Projet Kalaha

TD : GROUPE 1

CHARGE DE TD : DOUTRE SYLVIE

MEMBRE : CHAIMAA HADBI

WEIHUI TU

SOMMAIRE

I Introduction----------2

II Description des fonctions et procédures du module----------3

a) fonctions ----------3

b) procédures----------3

III Description des procédures de l'interface ----------4

IV Dialogue sous-jacent ----------4

1. Le bouton play ----------4
2. Le clic sur une case----------6
3. Si le dernier jeton répartie va dans le panier du joueur ----------6
4. Si la répartion des point se fait au-delà de la case 14 ----------7
5. La réparation des points se fait sur deux tours du tableau----------8
6. La partie est terminé----------9

V Structure des données ----------10

VI Les problèmes et solutions----------10

1. Problème au niveau de l'initialisation ----------10
2. Comment gérer le clic des boutons dans une même procédure
   Pour les deux joueurs ----------10
3. On a testé l'application à plusieurs reprises il y'a eu des bugs----------11
4. Quand le joueur 1 doit garder la main----------11

VII Conclusion ----------11

I. INTRODUCTION

Dans le cadre de notre master nous devons réaliser un projet de programmation où il faut créer une application. Cela nous permettra d'améliorer nos compétences et mettre en pratique la théorie et tous ce qu'on a appris tout au long de nos TP.

Notre projet de programmation consiste à réaliser une application qui permettrait de jouer au célèbre jeu africain Kalaha. C'est un jeu stratégique qui se joue à deux personnes : chaque joueur a 6 cases et un panier, dans chacune des cases on a le même nombre de jetons qu'il faudra répartir dans les autres cases et accumuler le maximum de jetons dans son panier.

A l'aide de Visual Basic.Net on va développer cette application, elle devra permettre au joueur de retrouver les mêmes éléments qu'en réalité. Avec la même flexibilité de pouvoir établir des stratégies. L'application doit être intuitive, savoir à qui est le tour, comptabiliser les points et les enregistrer jusqu'à rendre le résultat final. Pour cela il nous faut créer une interface qu'il faudra séparer du noyau fonctionnel c'est-à-dire que si l'on décide de modifier l'interface à tout moment cela ne doit pas impacter le noyau qui sera contenu dans un module, on n'aura pas besoin de le modifier. La partie qui peut changer concerne seulement ce qui est exploité par l'interface.

Il faut qu'au niveau de l'application tout soit explicite pour le joueur, que l'interaction lui soit facilité. Des messages peuvent apparaitre tout au long de la partie pour indiquer les différentes étapes. Le code implémenté dans le noyau doit être exploité de façon à ce que

On verra tout d'abord les différentes parties qui composent notre module puis ceux composant notre interface. Dans un deuxième temps nous décrirons en détail l'interaction entre les joueur et l'application. Pour finir nous évoquerons quelques démarches que nous avons dû faire pour aboutir au résultat final.

II. Description des fonction et procédures du module

a) Les fonctions

On a deux fonctions dans le module qui sont déclaré PUBLIC car on aura besoin qu'elles soient accessibles de façon illimité par le code qui interagit avec.

1. Public Function Initial_interface(ByVal i As Integer)

Cette fonction sera appelée lors de l'initialisation du jeu; elle permettra de répartir le bon nombre de jetons dans chaque case.

2. Public Function joueur_choisi(ByVal nb As Byte) As Boolean

Cette fonction permet de savoir quel joueur à la main, qui vient de jouer. Elle donne l'etat du joueur (vrai ou faux, cases activent ou non) et plus tard elle sera appelée par une procédure du module.

Vous pouvez retrouver le commentaire détaillé de ces fonctions dans le code du module.

b) Les procédures

Les procédures sont également en PUBLIC pour qu'on puisse y avoir accès avec le code de l'interface.

1. Public Sub KalahaGame(ByVal nb As Byte, ByVal tabdebut As Button(), ByVal valeur As Byte)

Cette procédure nous permet de considérer les différents cas de distributions de jetons lorsqu'il y'a plusieurs tours.
Un premier cas où lorsqu'on clique sur un bouton la distribution des jetons se fait sur un tour sans dépasser la 14ème case en effet la valeur à distribuer plus la position du bouton où a cliqué le joueur est inférieur à 14 donc il n'y a pas de problème les cases suivantes sont incrémenté d'un point.

Dans le deuxième cas le nombre de jetons à distribuer plus l'indice de la case dépasse la 14ème après la distribution il faut indiquer qu'il faut reprendre la distribution à partir du bouton 1.

Le troisième cas c'est lorsque la valeur à distribuer doit se faire sur trois tours c'est-à-dire qu'on démarre de la case 13 donc on ajoute 1 point au panier ensuite on reprend la distribution à partir du bouton 1 puis on fait le tour et on revient à la case 14 et on indique qu'on reprend à partir du bouton 1.

2. Public Sub joueur_Etat(ByVal nb As Byte, ByVal valeur As Byte, ByVal tabdebut As Button(), ByRef res As Byte, ByRef resultat As Byte)

Cette procédure indique dans quel cas le joueur qui vient de jouer pourra rejouer ou non. Elle fait appel à la fonction joueur_choisi dans le module pour savoir si c'est le joueur 1 ou 2 qui vient de jouer

ensuite elle vérifie si le joueur a en sa possession des jetons pour qu'il puisse rejouer, et s'il vient répartie son dernier jeton dans le panier ou si le joueur adverse n'a plus de jetons pour jouer et donc il garde la main.

Pour finir elle vérifie s'il reste des jetons en jeu pour continuer la partie, sinon si toutes les cases du joueur 1 et 2 sont vide elle déclare la fin de la partie en envoyant un résultat=1 à l'interface.

III. Description des procédures de l'interface

Les procédures de l'interface sont en PRIVATE ce qui implique que la portée ne dépasse pas le cadre de la class. Cependant comme la CLASS est en public si on rajoute des procédures publiques pourront être utilisé en dehors de la class. On a trois procédures en tout une qui gère le chargement du jeu, une pour l'initialisation et pour finir une qui gère les différents clics des boutons représentant les cases du tableau.

1. Private Sub Kalaha_load(sender As Object, e As EventArgs) Handles MyBase.Load

Indique que lorsque la page est chargé seul le bouton Play est activé et tous les autres inactivé, à ce moment-là aucune valeur n'apparait sur les boutons.

2. Private Sub btnplay_Click(sender As Object, e As EventArgs) Handles btnPlay.Click

Dans cette procédure on gère le bouton Play ; lorsqu'on clique dessus la fonction Initial_interface du module qui permet l'initialisation de tous les boutons est utilisé. Ensuite les boutons du joueur 2 sont désactivé et ceux du joueur 1 sont activé.

3. Private Sub joueur_Click(sender As Object, e As EventArgs) Handles btn1.Click, btn2.Click, btn3.Click, btn4.Click, btn5.Click, btn6.Click, btn8.Click, btn9.Click, btn10.Click, btn11.Click, btn12.Click, btn13.Click

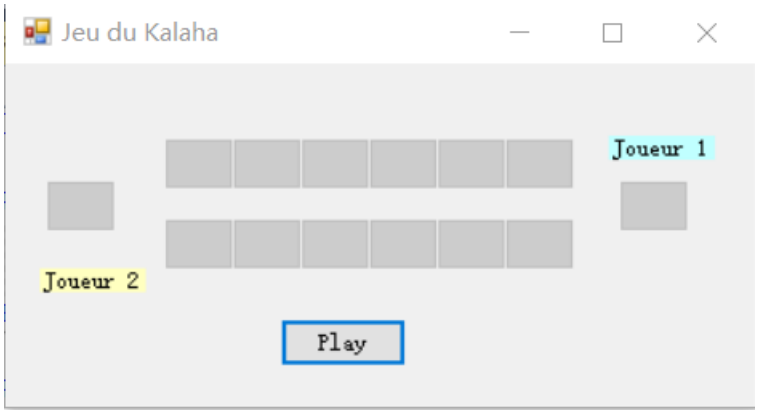
C'est la procédure qui gère la clique des boutons, elle fait appel à deux procédures du module celle du kalahaGame pour la répartition des jetons lorsqu'on clique sur un bouton et joueur_Etat qui permet de dire si le joueur à la main pour jouer, s'il peut rejouer ou non et pour finir elle annonce la fin du jeu avec le gagnant.

Le fonctionnement de ces procédures est détaillé plus amplement sur VB.

IV. Dialogue sous-jacent

1. Le bouton Play.

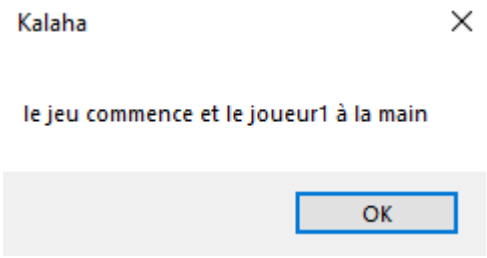
Avant le clic du bouton :



Lors du chargement de la page c'est le seul bouton activé le joueur peut appuyer uniquement sur ce bouton.
C'est la procédure Private Sub Kalaha_load(sender As Object, e As EventArgs) Handles MyBase.Load qui est utilisé .

Après le clic :

Ici c'est procédure btnplay_click de l'interface qui va réagir, elle fait tout d'abords apparaitre une MsgBox pour indiquer quel joueur aura la main, il faut cliquer sur « ok ».



Elle utilisera ensuite la fonction du module Public Function Initial_interface(ByVal i As Integer) qui initialisera toutes les cases en leurs donnant la bonne valeur de jetons (3 dans chaque case). Pour finir elle activera les boutons pour le joueur 1 et les désactive pour le joueur 2.

Le résultat final après cette procédure :

2. Le clic sur une case

Après le clic :

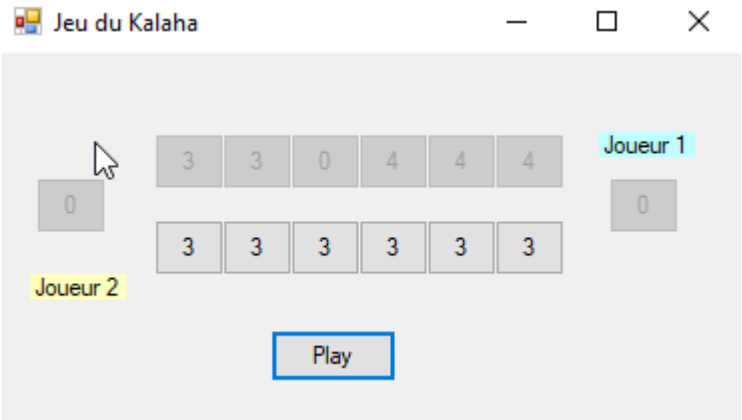
Le joueur 1 a cliqué sur le bouton représentant la case 3 ; la valeur de la case passe à zéro et les autres cases ont un jeton de plus.

Ici c'est la procédure joueur_click qui se déclenche (elle met la case à zéro) ; elle fait également appel à deux procédures du module.

Elle fait d'abord appel à la procédure kalahaGame qui va répartir les points dans les autres cases. Dans notre cas le joueur 1 va cliquer sur la case 3 qui possède 3 jetons, les jetons seront répartient un à un dans chaque case qui suit donc y'aura 4 jetons dans les cases 4, 5 et 6 et toujours zéro jeton dans le panier du joueur 1.

Puis la procédure joueur_click fait appel à la procédure joueur_Etat, cette dernière fait appel à la fonction joueur_choisi du module ici elle lui indique « état = vrai » c'est-à-dire c'est le joueur 1 qui vient de jouer. A partir de là la procédure joueur_Etat vérifie si ce joueur peut rejouer ou si c'est au tour du joueur 2. Elle envoie un res=1 si le joueur1 rejoue ou res=0 si c'est le joueur 2 prend la main. Dans notre cas le joueur 1 n'a pas réparti son dernier jeton dans le panier et le joueur 2 à tous ses jetons pour jouer donc res = 0. La procédure joueur_click recupère ce res, désactive les boutons du joueur 1 et active ceux du joueur 2 qui pourra alors jouer.

Résultat final :

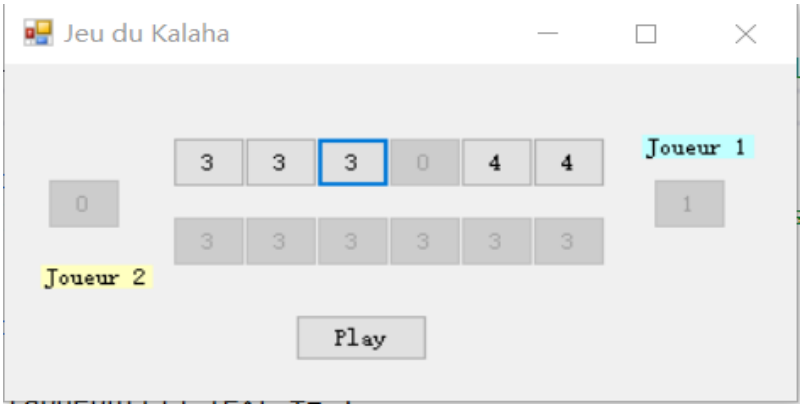


3. Si le dernier jeton répartie va dans le panier du joueur.

Premier cilc :

Le joueur garde la main, la fonction du module joueur_choisi retourne « etat=1 » à la procédure du module joueur_Etat qui elle renvoie un res = 1 (car le joueur 1 peut rejouer) à la procédure de l'interface joueur_Click qui maintient les boutons du joueur 1 activés.

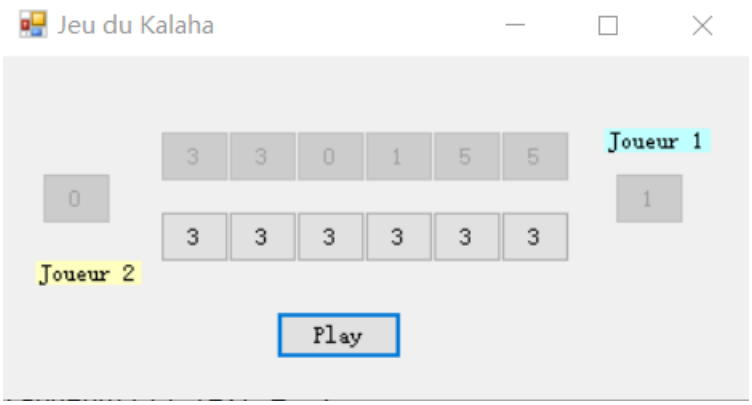
Ici le joueur a cliqué sur la case 4 qui avait trois jetons ils sont répartie dans la case 5, 6 et le dernier jetons dans le panier.



Deuxième clic :

Il peut rejouer il clique sur la case 3, ce qui entraine une répartition de ses trois jetons et il passe la mains au joueur 2 (le dernier jeton répartie ne va pas dans le panier mais dans la case 6) le res renvoyé par le module est égal à 0.

résultat final :



4. Si la répartion des point se fait au-delà de la case 14.

Avant le clic :

Si le joueur 2 décide de cliquer sur la case 13, la répartition des 3 jetons devra se faire dans la case 14 puis on reprend à la case 1 et 2 pour les deux derniers jetons.

La procédure du module KalahaGame considère ce cas-là et indique au tableau qu'il faut reprendre la répartition à la case 1. (cf deuxième cas dans le code de la procédure qu'on a détaillé en commentaire dans vb). Cette procédure est ensuite appelée par l'interface via joueur_click.

Après le clic :

5. La réparation des points se fait sur deux tours du tableau.

Avant le clic :

Ici si le joueur deux clic sur le bouton de la case 13 la réparation des jetons dépasse deux fois la dernière case du tableau. Donc à deux reprises il faut indiquer au tableau que la réparation des jetons restant reprend à la case 1.

Après le clic :

Cette possibilité est prise en compte dans le code de La procédure kakahaGame c'est géré par le cas 3 du code (à voir plus en détail en commentaire sur VB).

Resultat final :

On a deux points de plus dans le panier du joueur 2.

6. La partie est terminé

Avant le clic :

Il ne reste plus qu'un jeton à jouer. Après avoir cliqué sur ce bouton le joueur 1 n'aura plus de jetons à jouer ainsi que le joueur 2.

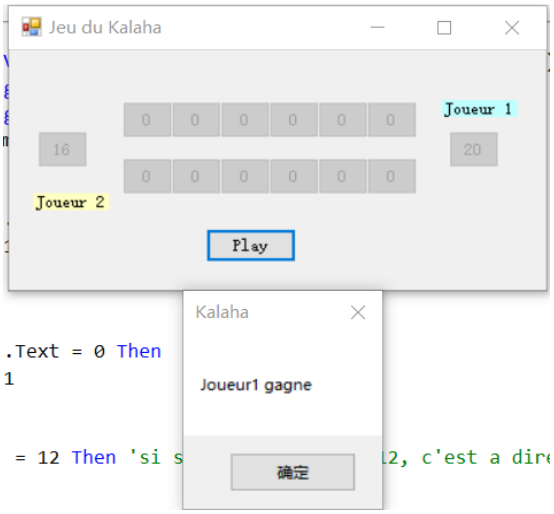
Après le clic :

La procédure joueur_click appel la procédure joueur_Etat celle-ci appel la fonction joueur_choisi du module qui lui envoie « état = 1 » car c'est le joueur 1 qui vient de jouer. La procédure joueur_Etat regarde si le joueur 1 peut rejouer, ici il a réparti sont dernier jeton dans le panier mais lorsuqu'elle fait la somme de ses cases vide c'est égal à 6 ce qui veut dire qu'il n'a plus de jetons à jouer elle passe alors au joueur 2 mais en faisant la somme de ses cases vide c'est égal à 6 aussi donc il n'a plus de

jetons non plus. On a un total de 12 cases vides qui représentent les jetons à jouer. La procédure joueur_Etat envoie un « résultat = 1 » à la procédure joueur_click de l'interface.

La procédure de l'interface joueur_click désactive tous les boutons puis compare les valeurs des deux paniers donc entre la case 7 et 14 (btn7 et btn14) et fait apparaitre une MsgBox pour indiquer le gagnant c'est-à-dire celui qui a le plus grand nombre dans son panier.

Résultat final :



V. Structure des données

On a tout d'abords un tableau indexé de 1 à 14 chaque case contient les jetons des différents joueurs (3 par case). Les jetons du joueur 1 qu'il peut jouer se trouvent dans les cases de 1 à 6 et ceux du joueur 2 dans les cases de 8 à 13. La case 7 sert de panier pour le joueur 1 elle sera toujours inactivé où il pourra accumuler ses jetons gagnés. La case 14 est le panier du joueur 2.

Les joueurs se sont deux variables booléennes qui permette au joueur de jouer quand c'est vrai ou bien de ne pas jouer quand c'est faux.

VI. Les problèmes et solutions

1. Problème au niveau de l'initialisation

Au début on l'avait faite dans l'interface et pour chaque bouton on a attribué une valeur donc on avait une ligne par bouton. Puis on a pensé à faire une fonction dans le module pour créer un tableau à 14 cases qui stock nos données c'est-à-dire les jetons, et ensuite il suffisait d'appeler la fonction dans l'interface qui initialise le jeu.

2. Comment gérer le clic des boutons dans une même procédure pour les deux joueurs.

D'abords on a fait une procédure pour chaque bouton il y'avait beaucoup trop de ligne qui se répétaient. Puis on a pensé à faire une procédure par joueur chaque procédure gérait les boutons d'un des deux joueurs, une qui gérait les boutons de 1 à 6 et l'autre de 8 à 13. Mais comme les deux

procédures avait le même code mais appelait deux fonctions différente dans le module il en fallait une pour le joueur 1 qui était joueur1_etat et une autre pour le joueur 1 qui était joueur2_etat alors que les deux fonctions exécutaient la même chose.

Donc pour finir on a pensé à mettre les deux joueurs dans la même procédure joueur_click dans l'interface qui gère les clics et qui appel une seule fonction dans le module joueur_etat ; tout ceci dans le but d'optimiser le code et avoir moins de lignes qui se répètent.

3. On a testé l'application à plusieurs reprises il y'a eu des bugs

Le premier c'était quand la répartition des jetons d'une case se faisait sur trois tours. Et donc dépassé la dernière case du tableau 2 fois. Donc on a ajouté une troisième condition dans l'algorithme de kalahaGame.

4. Quand le joueur 1 doit garder la main.

Lorsqu'il a mis son dernier jeton dans le panier il ne pouvait pas rejouer, donc il fallait rajouter une condition pour cela mais aussi préciser que s'il n'avait plus de jeton il pouvait pas rejouer. Donc il faut que l'algorithme puisse compter les cases vide à l'aide d'une boucle.

VII. CONCLUSION

La réalisation de notre application mettant en œuvre le fameux jeu kalaha nous a permis de développer de nouvelles compétences. Le fait de travailler en équipe permet de voir le problème sous différents angles, et de se compléter. En premier lieu il n'a pas été difficile de penser à un algorithme exécuterai l'application, cependant quand on l'a testé sur Visual Basic on a rencontré quelques bugs au niveau des règles de jeu à respecter donc il a fallu le modifier à deux reprises pour prendre en compte tous les cas possibles. Une fois que notre application fonctionnait on a modifié l'interface plusieurs fois pour l'optimiser au maximum. On a fusionné certaines procédures ou rajouter des fonctions tout en prenant soins que cela n'impacte pas son bon fonctionnement. Le module de notre application est fonctionnel indépendamment de l'interface. Donc si plus tard on voulait remplacer nos boutons par des labels par exemple ce serait possible sans avoir à modifier le module.