Université Toulouse 1 – Science sociale

M1 SIIO
Année 2008 – 2009

Projet :
Algorithme et programmation événementielle

Etudiants : PHAM Le Nguyen
Hamza SMAIL

Plan

Introduction

Ce document contient un rapport sur la réalisation de projet de programmation événementielle dans le cadre des travaux dirigés.

La problématique de ce projet est de réaliser un programme Visual Basic permettant de trouver la distance entre deux villes données. Cette distance correspond au nombre de villes intermédiaires et aussi d'afficher le nom des villes qui représente cette distance avec distance=1.

Pour ce faire on a choisit de faire le projet en deux version. La première version est celle où on n'a définit nous même les connexions entre les villes (nombre de villes est de 10), ainsi l'utilisateur n'a qu'à choisir les deux villes de départ et d'arrivée pour avoir les résultats. Dans la deuxième version, l'utilisateur, a la possibilité d'initialiser les connections entre les villes, le reste de l'application ayant le même fonctionnement que la première version.

1. Première version (connections initialisées) :

1.1. Initialisation de table villes :

La public function Villes définie dans le module, nous permet à travers l'instruction alternatives select case…end select de définir chaque nom de ville, cette fonction servira à remplir le tableau des villes dans le form load.

```
Public Function Villes (Nombre As Integer) As String
'Récupérer les noms de villes
   Select Case Nombre
      Case 1: Villes = "Toulouse"
      Case 2: Villes = "Bordeaux"
      Case 3: Villes = "Montpellier"
      Case 4: Villes = "Marseille"
      Case 5: Villes = "Poitier"
      Case 6: Villes = "Limoges"
      Case 7: Villes = "Paris"
      Case 8: Villes = "Rouen"
      Case 9: Villes = "Nice"
      Case 10: Villes = "Dijon"
   End Select
End Function
```

1.2. Initialisation de la table des connections :

La public sub (procédures) TBCN, est pour définir les connections entre les villes, on a utilisé une procédure pour récupérer facilement ces connections dans le form load en les chargeant dans le tableau des connections.

```
Public Sub TBCN(TabConnections() As Boolean)

'Définir les connections entre les villes

TabConnections(1, 1) = False
TabConnections(1, 2) = True
TabConnections(1, 3) = True
TabConnections(1, 4) = False
TabConnections(1, 5) = False
TabConnections(1, 6) = False
TabConnections(1, 7) = False
TabConnections(1, 8) = False
TabConnections(1, 9) = False
TabConnections(1, 10) = False


TabConnections(2, 1) = True
TabConnections(2, 2) = False
TabConnections(2, 3) = False
```

TabConnections(2, 4) = False
TabConnections(2, 5) = True
TabConnections(2, 6) = False
TabConnections(2, 7) = False
TabConnections(2, 8) = False
TabConnections(2, 9) = False
TabConnections(2, 10) = False

TabConnections(3, 1) = True
TabConnections(3, 2) = False
TabConnections(3, 3) = False
TabConnections(3, 4) = True
TabConnections(3, 5) = False
TabConnections(3, 6) = False
TabConnections(3, 7) = False
TabConnections(3, 8) = False
TabConnections(3, 9) = False
TabConnections(3, 10) = False

TabConnections(4, 1) = False
TabConnections(4, 2) = False
TabConnections(4, 3) = True
TabConnections(4, 4) = False
TabConnections(4, 5) = False
TabConnections(4, 6) = False
TabConnections(4, 7) = False
TabConnections(4, 8) = False
TabConnections(4, 9) = True
TabConnections(4, 10) = False

TabConnections(5, 1) = False
TabConnections(5, 2) = True
TabConnections(5, 3) = False
TabConnections(5, 4) = False
TabConnections(5, 5) = False
TabConnections(5, 6) = True
TabConnections(5, 7) = True
TabConnections(5, 8) = False
TabConnections(5, 9) = False
TabConnections(5, 10) = False

TabConnections(6, 1) = False
TabConnections(6, 2) = False
TabConnections(6, 3) = False
TabConnections(6, 4) = False
TabConnections(6, 5) = True
TabConnections(6, 6) = False
TabConnections(6, 7) = True

```
TabConnections(6, 8) = False
TabConnections(6, 9) = False
TabConnections(6, 10) = True


TabConnections(7, 1) = False
TabConnections(7, 2) = False
TabConnections(7, 3) = False
TabConnections(7, 4) = False
TabConnections(7, 5) = True
TabConnections(7, 6) = True
TabConnections(7, 7) = False
TabConnections(7, 8) = True
TabConnections(7, 9) = False
TabConnections(7, 10) = True

TabConnections(8, 1) = False
TabConnections(8, 2) = False
TabConnections(8, 3) = False
TabConnections(8, 4) = False
TabConnections(8, 5) = False
TabConnections(8, 6) = False
TabConnections(8, 7) = True
TabConnections(8, 8) = False
TabConnections(8, 9) = False
TabConnections(8, 10) = False


TabConnections(9, 1) = False
TabConnections(9, 2) = False
TabConnections(9, 3) = False
TabConnections(9, 4) = True
TabConnections(9, 5) = False
TabConnections(9, 6) = False
TabConnections(9, 7) = False
TabConnections(9, 8) = False
TabConnections(9, 9) = False
TabConnections(9, 10) = False


TabConnections(10, 1) = False
TabConnections(10, 2) = False
TabConnections(10, 3) = False
TabConnections(10, 4) = False
TabConnections(10, 5) = False
TabConnections(10, 6) = True
TabConnections(10, 7) = True
TabConnections(10, 8) = False
TabConnections(10, 9) = False
TabConnections(10, 10) = False
End Sub
```

1.3. Code de l'algorithme général :

Pour l'algorithme général appelé Noyau on a pris comme paramètres :

Les données:

N: qui est un nombre entier qui est le nombre maximum des éléments du tableau des villes

X : qui correspond à la ville de départ.

Y : qui correspond à la ville d'arrivée.

TabC () : qui correspond au tableau des connections.

TabV () : qui correspond au tableau des villes.

Les résultats :

Distance : qui est la distance entre deux villes.

NbreV : qui est le nombre de villes intermédiaires et qui va servir à récupérer le nombre des villes intermédiaires.

TabVillesInters () : qui reçoit le nom des villes intermédiaires.

Le fonctionnement :

Pour faire fonctionner l'algorithme on a définit dans un premier temps des variables « l », « c » et « ArreterDeChercher ». En suite on a initialisé le nombre de ville à 0.

Au début, l'algorithme vérifie si la connexion entre deux villes données est vraie, si c'est le cas le résultat sera que la distance égale à 0.

Dans un deuxième temps si la première condition n'est pas vérifiée, on vérifie d'abord si l'indice des deux villes (x et y) a la même valeur donc logiquement pas de connexion entre une ville et elle-même, le résultat renvoyé est un MsgBox demandant de choisir une ville de destination. Dans le cas où x est différent de y, on vérifie la condition qui permet de donner une connexion indirecte entre deux villes données avec une seule ville intermédiaire. Le résultat sera que la distance reçoit 1 et pour afficher le nom des villes à distance 1, on utilise l'indice du nombre de villes (NbreV) qui croit de 1 à chaque fois que la condition est vérifiée, il permet de positionner la première ville trouvée à la position 1 du tableau des villes intermédiaires et ainsi de suite. Pour la condition « ArreterDeChercher =Vrai » on a été obligé de l'utiliser car elle permet d'arrêter le calcul au niveau d'une seule ville intermédiaires même s'il existe une connexion en passant par deux villes.

En fin, si entre les deux villes données (départ et arrivée) il n'existe pas de villes intermédiaire en passant que par une seule ville, on va voir s'il y a deux villes par lesquelles il faut passer pour aller d'une ville à une autre. Ça fonctionne comme suit : on donne une ville de départ « x », s'il y a connexion avec une ville d'indice « l », l'algorithme vérifie s'il y a

connexion entre la ville trouvée d'indice « l » et une autre d'indice « c », si c'est vrai la dernière étape est de voir si cette ville d'indice « c » est connectée avec la ville d'arrivée « y » et dans le cas où cette condition est vérifiée le résultat est « Distance=2 ».

Le code est comme suit :

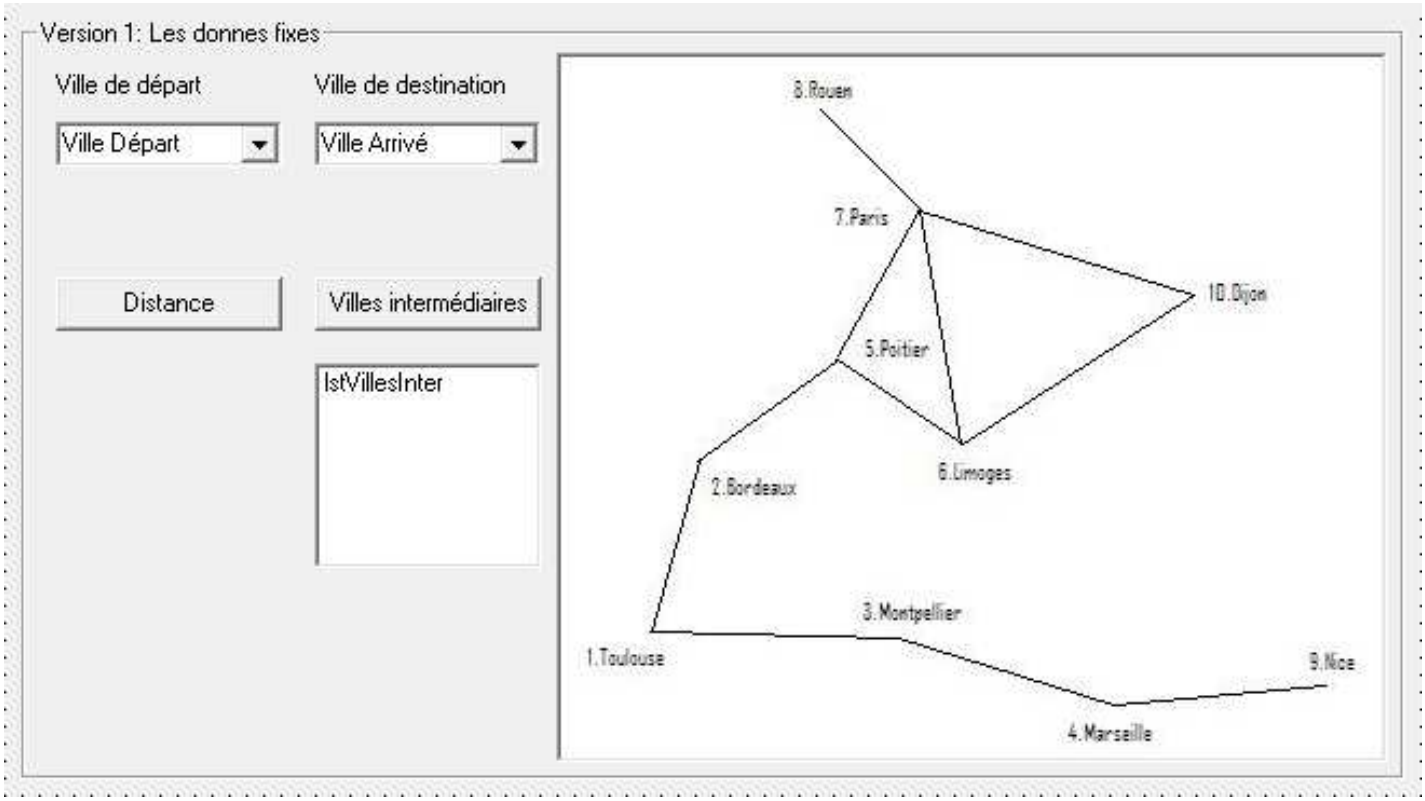
```
Public Sub Noyau(ByVal n As Integer, ByVal x As Integer, ByVal y As Integer, ByRef TabC() As Boolean, ByRef TabV() As String, ByRef Distance As Variant, ByRef NbreV As Integer, ByRef TabVillesInters() As String)

 Dim l As Integer
 Dim c As Integer
 Dim ArreterDeChercher As Boolean


 NbreV = 0
   If TabC(x, y) = True Then
      Distance = 0
   Else
      If x = y Then
         MsgBox "Choisir une autre ville de destination"
      Else
        ArreterDeChercher = False
         For l = 1 To n
            If TabC(x, l) = True And TabC(l, y) = True Then
               Distance = 1
               NbreV = NbreV + 1
               TabVillesInters(NbreV) = TabV(l)
               ArreterDeChercher = True
            ElseIf ArreterDeChercher = False Then
               For c = 1 To n
                  If TabC(x, c) = True And TabC(c, l) = True And TabC(l, y) = True Then
                     Distance = 2
                  End If
               Next c
            End If
         Next l
      End If
   End If
End Sub
```

1.4. L'Interface :

Elle est comme suit :



1.5. Code Form :

```
Const MaxNbVilles = 25
Dim NbVilles As Integer
Dim TabVilles(1 To MaxNbVilles) As String
Dim Connections(0 To MaxNbVilles, 0 To MaxNbVilles) As Boolean
Dim TabVillesInters(1 To MaxNbVilles) As String

Private Sub Form_Load()
    Dim i As Integer

    frmVersion1.Visible = False
    frm1Version2.Visible = False
    frm2Version2.Visible = False
    frm3Version2.Visible = False

'Initialise Tableaux Ville
    For i = 1 To MaxNbVilles
        TabVilles(i) = Villes(i)
    Next i

'Initialise Tableaux Connections entre les villes
    Call TBCN (Connections())
```

```
'Charger les valeurs pour les ComboBox de villes départ et villes d'arrivée
   For i = 1 To MaxNbVilles
      cmbDepart.AddItem TabVilles(i)
      cmbArrive.AddItem TabVilles(i)
   Next i
'Déactiver les buttons
   cmdDistance.Enabled = False
   cmdVillesInters.Enabled = False

End Sub
```

Private Sub cmbArrive_Click()

```
'Activer button Distance
   If cmbDepart.ListIndex >= 0 Then
      cmdDistance.Enabled = True
   End If
End Sub
```

Private Sub cmbDepart_Click()

```
'Activer button Distance
   If cmbArrive.ListIndex >= 0 Then
      cmdDistance.Enabled = True
   End If
End Sub
```

Private Sub cmdDistance_Click()

```
   Dim x As Integer
   Dim y As Integer
   Dim Distance As Variant

'Effacer List villes intermédiaire
   lstVillesInter.Clear

'Récupérer les valeurs dans les ComboBox
   x = cmbDepart.ListIndex + 1
   y = cmbArrive.ListIndex + 1

'Lancer la recherche en faisant appel à l'algorithme général
   Call Noyau (MaxNbVilles, x, y, Connections(), TabVilles(), Distance, NbVilles,
TabVillesInters())

'Afficher la distance entre deux villes
   lblDistance.Caption = Distance

'Activer button Villes Intermédiaires
   If Distance = 1 Then
```

```
        cmdVillesInters.Enabled = True
    End If

'Déactiver button Distance
    cmdDistance.Enabled = False

End Sub

Private Sub cmdVillesInters_Click()
    Dim i As Integer

'Afficher les villes intermédiaires
    For i = 1 To NbVilles
        lstVillesInter.AddItem TabVillesInters(i)
    Next i

'Déactiver les buttons
cmdDistance.Enabled = False
cmdVillesInters.Enabled = False

End Sub
```

2. Deuxième version (initialisation des villes par l'utilisateur) :

2.1. Initialisation de table villes :

2.1.1. Code :

On utilise le code suivant :

```
'A partir d'ici on a les codes de version 2, l'utilisateur peut modifier son réseau de villes

Private Sub cmdInsererVille_Click()

'Remplir le listbox de ville
    lstResauxVille.AddItem (txtInsererVilles.Text)
    txtInsererVilles.Text = ""
    txtInsererVilles.SetFocus

 'Déactiver cmdInsererVille
    cmdInsererVille.Enabled = False
    'Activer button contruire tableau de ville
    If lstResauxVille.ListCount >= 2 Then
        cmdTabVilles.Enabled = True
    End If
End Sub
```

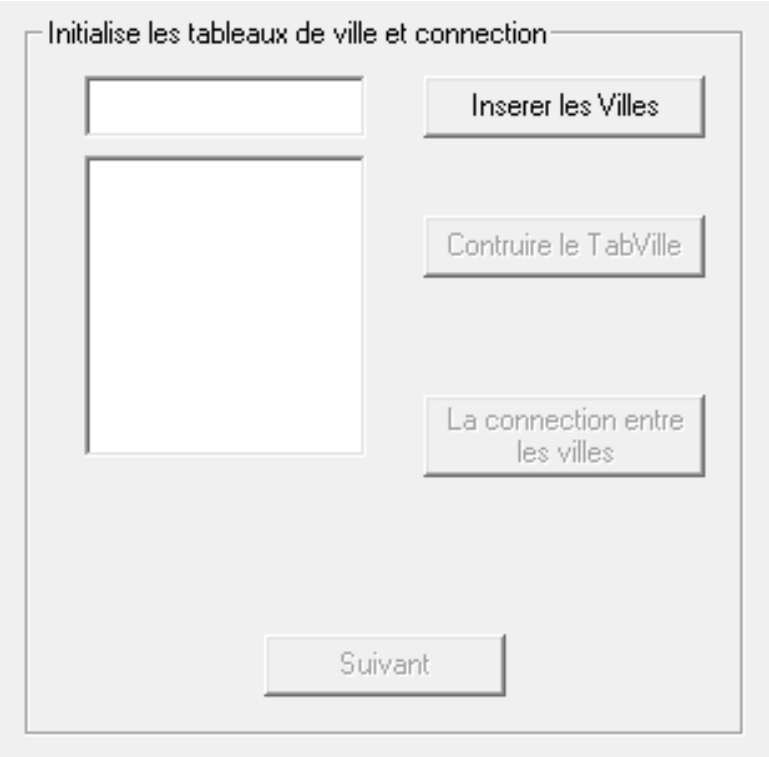
Private Sub cmdTabVilles_Click()

```
'Remplir le table ville
   Dim i As Integer

   For i = 1 To lstResauxVille.ListCount
      TabVilles(i) = lstResauxVille.List(i - 1)
   Next i

 'Activer button construire tableau de connection entre les villes
   cmdConnections.Enabled = True
End Sub
```

2.1.2. Interface :



2.2. Initialisation de table connections :

2.2.1 Code :

Private Sub cmdConnections_Click()

```
  Dim i As Integer
  Dim j As Integer

'Remplir le table connections
   For i = 1 To lstResauxVille.ListCount
      For j = i + 1 To lstResauxVille.ListCount
      If MsgBox("Il y a une connection direct entre " & TabVilles(i) & " et " & TabVilles(j),
vbYesNo) = vbYes Then
         Connections(i, j) = True
```

```
            Connections(j, i) = True
        Else
            Connections(i, j) = False
            Connections(j, i) = False
        End If
        Next j
    Next i

'Activer button suivant1
    cmdSuivant1.Enabled = True
End Sub
```

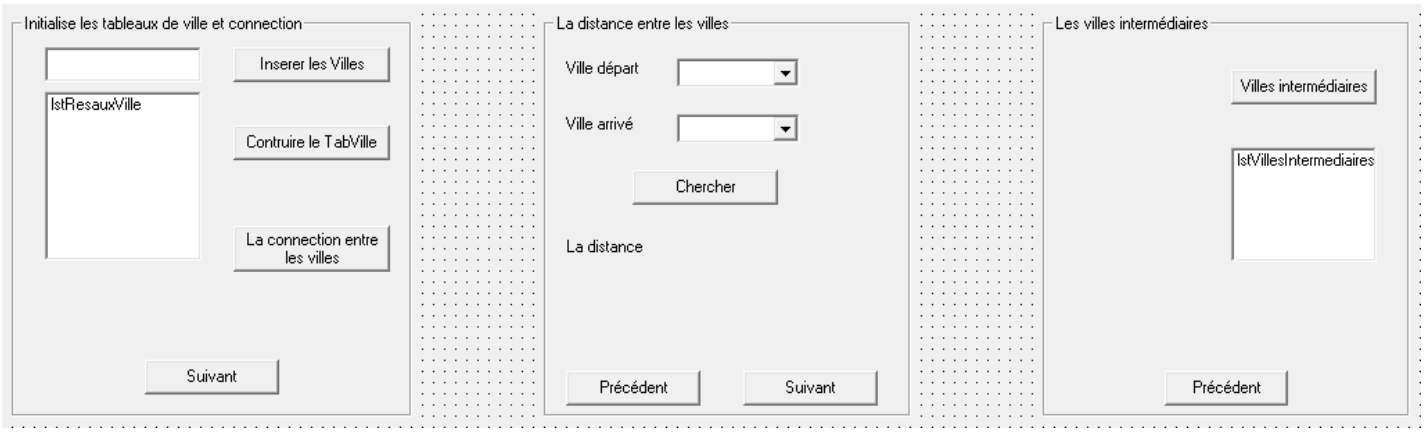
2.2.2 Interface:

On utilise un message box pour établir la connections entre les villes :

2.3. Code de l'algorithme général:

On utilise le même algorithme que la version 1

2.4. Interface :



2.5. Code Form :

```
Private Sub cmdSuivant1_Click()
   Dim i As Integer

' Remplir les combobox pour les villes de départ et les villes d'arrivé
      For i = 1 To lstResauxVille.ListCount
         cmbDepartVersion2.AddItem TabVilles(i)
         cmbArriveVersion2.AddItem TabVilles(i)
      Next i

 'Activer frame 2 de version 2
   frm2Version2.Visible = True
End Sub

Private Sub cmdChercher_Click()
   Dim x As Integer
   Dim y As Integer
   Dim Distance As Variant

'Chercher la distance entre les villes

         'Effacer la listbox chaque fois click sur le button
                  lstVillesIntermediairesVersion2.Clear

          'Récupérer les valeurs dans les ComboBox
                   x = cmbDepartVersion2.ListIndex + 1
                   y = cmbArriveVersion2.ListIndex + 1

          'Récupérer le calcule dans l'algorithme général
                  Call Noyau(MaxNbVilles, x, y, Connections(), TabVilles(), Distance,
                  NbVilles, TabVillesInters())

           'Afficher la distance entre deux villes
                   lblDistanceVersion2.Caption = Distance

            'Activer button Villes Intermédiaires
                   If Distance = 1 Then
                   cmdVillesInterVersion2.Enabled = True
   End If

   'Déactiver button Distance
   cmdDistance.Enabled = False

   'Activer button précédent et suivant
   cmdPrecedent1.Enabled = True
   cmdSuivant2.Enabled = True
End Sub
```

```
Private Sub cmdVillesInterVersion2_Click()
   Dim i As Integer

' Afficher les villes intermédiaires
      For i = 1 To NbVilles
         lstVillesIntermediairesVersion2.AddItem TabVillesInters(i)
      Next i
End Sub

' Les divers codes pour activer et désactiver les buttons


Private Sub cmbArriveVersion2_Click()

 'Activer button chercher
    If cmbArriveVersion2.ListIndex >= 0 Then
    cmdChercher.Enabled = True
    End If
End Sub

Private Sub cmbDepartVersion2_Click()

'Activer button chercher
   If cmbDepartVersion2.ListIndex >= 0 Then
      cmdChercher.Enabled = True
   End If
End Sub

Private Sub cmdPrecedent1_Click()

'Déactiver frame 2 et frame 3 de version 2
   frm2Version2.Visible = False
   frm3Version2.Visible = False
End Sub

Private Sub cmdPrecedent2_Click()

'Déactiver frame 3 de version 2
   frm3Version2.Visible = False
End Sub

Private Sub cmdSuivant2_Click()

'Activer frame 3 de version 2
   frm3Version2.Visible = True
End Sub
```

Private Sub cmdVersion1_Click()

```
'Activer version 1
   frmVersion1.Visible = True

'Déactiver version 2
   frm1Version2.Visible = False
   frm2Version2.Visible = False
   frm3Version2.Visible = False

End Sub
```

Private Sub cmdVersion2_Click()

```
'Effacer list réseau ville
   lstResauxVille.Clear
   'Acitiver Version 2
   frm1Version2.Visible = True

'Déactiver version 1
   frmVersion1.Visible = False

'Déactiver les buttons de version 2
   cmdInsererVille.Enabled = False
   cmdTabVilles.Enabled = False
   cmdConnections.Enabled = False
   cmdSuivant1.Enabled = False
   cmdChercher.Enabled = False
   cmdPrecedent1.Enabled = False
   cmdSuivant2.Enabled = False
End Sub
```

Private Sub txtInsererVilles_Change()

```
'Activer button cmdInsererVille
   If txtInsererVilles.Text <> "" Then
      cmdInsererVille.Enabled = True
   End If
End Sub
```

Conclusion

Nous pensons que nous avons accomplit note projet avec un petits succès. Nous avons bien cerné le projet en faisant beaucoup d'efforts. Grâce au projet, on a élargi nos connaissances notamment en travaillant en équipe.

On a rédigé notre projet en expliquant clairement la relation entre l'algorithme et Visual Basic, notamment la manière dont on a utilisé Visual Basic. Il existe encore des améliorations à effectuer. On espère que notre travaille saura vous satisfaire.

En fin, nous vous remercions beaucoup de votre attention et de nous excuser pour les quelques erreurs qui peuvent exister.