



PROJET DE PROGRAMMATION STRUCTUREE

12 /2017

Sommaire

II. INTRODUCTION	2
- Règles du jeu :	2
- Mise en place du jeu :	2
III. INTERFACE	2
IV. MODULE	5
- Variables globales :	5
- Les Fonctions :	5
V. RELATION INTERFACE-MODULE (FORM)	9

II. Introduction

Pour la réalisation de notre projet on a utilisé les notions vues en cours, en TP mais aussi on a cherché sur Internet des informations permettant de réussir.

L'idée de ce dossier est de présenter le sujet et de quelle façon on l'a abordé, ayant comme résultat final le jeu que vous avez eu le privilège de tester.

- Règles du jeu :
Le jeu de Kakuha consiste à prendre les graines d'une case et les répartir, sauf une graine qu'il va conserver, dans les cases suivantes, toujours dans le sens inverse des aiguilles d'une montre.
Dans un même tour, le joueur peut jouer et prendre les graines des différentes cases tant qu'il arrive dans une case où il y a des graines. Dès qu'il tombe dans une case vide, c'est à l'autre joueur de jouer.
- Mise en place du jeu :
Pour mettre en place le jeu de Kakuha, on a créé une interface permettant sur Visual Studio avec 6 cases pour chaque joueur.
Le jeu se déroule grâce un code que nous avons construit et qu'on va le présenter dans la suite de ce dossier.

III. Interface

On commence l'explication du travail fourni par l'interface.

On a considéré de faire en premier l'interface pour avoir déjà une idée claire du produit final qu'on veut montrer.

On doit faire la différence entre les éléments de bases demandés par le sujet et les éléments qu'on a rajouté pour rendre l'interface plus amicale.

Comme c'est logique, on a mis 6 cases pour chaque joueur c'est-à-dire 12 en total, ces cases sont représentées par des boutons car on cherche l'interaction de l'utilisateur avec le jeu.

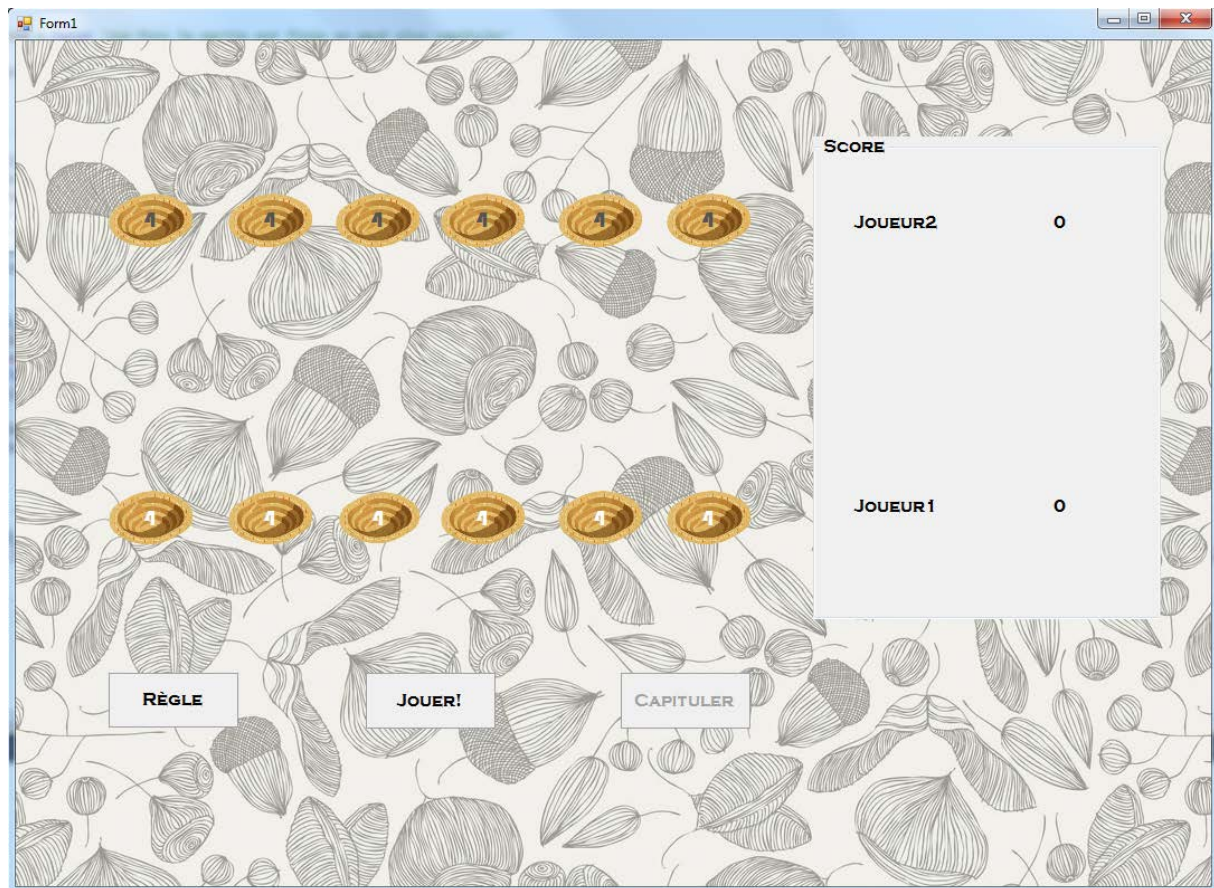
Ensuite on a créé un bouton « Jouer ! » permettant aux joueurs d'initialiser la partie.

Finalement on a rajouté un groupbox « Score » regroupant 4 labels, 2 labels indiquant le numéro de joueur et 2 labels pour afficher les scores des joueurs.

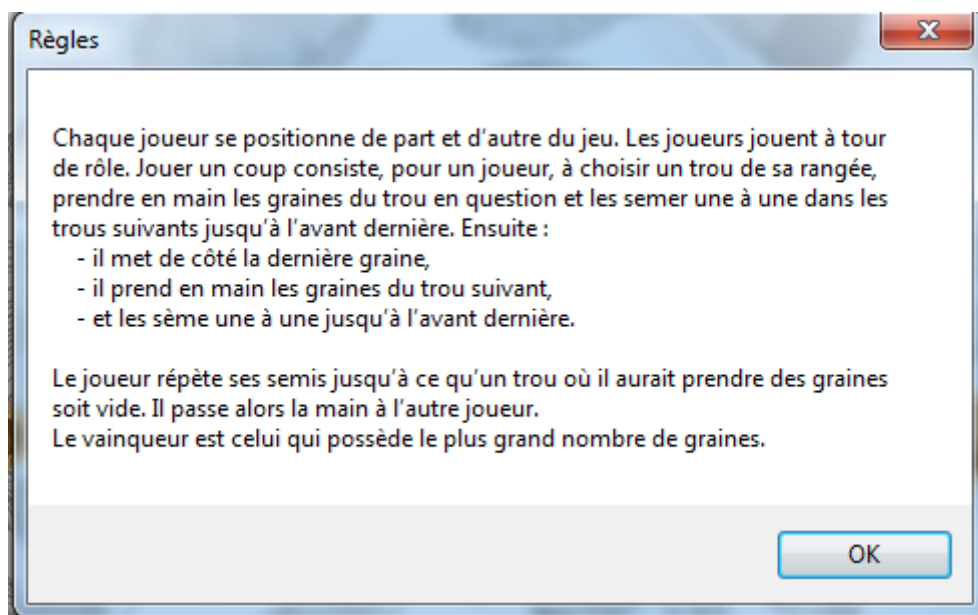
On a rajouté des éléments pour faire notre application plus intéressante et amicale pour l'utilisateur.

Concrètement on a mis deux boutons supplémentaires, « Règle » et « Capituler ».

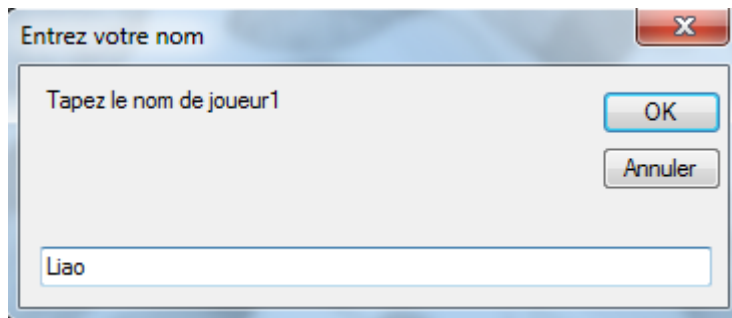
Ici on montre notre interface avant d'initialiser le jeu.



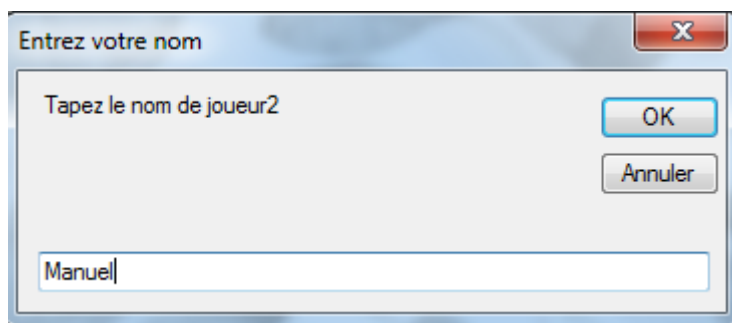
Avant de commencer, si les joueurs souhaitent lire les règles, ils peuvent cliquer sur le bouton « Règle » et les règles s'afficheront dans une nouvelle fenêtre. On peut accéder aux règles n'importe à quel moment de la partie.



Lors de l'initialisation du jeu, quand le joueur clique sur « Jouer ! » il est demandé d'insérer les noms des joueurs.

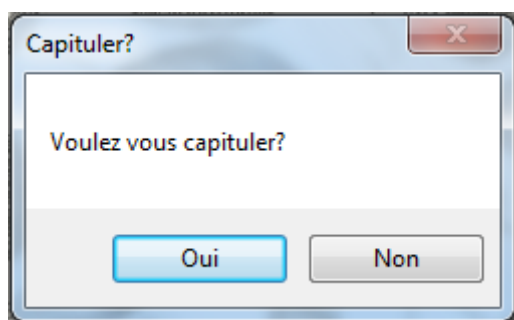


A dialog box titled "Entrez votre nom" with a close button (X) in the top right corner. It contains a text input field with the placeholder "Tapez le nom de joueur1". Below the input field are two buttons: "OK" and "Annuler". The input field contains the text "Liao".



A dialog box titled "Entrez votre nom" with a close button (X) in the top right corner. It contains a text input field with the placeholder "Tapez le nom de joueur2". Below the input field are two buttons: "OK" and "Annuler". The input field contains the text "Manuel".

Si un des joueurs souhaite abandonner la partie il peut appuyer sur le bouton « Capituler », un message s'affichera pour savoir s'il veut réellement quitter la partie.



A dialog box titled "Capituler?" with a close button (X) in the top right corner. It contains a text input field with the placeholder "Voulez vous capituler?". Below the input field are two buttons: "Oui" and "Non".

A la fin de la partie, si les joueurs veulent rejouer une partie, ils peuvent cliquer sur le bouton « Rejouer ».



IV. Module

Sur notre module on a défini les variables globales, c'est-à-dire celles, qu'on va utiliser toujours durant l'exécution du programme. Et on a ensuite écrit nos fonctions qu'utilisent ces variables.

- Variables globales :

Dans notre module on a défini les variables suivantes :

- Trou(12), le tableau comportant les 12 cases du jeu, 6 pour chaque joueur
- Graines, représentant les graines que les joueurs utilisent durant leur partie
- Joueur, représentant la personne jouant une partie

- Les Fonctions :

1. La première fonction qu'on a créée est « initialise jeu »

```
Public Sub InitialiserJeu()
    For i = 1 To 12
        trou(i) = 4

    Next i

End Sub
```

La variable i représente le numéro de la case du tableau trou(12).

On applique la boucle «for » pour prendre toutes les cases, de 1 à 12, et leur donner la même valeur, 4.

C'est comme ça qu'on souhaite initialiser le jeu.

2. La deuxième fonction qu'on utilise est « jouer »

```
Public Sub Jouer(ByVal i As Integer) 'i est le numéro de la case'
    Dim j As Integer 'j c'est la case ciblé, où la graine est destinée'
    graines = trou(i)
    trou(i) = 0 'On veut que le trou(i) = 0 après avoir pris toutes les graines'
    If i + graines - 1 <= 12 Then 'On peut jouer dans les 12 cases sans retourner
à la ligne'
        For j = i + 1 To i + graines - 1 'On explique ici quelle est la première
case dont on distribue jusqu'à la dernière case où on dépose les graines'
            trou(j) = trou(j) + 1
        Next
    Else
        'A partir de ce point, on considère la possibilité de retourner à la
ligne'
        If i + graines - 1 <= 24 Then 'Il existe ensuite la possibilité qu'en
fonction des graines dans une case, on puisse retourner à la ligne'
            For j = i + 1 To 12 'distribution de la première case où on est censé
jusqu'à la numéro 12'
                trou(j) = trou(j) + 1
            Next
            For j = 1 To (graines - 1) - (12 - i) 'On fait le retour et on
distribue les graines restantes'
                trou(j) = trou(j) + 1
            Next
        Else
```

```

        'Deux retours à la ligne'
        For j = i + 1 To 12 'distribution de la première case où on est censé
jusqu'à la numéro 12'
            trou(j) = trou(j) + 1
        Next
        For j = 1 To 12 'distribution de la première case où on est censé
jusqu'à la numéro 12'
            trou(j) = trou(j) + 1
        Next
        For j = 1 To (graines - 1) - 12 - (12 - i) 'On fait le retour et on
distribue les graines restantes'
            trou(j) = trou(j) + 1
        Next
    End If
End If

End Sub

```

Dans cette fonction on détermine la variable « i » que représente le numéro de la case et la variable « j » que représente la case cible, c'est-à-dire la case où l'avant-dernière graine est destinée car la dernière graine est gardée par le joueur dans son score.

Cette fonction a pour objectif de réaliser la distribution des graines en fonction du trou(i) et du nombre de graines. Aussi elle oblige aux joueurs de prendre toujours la case j+1 pour les pas intermédiaires.

On a fait la distinction de quand les joueurs font un coup entre les 12 premières cases et quand ils doivent retourner à la ligne, c'est-à-dire quand le nombre de graines est tel qu'il passe la case 12 et doit continuer dans la case 1.

On a rajouté la possibilité que si le joueur a un grand suffisant de graines, il puisse faire le tour deux fois, c'est-à-dire passer par la case 1 deux fois dans un seul coup.

3. La troisième fonction est « PuetJouer »

```

Public Function PeutJouer(ByVal i As Integer) As Boolean
    Dim continuer As Boolean = True
    Dim j As Byte 'c'est la case cible'
    If i + graines - 1 <= 11 Then 'dans cette ligne on veut toujours comparer la
avant-dernière case, de la case cible'
        j = i + graines
        If trou(j) = 0 Then
            continuer = False
            If joueur = 1 Then
                'Dans ces lignes, on considère la possibilité qu'un des deux
joueurs a tous ces trous vides donc si l'autre joueur tombe dans une case vide, il
peut rejouer '
                If Not (trou(7) = 0 And trou(8) = 0 And trou(9) = 0 And trou(10) =
0 And trou(11) = 0 And trou(12) = 0) Then
                    joueur = 3 - joueur
                End If
            Else
                If Not (trou(1) = 0 And trou(2) = 0 And trou(3) = 0 And trou(4) =
0 And trou(5) = 0 And trou(6) = 0) Then
                    joueur = 3 - joueur
                End If
            End If
        End If
    Else

```

```

    'Idem, mais on considère aussi la possibilité que le joueur fasse un
    retour à la ligne'
    If i + graines - 1 <= 23 Then
        j = graines - (12 - i)
        If trou(j) = 0 Then
            continuer = False
            If joueur = 1 Then
                If Not (trou(7) = 0 And trou(8) = 0 And trou(9) = 0 And
trou(10) = 0 And trou(11) = 0 And trou(12) = 0) Then
                    joueur = 3 - joueur
                End If
            Else
                If Not (trou(1) = 0 And trou(2) = 0 And trou(3) = 0 And
trou(4) = 0 And trou(5) = 0 And trou(6) = 0) Then
                    joueur = 3 - joueur
                End If
            End If
        End If
    Else
        'Ici c'est la même chose mais avec deux retours à la ligne'
        j = graines - 12 - (12 - i)
        If trou(j) = 0 Then
            continuer = False
            If joueur = 1 Then
                If Not (trou(7) = 0 And trou(8) = 0 And trou(9) = 0 And
trou(10) = 0 And trou(11) = 0 And trou(12) = 0) Then
                    joueur = 3 - joueur
                End If
            Else
                If Not (trou(1) = 0 And trou(2) = 0 And trou(3) = 0 And
trou(4) = 0 And trou(5) = 0 And trou(6) = 0) Then
                    joueur = 3 - joueur
                End If
            End If
        End If
    End If

    Return continuer
End Function

```

Dans cette fonction on définit les variables « i » comme le numéro de la case, j comme la case cible et « continuer », variable logique, comme la variable permettant de continuer la main ou non.

Cette fonction a pour objectif de permettre aux joueurs de faire leurs mouvements que dans les cases où il y a encore de graines, sinon ils ne peuvent pas continuer avec leur coup. On a rajouté la possibilité que si un joueur a tous ces cases vides quand son tour arrive, alors c'est l'autre joueur qui joue encore.

4. Fonction « JeuTermine »

```
Public Function JeuTermine(ByVal score1 As Byte, score2 As Byte) As Boolean
```



```
    Dim termine As Boolean = False
    If score1 > 24 Or score2 > 24 Or (score1 = 24 And score2 = 24) Then 'Comme il
y a 48 graines, le premier joueur à atteindre la moitié +1 gagne automatiquement, on
considère aussi la possibilité d'un match nul'
        termine = True
    End If
    Return termine
End Function
```

Dans cette fonction on définit les variables « score1 » et « score2 » comme les scores correspondant aux joueurs et « termine », variable logique, déterminant quand les joueurs ne peuvent plus faire des mouvements.

On a défini qu'une partie est finie qu'en deux situations possibles.

La première c'est quand un des deux joueurs atteint plus de 24 graines dans leur score.

Et la deuxième possibilité c'est quand il y a une égalité, c'est-à-dire quand les deux joueurs arrivent à 24 graines chacun.

V. Relation Interface-Module (Form)

On va maintenant expliquer comment c'est que l'interface est reliée au module, c'est-à-dire comment fait-on pour exécuter le jeu.

On explique que doit faire le joueur dans l'interface pour activer le module et quels sont les effets visibles sur l'interface.

1. Le premier pas qu'on a fait, c'est définir les tableaux qu'on utilise, on a pris des tableaux parce que on définit un nombre fixe de joueurs, de scores, de boutons et de noms de joueurs.

'On définit les tableaux qu'on va utiliser pendant toute l'exécution'

```
Public button(12) As Button
Public score(2) As Label
Public numjoueur(2) As Label
Public nom(2) As Label
```

2. Ensuite on a défini une procédure

Public Sub dimcontrol() 'regroupe les différents tableaux et leurs connexion à l'interface'

```
    button(1) = btn1
    button(2) = btn2
    button(3) = btn3
    button(4) = btn4
    button(5) = btn5
    button(6) = btn6
    button(7) = btn7
    button(8) = btn8
    button(9) = btn9
    button(10) = btn10
    button(11) = btn11
    button(12) = btn12
```

```
    score(1) = lb3
    score(2) = lb4
```

```
    numjoueur(1) = lb1
    numjoueur(2) = lb2
```

```
    indique(1) = lb5
    indique(2) = lb6
```

```
    nom(1) = lblnom1
    nom(2) = lblnom2
```

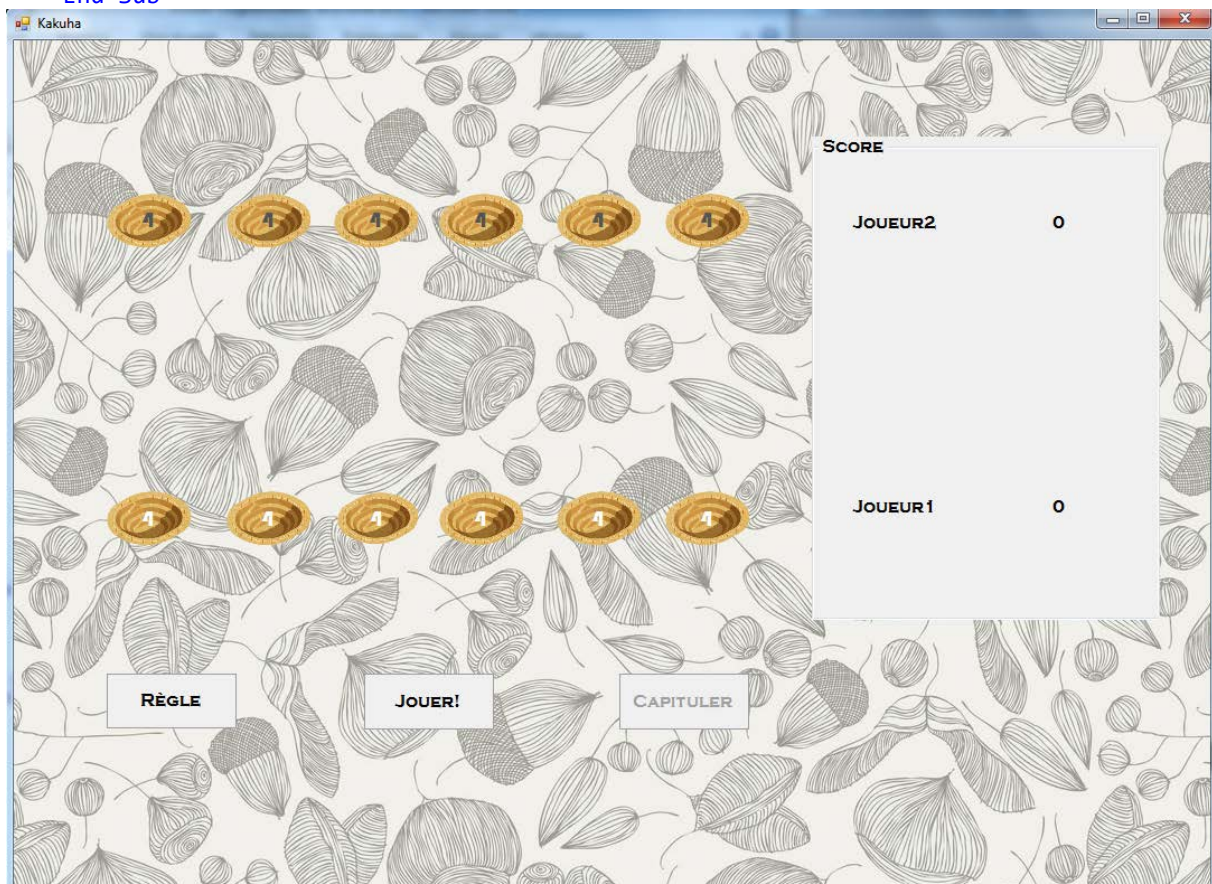
```
End Sub
```

3. On a créé cette procédure permettant d'initialiser le jeu car elle appelle notre fonction InitialiseJeu() vu auparavant.

```
Public Sub InitialiseBouton() 'Pour mettre tous les paramètre souhaités avant le
premier tour'
    Call InitialiseJeu() 'On appelle la fonction Initialisejeu() pour créer la
table trou(i)'
    For i = 1 To 12
        button(i).Enabled = False
    Next
    For i = 1 To 12

        button(i).Text = trou(i) 'tous les cases au début du jeu on 4 graines'
    Next i

    score(1).Text = 0 'On initialise les scores des joueurs à 0'
    score(2).Text = 0
    joueur = 1
    numjoueur(joueur).Text = "Joueur" & joueur 'On affiche le numéro du joueur'
    indique(joueur).Text = ">>>>"
    numjoueur(3 - joueur).Text = "Joueur" & (3 - joueur)
    indique(3 - joueur).Text = ""
    For j = 1 To 6
        button(j).Enabled = True 'Les boutons sont actives'
        button(j).ForeColor = Color.White
    Next
    For j = 7 To 12
        button(j).Enabled = False 'Les boutons sont désactivés'
        button(j).ForeColor = Color.White
    Next
End Sub
```



Comme c'est montré dans l'image, quand on initialise le jeu c'est toujours le joueur 1 qui commence la partie, c'est pour cela que ces cases affichent le nombre de graines en blanc. On bloque les cases du joueur 2 et on met le nombre de de graines dans une couleur différente.

On peut observer les scores de joueurs et leurs noms.

Finalement on voit les trois boutons, « Règle », « Jouer ! » et « Capituler ». Comme la partie n'est pas encore commencée, « Capituler » est désactivé et « Jouer ! » est encore active.

4. On a créé une procédure pour bloquer les cases où le joueur ne peut pas cliquer durant le coup, comme ça on lui montre la séquence logique pas à pas.
On a défini les variable « termine » et « continuer » comme de variables logiques représentant l'état d'un coup, puisque on veut donner la possibilité aux joueurs de regarder comment ces graines sont distribuées.
Cette procédure oblige aussi au joueur qui va commencer son coup de jouer uniquement que dans les cases que lui appartiennent.
Finalement, on cherche que quand une partie est fini toutes les cases soient désactivées, également pour le bouton « Capituler » car la partie est déjà fini. Au même temps on active le bouton « Rejouer » pour recommencer une nouvelle partie.

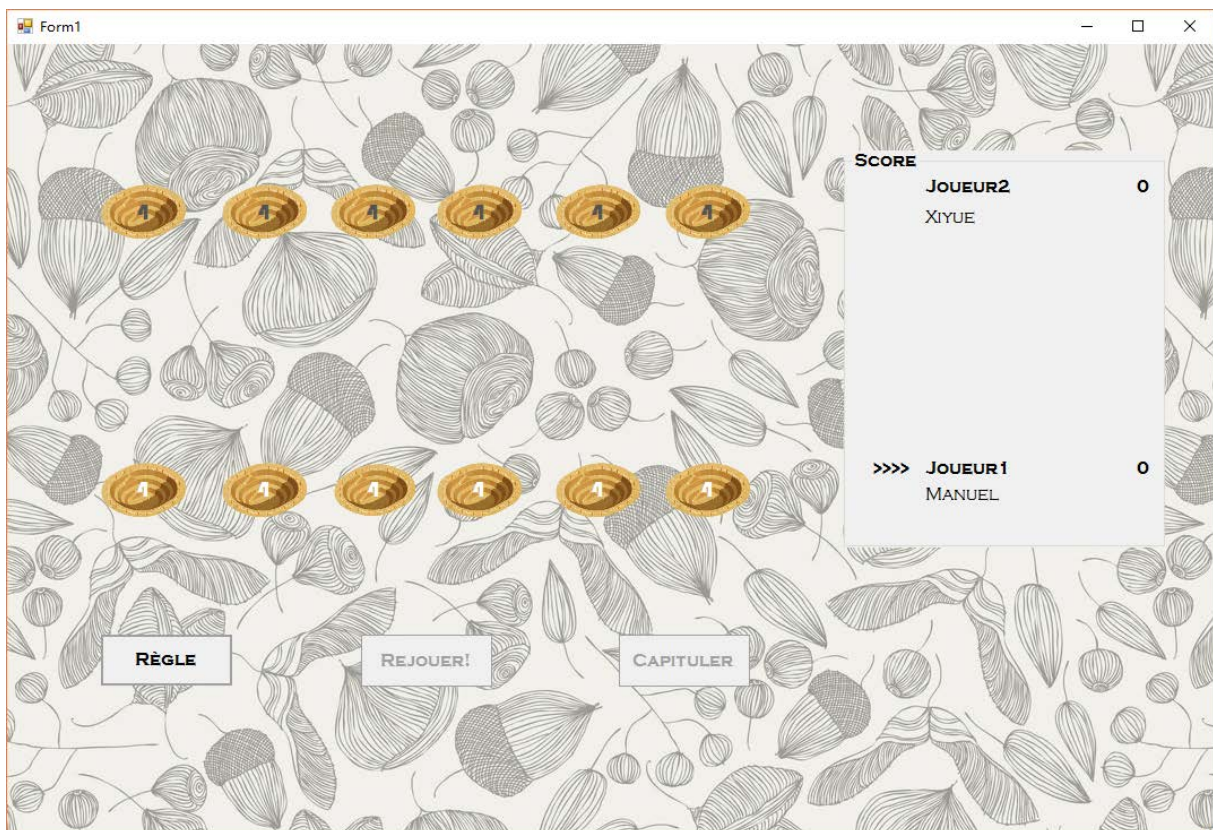
```
Public Sub validtrou(ByVal termine As Boolean, ByVal continuer As Boolean, ByVal i As Integer) 'Pour montrer quelles sont les cases dans lesquelles on peut pas cliquer'
    If termine = False Then
        If continuer = True Then
            For j = 1 To 12
                button(j).Enabled = False 'Ici on boucle toutes les cases'
            Next
            'Mais on laisse activée la case dont on doit cliquer'
            If i + graines - 1 <= 11 Then
                button(i + graines).Enabled = True
                button(i + graines).ForeColor = Color.Blue
            Else
                If i + graines - 1 <= 23 Then
                    button(graines - (12 - i)).Enabled = True
                    button(graines - (12 - i)).ForeColor = Color.Blue
                Else
                    button(graines - 12 - (12 - i)).Enabled = True
                    button(graines - 12 - (12 - i)).ForeColor = Color.Blue
                End If
            End If
        End If
    Else
        'Quand le coup est fini on veut que le joueur ayant la main puisse jouer que dans ces cases'
        If joueur = 1 Then
            For j = 1 To 6
                If button(j).Text <> 0 Then
                    button(j).Enabled = True
                    button(j).ForeColor = Color.White
                Else
                    button(j).Enabled = False
                End If
            Next
            For j = 7 To 12
                button(j).Enabled = False
            Next
        Else
            For j = 1 To 6
                button(j).Enabled = False
            Next
        End If
    End If
End Sub
```

```

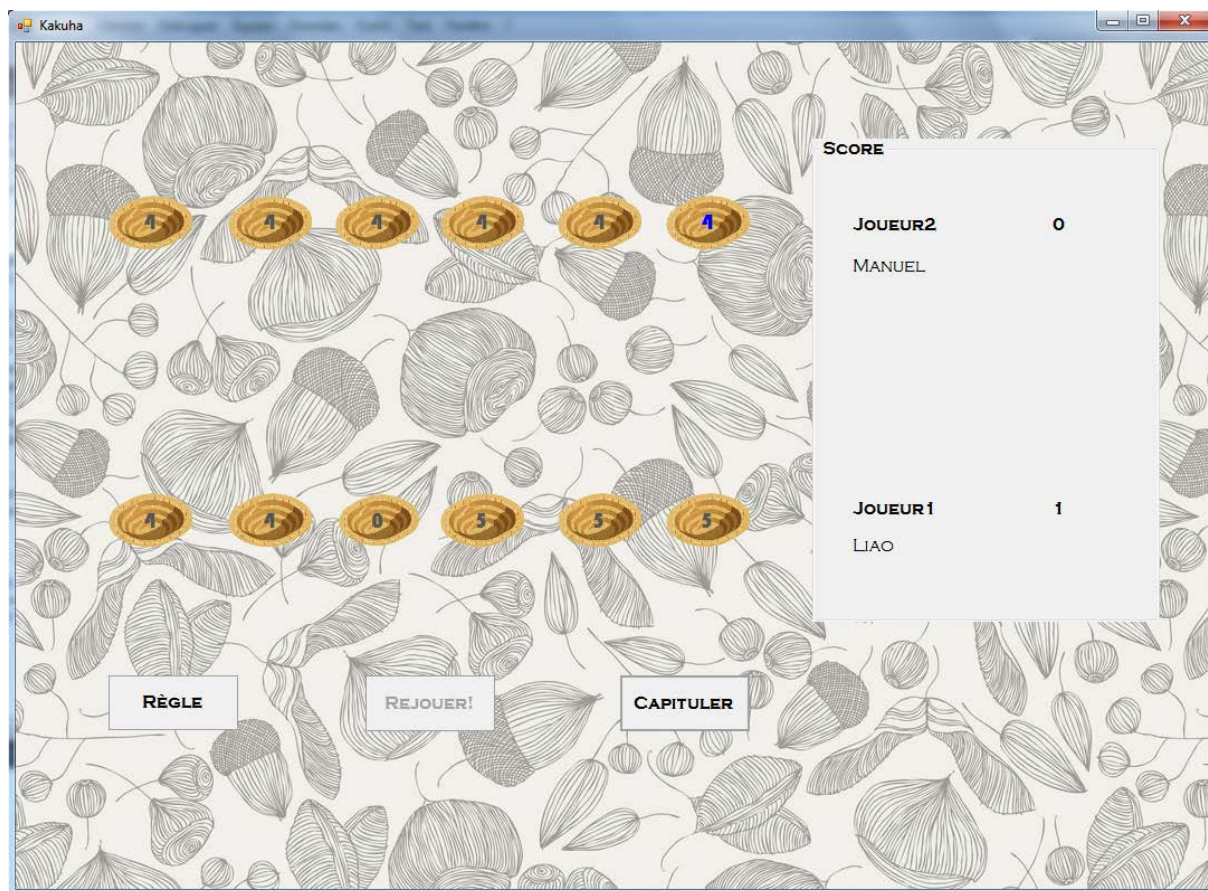
Next
For j = 7 To 12
    If button(j).Text <> 0 Then
        button(j).Enabled = True
        button(j).ForeColor = Color.White
    Else
        button(j).Enabled = False
    End If
Next
End If
End If
Else
    For j = 1 To 12
        button(j).Enabled = False 'Quand le jeu est fini, les cases sont
bloquées'
    Next
    btn13.Enabled = True 'Rejouer est activé'
    btn14.Enabled = False 'Capituler est désactivé'
End If
End Sub

```

Avant jouer:

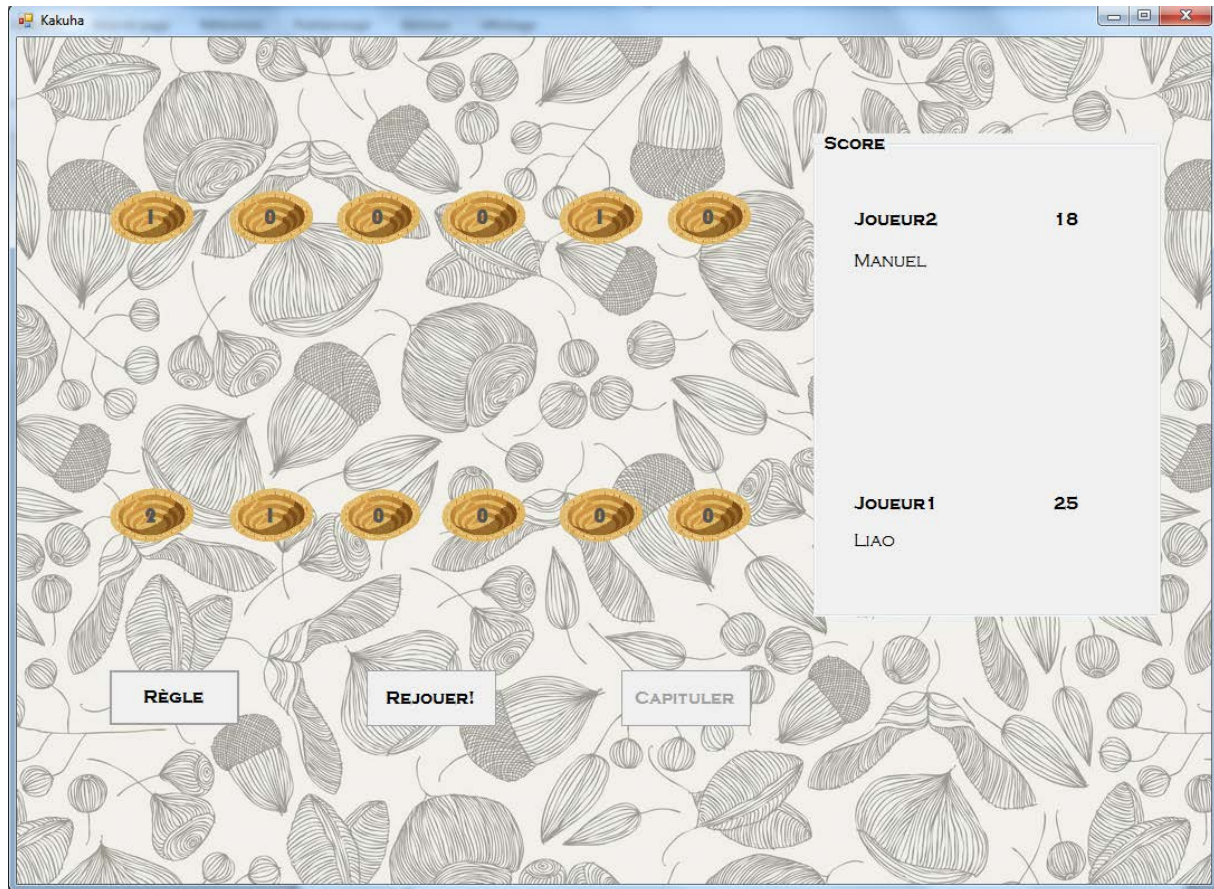


Pendant un coup (Pas à pas):



Ici on voit le trou(3)=0 et que la case dont le joueur 1 doit cliquer pour continuer c'est trou(7), avec le nombre de graines afficher en bleu. Le score de Liao a passé à 1 et le bouton « Rejouer ! » est désactivé pendant que le bouton « Capituler » est activé.

Fin de jeu :



Quand la partie est finie, toutes les cases sont bloquées alors que les boutons « Rejouer ! » et « Capituler » sont activés et désactivés respectivement.

5. On charge les paramètres initiaux du jeu.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load 'Tous les boutons sont bloqués avant d'initialiser le jeu'
    dimcontrol()
    InitialiseBouton() 'Chargé les paramètres initiales du jeu'

    btn14.Enabled = False
End Sub
```

6. On explique quels sont les effets quand on clique les boutons représentant les trous.

```
'On cherche à seulement écrire les effets des boutons des cases une seule fois'
Private Sub btn1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btn1.Click, btn2.Click, btn3.Click, btn4.Click, btn5.Click, btn6.Click,
btn7.Click, btn8.Click, btn9.Click, btn10.Click, btn11.Click, btn12.Click
    Dim Numtrou As Byte = CByte(Mid(sender.name, 4))
    Dim continuer As Boolean
    Dim terminer As Boolean
```

```

        btn14.Enabled = True 'On laisse le bouton capituler active pendant toute la
partie'
        Jouer(Numtrou)
        For i = 1 To 12
            button(i).Text = trou(i) 'On affiche la valeur de la case, c'est-à-dire le
nombre de graines'
            Next
            score(joueur).Text = CByte(score(joueur).Text) + 1 'On rajoute une graine a
chaque fois qu'on clique'
            terminer = JeuTermine(CByte(score(joueur).Text), CByte(score(3 -
joueur).Text)) 'Le jeu termine si un des joueurs a plus de 24 graines ou s'il y a une
partie nulle'
            continuer = PeutJouer(Numtrou)
            If terminer = False Then
                If continuer = False Then
                    If Not (trou(7) = 0 And trou(8) = 0 And trou(9) = 0 And trou(10) = 0
And trou(11) = 0 And trou(12) = 0) Then 'Affichage de changement de joueur'
                        MsgBox("Changer le joueur!")
                        numjoueur(3 - joueur).Text = "Joueur" & (3 - joueur)
                        indique(3 - joueur).Text = ""
                        numjoueur(joueur).Text = "Joueur" & joueur
                        indique(joueur).Text = ">>>>"
                    Else
                        numjoueur(3 - joueur).Text = "Joueur" & (3 - joueur)
                        indique(3 - joueur).Text = ""
                        numjoueur(joueur).Text = "Joueur" & joueur
                        indique(joueur).Text = ">>>>"
                    End If
                End If
            Else
                If score(1).Text = 24 And score(2).Text = 24 Then
                    MsgBox("Il n'y pas de gagnant", , "Wow!") 'Affichage de la partie
nulle'
                Else
                    If continuer = False Then
                        'Affichage du vainqueur'
                        MsgBox("Joueur " & (3 - joueur) & " a gagné !", ,
"Félicitations!")
                    Else
                        MsgBox("Joueur " & joueur & " a gagné !", , "Félicitations!")
                    End If
                End If
            End If
            btn14.Enabled = False 'Une fois la partie est finie on peut plus
capituler'
        End If
        validtrou(terminer, continuer, Numtrou)
    End Sub

```

7.

```

Private Sub btn13_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn13.Click 'Pour commencer le jeu'
    InitialiseBouton()
    btn13.Text = "Rejouer!" 'Une fois la partie commence on change le jouer par
rejouer'
    btn13.Enabled = False
    btn14.Enabled = False
    nom(1).Text = InputBox("Tapez le nom de joueur1", "Entrez votre nom") 'Les
joueurs peuvent rentrer leurs nom'
    nom(2).Text = InputBox("Tapez le nom de joueur2", "Entrez votre nom")
End Sub

```


8.

```

Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles
btn14.Click 'pour le bouton Capituler'
    If MsgBox("Voulez vous capituler?", vbYesNo, "Capituler?") = vbYes Then 'On
donne l'option au joueur s'il est vraiment sure de capituler'
        MsgBox("Joueur" & (3 - joueur) & " a gagné !") 'Affichage du joueur
gagnant'
        For i = 1 To 12
            button(i).Enabled = False 'Les cases deviennent alors désactivées'
        Next
        btn13.Enabled = True 'On peut rejouer après la partie est finie'
        btn14.Enabled = False 'Comme la partie est finie le bouton Capituler est
désactivé'
    End If

End Sub

```

9.

```

Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click 'On affichage les règles du jeu'
    MsgBox("Chaque joueur se positionne de part et d'autre du jeu. Les joueurs
jouent à tour de rôle. Jouer un coup consiste, pour un joueur, à choisir un trou de sa
rangée, prendre en main les graines du trou en question et les semer une à une dans
les trous suivants jusqu'à l'avant dernière. Ensuite : " & vbCrLf & "    - il met de côté
la dernière graine," & vbCrLf & "    - il prend en main les graines du trou suivant," &
vbCrLf & "    - et les sème une à une jusqu'à l'avant dernière." & vbCrLf & vbCrLf & "Le
joueur répète ses semis jusqu'à ce qu'un trou où il aurait prendre des graines soit
vide. Il passe alors la main à l'autre joueur." & vbCrLf & "Le vainqueur est celui qui
possède le plus grand nombre de graines.", , "Règles")
End Sub

```