Projet Boids

Dossier du Code

FLOCH-ROBACH

A travers ce dossier de programmation nous afficherons le traitement du code mais nous le détaillerons explicitement à travers de nombreux commentaires. Nous diviserons le code de programmation en 3 parties : graphisme, formules et comportement dans lequel sont expliqués les algorithmes décrivant le comportement d'un boid.

GRAPHISME

➢ FORMULAIRE : PLATEAU DE JEU : *FORMPLATEAU*

Nous retrouvons ici les algorithmes nécessaires à la création de l'interface graphique mais également à la création des obstacles et du boid.

Interface graphique

```
Private Sub PlateauPictureBox_Click(sender As Object, e As EventArgs) Handles PlateauPictureBox.Click
    Dim feuille As Bitmap = New Bitmap(plateauX, plateauY)
    Dim dessin As Graphics = Graphics.FromImage(feuille)
End Sub
```

Conception d'un Arbre

```
Private Sub dessinerarbres(ByVal bmp As Bitmap)
    Using g As Graphics = Graphics.FromImage(bmp)
        Using k As Graphics = Graphics.FromImage(bmp)

            Dim q As Pen
            q = New Pen(Brushes.DarkGreen)
            q.Width = 4

            Dim p As Pen
            p = New Pen(Brushes.DarkRed)
            p.Width = 1

            Dim t As SolidBrush
            t = New SolidBrush(Color.Green)

            Dim h As SolidBrush
            h = New SolidBrush(Color.Pink)

            Dim i As Integer
            Dim j As Integer
            Dim c As PointF
            Dim a As PointF


            For i = 0 To nbarbres - 1
```

```
            arbres(i).rayon = 20
            c.X = arbres(i).x - Math.Sqrt(arbres(i).rayon ^ 2)
            c.Y = arbres(i).y - Math.Sqrt(arbres(i).rayon ^ 2)
            g.FillEllipse(t, New RectangleF(New PointF(c.X, c.Y), New Size(2 * arbres(i).rayon, 2 *
arbres(i).rayon)))
            g.DrawEllipse(q, New RectangleF(New PointF(c.X, c.Y), New Size(2 * arbres(i).rayon, 2 *
arbres(i).rayon)))
            For j = 0 To arbres(i).nbcerises - 1
               arbres(i).cerises(j).rayon = 5
               a.X = arbres(i).cerises(j).x - Math.Sqrt(arbres(i).cerises(j).rayon ^ 2)
               a.Y = arbres(i).cerises(j).y - Math.Sqrt(arbres(i).cerises(j).rayon ^ 2)
               k.FillEllipse(h, New RectangleF(New PointF(a.X, a.Y), New Size(2 * arbres(i).cerises(j).rayon, 2 *
arbres(i).cerises(j).rayon)))
               k.DrawEllipse(p, New RectangleF(New PointF(a.X, a.Y), New Size(2 * arbres(i).cerises(j).rayon, 2 *
arbres(i).cerises(j).rayon)))
            Next
         Next

      End Using
   End Using
End Sub
```

Conception d'un Obstacles

```
Private Sub dessinerobstacles(ByVal bmp As Bitmap)
   Using g As Graphics = Graphics.FromImage(bmp)

      Dim b As SolidBrush
      b = New SolidBrush(Color.DarkRed)

      Dim i As Integer
      Dim c As PointF

      For i = 0 To nbobstacles - 1
         c.X = obstacles(i).x
         c.Y = obstacles(i).y
         g.FillRectangle(b, New RectangleF(New PointF(c.X, c.Y), New SizeF(35, 35)))
      Next

   End Using
End Sub
```

Conception d'un Boid

```
Private Sub dessinerboids(ByVal bmp As Bitmap)
   Using g As Graphics = Graphics.FromImage(bmp)

      Dim p As Pen
```

```
        p = New Pen(Brushes.Gray)
        p.Width = 0.5

        Dim b As SolidBrush
        b = New SolidBrush(Color.LightGoldenrodYellow)

        Dim i As Integer

        For i = 0 To nbboids - 1
            Dim Q As boid
            Q.d.x = boids(i).d.x + boids(i).acc.x
            Q.d.y = boids(i).d.y + boids(i).acc.y
            normaliser(Q.d)
            boids(i).d = Q.d
            Dim point1 = New PointF(boids(i).p.x + 10 * boids(i).d.x, boids(i).p.y + 10 * boids(i).d.y)
            Dim point2 = New PointF(boids(i).p.x + boids(i).masse * rotation(boids(i).d, -2 * PI / 3).x, boids(i).p.y +
boids(i).masse * rotation(boids(i).d, -2 * PI / 3).y)
            Dim point3 = New PointF(boids(i).p.x, boids(i).p.y)
            Dim point4 = New PointF(boids(i).p.x + boids(i).masse * rotation(boids(i).d, 2 * PI / 3).x, boids(i).p.y +
boids(i).masse * rotation(boids(i).d, 2 * PI / 3).y)
            Dim poly = {point1, point2, point3, point4}
            'on choisit de representer les boids plus gros en fonction de leur masse

            g.FillPolygon(b, poly)
            g.DrawPolygon(p, poly)
        Next
    End Using
End Sub
```

```
Private Sub dessinerrapaces(ByVal bmp As Bitmap)
    Using g As Graphics = Graphics.FromImage(bmp)

        Dim p As Pen
        p = New Pen(Brushes.Black)
        p.Width = 0.5

        Dim b As SolidBrush
        b = New SolidBrush(Color.DarkGray)

        Dim i As Integer

        For i = 0 To nbrapaces - 1
            Dim R As rapace
            R.d.x = comportementcible(rapaces(i)).x + rapaces(i).acc.x
            R.d.y = comportementcible(rapaces(i)).y + rapaces(i).acc.y
```

```
            normaliser(R.d)
            rapaces(i).d = R.d
            Dim point1 = New PointF(rapaces(i).p.x + 10 * rapaces(i).d.x, rapaces(i).p.y + 10 * rapaces(i).d.y)
            Dim point2 = New PointF(rapaces(i).p.x + rapaces(i).masse * rotation(rapaces(i).d, -2 * PI / 3).x,
rapaces(i).p.y + rapaces(i).masse * rotation(rapaces(i).d, -2 * PI / 3).y)
            Dim point3 = New PointF(rapaces(i).p.x, rapaces(i).p.y)
            Dim point4 = New PointF(rapaces(i).p.x + rapaces(i).masse * rotation(rapaces(i).d, 2 * PI / 3).x,
rapaces(i).p.y + rapaces(i).masse * rotation(rapaces(i).d, 2 * PI / 3).y)
            Dim poly = {point1, point2, point3, point4}
            'on choisit de representer les rapaces plus gros en fonction de leur masse

            g.FillPolygon(b, poly)
            g.DrawPolygon(p, poly)
        Next
    End Using
End Sub
```

```
Private Sub FormPlateau_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    PlateauPictureBox.Top = 0
    PlateauPictureBox.Left = 0
    PlateauPictureBox.Width = plateauX
    PlateauPictureBox.Height = plateauY
    Console.Show()
End Sub

Private Sub FormPlateau_Paint(sender As Object, e As PaintEventArgs) Handles Me.Paint
    If PlateauPictureBox.Image IsNot Nothing Then
        PlateauPictureBox.Image.Dispose()
    End If

    Dim bmp As New Bitmap(PlateauPictureBox.Width, PlateauPictureBox.Height)
    dessinerobstacles(bmp)
    dessinerarbres(bmp)
    dessinerboids(bmp)
    dessinerrapaces(bmp)
    PlateauPictureBox.Image = bmp
End Sub
```

➢ FORMULAIRE : CONSOLE DE COMMANDE : *CONSOLE*

Nous retrouvons ici les parties assurant le bon fonctionnement de l'interface graphique mais qui assurent égelement avec les algorithmes.

```
Private Sub btncreerboids_Click(sender As Object, e As EventArgs) Handles btncreerboids.Click
```

```
    Dim n As Integer
    n = Val(hsnbboids.Value)
    creerboids(n)
    FormPlateau.Refresh()
  End Sub
```

```
Private Sub btncreerarbres_Click(sender As Object, e As EventArgs) Handles btncreerarbres.Click
    Dim n As Integer
    n = Val(hsnbarbres.Value)
    creerarbres(n)
    FormPlateau.Refresh()
  End Sub
```

```
Private Sub btncreerobstacles_Click(sender As Object, e As EventArgs) Handles btncreerobstacles.Click
    Dim n As Integer
    n = Val(hsnbobstacles.Value)
    creerobstacles(n)
    FormPlateau.Refresh()
  End Sub
```

```
Private Sub btnrapaces_Click(sender As Object, e As EventArgs) Handles btnrapaces.Click
    Dim n As Integer
    n = Val(hsnbrapaces.Value)
    Dim v As Double
    v = Val(hsvrapaces.Value)
    creerrapaces(n, v)
    FormPlateau.Refresh()
  End Sub
```

```
  Private Sub Console_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    FormPlateau.Show()
  End Sub
```

➢ MODULE

Nous avons fait le choix de ne pas créer de structure pour les vecteurs. Un vecteur sera donc défini par un point, on pourra déterminer sa norme grâce à la fonction distance appliquée à ce point et à l'origine du repère.

Nous retrouvons ici toutes les structures que nous avons créées pour notre projet :

```
Public Structure point
    Dim x As Double
    Dim y As Double
End Structure
```

```
Public Structure base_arbre
    Dim x As Double
    Dim y As Double
    Dim rayon As Double
    Dim cerises() As cerise
    Dim nbcerises As Integer
End Structure
```

```
Public Structure cerise
    Dim x As Double
    Dim y As Double
    Dim rayon As Double
End Structure
```

```
Public Structure rapace
    Dim masse As Integer 'sera un entier compris entre 6 et 10 defini aleatoirement pour chaque boid
    Dim p As point 'position du rapace
    Dim d As point 'vecteur direction du rapace
    Dim id As Integer 'identite du rapace definie par un numero
    Dim acc As point 'vecteur acceleration du rapace
    Dim vitesseR As Double 'vitesse du rapace
    Dim cible As Integer 'indice du boid cible : les rapaces mangent les boids
End Structure
```

```
Public Structure base_obstacle
    Dim x As Double
    Dim y As Double
End Structure
```

```
Public Structure boid
    Dim masse As Integer ' la masse sera un entier compris entre 3 et 5 défini aléatoirement pour chaque boid
    Dim p As point 'position du boid
    Dim d As point 'vecteur direction du boid
    Dim id As Integer 'identité du boid définie par un numéro
    Dim acc As point 'vecteur accélération du boid
    Dim vitesse As point 'vecteur vitesse du boid
    Dim repas As Integer 'compte le nombre de cerises mangées
End Structure
```

```
    Public nbobstacles As Integer
    Public nbarbres As Integer
    Public nbboids As Integer
    Public nbrapaces As Integer
    'on pourra mettre en jeu jusqu' à 10 arbres et 10 maisons
    Public rapaces() As rapace
    Public obstacles() As base_obstacle
    Public arbres() As base_arbre
    Public boids() As boid 'on met un nombre boids en jeu, ce nombre pouvant être modifié à l'aide d'un
curseur sur le plateau de jeu


    Public Const plateauX As Integer = 700      'La dimension en x du plateau (largeur)
    Public Const plateauY As Integer = 700      'La dimension en y du plateau (hauteur)
```

Les structures sont suivies des principaux modules assurant le bon fonctionnement du projet :

```
Sub creerobstacles(ByVal nbr As Integer)
        nbobstacles = nbr
        If nbr > 10 Then
            MsgBox("le nombre d'obstacles doit être inferieur ou égal a 10")
        Else
            Dim i As Integer
            ReDim obstacles(nbr - 1)
            Dim rdm As Random
            rdm = New Random()
            For i = 0 To nbobstacles - 1
                'on fait - 40 pour éviter qu'un obstacle se dessine au bord
                obstacles(i).x = (plateauX - 40) * rdm.NextDouble()
                obstacles(i).y = (plateauY - 40) * rdm.NextDouble()
            Next
        End If
    End Sub
```

```
    Sub creerarbres(ByVal nbr As Integer)
        nbarbres = nbr
        If nbr > 10 Then
            MsgBox("le nombre d obstacles doit être inférieur ou égal a 10")
        Else
            Dim i As Integer
            ReDim arbres(nbr - 1)
            Dim rdm As Random
            rdm = New Random()
            For i = 0 To nbarbres - 1
```

```
            'on fait - 40 pour éviter qu'un arbre se dessine au bord
            arbres(i).x = (plateauX - 40) * rdm.NextDouble()
            arbres(i).y = (plateauY - 40) * rdm.NextDouble()
            arbres(i).rayon = 20
            Dim j As Integer
            Randomize()
            'nbcerises est le nombre de cerises par arbres
            arbres(i).nbcerises = Int(8 * Rnd()) + 2 'définition d un nombre aléatoire entier entre 2 et 8
            ReDim arbres(i).cerises(arbres(i).nbcerises - 1)
            For j = 0 To arbres(i).nbcerises - 1
              'position initiale définie aléatoirement sur chaque arbre
              Randomize()
              arbres(i).cerises(j).x = arbres(i).x - 10 + 10 * rdm.NextDouble()
              arbres(i).cerises(j).y = arbres(i).y + 10 - 10 * rdm.NextDouble()
              arbres(i).cerises(j).rayon = 5
            Next
          Next
        End If
      End Sub
```

```
Sub creerboids(ByVal nbr As Integer)
    nbboids = nbr
    If nbr > 100 Then
      MsgBox("le nbr de boids doit etre inferieur ou egal a 100")
    Else
      Dim i As Integer
      ReDim boids(nbr - 1)
      Dim rdm As Random
      rdm = New Random()
      For i = 0 To nbboids - 1
        'position initiale definie aleatoirement sur le plateau
        boids(i).p.x = plateauX * rdm.NextDouble()
        boids(i).p.y = plateauY * rdm.NextDouble()
        boids(i).d.x = rdm.NextDouble()
        boids(i).d.y = rdm.NextDouble()
        boids(i).vitesse.x = rdm.NextDouble()
        boids(i).vitesse.y = rdm.NextDouble()
        boids(i).acc.x = 1
        boids(i).acc.y = 1
        boids(i).repas = 0
        boids(i).id = i
        Randomize()
        boids(i).masse = Int(5 * Rnd()) + 3 'definition d un nombre aléatoire entier entre 3 et 5
      Next
    End If
```

```
End Sub
```

```
Sub creerrapaces(ByVal nbr As Integer, ByVal v As Double)
    nbrapaces = nbr
    Dim i As Integer
    ReDim rapaces(nbr - 1)
    Dim rdm As Random
    rdm = New Random()
    For i = 0 To nbrapaces - 1
      'position initiale definie aleatoirement sur le plateau
      rapaces(i).p.x = plateauX * rdm.NextDouble()
      rapaces(i).p.y = plateauY * rdm.NextDouble()
      rapaces(i).d.x = rdm.NextDouble()
      rapaces(i).d.y = rdm.NextDouble()
      rapaces(i).vitesseR = v
      rapaces(i).acc.x = 1
      rapaces(i).acc.y = 1
      rapaces(i).id = i
      rapaces(i).cible = i
      Randomize()
      rapaces(i).masse = Int(10 * Rnd()) + 6 'definition d un nombre aléatoire entier entre 6 et 10
    Next
  End Sub
```

FORMULES NECESSAIRES

Notre projet repose sur un grand nombre de formule mathématiques, ces formules sont indispensables pour les différents modules et fonctions :

➢ MODULE

```
Public Const PI As Double = 3.14159265358979

  'Donne la valeur de l'angle entre 2 points (1 et 2) en degré :
  Public Function calculangle(ByVal x1 As Double, ByVal x2 As Double, ByVal y1 As Double, ByVal y2 As
Double) As Double
    Return Math.Acos((x1 * x2 + y1 * y2) / (Math.Sqrt(x1 ^ 2 + y1 ^ 2) * Math.Sqrt(x2 ^ 2 + y2 ^ 2))) * 180 / PI
  End Function
```

```
  'Normalise le vecteur de direction d un boid :
  'b : le boid dont on doit normaliser le vecteur de direction
  Public Sub normaliser(ByRef v As point)
    Dim n As Double
    n = Math.Sqrt(v.x ^ 2 + v.y ^ 2)
    v.x = v.x / n
    v.y = v.y / n
  End Sub
```

```
  'calcule la distance entre 2 points : p1 et p2
  Public Function distance(ByVal p1x As Double, ByVal p1y As Double, ByVal p2x As Double, ByVal p2y As
Double) As Double
    Return Math.Sqrt((p2x - p1x) ^ 2 + (p2y - p1y) ^ 2)
  End Function
```

```
  'la fonction rotation fait la rotation d un vecteur défini par le point p selon l angle a (en radian)
  'visual studio utilise les radians comme unite
  Public Function rotation(ByVal p As point, ByVal a As Double) As point
    Dim q As point
    q.x = p.x * Math.Cos(a) - p.y * Math.Sin(a)
    q.y = p.x * Math.Sin(a) + p.y * Math.Cos(a)
    Return q
  End Function
```

COMPORTEMENT DE BASE

➢ MODULE

Nous avons 4 méthodes à écrire pour les forces :

- ❖ Répulsion
- ❖ Alignement
- ❖ Séparation
- ❖ Cohésion

Chacune de ces méthodes doit ressortir un vecteur. De plus nous devons ensuite écrire différentes méthodes pour calculer le déplacement des boids au cours du temps :

- ❖ Une formule pour la position
- ❖ Une formule pour la vitesse

On ajoute une fonction qui ressort vrai ou faux en fonction de si un boid est dans le champ de vision d'un autre boid ou pas. Ici G un boid, B un autre boid, la fonction retourne vrai si le boid B est dans le champ de vision du boid G, dont le rayon du champ est spécifié dans "champ".

```
Public Function vision(ByVal G As boid, ByVal B As boid, ByVal champ As Integer) As Boolean
    If G.id <> B.id Then
        'On vérifie que la distance entre un boid et les autres est inferieure au rayon du champ de vision
        If distance(B.p.x, B.p.y, G.p.x, G.p.y) < champ Then
            'ensuite on vérifie que l'angle entre le vecteur qui va de notre boid à l'autre boid et le vecteur de
direction du boid est inferieur à 135 degrés
            If Math.Abs(calculangle(G.d.x, B.p.x - G.p.x, G.d.y, B.p.y - G.p.y)) < 135 Then
                Return True
            Else
                Return False
            End If
        Else
            Return False
        End If
    Else
        Return False
    End If
End Function
```

```
'mouvement est une fonction qui retourne le vecteur position du boid
Function mouvement(ByVal b As boid) As point
    Dim Xp As point
    Xp.x = (b.p.x + vitesse(b).x)
```

```
    Xp.y = (b.p.y + vitesse(b).y)
    If Xp.x < 0 Then
      Xp.x = (plateauX + (b.p.x + vitesse(b).x)) Mod plateauX
    Else
      Xp.x = (b.p.x + vitesse(b).x) Mod plateauX
    End If
    If Xp.y < 0 Then
      Xp.y = (plateauY + (b.p.y + vitesse(b).y)) Mod plateauY
    Else
      Xp.y = (b.p.y + vitesse(b).y) Mod plateauY
    End If
    Return Xp
    'Xp est le vecteur position du boid b
  End Function
```

```
  'vitesse est une fonction qui retourne le vecteur vitesse du boid
  Function vitesse(ByVal b As boid) As point
    Dim Xv As point
    Xv.x = b.vitesse.x + b.acc.x
    Xv.y = b.vitesse.y + b.acc.y
    normaliser(Xv)
    Return Xv
  End Function
```

```
  'alignement fait la moyenne des vecteurs directions des boids autour
  'cette fonction ressort un vecteur de force d'alignement (qui est le vecteur moyen + le vecteur de direction du boid)
  Function alignement(ByVal B As boid) As point
    Dim R As point
    Dim i As Integer
    Dim cpt As Integer = 1 'compte le nombre de boids présents dans le champ de vision du boid
    Dim cptx As Double = B.d.x 'fait la somme des x des vecteurs directions des boids
    Dim cpty As Double = B.d.y 'fait la somme des y des vecteurs directions des boids
    For i = 0 To nbboids - 1
      'force d'alignement : rayon du champ 60
      If vision(B, boids(i), 60) = True Then
        normaliser(boids(i).d)
        cptx = cptx + boids(i).d.x
        cpty = cpty + boids(i).d.y
        cpt = cpt + 1
      End If
    Next
    'cpt <> 1 signifie que le boid n'est pas seul : auquel cas la force s'applique
    If cpt <> 0 And cpt <> 1 Then
      R.x = (cptx / cpt) + B.d.x
```

```
        R.y = (cpty / cpt) + B.d.y
        Return R
      Else
        R.x = 0
        R.y = 0
        Return R
      End If
    End Function
```

```
    'cohésion calcule le vecteur qui va du boid au centre du groupe de boids
    'cohésion ressort un vecteur de force de cohésion (qui est le vecteur entre la direction de B et le centre du groupe)
    Function cohesion(ByRef B As boid) As point
      Dim R As point
      Dim i As Integer
      Dim cpt As Integer = 0 'compte le nombre de boids présents dans le champ de vision d'un boid
      Dim cptx As Double = 0 'fait la somme des x des vecteurs positions des boids
      Dim cpty As Double = 0 'fait la somme des x des vecteurs positions des boids
      For i = 0 To nbboids - 1
        'force de cohésion : rayon du champ 80
        If vision(B, boids(i), 80) = True Then

          cptx = cptx + boids(i).p.x
          cpty = cpty + boids(i).p.y
          cpt = cpt + 1

        End If

        If cpt <> 0 Then
          'si la force de séparation s'applique parce que les boids sont trop proches, il ne faut pas que la cohésion intervienne
          If distance(B.p.x, B.p.y, boids(i).p.x, boids(i).p.y) > 30 Then
            'on crée le point g : barycentre des positions des boids dans le champ
            Dim g As point
            g.x = cptx / cpt
            g.y = cpty / cpt
            R.x = g.x - B.d.x
            R.y = g.y - B.d.y
            Return R
          Else
            R.x = 0
            R.y = 0
            Return R
          End If
        Else
          R.x = 0
          R.y = 0
```

```
            Return R
        End If
    Next
End Function
```

```
'séparation ressort un vecteur de force de séparation (qui est l'opposé du vecteur entre B et le boid + le vecteur direction du boid)
Function separation(ByVal B As boid) As point
    Dim R As point
    Dim i As Integer
    Dim j As Integer
    Dim min As Integer 'min représentera l'indice du boid le plus proche
    Dim tmp As Double
    tmp = 0
    Dim cpt As Double 'cpt va compter le nombre de boids vus par B
    cpt = 0
    Dim T(,) As Double 'tableau à 2 lignes une pour l'indice du boid, une pour la distance qu'il a avec B
    ReDim T(1, 99)
    For i = 0 To nbboids - 1
        'on va remplir un tableau des boids vus par B : le tableau T
        'force de séparation : rayon du champ 25
        If vision(B, boids(i), 25) = True Then
            T(0, cpt) = i
            T(1, cpt) = distance(B.p.x, B.p.y, boids(i).p.x, boids(i).p.y)
            cpt = cpt + 1
        End If
    Next
    'en sortie de cette boucle if : cpt représente le nombre de boids vus par B
    'maintenant on cherche le boid le plus proche pour créer une force de séparation avec ce dernier :
    If cpt = 0 Then 'B ne voit aucun boid autour de lui donc aucune force de séparation ne s applique
        R.x = 0
        R.y = 0
    Else
        For j = 1 To cpt - 1
            If T(1, j) < T(1, tmp) Then
                tmp = j
            End If
        Next
        'en sortie de cette boucle tmp représente l'indice de la colonne ou se trouve le boid le plus proche dans le tableau T
        min = T(0, tmp)
        R.x = -(boids(min).p.x - B.p.x) + (B.d.x)
        R.y = -(boids(min).p.y - B.p.y) + (B.d.y)
    End If

    Return R
End Function
```

```
  'répulsion ressort un vecteur de force de repulsion (qui est l oppose du vecteur entre le boid et l obstacle + le vecteur direction du boid)
  Function repulsion(ByVal B As boid) As point
    Dim R As point
    Dim i As Integer
    Dim j As Integer
    Dim min As Integer
    Dim tmp As Double
    tmp = 0
    Dim T(,) As Double 'tableau à 2 lignes une pour l'indice de l'obstacle, une pour la distance qu'il a avec B
    ReDim T(1, 9)
    For i = 0 To nbobstacles - 1
      T(0, i) = i
      T(1, i) = distance(B.p.x + B.d.x, B.p.y + B.d.y, obstacles(i).x, obstacles(i).y)
    Next
    For j = 1 To nbobstacles - 1
      If T(1, j) < T(1, tmp) Then
        tmp = j
      End If
    Next
    min = T(0, tmp)
    'en sortie de cette boucle tmp représente l'indice de la colonne ou se trouve l'obstacle le plus proche dans le tableau T
    'min est l'indice de l'obstacle le plus proche
    If T(1, tmp) < 40 Then
      R.x = -(obstacles(min).x - B.p.x) + (B.d.x)
      R.y = -(obstacles(min).y - B.p.y) + (B.d.y)
    Else
      R.x = 0
      R.y = 0
    End If
    Return R
  End Function
```

➢ FORMULAIRE : CONSOLE DE COMMANDE : *CONSOLE*

```
Private Sub Timer_pas_Tick(sender As Object, e As EventArgs) Handles Timer_pas.Tick

    Dim i As Integer
    For i = 0 To nbboids - 1
        boids(i).acc.x = (Val(hssep.Value) * separation(boids(i)).x + Val(hsco.Value) * cohesion(boids(i)).x +
Val(hsali.Value) * alignement(boids(i)).x + Val(hsrep.Value) * repulsion(boids(i)).x) / boids(i).masse
        boids(i).acc.y = (Val(hssep.Value) * separation(boids(i)).y + Val(hsco.Value) * cohesion(boids(i)).y +
Val(hsali.Value) * alignement(boids(i)).y + Val(hsrep.Value) * repulsion(boids(i)).y) / boids(i).masse

        Dim B As boid
        B.d.x = boids(i).d.x + boids(i).acc.x
        B.d.y = boids(i).d.y + boids(i).acc.y
        normaliser(B.d)
        boids(i).d = B.d

        boids(i).vitesse = vitesse(boids(i))
        boids(i).p = mouvement(boids(i))
        Dim k As Integer
        For k = 0 To nbarbres - 1
            mange(boids(i), arbres(k))
        Next
        mort_repas(boids(i))

        lblboidsmorts.Text = Str(boidsmorts)
        lblnbboids.Text = Str(hsnbboids.Value)
        lblboids.Text = Str(hsnbboids.Value - boidsmorts)
    Next

    Dim j As Integer
    For j = 0 To nbrapaces - 1
        rapaces(j).acc.x = (Val(hssep.Value) * separationR(rapaces(j)).x + Val(hsrep.Value) *
repulsionR(rapaces(j)).x + Val(hsali.Value) * alignementR(rapaces(j)).x) / rapaces(j).masse
        rapaces(j).acc.y = (Val(hssep.Value) * separationR(rapaces(j)).y + Val(hsrep.Value) *
repulsionR(rapaces(j)).y + Val(hsali.Value) * alignementR(rapaces(j)).y) / rapaces(j).masse

        Dim R As rapace
        R.d.x = comportementcible(rapaces(j)).x + rapaces(j).acc.x
        R.d.y = comportementcible(rapaces(j)).y + rapaces(j).acc.y
        normaliser(R.d)
        rapaces(j).d = R.d
        rapaces(j).p = mouvementR(rapaces(j))
        mangeboid(rapaces(j), boids)
```

```
        lblboidsmorts.Text = Str(boidsmorts)
        lblnbboids.Text = Str(hsnbboids.Value)
        lblboids.Text = Str(hsnbboids.Value - boidsmorts)
    Next


    FormPlateau.Refresh()
End Sub
```

```
Private Sub btnstart_Click(sender As Object, e As EventArgs) Handles btnstart.Click
    Timer_pas.Interval = 5
    Timer_pas.Enabled = True
    Timer_pas.Start()

End Sub
```

COMPORTEMENT APPROFONDI

➢ MODULE

On créé des rapaces qui vont manger les boids.

```
'retounre le vecteur direction du boid
Function comportementcible(ByRef r As rapace) As point
    Dim D As point
    'Aller vers la cible
    D.x = boids(BoidLePlusProche(r)).p.x - r.p.x
    D.y = boids(BoidLePlusProche(r)).p.y - r.p.y
    Return D
End Function
```

```
'Retourne imin l indice du boid le plus proche
Public Function BoidLePlusProche(ByVal r As rapace) As Integer
    Dim iMin As Integer
    Dim i As Integer
    Dim q As Integer
    q = r.cible
```

```
    'si il existe une cible + proche que celle de r alors la fonction retournera cette nouvelle cible :
    iMin = q
    For i = 0 To nbboids - 1
      If i <> q Then
        If distance(r.p.x, r.p.y, boids(i).p.x, boids(i).p.y) <= distance(r.p.x, r.p.y, boids(iMin).p.x,
boids(iMin).p.y) Then
          iMin = i
        End If
      End If
    Next
    Return iMin
  End Function
```

```
'G un rapace, B un autre rapace
  'la fonction retourne vrai si le rapace B est dans le champ de vision du rapace G, dont le rayon du champ est
specifie dans "champ"
  Public Function visionR(ByVal G As rapace, ByVal B As rapace, ByVal champ As Integer) As Boolean
    If G.id <> B.id Then
      'on verifie que la distance entre un rapace et les autres est inferieure au rayon du champ de vision
      If distance(B.p.x, B.p.y, G.p.x, G.p.y) < champ Then
        'ensuite on verifie que l angle entre le vecteur qui va de notre rapace a l autre rapace et le vecteur de
direction du rapace est inferieur a 135 degres
        If Math.Abs(calculangle(G.d.x, B.p.x - G.p.x, G.d.y, B.p.y - G.p.y)) < 135 Then
          Return True
        Else
          Return False
        End If
      Else
        Return False
      End If
    Else
      Return False
    End If
  End Function
```

```
'separation ressort un vecteur de force de separation (qui est l oppose du vecteur entre R et un autre rapace +
le vecteur direction du rapace)
  Function separationR(ByVal R As rapace) As point
    Dim B As point
    Dim i As Integer
    Dim j As Integer
    Dim min As Integer
    Dim tmp As Double
    tmp = 0
```

```
    Dim cpt As Double 'cpt va compter le nbr de rapaces vus par R
    cpt = 0
    Dim T(,) As Double 'tableau a 2 lignes une pour l indice du rapace, une pour la distance qu il a avec R
    ReDim T(1, 9)
    For i = 0 To nbrapaces - 1
      'on va remplir un tableau des rapaces vus par R : le tableau T
      'force de separation : rayon du champ 25
      If visionR(R, rapaces(i), 25) = True Then
        T(0, cpt) = i
        T(1, cpt) = distance(R.p.x, R.p.y, rapaces(i).p.x, rapaces(i).p.y)
        cpt = cpt + 1
      End If
    Next
    'en sortie de cette boucle if : cpt represente le nbr de rapaces vus par R
    'maintenant on cherche le rapace le plus proche pour creer une force de separation avec ce dernier :
    If cpt = 0 Then 'R ne voit aucun rapace autour de lui donc aucune force de separation ne s applique
      B.x = 0
      B.y = 0
    Else
      For j = 1 To cpt - 1
        If T(1, j) < T(1, tmp) Then
          tmp = j
        End If
      Next
      'en sortie de cette boucle tmp represente l indice de la colonne ou se trouve le rapace le plus proche
dans le tableau T
      'min representera l indice du rapace le plus proche
      min = T(0, tmp)
      B.x = -(rapaces(min).p.x - R.p.x) + (R.d.x)
      B.y = -(rapaces(min).p.y - R.p.y) + (R.d.y)
    End If

    Return B
  End Function
```

```
'cette fonction ressort un vecteur de force d alignement (qui est le vecteur direction de la cible + le vecteur de
direction du rapace)
  Function alignementR(ByVal R As rapace) As point
    Dim B As point
    B.x = boids(R.cible).p.x + R.d.x
    B.y = boids(R.cible).p.y + R.d.y

  End Function
```

```
'repulsion ressort un vecteur de force de repulsion (qui est l oppose du vecteur entre le rapace et l obstacle + le vecteur direction du rapace)
   Function repulsionR(ByVal R As rapace) As point
      Dim B As point
      Dim i As Integer
      Dim j As Integer
      Dim min As Integer
      Dim tmp As Double
      tmp = 0
      Dim T(,) As Double 'tableau a 2 lignes une pour l indice de l arbre, une pour la distance qu il a avec R
      ReDim T(1, 9)
      For i = 0 To nbarbres - 1
         T(0, i) = i
         T(1, i) = distance(R.p.x + R.d.x, R.p.y + R.d.y, arbres(i).x, arbres(i).y)
      Next
      For j = 1 To nbarbres - 1
         If T(1, j) < T(1, tmp) Then
            tmp = j
         End If
      Next
      min = T(0, tmp)
      'en sortie de cette boucle tmp represente l indice de la colonne ou se trouve l arbre le plus proche dans le tableau T
      'min est l indice de l arbre le plus proche
      If T(1, tmp) < 40 And nbarbres <> 0 Then
         B.x = -(arbres(min).x - R.p.x) + (R.d.x)
         B.y = -(arbres(min).y - R.p.y) + (R.d.y)
      Else
         B.x = 0
         B.y = 0
      End If
      Return B
   End Function
```

Les rapaces n'auront pas les mêmes formules pour le mouvement :

```
'mouvement est une fonction qui retourne le vecteur position du rapace
   Function mouvementR(ByVal r As rapace) As point
      Dim Xp As point
      Xp.x = r.p.x + r.d.x * r.vitesseR
      Xp.y = r.p.y + r.d.y * r.vitesseR
      If Xp.x < 0 Then
```

```
        Xp.x = (plateauX + (r.p.x + r.d.x * r.vitesseR)) Mod plateauX
    Else
        Xp.x = (r.p.x + r.d.x * r.vitesseR) Mod plateauX
    End If
    If Xp.y < 0 Then
        Xp.y = (plateauY + (r.p.y + r.d.y * r.vitesseR)) Mod plateauY
    Else
        Xp.y = (r.p.y + r.d.y * r.vitesseR) Mod plateauY
    End If
    Return Xp
    'Xp est le vecteur position du rapace r
  End Function
```

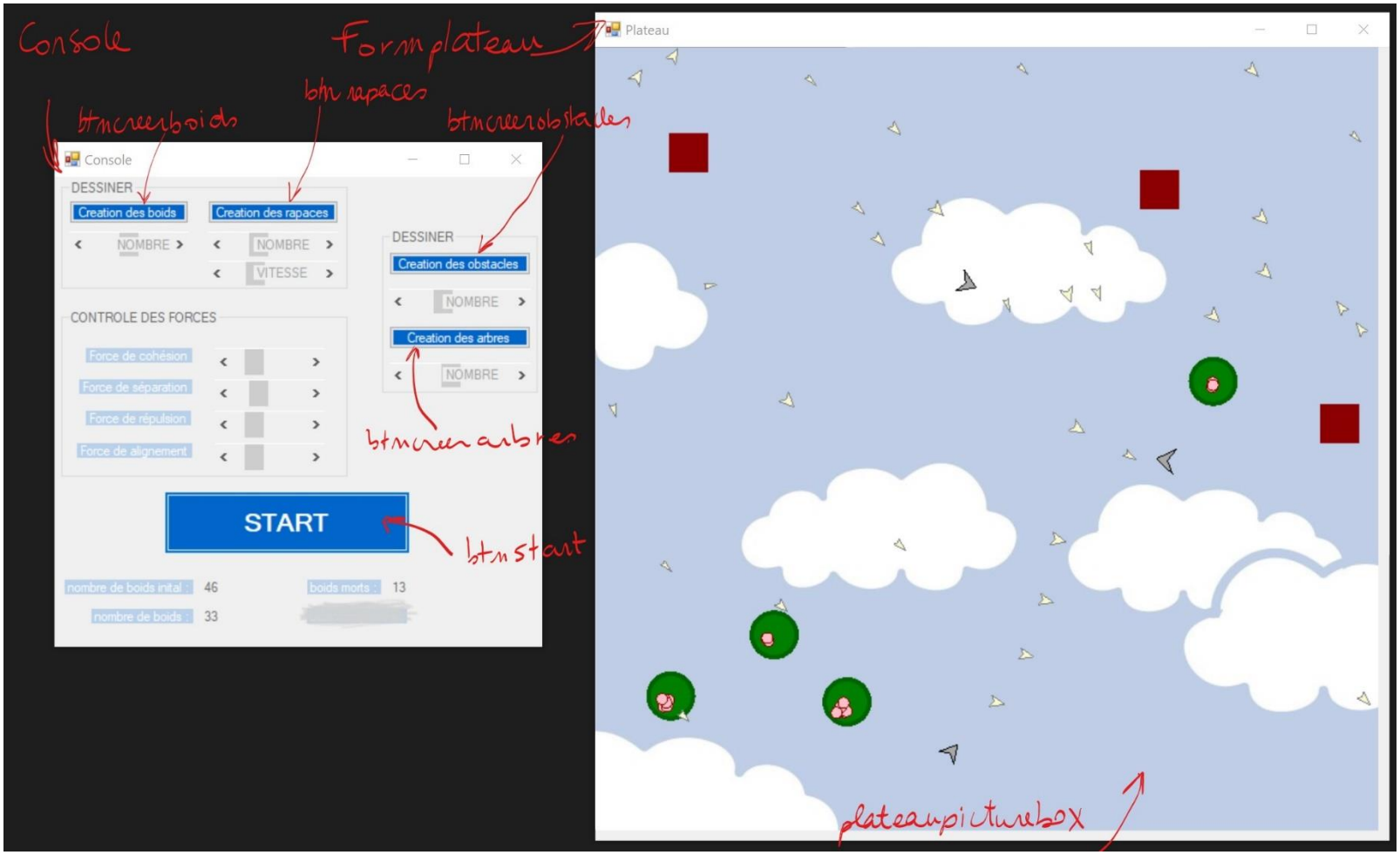
```
Public boidsmorts As Integer = 0

  'en passant sur un boid le rapace le mange
  'cette procedure modifie le tableau de boids :
  Sub mangeboid(ByRef r As rapace, ByRef boids() As boid)
    Dim j As Integer
    For j = 0 To nbboids - 1
      'si un rapace passe sur un boid alors il le mange et celui ci disparait :
      If r.p.x >= boids(j).p.x - 3 And r.p.x <= boids(j).p.x + 3 Then
        If r.p.y <= boids(j).p.y + 3 And r.p.y >= boids(j).p.y - 3 Then


          'a la place du boid mort on met le dernier boid du tableau :
          boids(j) = boids(nbboids - 1)


          boidsmorts = boidsmorts + 1
          nbboids = nbboids - 1
        End If
      End If
    Next
  End Sub
```

Et on créé des cerises qui seront mangées par les boids. On ajoute une règle : Les boids qui mangeront plus de cerises que la valeur de leur masse mourront.

```
'en passant sur une cerise un boid la mange
  'la cerise disparait
  'un boid peut manger plusieurs cerises a la fois
  Sub mange(ByRef b As boid, ByRef A As base_arbre)
```

```
    Dim d As Integer
    d = A.nbcerises
    Dim j As Integer
    For j = 0 To A.nbcerises - 1
      'si un boid passe sur une cerise alors il la mange et celle ci disparait :
      'on modifie le tableau des cerises de l arbre A en supprimant la case de la cerise
      If b.p.x >= A.cerises(j).x - 3 And b.p.x <= A.cerises(j).x + 3 Then
        If b.p.y <= A.cerises(j).y + 3 And b.p.y >= A.cerises(j).y - 3 Then
          'a la place de la cerise mangee on met la derniere cerise du tableau de cerises de l arbre :
          A.cerises(j) = A.cerises(A.nbcerises - 1)
          A.nbcerises = A.nbcerises - 1
        End If
      End If
    Next
    Dim c As Integer
    c = d - A.nbcerises
    b.repas = b.repas + c
  End Sub
```

```
'on ajoute dans la structure boid : Dim repas As Integer 'compte le nbr de cerises mangees
  'un boid ne doit pas avoir mange + de cerises que sa masse sinon il meurt
  'qd un boid meurt : un autre nait
  Sub mort_repas(ByVal b As boid)
    If b.repas >= b.masse Then
      Dim indice As Integer
      indice = b.id
      'le boid meurt :
      'a la place du boid mort on met le dernier boid du tableau :
      boids(indice) = boids(nbboids - 1)
      boidsmorts = boidsmorts + 1
      nbboids = nbboids - 1
    End If
  End Sub
```


Console
Form plateau
btn creer boids
btn rapaces
btn creer obstacles
Console
DESSINER
Creation des boids
NOMBRE
Creation des rapaces
NOMBRE
VITESSE
DESSINER
Creation des obstacles
NOMBRE
Creation des arbres
NOMBRE
CONTROLE DES FORCES
Force de cohésion
Force de séparation
Force de répulsion
Force de alignement
START
btn creer arbres
btn start
nombre de boids initial :
46
boids morts :
13
nombre de boids :
33
Plateau
plateau picture box