LIAO Xiyue
CASTILLO Manuel

Groupe TD3

Projet de programmation structurée
Semestre 2

28 avril 2018

Chargé de TD : Mme DOUTRE

Sommaire

Introduction

Pour la réalisation de notre projet nous avons utilisés les notions vues en cours, durant les séances de travaux pratiques (TP) mais aussi nous sommes allés chercher sur Internet des informations pertinentes nous permettant de réussir notre projet.

Le but de ce dossier est de présenter le sujet et de quelle façon nous l'avons interprété puis ensuite résolu.

Règles du jeu

Le jeu 2048 se compose de 16 cases pouvant être déplacées tout au long d'une grille, chaque case remplie affiche un numéro. L'objectif du jeu consiste à faire fusionner les cases ayant le même numéro et se trouvant l'une à côté de l'autre.
Le joueur a la possibilité de revenir un coup en arrière s'il souhaite faire une action différente.
Le jeu se termine quand le joueur le joueur ne peut pas fusionner aucune case et toute la grille est remplie.
Quand le joueur atteint 2048 il gagne néanmoins il peut continuer à jouer s'il le veut.

Développement du jeu

Nous avons utilisé le logiciel Visual Studio 2010 pour développer notre jeu avec une interface très similaire à la version existante pour le jeu sur iPhone.
Le jeu se déroule grâce au code que nous avons créé et que nous allons vous présenter dans la suite de ce dossier.

Interface

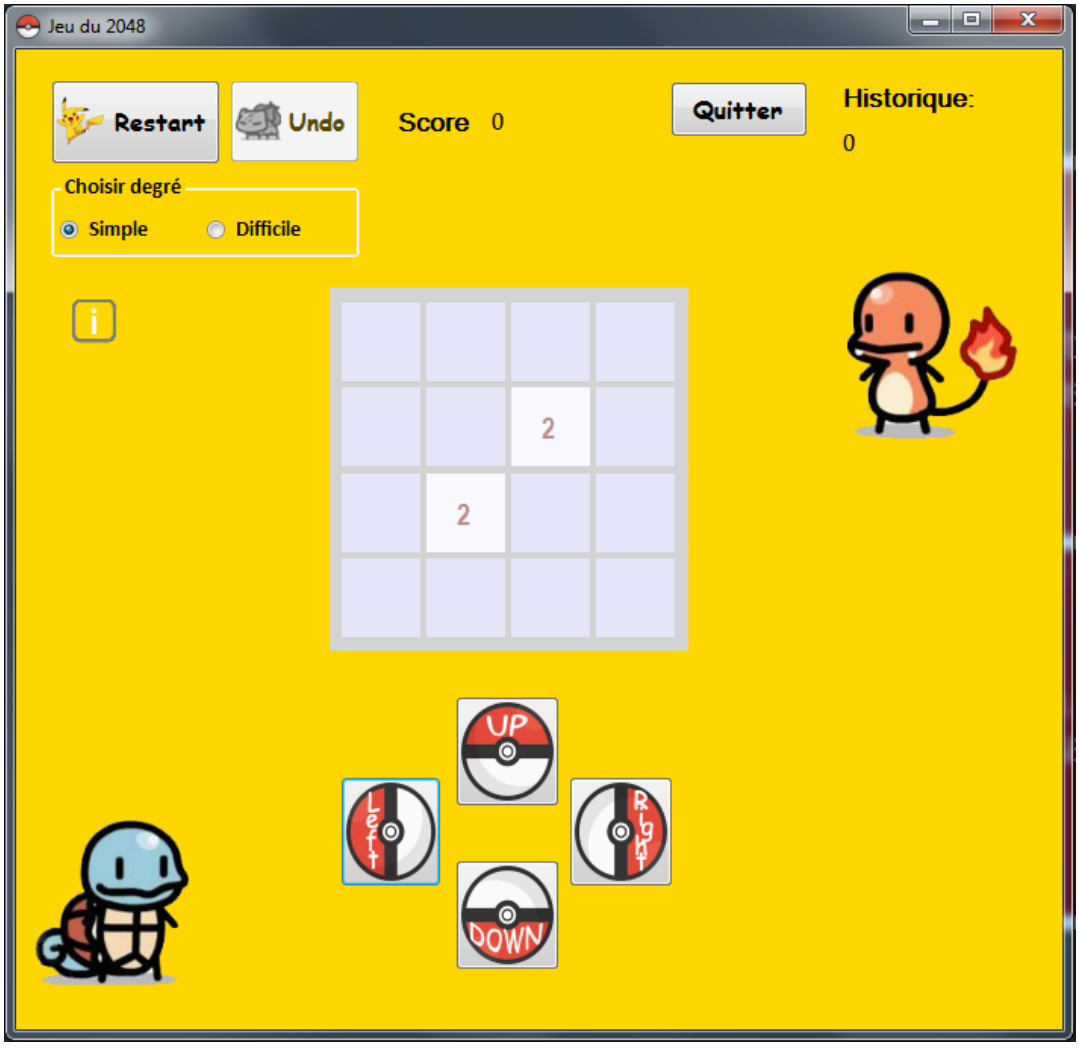
Nous avons commencé par faire l'interface jeu, cela permet d'avoir une idée plus précise des mécanismes du jeu et quels sont les fonctions et procédures nécessaires. En plus nous pouvons avoir une première image du produit final que nous allons présenter.
Il faut expliquer aussi que nous voulions faire un projet différent dans le sens que la plupart des autres groupes se sont inspirés de la même interface (iPhone) donc nous avons aménagé notre interface pour qu'elle soit unique.

Notre interface se compose de plusieurs éléments, ceux dont le jeu précise et ceux dont nous avons pris la liberté de rajouter.

Suivant les consignes du jeu nous avons placé une grille avec les 16 cases qui va utiliser le joueur. Nous avons rajouté aussi le bouton « restart » permettant de redémarrer la partie et

le bouton « undo » permettant de revenir un coup en arrière pendant le jeu. Vous trouverez aussi le label « score » permettant de calculer le score du joueur.

De notre nous avons rajouté le bouton « quitter » permettant d'abandonner une partie et quitter le jeu sans besoin d'atteindre 2048 ou d'arriver au « Game Over ». Ainsi que des boutons permettant d'émuler les flèches du clavier, la consigne étant de permettre à l'utilisateur d'utiliser les flèches du clavier nous estimons que c'est important de lui donner une autre façon de jouer.
Par rapport au score nous sommes conscients qu'un joueur veut toujours battre son score les plus élevé, alors nous avons mis un label « historique » stockant le meilleur score réalisé par l'utilisateur pendant une session.
De plus nous avons mis à disposition un onglet information « i » renseignant le joueur de la jouabilité de notre application.
Finalement nous pensons que c'est plus intéressant pour un joueur d'avoir plus d'un niveau de difficulté donc il peut choisir entre deux niveaux : « simple » et « difficile ».



Interface du jeu 2048

Module

Notre module c'est la feuille où nous écrivons le code permettant d'exécuter le jeu, c'est le noyau de notre projet.
Ici nous définissons les variables qu'on utilisera tout au long du projet ainsi que les fonctions et les procédures que nous appelons dans la forme.

Variables définies « public »

Nous avons défini les variables suivantes :

```
    Public n As Byte = 4
    Public Jeu (n, n) As Integer
    Public Undo (n, n) As Integer
    Public verifier (n, n) As Integer
    'Pour la score on a changé son type à Integer car on a besoin de valeurs plus
     grandes'
    Public Score As Integer = 0
    Public UndoScore As Integer
    Public verifierScore As Integer
    Public Historique As Integer = 0
    Public memoire (4) As Integer
    Public direction As String
```

Fonctions et procédures à utiliser

Procédure Initialiser

La première procédure que nous utilisons est Initialiser dans laquelle nous avons inséré deux fonctions, RandomCase et RandomValeur :

```
    'Procédure initialisant le jeu et permettant de remplir deux cases pour commencer
à jouer'
    Public Sub Initialiser (ByVal degre)

        Dim x As Integer
        Dim y As Integer
        Dim z As Integer
        Dim i As Byte = 1
        'On initialise le score à 0'
        Score = 0
        'On définit les variables x et y comme lignes et colonnes'
        For x = 1 To n
            For y = 1 To n
                Jeu (x, y) = 0
            Next
        Next
        'On appelle la fonction nous permettant de choisir les cases aléatoirement
pour les lignes comme pour les colonnes'
        While i <= 2
            x = RandomCase()
            y = RandomCase()
            If Jeu (x, y) = 0 Then
                '
```

```
La valeur z est égal soit à 2 ou 4 puisque on appelle la fonction RandomValeur'
                z = RandomValeur(degre)
                Jeu (x, y) = z
                i = i + 1
            End If
        End While
    End Sub
```

Cette procédure nous permet de charger les paramètres principaux du jeu, le score affiché est égal à 0 et dans la grille il y a deux cases remplisses avec des 2 ou des 4. Grâce aux fonctions RandomCase et RandomValeur que nous appelons, la procédure va choisir aléatoirement les deux cases et les valeurs correspondants.

Procédure Tirer Aléatoirement

La procédure définie à continuation nous nous servons à chaque coup jouer, c'est la procédure TirerAleatoirement :

```
    'Cette procédure permet de placer une valeur aléatoirement (2 ou 4) dans une case
vide'
    Public Sub TirerAleatoirement(ByVal degre)
        Dim x As Integer
        Dim y As Integer
        Dim z As Integer
        Dim ajouter As Boolean = False
        While ajouter = False
            x = RandomCase()
            y = RandomCase()
            If Jeu(x, y) = 0 Then
                z = RandomValeur(degre)
                Jeu(x, y) = z
                ajouter = True
            End If
        End While
    End Sub
```

Après chaque coup joué il doit apparaitre automatiquement dans une des cases libres restantes soit un 2 soit un 4. Comme pour la procédure permettant d'initialiser le jeu, nous appellons les fonctions RandomCase et RandomValeur pour que le programme choisit la case et le chiffre aléatoirement.

Fonction valeur aléatoire

La fonction suivante nous montre qu'il existe deux niveaux de difficultés différents, nous considérons que c'est une fonctionnalité très intéressante pour un joueur qui domine déjà bien le jeu de pouvoir tester un niveau de difficulté supérieur :

```
'Cette fonction permet d'avoir deux niveaux différents de difficultés, simple et
difficile'
    Public Function RandomValeur(ByVal degre)
        Dim valeur As Integer
        Dim curseur As Integer
        Randomize()
        'Quand l'utilisateur choisit simple il a 90% de chance qu'il lui apparaisent
plus de 2 après chaque coup joué que de 4'
        If degre = "simple" Then
            curseur = Int((Rnd() * 100) + 1)
            If curseur <= 90 Then
                valeur = 2
            Else
                valeur = 4
            End If
            'Quand c'est en "difficile" il y a 50% de possibilités d'avoir un 2 ou un
4 après chaque coup joué'
        ElseIf degre = "difficile" Then
            valeur = Int((2 - 1 + 1) * Rnd() + 1) * 2
        End If
        Return valeur
    End Function
```

Cette fonction détermine que pour le niveau simple il va y avoir une probabilité de 90% que la valeur aléatoire qui va être choisie soit un deux. Alors que pour le niveau simple il y a une probabilité de 50% pour la valeur 2 et 50% pour la valeur 4.

Fonction case aléatoire

La fonction qui nous permet de choisir aléatoirement une case c'est :

```
    'Cette fonction permet de choisir aléatoirement la case'
    Public Function RandomCase()

        Dim box As Integer

        Randomize()
        box = Int((n - 1 + 1) * Rnd() + 1)

        Return box

    End Function
```

Procédure le meilleur score

Nous avons fait une procédure permettant que le meilleur score pendant une même session soit enregistré. Quand nous parlons de session c'est quand l'utilisateur lance le jeu jusqu'à qui le ferme.

```
'Cette procédure permet d'enregistrer le meilleur score réaliser par le joueur'
Public Sub historiqueScore()
    If Historique < Score Then
        Historique = Score
    End If
End Sub
```

Si le score réalisé pendant une partie est supérieur au score enregistré dans le label alors celui-ci est remplacé par le score de la partie en cours.

Procédure Undo

Une des fonctionnalités du jeu demandées est celle de pouvoir revenir toujours un coup en arrière, cette procédure est un pas intermédiaire :

```
'Cette procédure permet d'enregistrer le dernier coup'
Public Sub undoJeu()
    For x = 1 To n
        For y = 1 To n
            Jeu(x, y) = Undo(x, y)
        Next
    Next
    Score = UndoScore
End Sub
```

À chaque coup joué nous enregistrons le jeu dans un tableau Undo(x,y) ainsi que le score à chaque tour.

Procédure charger Undo

Une des fonctionnalités du jeu demandées est celle de pouvoir revenir toujours un coup en arrière. Nous pouvons répondre à cette fonctionnalité grâce à la procédure suivante :

```
'Cette procédure est utilisé pour remettre le dernier coup enregistré'
Public Sub enregistrerUndo()
    For x = 1 To n
        For y = 1 To n
            Undo(x, y) = Jeu(x, y)
        Next
    Next
    UndoScore = Score
End Sub
```

Nous récupérons le jeu enregistré dans le tableau Undo et nous l'insérons à nouveau dans le tableau jeu.

Procédure pour les déplacements

La procédure déplacer est la plus importante de notre projet car elle nous permet de réaliser tous les déplacements nécessaires pour le bon fonctionnement du jeu :

```
    Public Sub deplacer(ByVal direction)

        'Déclaration des variables utilisées dans cette procédure'
        Dim x As Integer 'Indicateur de ligne de table Jeu'
        Dim y As Integer 'Indicateur de colonne de table Jeu'
        'a et b indiquent l'ordre de calcul. Pour chaque ligne ou chaque colonne nous
faisons le calcul de a jusqu'à b'

        Dim a As Integer
        Dim b As Integer
        Dim s As Integer 's indique la position dans la boucle du calcul '
        Dim i As Byte

        'Définition l'ordre de calcul, si le déplacement est vers "up" le calcul des
valeurs pour chaque colonne y depuis 1(a) à n(b) est step(s) = 1,
        'si le déplacement est vers "left" le calcul des valeurs pour chaque ligne x
depuis 1(a) jusqu'à n(b) est step(s) = 1'

        If CStr(direction) = "up" Or CStr(direction) = "left" Then
            a = 1
            b = n
            s = 1
            'Le déplacement "down" est l'inverse du déplacement "up", le déplacement
"right" est l'inverse de "left" c'est-à-dire step = -1'

        ElseIf CStr(direction) = "down" Or CStr(direction) = "right" Then
            a = n
            b = 1
            s = -1
        End If

        'Si le déplacement est vers "up" ou "down", on fait calcul pour chaque ligne,
donc la boucle principale est sur les colonnes (y)'

        If CStr(direction) = "up" Or CStr(direction) = "down" Then
            For y = 1 To n
                i = 0

                'Initialisation du tableau memoire, mettant tous les valeurs à 0'
                For j = 1 To n
                    memoire(j) = 0
                Next
                'On met les valeurs diffèrents à 0 de cette colonne dans un nouveau
tableau memoire de 1 dimension'

                For x = a To b Step s
                    If Jeu(x, y) <> 0 Then
                        i = i + 1
                        memoire(i) = Jeu(x, y)
                        Jeu(x, y) = 0
                    End If
                Next
```

```
                Call CalMemoire()

                'On remet les valeurs de table memoire dans la colonne du table Jeu'
                i = 1
                For x = a To b Step s
                    Jeu(x, y) = memoire(i)
                    i = i + 1
                Next
            Next
            'Si le déplacement est vers "left" ou "right" il y a un calcul pour chaque
colonne, donc la boucle principal est sur les lignes (x)'

        ElseIf CStr(direction) = "left" Or CStr(direction) = "right" Then
            For x = 1 To n
                i = 0
                For j = 1 To n
                    memoire(j) = 0
                Next
                For y = a To b Step s
                    If Jeu(x, y) <> 0 Then
                        i = i + 1
                        memoire(i) = Jeu(x, y)
                        Jeu(x, y) = 0
                    End If
                Next

                Call CalMemoire()

                i = 1
                For y = a To b Step s
                    Jeu(x, y) = memoire(i)
                    i = i + 1
                Next
            Next
        End If
    End Sub
```

Procédure tableau intermédiaire calculs

Pour réaliser les additions des lignes et des colonnes, nous avons créé un tableau intermédiaire qui permet de stocker les valeurs de calculs effectués à chaque coup. Et la procédure à continuation est celle qui nous permets de réaliser les calculs souhaités :

```
    Sub CalMemoire()
        'Cette fonction a pour objectif de faire le calcul de la table memoire, on a
tiré les valeurs différentes à 0 dans cette table depuis une ligne ou une colonne du
tableau Jeu'
        'on veut calculer les valeurs dans cette table. On remet les résultats
différents à 0 dans l'ordre dans ce tableau. Chaque fois on combine deux valeurs, on
calcule le score.'

        If memoire (1) <> 0 Then
            If memoire (1) = memoire (2) Then
                memoire (1) = memoire (1) + memoire (2)
                memoire (2) = 0
                Score = Score + memoire (1)
                If memoire (3) <> 0 Then
                    If memoire (3) = memoire (4) Then
                        memoire (2) = memoire (3) + memoire (4)
                        memoire (3) = 0
                        memoire (4) = 0
                        Score = Score + memoire (2)
                    Else
                        memoire (2) = memoire (3)
                        memoire (3) = memoire (4)
                        memoire (4) = 0
                    End If
                End If
            Else
                If memoire (2) <> 0 Then
                    If memoire (2) = memoire (3) Then
                        memoire (2) = memoire (2) + memoire (3)
                        memoire (3) = memoire (4)
                        memoire (4) = 0
                        Score = Score + memoire (2)
                    Else
                        If memoire (3) <> 0 Then
                            If memoire (3) = memoire (4) Then
                                memoire (3) = memoire (3) + memoire (4)
                                memoire (4) = 0
                                Score = Score + memoire (3)
                            End If
                        End If
                    End If
                End If
            End If
        End If
    End Sub
```

Fonction mouvements restants

Cette fonction permet de connaître s'il y a encore de mouvements possibles pour un joueur. Elle est importante surtout pour savoir quand une partie est finie.

```
    'Cette fonction permet de vérifier que le joueur a encore des actions possibles à
faire'
    Public Function verifierdeplacer(ByVal direction) As Boolean
        Dim OK As Boolean = False
        For x = 1 To n
            For y = 1 To n
                verifier(x, y) = Jeu(x, y)
            Next
        Next
        verifierScore = Score
        deplacer(direction)
        For x = 1 To n
            For y = 1 To n
                If verifier(x, y) <> Jeu(x, y) Then
                    OK = True
                End If
            Next
        Next
        For x = 1 To n
            For y = 1 To n
                Jeu(x, y) = verifier(x, y)
            Next
        Next
        Score = verifierScore
        Return OK
    End Function
```

Fonction partie gagnée

Nous avons introduit une fonction pour alerter le joueur qu'il a gagné quand il arrive à 2048 :

```
'Cette fonction on l'utilise pour décider si le joueur a gagné c'est-à-dire arriver à 2048'

    Public Function checkwin() As Boolean
        Dim win As Boolean = False
        Dim i As Byte
        For x = 1 To n
            For y = 1 To n
                If Jeu(x, y) = 2048 Then
                    i = i + 1
                End If
            Next
        Next
        If i = 1 Then
            win = True
        End If
        Return win
    End Function
```

Fonction partie terminée

Nous avons aussi fait la fonction indiquant quand le jeu peut pas continuer, c'est-à-dire quand il est fini.

```
'Cette fonction on l'utilise pour décider si le jeu est terminé (on ne peut plus déplacer)

    Public Function JeuTerminer() As Boolean
        Dim terminer As Boolean = False
        If verifierdeplacer("up") = False And verifierdeplacer("down") = False And
        verifierdeplacer("right") = False And verifierdeplacer("left") = False Then
            terminer = True
        End If
        Return terminer
    End Function
```

Relation interface-module

Nous expliquerons maintenant comment l'interface est reliée au module, c'est-à-dire comment c'est que le jeu s'exécute.
Nous montrerons comment le jeu se déroule du point de vue du joueur et les modules qu'il appelle par rapport aux actions qu'il fait.

Définitions des variables type « public »

Nous avons commencés par définir les variables que nous allons utiliser tout au long de la feuille form en « public » :

```
Public lbl(16) As Label
Public win As Boolean = False ' Cette variable est pour évaluer la condition de
                              gagner'
Public degre As String = "simple" 'Cette variable est pour transférer le degré de
                                   difficulté choisi par le joueur'
```

Redefinition tableau

Ensuite nous avons nommé les labels que nous utilisons tous au long du jeu :

```
    'Cette procédure on l'utilise pour définir les labels'
    Private Sub definirLabel()
        lbl(1) = lbl1
        lbl(2) = lbl2
        lbl(3) = lbl3
        lbl(4) = lbl4
        lbl(5) = lbl5
        lbl(6) = lbl6
        lbl(7) = lbl7
        lbl(8) = lbl8
        lbl(9) = lbl9
        lbl(10) = lbl10
        lbl(11) = lbl11
        lbl(12) = lbl12
        lbl(13) = lbl13
        lbl(14) = lbl14
        lbl(15) = lbl15
        lbl(16) = lbl16
    End Sub
```

Relation jeu et touches du clavier

Après nous activons les touches du clavier pour que le joueur puisse avoir une interaction plus souple avec notre jeu :

```
'Cette fonction nous l'utilisons pour utiliser les touches du clavier "up", "down",
"left", "right"'
    Protected Overrides Function ProcessDialogKey(ByVal keyData As Keys) As Boolean
        If keyData = Keys.Up Or keyData = Keys.Down Or keyData = Keys.Left Or
           keyData = Keys.Right Then
            Return False
        Else
```

```
            Return MyBase.ProcessDialogKey(keyData)
        End If
    End Function

'Cette procédure sert à connecter les boutons de clavier avec les boutons de
l'interface'
    Private Sub Form2048_KeyDown(sender As Object, e As KeyEventArgs) Handles
Me.KeyDown

        'On donne la possibilité de jouer avec les touches ZQSD'
        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

        'On donne la possibilité de jouer avec les flèches du clavier'
        If e.KeyData = Keys.Up Then
            btnUp_Click(sender, e)
        End If
        If e.KeyData = Keys.Down Then
            btnDown_Click(sender, e)
        End If
        If e.KeyData = Keys.Right Then
            btnRight_Click(sender, e)
        End If
        If e.KeyData = Keys.Left Then
            btnLeft_Click(sender, e)
        End If
        GroupDegre.Enabled = True
    End Sub
```

Procédures feuille form

Après avoir défini les variables et la fonctionnalité du clavier nous montrerons l'interaction interface-module.

Chargement du jeu

On charge en premier tous les paramètres du jeu :

```
    Private Sub Form2048_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load

        Call definirLabel()
        'On active l'utilisation de clavier'

        Me.KeyPreview = True
        btnUndo.Enabled = False

        'Pour initialiser le jeu la fonction Initialiser charge paramètres initiales'
        Call Initialiser(degre)
```

```
        'La fonction Afficher permet de visualiser les paramètres initiaux du jeu'
        Call Afficher()

    End Sub
```

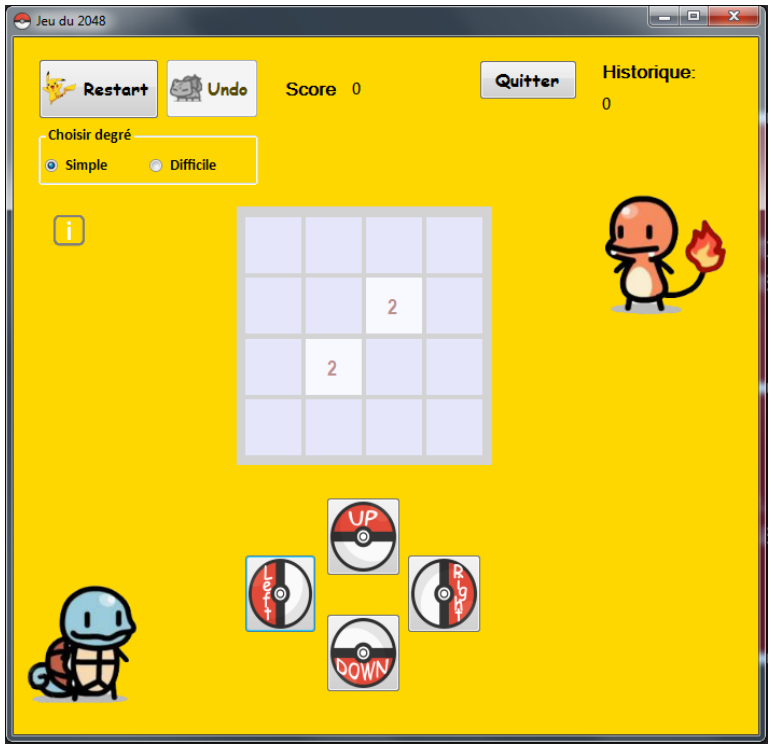
Pour rendre notre jeu plus agréable nous avons mis des différentes couleurs en fonctions du chiffre se trouvant dans la case :

```
'Cette procédure on l'a fait pour changer la couleur de la case en fonction du chiffre
qui s'affiche dans la case'

    Private Sub ChangerCouleur()
        For i = 1 To 16
            If lbl(i).Text = "" Then
                lbl(i).BackColor = Color.Lavender
            ElseIf CInt(lbl(i).Text) = 2 Then
                lbl(i).BackColor = Color.GhostWhite
                lbl(i).ForeColor = Color.RosyBrown
            ElseIf CInt(lbl(i).Text) = 4 Then
                lbl(i).BackColor = Color.Cornsilk
                lbl(i).ForeColor = Color.RosyBrown
            ElseIf CInt(lbl(i).Text) = 8 Then
                lbl(i).BackColor = Color.PeachPuff
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 16 Then
                lbl(i).BackColor = Color.LightSalmon
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 32 Then
                lbl(i).BackColor = Color.Salmon
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 64 Then
                lbl(i).BackColor = Color.Tomato
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 128 Then
                lbl(i).BackColor = Color.Khaki
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 256 Then
                lbl(i).BackColor = Color.Gold
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 512 Then
                lbl(i).BackColor = Color.Goldenrod
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 1024 Then
                lbl(i).BackColor = Color.Orange
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 2048 Then
                lbl(i).BackColor = Color.Yellow
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 4096 Then
                lbl(i).BackColor = Color.MediumAquamarine
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) = 8192 Then
                lbl(i).BackColor = Color.LightSkyBlue
                lbl(i).ForeColor = Color.White
            ElseIf CInt(lbl(i).Text) >= 16384 Then
                lbl(i).BackColor = Color.MediumPurple
                lbl(i).ForeColor = Color.White
            End If
        Next
    End Sub
```

Affichage du jeu

Pour la suivante étape nous définit une procédure qui nous permet d'afficher le jeu sur la grille :

```
    Private Sub Afficher()
'Elles vont être affichées seulement les cases avec les chiffres différents de 0 sinon rien s'affichera'

        Dim l As Byte
        Dim i As Byte
        Dim j As Byte
        For i = 1 To n
            For j = 1 To n
                l = (i - 1) * n + j
                If Jeu(i, j) <> 0 Then
                    lbl(l).Text = CStr(Jeu(i, j))
                Else
                    lbl(l).Text = ""
                End If
            Next
        Next
        'Le score et l'historique sont affichés'
        lblScore.Text = CStr(Score)
        Call historiqueScore()
        lblmaxscore.Text = CStr(Historique)

        'La procédure permettant de changer le couleur des cases est appelée'
        Call ChangerCouleur()

    End Sub
```

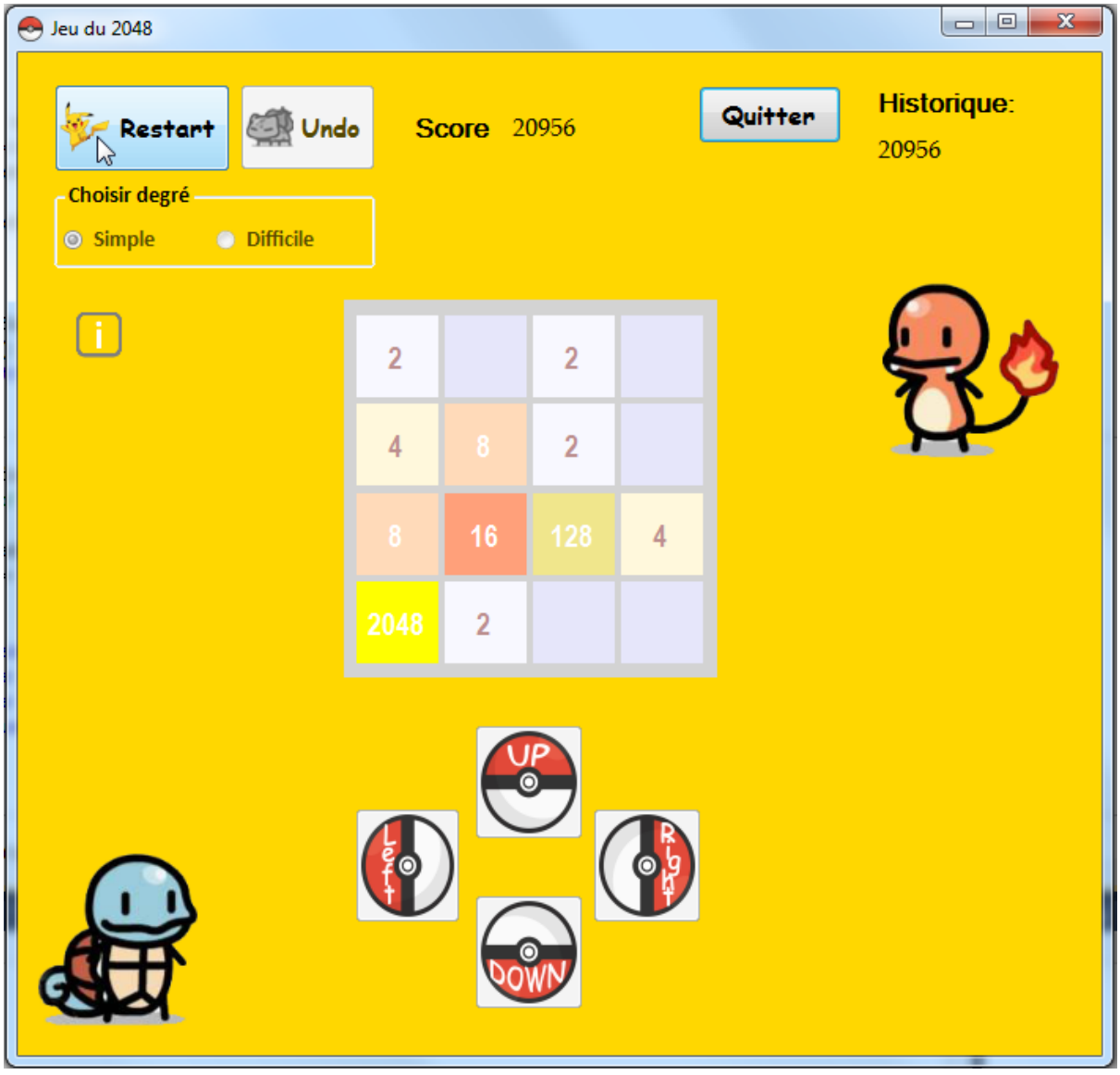


Affichage du jeu

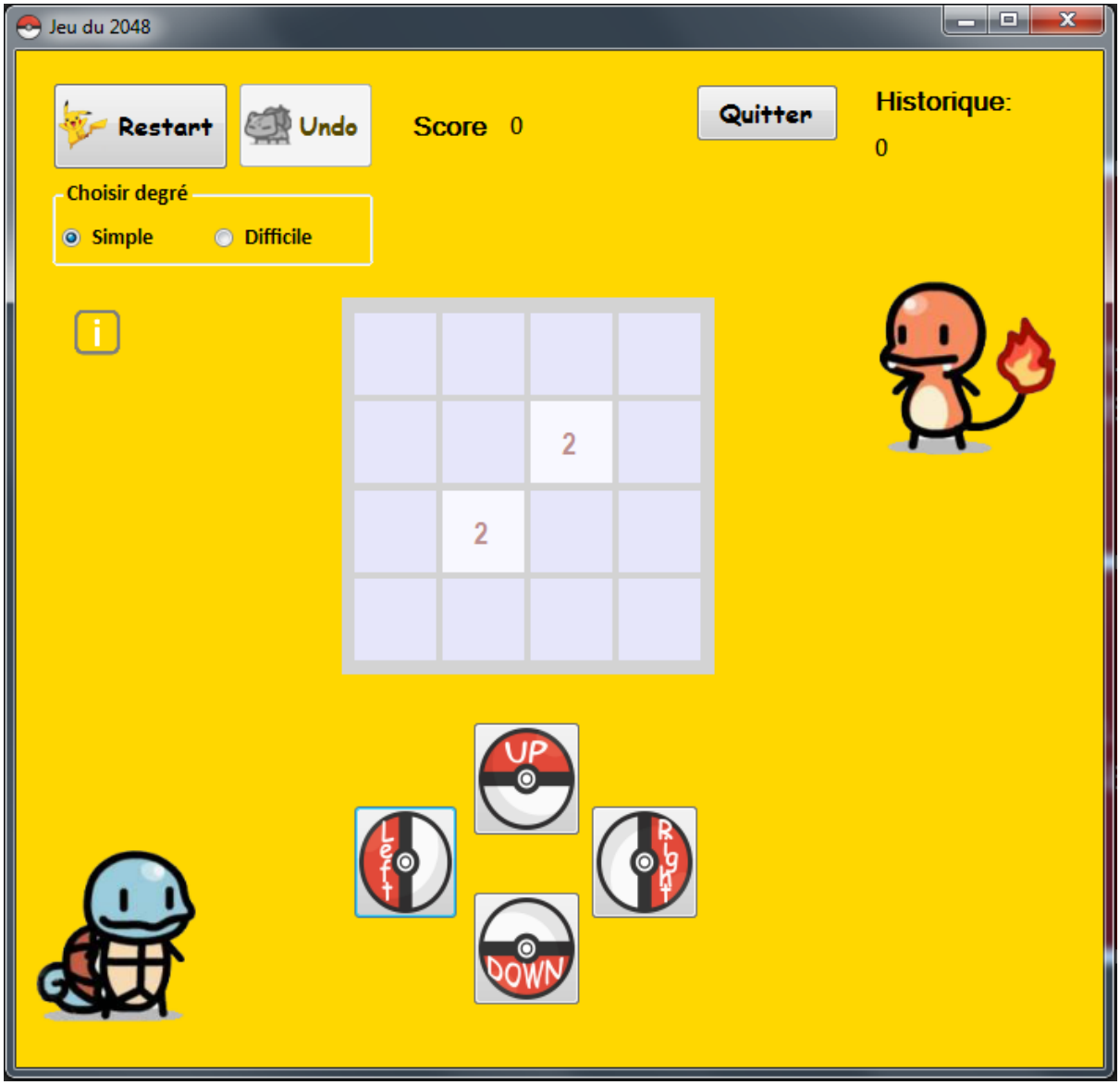
Réinitialisation du jeu

Quand le joueur clique sur le bouton restart on appelle la fonction Initialiser ainsi qu'afficher pour remettre le jeu dans l'état initial.

```
'Cette procédure permet au joueur de réinitialiser la partie'
    Private Sub btnRestart_Click(sender As System.Object, e As System.EventArgs)
Handles btnRestart.Click
        radio_difficile.Enabled = True
        radio_simple.Enabled = True
        Call Initialiser(degre)
        Call Afficher()
        btnUndo.Enabled = False
        btnDown.Enabled = True
        btnLeft.Enabled = True
        btnRight.Enabled = True
        btnUp.Enabled = True
        Me.KeyPreview = True
    End Sub
```



Jeu en cours

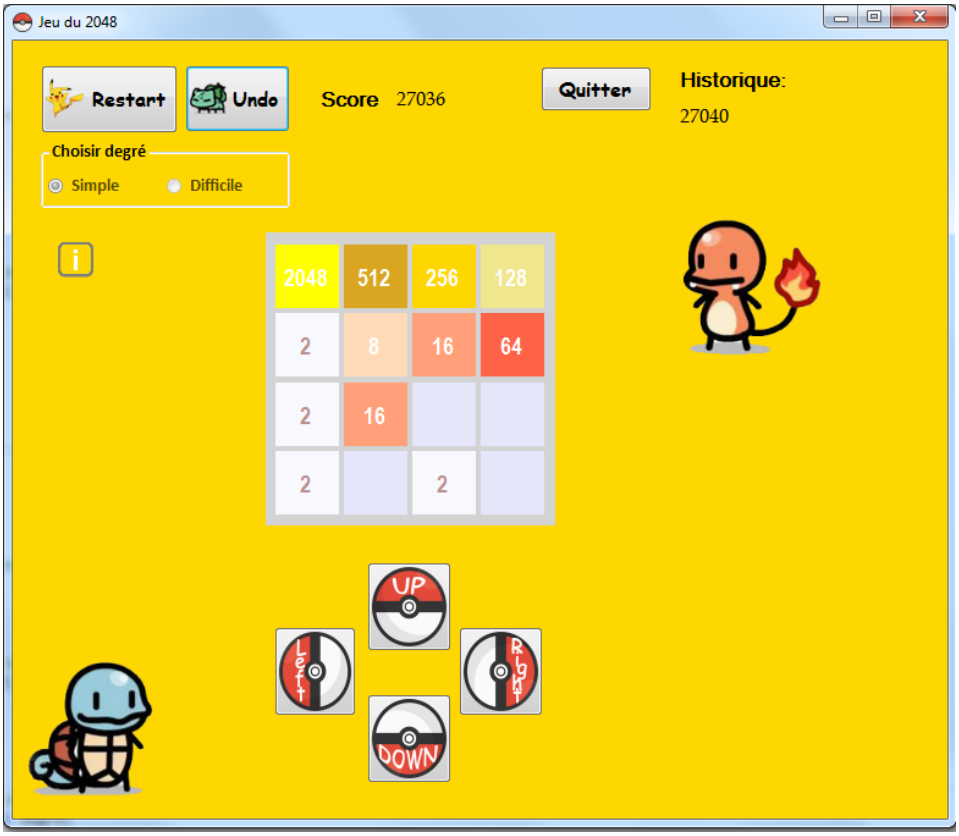


Jeu réinitialisé

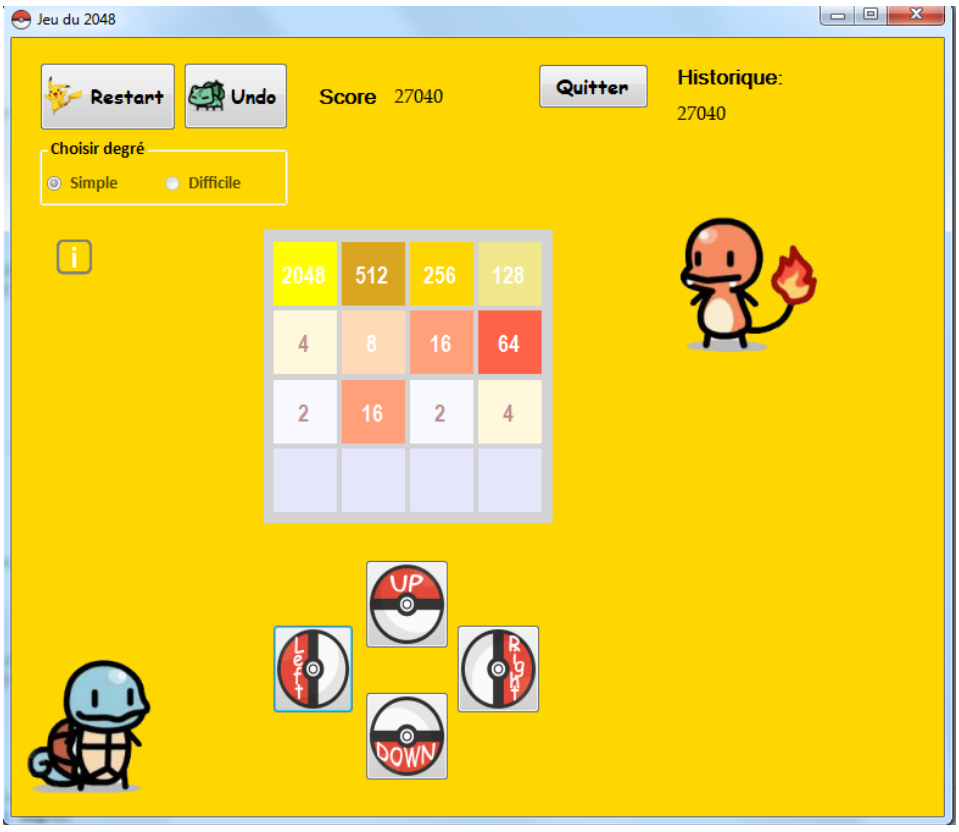
Retour en arrière

Quand le joueur clique sur le bouton Undo on retourne un pas avant, on appelle la procédure undoJeu puis on l'affiche.

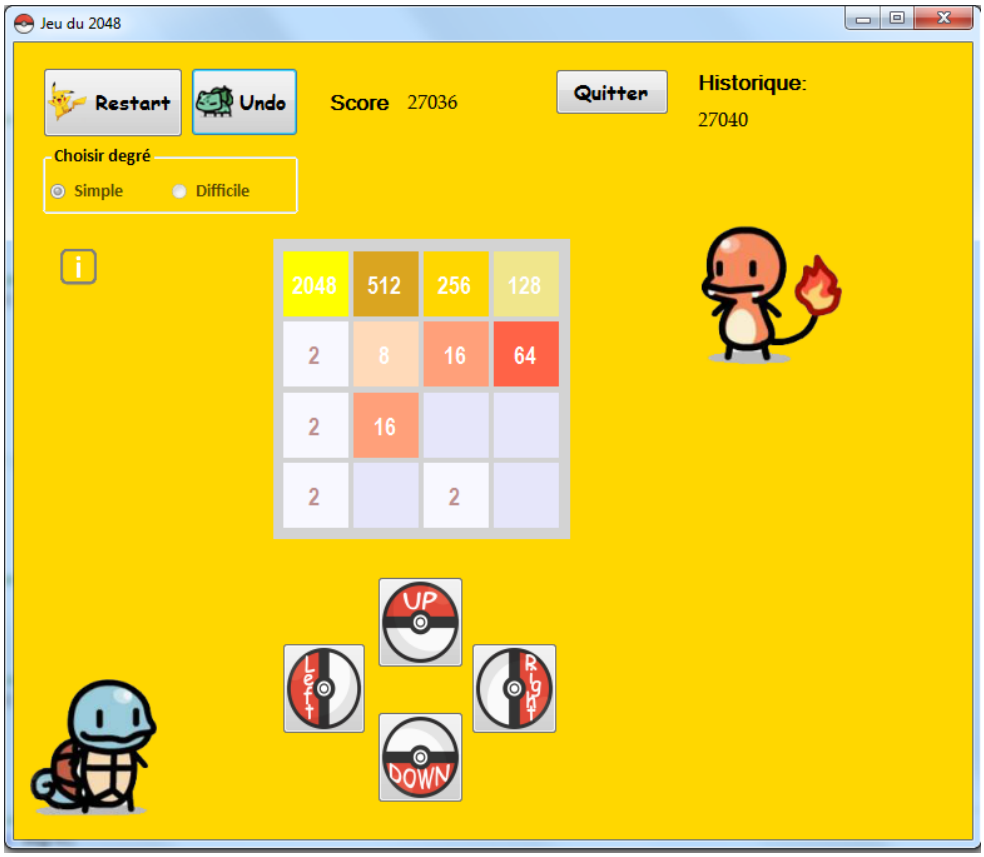
```
'Cette procédure permet d'appeler la procédure undoJeu du module'
    Private Sub btnUndo_Click(sender As System.Object, e As System.EventArgs) Handles
btnUndo.Click

        'Le dernier coup peut être rejoué'
        Call undoJeu()

        'Le dernier coup est affiché'
        Call Afficher()

    End Sub
```



Jeu à l'étape A



Jeu à l'étape B



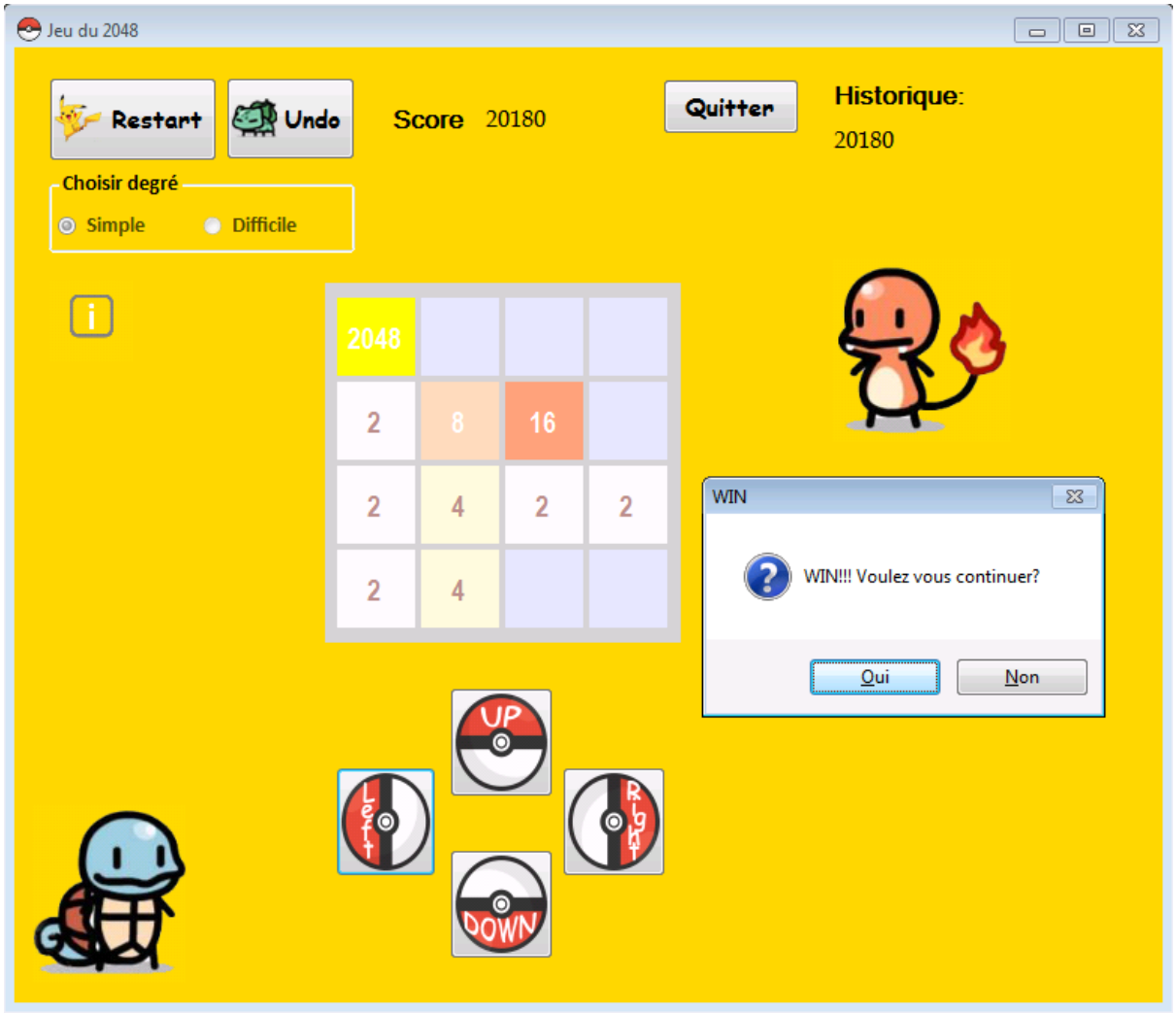
Retour à l'étape B

Finalisation du jeu

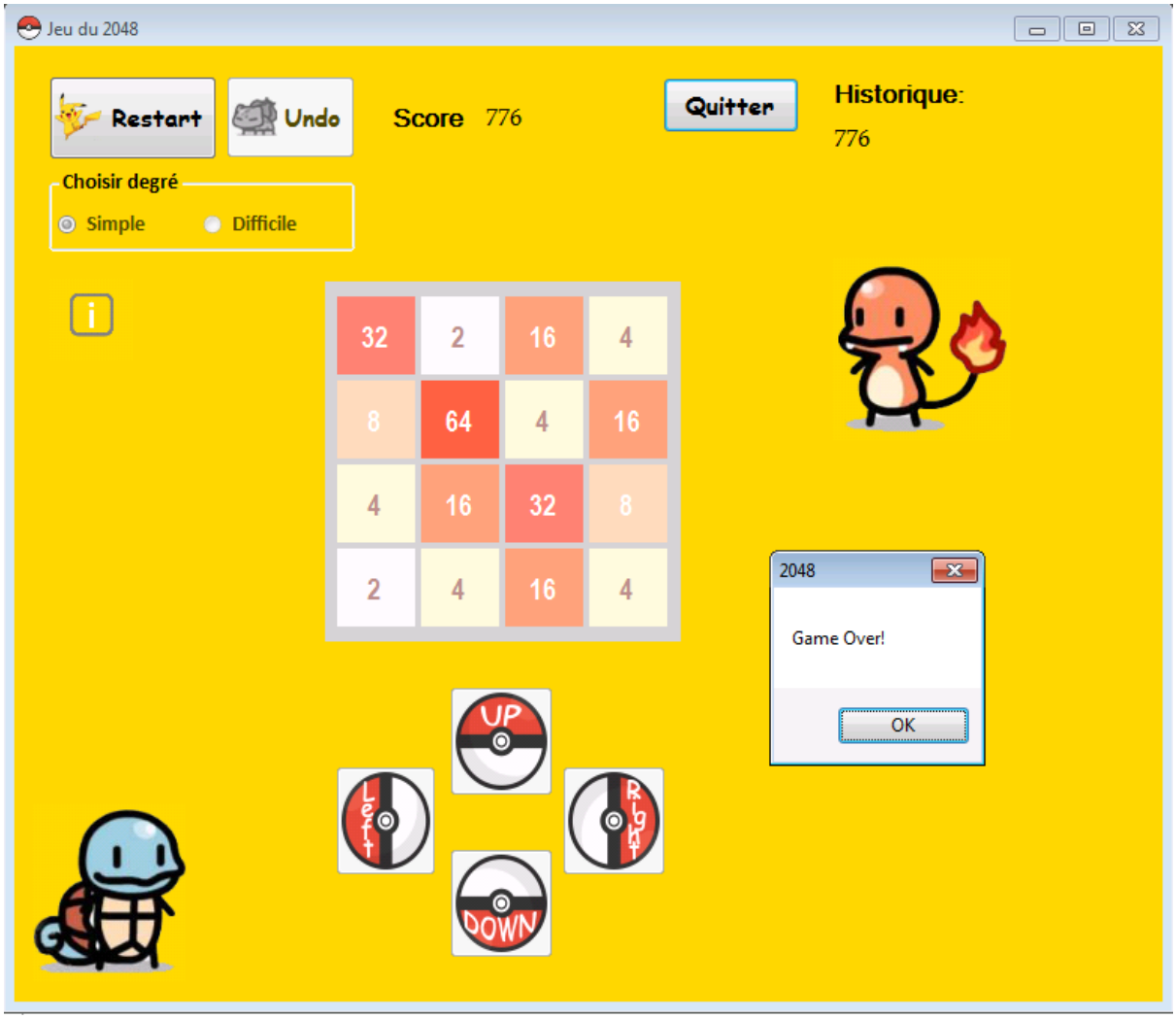
D'après les règles du jeu une partie est finie soit quand le joueur atteint un score de 2048 soit quand il est incapable de faire plus de mouvements :

```
'Cette procédure on l'utilise pour désactiver les boutons sur interface et les boutons de clavier quand le joueur décide de terminer le jeu après avoir gagné ou quand le jeu est terminé, c'est-à-dire le joueur ne peut plus déplacer'

    Private Sub Terminer()

        'Cette partie est pour désactiver les boutons quand le joueur ne veut pas continuer à jouer après arriver à 2048'

        If win = False Then
            win = checkwin()
            If win = True Then
                If MsgBox("WIN!!! Voulez vous continuer? ", vbYesNo + vbQuestion,
                          "WIN") = MsgBoxResult.No Then
                    btnUndo.Enabled = False
                    btnDown.Enabled = False
                    btnLeft.Enabled = False
                    btnRight.Enabled = False
                    btnUp.Enabled = False
                    Me.KeyPreview = False
                End If
```

```
            End If
        End If

        'Cette partie est pour désactiver les boutons quand le jeu est terminé'
        If JeuTerminer() = True Then
            btnUndo.Enabled = False
            btnDown.Enabled = False
            btnLeft.Enabled = False
            btnRight.Enabled = False
            btnUp.Enabled = False
            Me.KeyPreview = False
            MsgBox("Game Over!")
        End If
    End Sub
```



Partie gagnée



Partie perdue

Mouvements avec la souris

Nous avons décidé de donner au joueur la possibilité d'utiliser différents moyens pour jouer. Le premier c'est grâce aux touches du clavier et le deuxième c'est avec des boutons que nous avons placé sur l'interface.

```
    Private Sub btnLeft_Click(sender As System.Object, e As System.EventArgs) Handles
btnLeft.Click
        'gère le déplacement à gauche'

        'D'abord on vérifie si le joueur peut déplacer à gauche'
        Dim OK As Boolean = verifierdeplacer("left")

        'On active le bouton Undo chaque fois que le joueur déplace les cases'
        btnUndo.Enabled = True

        'Une fois le joueur déplace, on ne peut pas changer le degré de difficulté'
        radio_difficile.Enabled = False
        radio_simple.Enabled = False
```

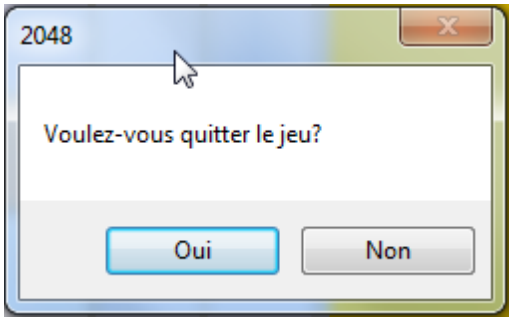
```
        'Si le joueur peut déplacer à gauche, nous enregistrons le tableau Undo pour
que nous puissions retourner à l'étape dernière'
        Nous calculons pour le déplacement puis nous affichons la nouvelle valeur dans
une nouvelle case aléatoirement'

        If OK = True Then
            Call enregistrerUndo()
            Call deplacer("left")
            Call TirerAleatoirement(degre)
            Call Afficher()
        End If

        'Vérifier si le jeu termine ou pas'
        Terminer()
    End Sub

    Private Sub btnRight_Click(sender As System.Object, e As System.EventArgs) Handles
btnRight.Click
        ' gère le déplacement à droite'

        Dim OK As Boolean = verifierdeplacer("right")
        btnUndo.Enabled = True
        radio_difficile.Enabled = False
        radio_simple.Enabled = False
        If OK = True Then
            Call enregistrerUndo()
            Call deplacer("right")
            Call TirerAleatoirement(degre)
            Call Afficher()
        End If
        Terminer()
    End Sub

    Private Sub btnUp_Click(sender As System.Object, e As System.EventArgs) Handles
btnUp.Click
        ' gère le déplacement vers le haut'

        Dim OK As Boolean = verifierdeplacer("up")
        btnUndo.Enabled = True
        radio_difficile.Enabled = False
        radio_simple.Enabled = False
        If OK = True Then
            Call enregistrerUndo()
            Call deplacer("up")
            Call TirerAleatoirement(degre)
            Call Afficher()
        End If
        Terminer()
    End Sub

    Private Sub btnDown_Click(sender As System.Object, e As System.EventArgs) Handles
btnDown.Click
        ' gère le déplacement vers le bas'

        Dim OK As Boolean = verifierdeplacer("down")
        btnUndo.Enabled = True
        radio_difficile.Enabled = False
        radio_simple.Enabled = False
        If OK = True Then
            Call enregistrerUndo()
            Call deplacer("down")
```

```
            Call TirerAleatoirement(degre)
            Call Afficher()
        End If
        Terminer()
    End Sub
```

Abandonner le jeu

Nous avons mis la possibilité que le joueur puisse abandonner une partie `n'importe quel moment de la partie.

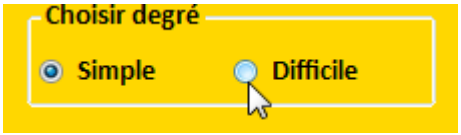
```
    Private Sub btn_quitter_Click(sender As System.Object, e As System.EventArgs)
Handles btn_quitter.Click
        'Cette procédure on l'utilise pour fermer la fenêtre quand le joueur clique
sur le bouton Quitter'
        If MsgBox("Voulez-vous quitter le jeu?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes
Then
            Me.Close()
        End If
    End Sub
```



Niveau de difficulté

Le niveau de difficulté peut être choisit grâce à deux radio boutons, chaque radio bouton appelle la variable degré.

```
    Private Sub radio_simple_CheckedChanged(sender As System.Object, e As
System.EventArgs) Handles radio_simple.CheckedChanged
        'Quand le joueur choisit simple en cliquant sur le radio bouton simple, on
donne à la variable degré un valeur texte simple'
        degre = "simple"
    End Sub

    Private Sub radio_difficile_CheckedChanged(sender As System.Object, e As
System.EventArgs) Handles radio_difficile.CheckedChanged
        'Quand le joueur choisit difficile en cliquant sur le radio bouton difficile,
on donne à la variable degre un valeur texte difficile
        degre = "difficile"
    End Sub
```



Icône information

Le joueur peut cliquer sur le symbole informations et il peut lire des instructions par rapport à la jouabilité.

```
Private Sub PictureBox3_Click(sender As System.Object, e As System.EventArgs) Handles
PictureBox3.Click
        'On affiche un mode d'emplois en cliquant sur le bouton i'
        MsgBox("Vous pouvez jouer avec les Pokéballs au-dessous ou avec le clavier
tapant en ""ZSQD"" ou ""up, down, left, right""")
    End Sub
```