey(keydata As
    System.Windows.Forms.Keys) As Boolean
27.        'détecter si les touches de direction sont pressées
28.        If keydata = Keys.Up Or keydata = Keys.Down Or keydata = Keys.Left
    Or keydata = Keys.Right Then
29.            Return False
30.        Else
```

```
31.             Return MyBase.ProcessDialogKey(keydata)
32.         End If
33.     End Function
34. ____________________________________________________________________
35.
36.     Private Sub Form2048_KeyDown(sender As Object, e As KeyEventArgs)
    Handles Me.KeyDown
37.         ' Connecter des boutons de clavier avec les boutons sur interface
38.         ' touches ←, →, ↑, ↓
39.         Select Case e.KeyCode
40.             Case Keys.Down
41.                 btnDown_Click(sender, e)
42.             Case Keys.Up
43.                 btnUp_Click(sender, e)
44.             Case Keys.Left
45.                 btnLeft_Click(sender, e)
46.             Case Keys.Right
47.                 btnRight_Click(sender, e)
48.         End Select
49.
50.         'touches Q D Z S
51.         Select Case Chr(e.KeyValue)
52.             Case "S"
53.                 btnDown_Click(sender, e)
54.             Case "Z"
55.                 btnUp_Click(sender, e)
56.             Case "Q"
57.                 btnLeft_Click(sender, e)
58.             Case "D"
59.                 btnRight_Click(sender, e)
60.         End Select
61.     End Sub
62. ____________________________________________________________________
63.     Private Sub Form2048_Load(sender As Object, e As EventArgs) Handles
    Me.Load
64. ' initialise la form
65. ' Au début, mettre toutes les variables dans le tableau à zéro et randomise
    2 chiffres de 2 (de probabilité 90%) ou 4(de probabilité 10%)
66.         Call initialiser()
67.         Call Affiche()
68.
69. ' Au début, on ne peut pas faire 'undo' car il n'y pas encore un mouvement
70.         btnUndo.Enabled = False
71.
```

```
72.         Me.KeyPreview = True
73.
74.     End Sub
75. ____________________________________________________________________
76.
77.     Private Sub Affiche()
78. ' Connecter les labels dans l'interface et les labels dans Form2048
79.         controlelabel()
80.
81. ' Connecter les textes de labels dans le form et les variables de jeu()
    dans la module et les afficher
82.         Dim i As Integer, j As Integer, l As Byte
83.  ' Les différentes valeurs sont correspondantes à différentes couleurs
84.         For i = 1 To n
85.             For j = 1 To n
86.                 l = (i - 1) * n + j
87.                 If Jeu(i, j) <> 0 Then
88.                     lbl(l).Text = CStr(Jeu(i, j))
89.                  Select Case lbl(l).Text = CStr(Jeu(i, j))
90.                       Case lbl(l).Text = 2
91.                         lbl(l).BackColor = Color.FromArgb(255, 234, 131)
92.                       Case lbl(l).Text = 4
93.                           lbl(l).BackColor = Color.FromArgb(249, 212, 36)
94.                       Case lbl(l).Text = 8
95.                         lbl(l).BackColor = Color.FromArgb(248, 152, 69)
96.                       Case lbl(l).Text = 16
97.                         lbl(l).BackColor = Color.FromArgb(252, 119, 5)
98.                       Case lbl(l).Text = 32
99.                         lbl(l).BackColor = Color.FromArgb(248, 142, 119)
100.                      Case lbl(l).Text = 64
101.                        lbl(l).BackColor = Color.FromArgb(251, 47, 26)
102.                      Case lbl(l).Text = 128
103.                        lbl(l).BackColor = Color.FromArgb(221, 99, 254)
104.                      Case lbl(l).Text = 256
105.                        lbl(l).BackColor = Color.FromArgb(145, 55, 169)
106.                      Case lbl(l).Text = 512
107.                        lbl(l).BackColor = Color.FromArgb(73, 198, 244)
108.                      Case lbl(l).Text = 1024
109.                        lbl(l).BackColor = Color.FromArgb(34, 163, 118)
110.                      Case lbl(l).Text = 2048
111.                          lbl(l).BackColor = Color.FromArgb(243, 13, 1)
112.                 End Select
113.             Else
114.                 lbl(l).Text = ""
```

```
115.                 lbl(l).BackColor = Color.FromArgb(150, 150, 150)
116.             End If
117.
118.         Next
119.     Next
120.
121. ' Afficher le score et le score maximum historique
122.     lblScore.Text = CStr(Score)
123.     lblScoreMax.Text = CStr(ScoreMax)
124.
125.  ' vérifiez que le jeu est terminé
126.     Call JeuBloquer()
127.
128.   End Sub
129.  _____________________________________________________________________
130.   Private Sub btnRestart_Click(sender As System.Object, e As
    System.EventArgs) Handles btnRestart.Click
131. ' gère le redémarrage d'une partie
132. ' Afin de faire face à une mauvaise opération inattendue, nous ajoutons
    un MsgBox pour demander si nous devons redémarrer le jeu.
133.     Dim restart As Byte
134.     restart = MsgBox("Vous êtes sûr?", 1, "Restart?????")
135.
136.     If restart = vbOK Then
137.         Call initialiser()
138.         Call Affiche()
139.         btnLeft.Enabled = True
140.         btnRight.Enabled = True
141.         btnUp.Enabled = True
142.         btnDown.Enabled = True
143.         btnUndo.Enabled = False
144.     End If
145.
146.     If restart = vbCancel Then
147.
148.     End If
149.   End Sub
150.  _____________________________________________________________________
151.   Private Sub btnUndo_Click(sender As System.Object, e As
    System.EventArgs) Handles btnUndo.Click
152. ' gère l'annulation du coup qui vient dêtre joué (on ne peut annuler qu'un
    coup)
153.     Annuler()   'Obtenez la valeur dont nous avons besoin pour afficher
154.     Affiche()   ', et après les affichez
```

```
155.
156.        btnUndo.Enabled = False
157.
158.    End Sub
159.   ____________________________________________________________________
160.
161.    Private Sub btnLeft_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnLeft.Click
162. ' gère le déplacement à gauche
163.
164. ' Après chaque déplacement, le bouton UNDO est disponible pour annuler
    ce coup
165.        btnUndo.Enabled = True
166. 'If e.KeyCode = Keys.Left Then
167. ' enregistrer les variables dans la table temporaire tem() avant le
    déplacement (Pour 'undo')
168.        Call tem()
169.        ' faire le déplacement à gauche
170.        Call Deplacer(1)
171.        ' comparer les valeurs avant et après
172.        Call SubComparer()
173. ' Lorsque les valeurs avant et après ne sont pas exactement les mêmes,
    le jeu peut générer un nombre aléatoire après le déplacement, et mettre
    à jour la valeur de la table d'annulation JeuAnnuler (Pour 'UnDo').
174.        If comparer = False Then
175.            ' randomise un nouveau chiffre dans le tableau jeu()
176.            Call TirerAleatoirement()
177.            Call TableAnnuler()
178.        End If
179. 'Lorsque les valeurs du jeu ne changent pas après le coup, le jeu ne génère
    pas de nouveaux nombres aléatoires. En même temps, la valeur de la table
    d'annulation pour UNDO ne sera pas mise à jour. Cela évite que si le mouvement
    est effectué plusieurs fois mais que les valeurs ne changent pas, cliquer
    sur UNDO ne retournera pas au résultat de l'étape qui changent les valeurs
    du jeu. C'est pourquoi nous avons d'abord stocké les valeurs du jeu dans
    la table temporaire tabletem() puis l'avons comparée, au lieu de les stocker
    directement dans la table d'annulation JeuAnnuler().
180.        ' afficher le résultat de déplacement
181.        Call Affiche()
182.
183.    End Sub
184.  ____________________________________________________________________
185.    Private Sub btnRight_Click(sender As System.Object, e As
    System.EventArgs) Handles btnRight.Click
```

```
186.        ' gère le déplacement à droite
187.        ' Les étapes sont les mêmes que les étapes vers la gauche
188.        btnUndo.Enabled = True
189.        Call tem()
190.        Call Deplacer(2)
191.        Call SubComparer()
192.        If comparer = False Then
193.            Call TirerAleatoirement()
194.            Call TableAnnuler()
195.        End If
196.        Call Affiche()
197.
198.    End Sub
199.  _____________________________________________________________________
200.    Private Sub btnUp_Click(sender As System.Object, e As
    System.EventArgs) Handles btnUp.Click
201.        ' gère le déplacement vers le haut
202.        ' Les étapes sont les mêmes que les étapes vers la gauche
203.        btnUndo.Enabled = True
204.        Call tem()
205.        Call Deplacer(3)
206.        Call SubComparer()
207.        If comparer = False Then
208.            Call TirerAleatoirement()
209.            Call TableAnnuler()
210.        End If
211.        Call Affiche()
212.
213.    End Sub
214.  _____________________________________________________________________
215.    Private Sub btnDown_Click(sender As System.Object, e As
    System.EventArgs) Handles btnDown.Click
216.        ' gère le déplacement vers le bas
217.        ' Les étapes sont les mêmes que les étapes vers la gauche
218.        btnUndo.Enabled = True
219.       Call tem()
220.        Call Deplacer(4)
221.        Call SubComparer()
222.        If comparer = False Then
223.            Call TirerAleatoirement()
224.            Call TableAnnuler()
225.        End If
226.        Call Affiche()
227.
```

```
228.    End Sub
229.  _____________________________________________________________________
230.    Private Sub JeuBloquer()
231.        'quand le Jeu termine, tous les boutons seront être bloqués sauf
    de bouton 'Restart'
232.        'Lorsque une valeur est égale à 2048, le jeu gagne et le jeu se
    termine.
233.        If JeuGagner() Then
234.            MsgBox("Vous avez bien atteint 2048 ! Le score est de " &
    lblScore.Text & " !")
235.            btnLeft.Enabled = False
236.            btnRight.Enabled = False
237.            btnUp.Enabled = False
238.            btnDown.Enabled = False
239.            btnUndo.Enabled = False
240.        End If
241.        'Quand il n'y a aucune possibilité de continuer à bouger, le jeu
    se termine.
242.        If JeuTerminer() Then
243.            MsgBox("Pas de mouvement possible ! Le score est de " &
    lblScore.Text & " !")
244.            btnLeft.Enabled = False
245.            btnRight.Enabled = False
246.            btnUndo.Enabled = False
247.            btnUp.Enabled = False
248.            btnDown.Enabled = False
249.        End If
250.
251.    End Sub
252. End Class
```

3) Version DRAGON BALL

```
1. Public Class Form2048
2.
3.     Public lbl(16) As Label
4.     Public button1 As Image
5.     Public button2 As Image
6.  ____________________________________________________________________
7.     Public Sub controlelabel()
8.         ' Connecter les labels dans l'interface et les labels dans Form2048
9.         lbl(1) = lbl1
10.        lbl(2) = lbl2
11.        lbl(3) = lbl3
12.        lbl(4) = lbl4
13.        lbl(5) = lbl5
14.        lbl(6) = lbl6
15.        lbl(7) = lbl7
16.        lbl(8) = lbl8
17.        lbl(9) = lbl9
18.        lbl(10) = lbl10
19.        lbl(11) = lbl11
20.        lbl(12) = lbl12
21.        lbl(13) = lbl13
22.        lbl(14) = lbl14
23.        lbl(15) = lbl15
24.        lbl(16) = lbl16
25.
26.    End Sub
27. ____________________________________________________________________
28. Protected Overrides Function processdialogkey(keydata As
    System.Windows.Forms.Keys) As Boolean
29.        'détecter si les touches de direction sont pressées
30.        If keydata = Keys.Up Or keydata = Keys.Down Or keydata = Keys.Left
    Or keydata = Keys.Right Then
31.            Return False
32.        Else
33.            Return MyBase.ProcessDialogKey(keydata)
34.        End If
35.    End Function
36. ____________________________________________________________________
37.    Private Sub Form2048_KeyDown(sender As Object, e As KeyEventArgs)
    Handles Me.KeyDown
38.        ' Connecter des boutons de clavier avec les boutons sur interface
39.        ' touches ←, →, ↑, ↓
```

```
        Select Case e.KeyCode
            Case Keys.Down
                btnDown_Click(sender, e)
            Case Keys.Up
                btnUp_Click(sender, e)
            Case Keys.Left
                btnLeft_Click(sender, e)
            Case Keys.Right
                btnRight_Click(sender, e)
        End Select

        'touches Q D Z S
        Select Case Chr(e.KeyValue)
            Case "S"
                btnDown_Click(sender, e)
            Case "Z"
                btnUp_Click(sender, e)
            Case "Q"
                btnLeft_Click(sender, e)
            Case "D"
                btnRight_Click(sender, e)
        End Select
    End Sub
 _____________________________________________________________________
    Private Sub Form2048_Load(sender As Object, e As EventArgs) Handles
Me.Load
        ' initialise la form
        'définir les styles différents de butons 'undo' quand il est bloqué
ou pas
        Call buttonstyle()
        ' Au début, mettre toutes les variables dans le tableau à zéro et
randomise 2 chiffres de 2 (de probabilité 90%) ou 4(de probabilité 10%)
        Call initialiser()
        Call Affiche()
        ' Au début, on ne peut pas faire 'undo' car il n'y pas encore un
mouvement
        btnUndo.Enabled = False
        'définir les styles différents de butons 'undo' quand il est bloqué
ou pas
        Me.KeyPreview = True

    End Sub
 _____________________________________________________________________
    Private Sub Affiche()
```

```
79.         'preparer des images différent pour des labels avec des valeurs
    différentes
80.
81.         Dim image1 As Image
82.         Dim image2 As Image
83.         Dim image3 As Image
84.         Dim image4 As Image
85.         Dim image5 As Image
86.         Dim image6 As Image
87.         Dim image7 As Image
88.         Dim image8 As Image
89.         Dim image9 As Image
90.         Dim image10 As Image
91.         Dim image11 As Image
92.         Dim image13 As Image
93.
94.         'extraire des images dans le ficher (path relative)
95.         image1 = Image.FromFile("..\..\images \01.png")
96.         image2 = Image.FromFile("..\..\images \02.png")
97.         image3 = Image.FromFile("..\..\images \03.png")
98.         image4 = Image.FromFile("..\..\images \04.png")
99.         image5 = Image.FromFile("..\..\images \05.png")
100.         image6 = Image.FromFile("..\..\images \06.png")
101.         image7 = Image.FromFile("..\..\images \07.png")
102.         image8 = Image.FromFile("..\..\images \08.png")
103.         image9 = Image.FromFile("..\..\images \09.png")
104.         image10 = Image.FromFile("..\..\images \10.png")
105.         image11 = Image.FromFile("..\..\images \11.png")
106.         image13 = Image.FromFile("..\..\images \13.png")
107.         button1 = Image.FromFile("..\..\images \button1.png")
108.         button2 = Image.FromFile("..\..\images \button2.png")
109.
110. ' Connecter les labels dans l'interface et les labels dans Form2048
111.         controlelabel()
112.
113. ' Connecter les textes de labels dans le form et les variables de jeu()
    dans la module et les afficher
114.         Dim i As Integer, j As Integer, l As Byte
115.  ' Les différentes valeurs sont correspondantes à différentes images
116.         For i = 1 To n
117.             For j = 1 To n
118.                 l = (i - 1) * n + j
119.                 If Jeu(i, j) <> 0 Then
120.                     lbl(l).Text = CStr(Jeu(i, j))
```

```
121.                     Select Case lbl(l).Text = CStr(Jeu(i, j))
122.                         Case lbl(l).Text = 2
123.                             lbl(l).Text = ""
124.                             lbl(l).Image = image1
125.                         Case lbl(l).Text = 4
126.                             lbl(l).Text = ""
127.                             lbl(l).Image = image2
128.                         Case lbl(l).Text = 8
129.                             lbl(l).Text = ""
130.                             lbl(l).Image = image3
131.                         Case lbl(l).Text = 16
132.                             lbl(l).Text = ""
133.                             lbl(l).Image = image4
134.                         Case lbl(l).Text = 32
135.                             lbl(l).Text = ""
136.                             lbl(l).Image = image5
137.                         Case lbl(l).Text = 64
138.                             lbl(l).Text = ""
139.                             lbl(l).Image = image6
140.                         Case lbl(l).Text = 128
141.                             lbl(l).Text = ""
142.                             lbl(l).Image = image7
143.                         Case lbl(l).Text = 256
144.                             lbl(l).Text = ""
145.                             lbl(l).Image = image8
146.                         Case lbl(l).Text = 512
147.                             lbl(l).Text = ""
148.                             lbl(l).Image = image9
149.                         Case lbl(l).Text = 1024
150.                             lbl(l).Text = ""
151.                             lbl(l).Image = image10
152.                         Case lbl(l).Text = 2048
153.                             lbl(l).Text = ""
154.                             lbl(l).Image = image11
155.                     End Select
156.                 Else
157.                     lbl(l).Text = ""
158.                     lbl(l).Image = image13
159.                 End If
160.             Next
161.         Next
162.
163.         'définir les styles différents de butons 'undo' quand il est
    bloqué ou pas
```

```
164.         Call buttonstyle()
165.         ' Afficher le score et le score maximum historique
166.         lblScore.Text = CStr(Score)
167.         lblScoreMax.Text = CStr(ScoreMax)
168.
169.         ' vérifiez que le jeu est terminé
170.         Call JeuBloquer()
171.
172.     End Sub
173. ____________________________________________________________________
174.     Private Sub btnRestart_Click(sender As System.Object, e As
    System.EventArgs) Handles btnRestart.Click
175.         ' gère le redémarrage d'une partie
176.         ' Afin de faire face à une mauvaise opération inattendue, nous
    ajoutons un MsgBox pour demander si nous devons redémarrer le jeu.
177.         Dim restart As Byte
178.         restart = MsgBox("Vous êtes sûr?", 1, "Restart?????")
179.
180.         If restart = vbOK Then
181.             Call initialiser()
182.             Call Affiche()
183.             btnLeft.Enabled = True
184.             btnRight.Enabled = True
185.             btnUp.Enabled = True
186.             btnDown.Enabled = True
187.             btnUndo.Enabled = False
188.         End If
189.
190.         If restart = vbCancel Then
191.
192.         End If
193.     End Sub
194. ____________________________________________________________________
195.     Private Sub btnUndo_Click(sender As System.Object, e As
    System.EventArgs) Handles btnUndo.Click
196.         ' gère l'annulation du coup qui vient dêtre joué (on ne peut
    annuler qu'un coup)
197.         Annuler()    'Obtenez la valeur dont nous avons besoin pour
    afficher
198.         Affiche()    ', et après les affichez
199.
200.         btnUndo.Enabled = False
201.         Call buttonstyle()
202.     End Sub
```

```
203. ____________________________________________________________________
204.     Private Sub btnLeft_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles btnLeft.Click
205.         ' gère le déplacement à gauche
206.         'If e.KeyCode = Keys.Left Then
207.         ' enregistrer les variables dans la table temporaire tem() avant
     le deplacement (Pour 'undo')
208. ' après chaque déplacement, on peut annuler qu'un coup
209.         btnUndo.Enabled = True
210.         Call tem()
211.         ' faire le deplacement à gauche
212.         Call Deplacer(1)
213.         ' comparer les valeurs avant et après
214.         Call SubComparer()
215.         ' Lorsque les valeurs avant et après ne sont pas exactement les
     même, le jeu peut générer un nombre aléatoire après le déplacement,
216.         'et mettre à jour la valeur de la table d'annulation JeuAnnuler
     (Pour 'UnDo').
217.         If comparer = False Then
218.             ' randomize un nouveau chiffre dans le tableau jeu()
219.             Call TirerAleatoirement()
220.             Call TableAnnuler()
221.         End If
222.         'Lorsque les valeurs du jeu ne changent pas après le coup, le jeu
     ne génère pas de nouveaux nombres aléatoires.
223.         'En même temps, la valeur de la table d'annulation pour UNDO ne
     sera pas mise à jour.
224.         'Cela évite que si le mouvement est effectué plusieurs fois mais
     que les valeurs ne changent pas, cliquer sur UNDO ne retournera pas
225.         'au résultat de l'étape qui changent les valeurs du jeu. C'est
     pourquoi nous avons d'abord stocké les valeurs du jeu dans la table
226.         'temporaire tabletem() puis l'avons comparée, au lieu de les
     stocker directement dans la table d'annulation JeuAnnuler().
227.         ' afficher le résultat de deplacement
228.         Call Affiche()
229.     End Sub
230. ____________________________________________________________________
231.     Private Sub btnRight_Click(sender As System.Object, e As
     System.EventArgs) Handles btnRight.Click
232.         ' gère le déplacement à droite
233.         ' Les étapes sont les mêmes que les étapes vers la gauche
234.         btnUndo.Enabled = True
235.         Call tem()
236.         Call Deplacer(2)
```

```
237.         Call SubComparer()
238.         If comparer = False Then
239.             Call TirerAleatoirement()
240.             Call TableAnnuler()
241.         End If
242.         Call Affiche()
243.
244.     End Sub
245. ____________________________________________________________________
246.     Private Sub btnUp_Click(sender As System.Object, e As
    System.EventArgs) Handles btnUp.Click
247.         ' gère le déplacement vers le haut
248.         ' Les étapes sont les mêmes que les étapes vers la gauche
249.         btnUndo.Enabled = True
250.         Call tem()
251.         Call Deplacer(3)
252.         Call SubComparer()
253.         If comparer = False Then
254.             Call TirerAleatoirement()
255.             Call TableAnnuler()
256.         End If
257.         Call Affiche()
258.
259.     End Sub
260. ____________________________________________________________________
261.     Private Sub btnDown_Click(sender As System.Object, e As
    System.EventArgs) Handles btnDown.Click
262.         ' gère le déplacement vers le bas
263.         ' Les étapes sont les mêmes que les étapes vers la gauche
264.         btnUndo.Enabled = True
265.         Call tem()
266.         Call Deplacer(4)
267.         Call SubComparer()
268.         If comparer = False Then
269.             Call TirerAleatoirement()
270.             Call TableAnnuler()
271.         End If
272.         Call Affiche()
273.
274.
275.     End Sub
276. ____________________________________________________________________
277.     Private Sub JeuBloquer()
```

```
278.         'quand le Jeu termine, tous les boutons seront être bloqués sauf
     de bouton 'Restart'
279.         'Lorsque une valeur est égale à 2048, le jeu gagne et le jeu se
     termine.
280.         If JeuGagner() Then
281.             MsgBox("Vous avez bien atteint 2048 ! Le score est de " &
     lblScore.Text & " !")
282.             btnLeft.Enabled = False
283.             btnRight.Enabled = False
284.             btnUp.Enabled = False
285.             btnDown.Enabled = False
286.             btnUndo.Enabled = False
287.         End If
288.         'Quand il n'y a aucune possibilité de continuer à bouger, le jeu
     se termine.
289.         If JeuTerminer() Then
290.             MsgBox("Pas de mouvement possible ! Le score est de " &
     lblScore.Text & " !")
291.             btnLeft.Enabled = False
292.             btnRight.Enabled = False
293.             btnUndo.Enabled = False
294.             btnUp.Enabled = False
295.             btnDown.Enabled = False
296.
297.         End If
298.     End Sub
299. ____________________________________________________________________
300.     Private Sub buttonstyle()
301.         'definir les styles différents de button 'undo' quand il est
     bloqué ou pas
302.         If btnUndo.Enabled = True Then
303.             btnUndo.BackgroundImage = button1
304.             btnUndo.ForeColor = Color.RoyalBlue
305.         ElseIf btnUndo.Enabled = False Then
306.             btnUndo.BackgroundImage = button2
307.             btnUndo.ForeColor = Color.Red
308.         End If
309.     End Sub
310.
311. ____________________________________________________________________
312. End Class
```

IV. Noyau fonctionnel

```
1.  Module Module2048
2.      Public n As Byte = 4
3.      Public Jeu(n, n) As Short
4.      Public Score As Short, ScoreAnnuler As Short, ScoreMax As Short,
    ScoreMaxAnnuler As Short
5.      Public tabletem(n, n) As Short
6.      Public scoretem As Short
7.      Public scoreMaxTem As Integer
8.      Public JeuAnnuler(n, n) As Short
9.      Public PointGagne As Integer = 2048
10.     Public bloquer As Boolean
11.     Public comparer As Boolean
12. ____________________________________________________________________
13.     Public Sub initialiser()
14.         ' mettre toutes les variables dans le tableau à zéro
15.         Dim i As Integer, j As Integer
16.
17.         For i = 1 To n
18.             For j = 1 To n
19.                 Jeu(i, j) = 0
20.             Next
21.         Next
22.
23.         Score = 0
24.         'dans deux cases du jeu, on met une tuile aléatoire de valeur 2 ou
    4
25.         TirerAleatoirement()
26.         TirerAleatoirement()
27.
28.     End Sub
29. ____________________________________________________________________
30.     Public Sub TirerAleatoirement()
31.         'choisir une case par la sélection aléatoire des indices de la table
    Jeu
32.         Dim random1 As New Random()
33.         Dim i As Byte, j As Byte
34.
35.         i = random1.Next(1, 5)
36.         j = random1.Next(1, 5)
37.
```

```
38.         'assurer la case choisie est vide, c'est-à-dire, si la case aléatoire
    n'est pas égale à 0, on resélectionnera une nouvelle case
39.         While Jeu(i, j) <> 0
40.             i = random1.Next(1, 5)
41.             j = random1.Next(1, 5)
42.         End While
43.
44.         'obtenir un paramètre entre 1 to 100 aléatoirement.
45.         'si ce paramètre est inférieur à 90, la valeur de cette case aléatoire
    est égale à 2, sinon la valeur est égale à 4.
46.         Dim rand As New Random()
47.         Dim proba As Integer
48.         proba = rand.Next(1, 101)
49.         If proba <= 90 Then
50.             Jeu(i, j) = 2
51.         Else
52.             Jeu(i, j) = 4
53.         End If
54.
55.     End Sub
56. ____________________________________________________________________
57.     Public Function tem()
58.         Dim i As Integer, j As Integer
59.         'Enregistrer la valeur du bloc et le score de chaque étape
60.         For i = 1 To n
61.             For j = 1 To n
62.                 tabletem(i, j) = Jeu(i, j)
63.             Next
64.         Next
65.         scoretem = Score
66.         scoreMaxTem = ScoreMax
67.         Return scoretem
68.         Return scoreMaxTem
69.     End Function
70. ____________________________________________________________________
71.     Public Sub SubComparer()
72.         'Les valeurs de la table temporaire et de la table de jeu déjà déplacée
    sont comparées une par une, et lorsqu'elles sont égales, on compte avec
    le paramètre k.
73.         Dim k As Integer
74.         k = 0
75.         For i = 1 To n
76.             For j = 1 To n
77.                 If Jeu(i, j) = tabletem(i, j) Then
```

```
78.                 k = k + 1
79.             End If
80.         Next
81.     Next
82.     'Lorsque k = 16, cela signifie que les deux tables sont complètement
    égales, il n'y a pas de changement avant et après le déplacement, et le
    paramètre comparer est True.
83.     If k = 16 Then
84.         comparer = True
85.     Else
86.         comparer = False
87.     End If
88.   End Sub
89. ____________________________________________________________________
90.
91.   Public Function TableAnnuler()
92.     'Pour revenir à l'étape précédente, nous obtenons toutes les valeurs
    qui doivent apparaître dans l'interface du jeu à partir de la table
    temporaire.
93.     Dim i As Integer, j As Integer
94.     For i = 1 To n
95.         For j = 1 To n
96.             JeuAnnuler(i, j) = tabletem(i, j)
97.         Next
98.     Next
99.     ScoreAnnuler = scoretem
100.     ScoreMaxAnnuler = scoreMaxTem
101.     Return ScoreAnnuler
102.     Return ScoreMaxAnnuler
103.
104.   End Function
105. ____________________________________________________________________
106.
107.   Public Function Annuler()
108.     'La valeur dans la table d'annulation est copiée dans la table de
    jeu afin d'afficher la valeur de l'étape précédente dans l'interface et
    revenir au dernier coup par le bouton Undo.
109.     For i = 1 To n
110.         For j = 1 To n
111.             Jeu(i, j) = JeuAnnuler(i, j)
112.         Next
113.     Next
114.     Score = ScoreAnnuler
115.     ScoreMax = ScoreMaxAnnuler
```

```
116.        Return Score
117.        Return ScoreMax
118.
119.    End Function
120.  ____________________________________________________________________
121.
122.    Public Function JeuGagner() As Boolean
123.        JeuGagner = False
124.        Dim i As Integer, j As Integer
125.        'Trouver s'il y a une valeur dans la table de jeu est égale à 2048.Si,
    le JeuGagner est correcte et on quitte le jugement. Sinon, on continue.
126.        For i = 1 To n
127.            For j = 1 To n
128.                If Jeu(i, j) = PointGagne Then
129.                    JeuGagner = True
130.                    Exit Function
131.                End If
132.            Next
133.        Next
134.    End Function
135.  ____________________________________________________________________
136.
137.    Public Function JeuTerminer() As Boolean
138.        'Le jeu se termine lorsqu'aucune des trois conditions suivantes
    n'est vraie. Trois conditions sont indispensables.
139.        'Les trois conditions sont:
140.        '- chaque valeur de la table est non nulle
141.        '- Gauche et droite ne peuvent pas bouger
142.        '- Impossible de monter et descendre
143.        If Not peutcontinue() And Not peutcolonne() And Not peutligne()
    Then
144.            JeuTerminer = True
145.        Else
146.            JeuTerminer = False
147.        End If
148.
149.        Return JeuTerminer
150.    End Function
151.  ____________________________________________________________________
152.
153.    Public Function peutcontinue() As Boolean
154.        'Vérifiez que chaque valeur de la table est non nulle
155.        Dim peutdepalcer As Boolean, NbpasZero As Byte
156.
```

```
157.        NbpasZero = 0
158.        For i = 1 To n
159.            For j = 1 To n
160.                If Jeu(i, j) <> 0 Then  'Lorsque la case n'est pas 0, compter
    dans le paramètre NbpasZero.
161.                    NbpasZero = NbpasZero + 1
162.                End If
163.            Next
164.        Next
165.        'Quand il y a une valeur de 0 dans la table, le jeu peut continuer.
    Lorsque les valeurs de 16 grilles sont non nulles, la première condition
    de fin du jeu est satisfaite. Si aucune direction ne peut bouger après,
    le jeu se terminera.
166.        If NbpasZero = 16 Then
167.            peutdepalcer = False
168.        Else
169.            peutdepalcer = True
170.        End If
171.
172.        Return peutdepalcer
173.    End Function
174. ____________________________________________________________________
175.
176.    Public Function peutligne() As Boolean
177. 'Déterminer si la gauche et la droite peuvent bouger
178.        Dim a As Boolean
179.        a = False
180.  'Quand il y a un nombre égal de valeurs adjacentes, cela signifie que
    le jeu peut encore bouger. Le jugement est correct, nous quittons la
    fonction.
181.       For j = 1 To 4
182.            For i = 1 To 3
183.                If Jeu(i, j) = Jeu(i + 1, j) Then
184.                    a = True
185.                    Return a
186.                    Exit Function
187.                End If
188.            Next
189.        Next
190.
191.        Return a
192.    End Function
193. ____________________________________________________________________
194.
```

```
195.    Public Function peutcolonne() As Boolean
196. 'Déterminez si vous pouvez monter et descendre.
197.        Dim a As Boolean
198.        a = False
199. 'Quand il y a un nombre égal de valeurs adjacentes, cela signifie que le
   jeu peut encore bouger. Le jugement est correct, nous quittons la fonction.
200.        For i = 1 To 4
201.            For j = 1 To 3
202.                If Jeu(i, j) = Jeu(i, j + 1) Then
203.                    a = True
204.                    Return a
205.                    Exit Function
206.                End If
207.            Next
208.        Next
209.
210.        Return a
211.    End Function
212. _____________________________________________________________________
213.
214.    Public Sub Deplacer(ByVal mode As Integer)
215.        Dim TableN() As Short
216.        Dim i As Integer, j As Integer, k As Integer, t As Integer
217.        Dim a As Integer, R As Integer, E As Integer, W As Integer
218. 'En contrôlant un nombre spécifique de paramètres, nous contrôlons un
   nombre spécifique de conditions pour atteindre l'algorithme de déplacement
   vers les différentes directions.
219. 'Utilisez le paramètre mode pour afficher 4 directions. Gauche 1, Droite
   2, Haut 3, Bas 4.
220.
221. 'étape 1 : Selon différentes directions, utilisez des paramètres
   différents.
222.        If mode < 3 Then
223.            If mode = 1 Then
224.                a = 1 : W = 0 : R = 1 : E = 4   'Gauche 1
225.            Else
226.                a = 4 : W = 5 : R = -1 : E = 1  'Droite 2
227.            End If
228.
229. 'étape 2 : Retirer des valeurs non zéro dans jeu() et les enregistrer dans
   TableN()
230.            For i = 1 To n
231.                ReDim TableN(n)
232.                k = 0
```

```
233.             For j = a To E Step R
234.                 If Jeu(i, j) <> 0 Then
235.                     k = k + 1
236.                     TableN(k) = Jeu(i, j)
237.                 End If
238.             Next
239.
240. 'étape 3 : Lorsqu'on obtient nez au moins deux valeurs non nulles dans
     cette ligne ou colonne, Faire les mouvements et calculer des valeurs
241.             If k > 1 Then
242.                 For j = 1 To k - 1
243.                     If TableN(j) = TableN(j + 1) Then
244.                         TableN(j) = 2 * TableN(j)
245.                         TableN(j + 1) = 0
246. 'compter le score et mise en jour le score maximal
247.                         Score = Score + TableN(j)
248.                         If ScoreMax < Score Then
249.                             ScoreMax = Score
250.                         Else
251.                             ScoreMax = ScoreMax
252.                         End If
253.                     End If
254.                 Next
255. 'étape 4 : mettre les résultats dans le nouvel ordre
256.                 t = W
257.                 For j = 1 To k
258.                     If TableN(j) <> 0 Then
259.                         t = t + R
260.                         Jeu(i, t) = TableN(j)
261.                     End If
262.                 Next
263.
264.                 For j = t + R To E Step R
265.                     Jeu(i, j) = 0
266.                 Next
267.
268.             ElseIf k = 1 Then
269.  'Pour les lignes ou colonnes avec une seule valeur différente de zéro,
     déplacez cette valeur vers la première position dans la direction.
270.                 Jeu(i, a) = TableN(k)
271.                 Jeu(i, a + R) = 0
272.                 Jeu(i, a + R + R) = 0
273.                 Jeu(i, a + R + R + R) = 0
274.             End If
```

```
275.          Next
276.       Else
277. 'étape 1 : Selon différentes directions, utilisez des paramètres
   différents.
278.          If mode = 3 Then
279.             a = 1 : W = 0 : R = 1 : E = 4   'Haut 3.
280.          Else
281.             a = 4 : W = 5 : R = -1 : E = 1  'Bas 4.
282.          End If
283. 'étape 2 : Retirer des valeurs non zéro dans jeu() et les enregistrer dans
   TableN()
284.          For i = 1 To n
285.             ReDim TableN(n)
286.             k = 0
287.             For j = a To E Step R
288.                If Jeu(j, i) <> 0 Then
289.                   k = k + 1
290.                   TableN(k) = Jeu(j, i)
291.
292.                End If
293.             Next
294. 'étape 3 : Lorsqu'on obtient nez au moins deux valeurs non nulles dans
   cette ligne ou colonne, Faire les mouvements et calculer des valeurs
295.             If k > 1 Then
296.                For j = 1 To k - 1
297.                   If TableN(j) = TableN(j + 1) Then
298.                      TableN(j) = 2 * TableN(j)
299.                      TableN(j + 1) = 0
300. 'compter le score et mise en jour le score maximal
301.                      Score = Score + TableN(j)
302.                      If ScoreMax < Score Then
303.                         ScoreMax = Score
304.                      Else
305.                         ScoreMax = ScoreMax
306.                      End If
307.                   End If
308.                Next
309. 'étape 4 : mettre les résultats dans le nouvel ordre
310.                t = W
311.                For j = 1 To k
312.                   If TableN(j) <> 0 Then
313.                      t = t + R
314.                      Jeu(t, i) = TableN(j)
315.                   End If
```

```
316.                 Next
317.
318.                 For j = t + R To E Step R
319.                     Jeu(j, i) = 0
320.                 Next
321.
322.             ElseIf k = 1 Then
323. 'Pour les lignes ou colonnes avec une seule valeur différente de zéro,
    déplacez cette valeur vers la première position dans la direction.
324.                 Jeu(a, i) = TableN(k)
325.                 Jeu(a + R, i) = 0
326.                 Jeu(a + R + R, i) = 0
327.                 Jeu(a + R + R + R, i) = 0
328.             End If
329.         Next
330.     End If
331.   End Sub
332.
333.
334. End Module
```

V. Conclusion

Grâce à ce projet, nous connaissons mieux l'utilisation de VB.NET. Ce jeu est différent du jeu dernier semestre, 2048 est un jeu plus célèbre et bien connu. Alors que nous améliorons constamment nos compétences en programmation et notre logique, nous avons également une meilleure compréhension du développement de certaines applications sur le marché grâce à ce projet.

Dans la collaboration de groupe, nous avons avancé de nombreuses hypothèses pour l'algorithme de base et effectué de nombreux tests. Enfin, nous avons choisi une solution optimale que nous pouvons atteindre. Maintenant que nous avons beaucoup de compréhension de la programmation collaborative du groupe, et le processus de coopération est également très agréable. En même temps, quand nous avons écrit l'algorithme, nous avons avancé quelques idées, mais notre capacité n'a pas pu réaliser ces idées. Nous devons également améliorer constamment nos capacités.