Université du Québec en Outaouais
Département d'informatique et d'ingénierie
Automne 2006

Karim El Jaouhari

Rapport final du projet synthèse :

Intégration d'une application Java pour la visualisation 3D d'un signal enregistré par un système de mesure de communication sans fil

Travail présenté à
Monsieur Larbi Talbi, Ph.D
Dans le cadre du cours
Projet synthèse Inf 4173

TABLE DES MATIÈRES

TABLE DES FIGURES

RÉSUMÉ

Le but principal de ce projet était l'intégration d'un graphique 3D dans un projet d'acquisition de données sans fil afin de visualiser un signal RF en fonction du trajet d'une antenne réceptrice sur le plan XY. L'utilité devient apparente lorsqu'on nécessite des mesures dans un environnement où l'interférence humaine est inacceptable et qu'on désire observer les résultats obtenus en temps réel. Ceci permet d'économiser du temps et donc de faire plus de tests et de consacrer notre temps à des tâches plus importantes.

Ma tache principale est d'implémenter une application qui représentera un graphique ayant l'allure 3D, c'est à dire avec des ombrages ou des reflets lumineux aidant à visualiser les graphiques comme déjà réalisé dans Matlab. Cela en utilisant le même code d'application JAVA existante, mais cette tâche c'est avérée plus difficile que je pensais. De ce fait, après le diagnostique, l'analyse et une étude approfondis du code existant, je me suis rendu compte que vaut mieux développer une application séparée indépendante. Cette dernière, recevra les données de l'application JAVA existant. (Le problème ici réside au faite que tout le code est en oriente Objet utilisant des classes, qui servent instancier des objet en temps real et non des fonctions qui sont pas prédéfinis pour la réalisation d'un graphe 3D en JAVA).

Finalement mon choix est tombé sur la programmation sous le Visuel Basic Version 6 comme langage de développement pour la future application (Raison de ma performance et ma maîtrise de ce langage de programmation et sa vaste possibilité dans le graphisme 3D).

1. INTRODUCTION

Le projet d'Automne que j'ai entrepris étant maintenant terminé, je présente le rapport final expliquant toutes les étapes que j'ai entrepris pour accomplir les tâches qui m'ont été assignées. Le titre de ce projet est ***Intégration d'une application Java pour la visualisation 3D d'un signal enregistré par un système de mesure de communication sans fil*** sous la supervision de M. Larbi Talbi.

Mon plan de travail et de progression a été suivi assez bien car j'ai pu accomplir toute la tâche demandée dans le temps alloué. De plus, j'ai même pu rapporté des améliorations qui ne m'ont pas étaient assignées dans la description de mes taches.

Malgré tous difficultés rencontrées au cours de cette session (Étude et compréhension du code de l'application existante, difficulté de comprendre et manipulé le matériel du projet), tout s'est déroulé relativement bien et j'ai même eu le temps d'entreprendre quelques tâches supplémentaires, comme l'ajout de l'interface de control des couleurs, et la manipulations du graphe avec possibilités de rotation sur les 3 axes de plan 3D (axe x ou y ou bien Z) et finalement le retraçage de la fonction qui écris le signal qui représente la puissance reçu à chaque position (X,Y) Fonction F(x,y,Puissance) et le déplacement de le véhicule F(x,y). Ainsi l'affichage des données par l'utilisateur.

Le présent document explique en détail toutes les étapes que j'ai entamé pour la réalisation durant ce projet ainsi que les difficultés que j'ai confrontées et les recherches que j'ai faites. Les figures référencées sont trouvées directement dans le texte et non en annexe afin de faciliter la compréhension. Vous trouverez en annexe toutes lignes de code Visuel Basic de l'application.

Les références trouvées dans le texte entre parenthèses sont soit à des figures (par exemple **fig. 2**) ou à des références bibliographiques (par exemple **[2]**).

2. IMPLÉMENTATION

2.1 Visualisation 3D du signal RF (Application JAVA existant et celle du la nouvelle version fait en Visuel Basic 6)

Ce projet a pour objectif de créer un graphique en trois dimensions qui se traçait en temps réel. Le graphique représente la valeur d'un signal RF en décibels (dB) en fonction de la position de l'antenne réceptrice dans le plan XY. Pour comprendre ce fonctionnement, je dois tout d'abord expliquer le fonctionnement de la PIAM (Plateforme Autonome Intelligente et Mobile), la machine sur laquelle je travaillais cet été.

Tout d'abord, la PIAM est une machine développée par Alexandre Perron et Vincent Bergeron, étudiants passés en génie informatique, ayant comme but de prendre des mesures de puissances dans des endroits hostiles, par exemple où l'interférence humaine est inacceptable. Le code de la machine, soit l'interface usager, est programmé en JAVA. Ce code communique avec le microcontrôleur de la machine (PIC) afin d'envoyer des données de position et de puissance. Voici une prise de vue de l'interface usager de la PIAM (**fig. 1**):

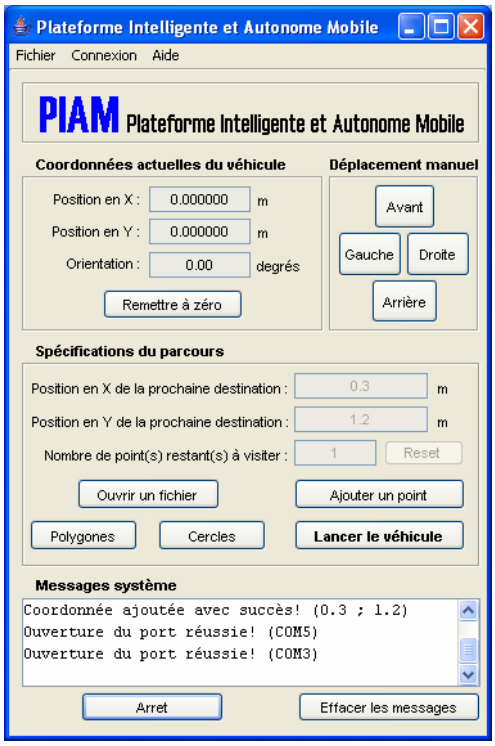

Fig. 1 Interface usager de la PIAM

Afin de faire fonctionner la machine, on doit tout d'abord établir la connexion avec la clé Bluetooth, ce qui permet à l'usager de communiquer avec la machine sans-fil (**fig. 2**). De plus, on doit spécifier sur quel port se trouvent les modules d'acquisition de données afin de recevoir les valeurs de puissance.

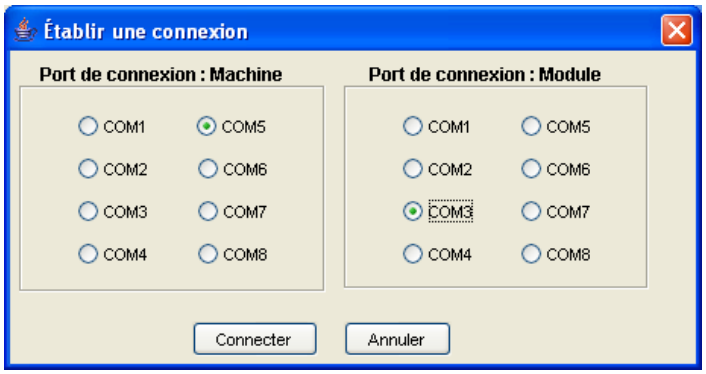

Fig. 2 Menu de connexion de la PIAM

Ensuite, on peut entrer une valeur en X et une valeur en Y comme coordonnées (**fig. 3 et fig. 4**) puis lancer la machine vers sa destination en lui spécifiant un nom de fichier dans lequel les données seront sauvegardées (**fig. 5**). Par défaut, le fichier est sauvegardé sous format *.xls, et placé dans le répertoire du code de la machine.

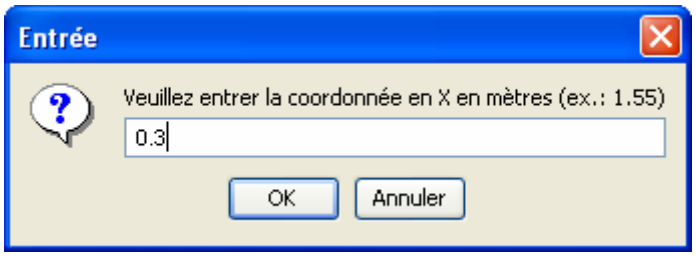

Fig. 3 Coordonnée en X

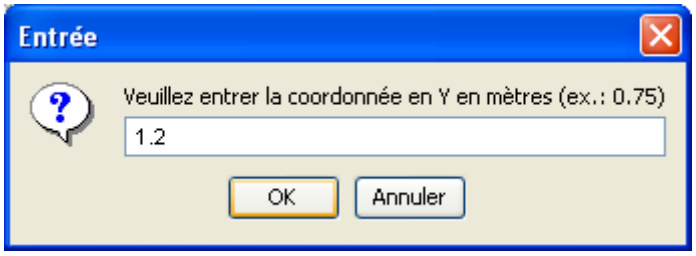

Fig. 4 Coordonnée en Y

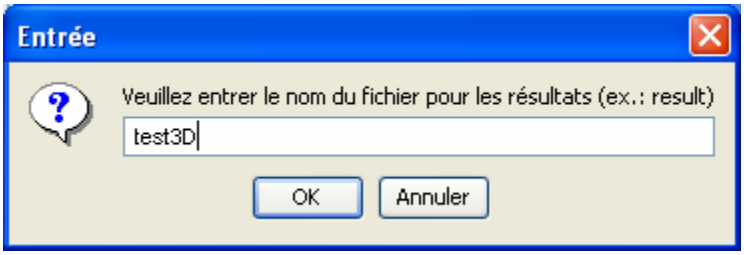

Fig. 5 Nom du fichier de sauvegarde

En cliquant sur OK de cette dernière fenêtre, on lance la machine vers sa destination puis c'est là que la nouvelle fenêtre contenant le graphique 3D s'ouvre.

Pour accomplir cette tâche, soit de créer le graphique 3D, les recherches à faire étaient nombreuses car JAVA n'est pas mon domaine d'expertise. J'ai donc dû faire des recherches sur le 3D en JAVA, ce qui est un domaine assez compliqué à comprendre pour un amateur. Après avoir cherché sur de nombreux sites Internet concernant ce sujet et avoir feuilleté à travers quelques livres portant sur JAVA, j'ai pu constater que je cherchais peut-être un peu trop fort pour quelque chose que je n'allais pas trouver. Je voulais trouver une classe, ou une utilité en Java, me permettant de faire directement un environnement 3D, c'est à dire un graphique comportant trois axes visibles afin de pouvoir lui passer des valeurs comme les coordonnées en X, en Y et aussi la puissance, et de pouvoir les dessiner directement en transformant ces valeurs en coordonnées à l'écran. Le concept est simple, comme coordonnées je reçois des valeurs en mètres et comme puissance je reçois des valeurs en Volts. Je devais alors les transformer en valeurs de pixels afin de bien visualiser les données.

Je soutiens encore qu'il doit y a voir un environnement 3D déjà créé en JAVA, où je pourrais visualiser le graphique et changer son point de vue, soit la position de la « caméra », en cliquant sur le graphique et en le déplaçant (*drag & drop*), mais je n'ai pas pu trouver comment faire avec les nombreuses recherches que j'ai faites, alors j'ai décidé de faire un graphique à trois axes, c'est-à-dire en 3D, à la main afin de visualiser la puissance relative des signaux captées par la machine en fonction de sa position dans le plan X-Y.

Pour ce faire, je me suis basé sur les graphiques 2D de l'an passé (**fig. 6 et fig. 7**) mais j'ai dû les modifier de façon extensive. Les données utilisées dans ce graphique proviennent de différentes classes alors j'ai dû m'assurer d'une bonne communication entre celles-ci. Au début de mon stage, j'ai essayé de rendre les données accessibles en créant une instance de leur classe mais je recevais toujours des zéros en prenant cette méthode. En faisant un peu plus de lecture [**7**], je me suis aperçu que je pouvais déclarer les variables en question comme étant *''static public''*. Ensuite, j'ai dû tout placer dans des listes chaînées séparées, c'est-à-dire une pour les X, une pour les Y et une pour la puissance. Le microcontrôleur de la machine retourne ces trois données au programme JAVA en même temps, à chaque 4 millimètres de son trajet. De cette façon, il est facile d'associer une valeur de puissance à une coordonnée. Ensuite, il s'agit juste de rafraîchir

l'espace de graphique (fonction *repaint()* de JAVA) à chaque fois qu'un nouveau point est ajouté aux listes chaînées.

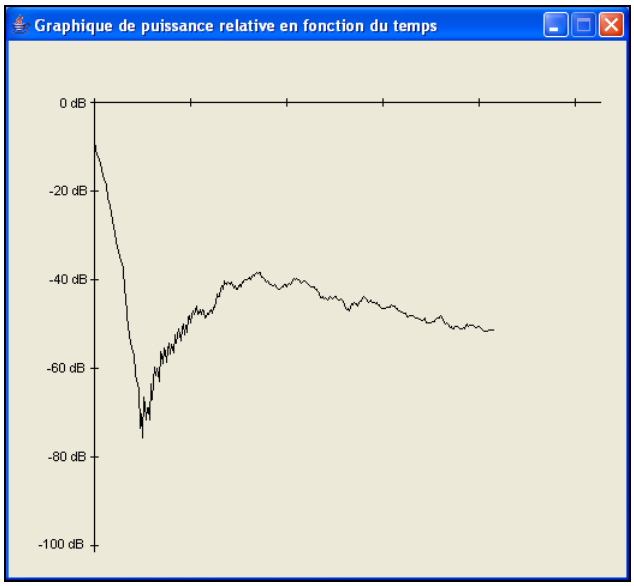

Fig. 6 Ancien graphe de la puissance relative en fonction du temps (temps réel)

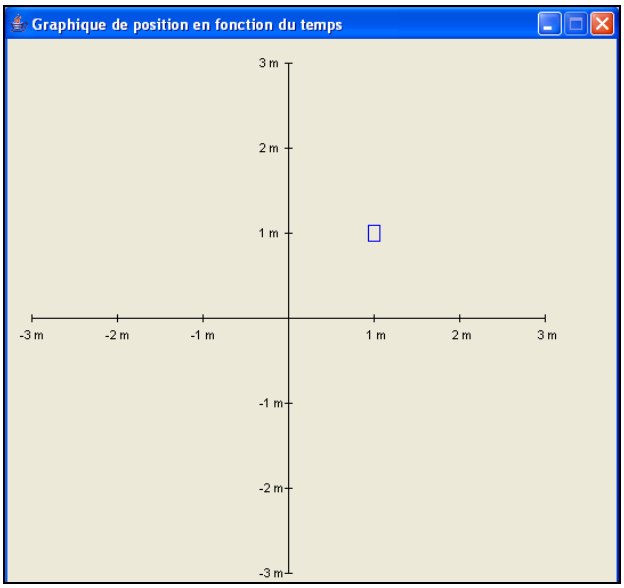

Fig. 7 Ancien graphe de la position (temps réel)

Le prochain problème était de rendre les données de position et de puissance en coordonnées à l'écran, car vu qu'on a maintenant trois axes à l'écran, le plan X-Y est incliné et donc les coordonnées à l'écran (les emplacements des pixels) ne correspondent pas aux coordonnées que je capte de la machine. J'ai donc fait un dessin à la main (**fig. 8**) afin de comprendre ce que j'étais en train d'aborder.

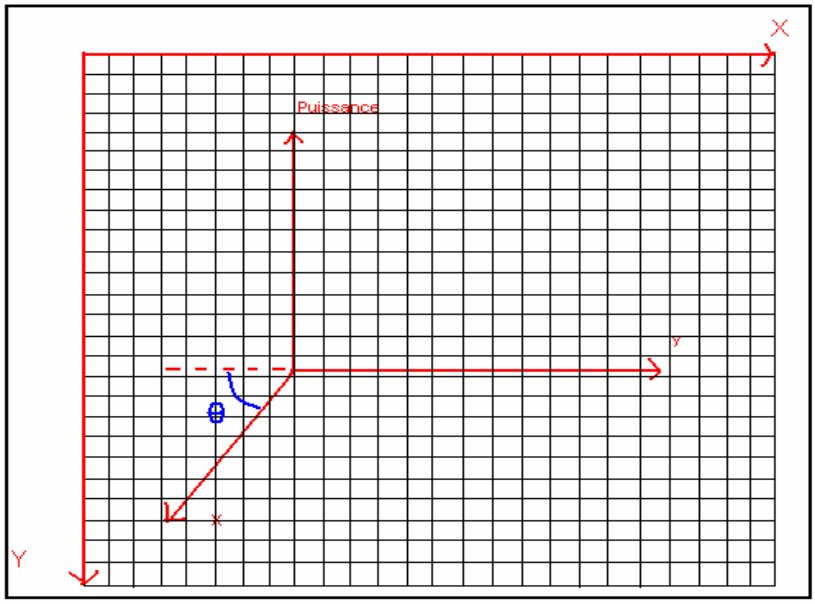

Fig. 8 Dessin des trois axes sur un dessin de l'écran

En connaissant l'angle thêta (Ɵ) et en sachant les dimensions (en pixels) de l'écran, je suis en mesure de savoir quelle proportion donner aux coordonnées reçues. En calculant les valeurs en fonction d'un écran de 520 x 480 pixels puis en fonction du placement de repère à trois axes désiré, j'ai trouvé les équations nécessaires. Elles peuvent être visualisées **en gras** dans la **section 6.1** de l'annexe. Ces équations représentent les valeurs de X et de Y visualisées à l'écran en fonction des valeurs des coordonnées et de la puissance reçues de la machine. De plus, j'ai rajouté des proportions aux équations qui ont été calculées de façon à ce que les coordonnées les plus éloignées de l'origine sont de 3m en X et de 3m en Y. Ceci me permet de toujours obtenir un graphique bien proportionné. Ensuite, j'ai rajouté une ligne en rouge représentant le trajet à faire afin de mieux visualiser en trois dimensions. L'ancien graphique est visualisé à la **figure 9**.

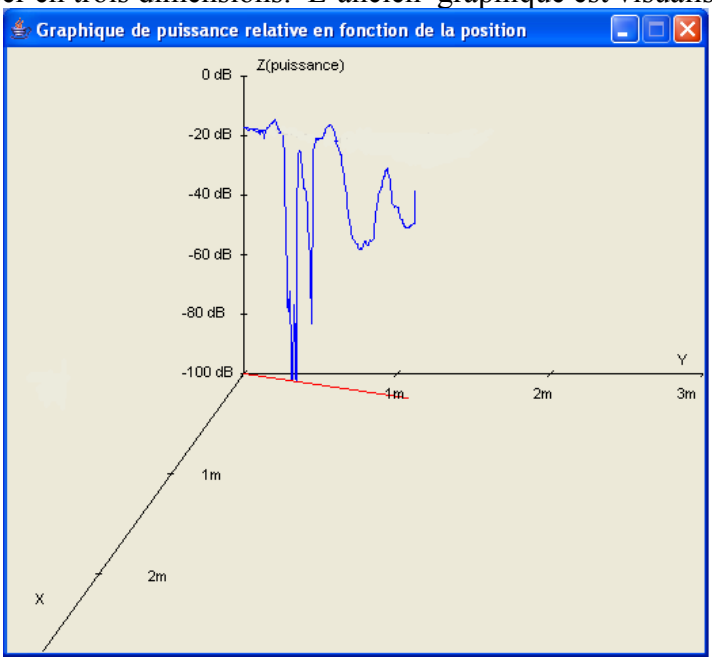

Fig. 9 Ancien graphique de la puissance relative en fonction de la position

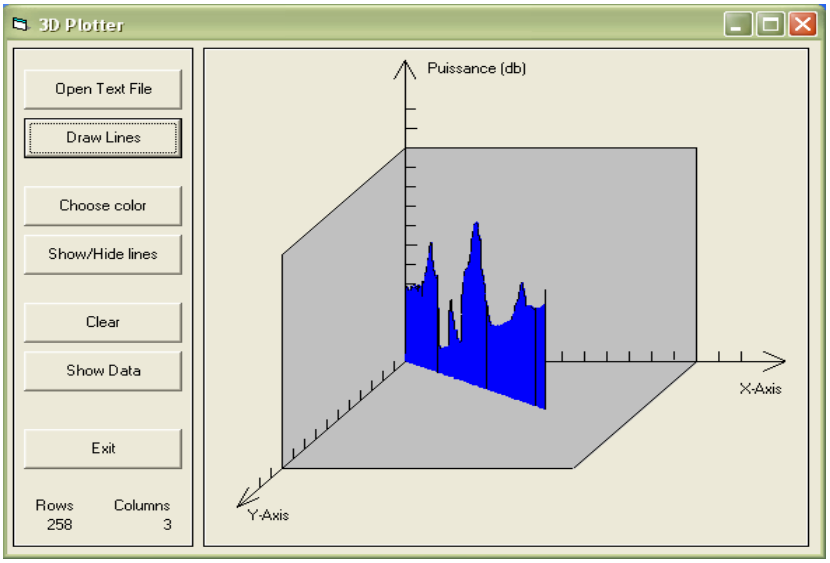

Fig. 10 Nouveau graphique de la puissance relative en fonction de la position

Cette partie n'était pas trop difficile mais plutôt longue car j'ai essayé plusieurs différentes solutions avant d'arriver à celle-ci, par exemple j'ai essayé des utilités distribuées par différentes compagnies pensant que j'allais trouver une solution idéale mais je pense que cette solution réponds à tous les besoins puis elle est facile à comprendre puis à modifier au besoin.

Un exemple du code utilisé pour rendre ce graphique peut être visualisé à la **section 6.1** de l'annexe.

2.2 Améliorations rapportés

Après avoir achevé avec succès les taches demandées, j'ai fait une présentation du travail réalisé à M. Larbi Talbi, qui m'a demandé de me concentrer sur les améliorations suggérées qui peuvent être très utiles pour le projet.
A ce niveau là, la programmation 3D est devenue mon grand défit.
- Implémentation d'une interface pour la manipulation du graphe (Comme Draw lines qui une option de trace le graphe Buttons de modifications des couleurs, affichage des données…)

Fig. 11 Menu de choix de couleur

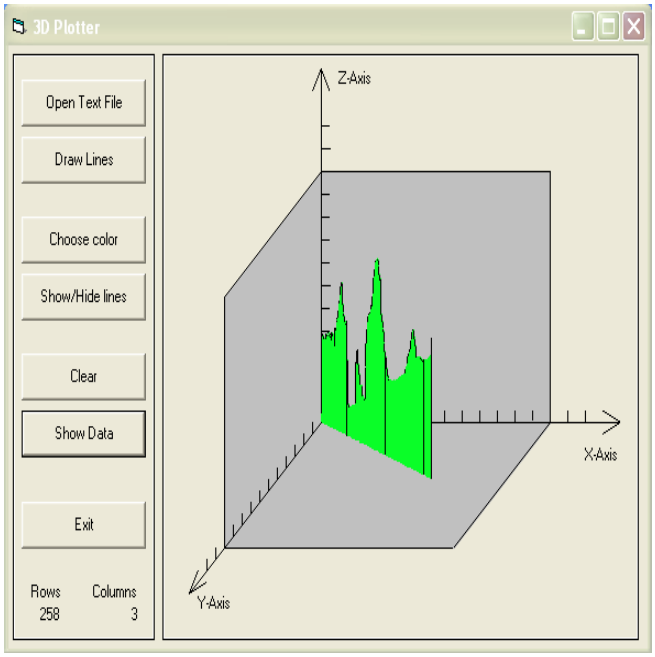

Fig. 12 Graphe avec couleur Verte

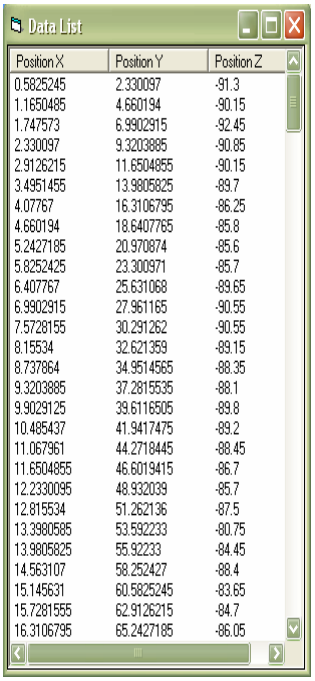

Fig. 13 Affichage des données

- Ajout du bouton de choix d'afficher (de la courbe de la puissance F(X, Y, P) et celle de déplacement F'(X, Y) seulement) ou bien de graphe 3D seulement.

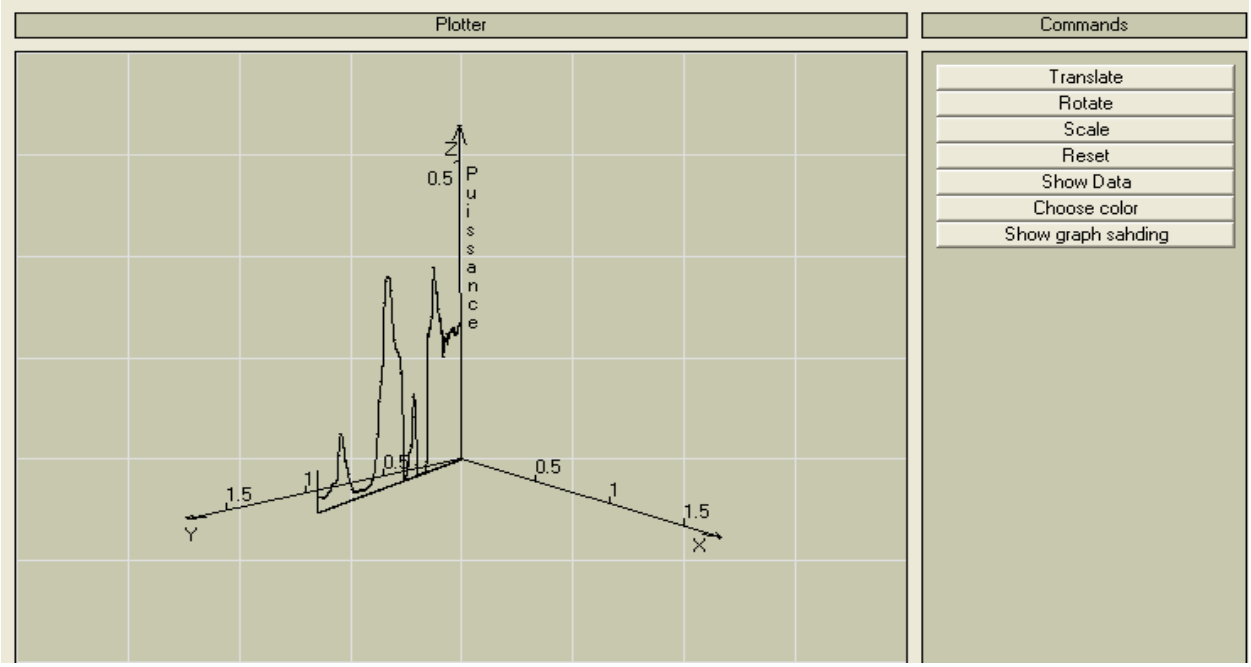

Fig. 14 Graphe de la puissance et de déplacement

La tache qui marquera le projet est bien celle qui suit et consiste à la Manipulation de la Rotation du graphe sur les 3 axes des environnement 3D (rotation sur le X, Y et le Z)

- Translation, Rotation ainsi que scaler le graphe.)

-Recherche approfondis de la programmation des fonctions Mathématiques et graphiques

-Opérations arithmétiques sur les Matrices (Multiplication, addition, division…)

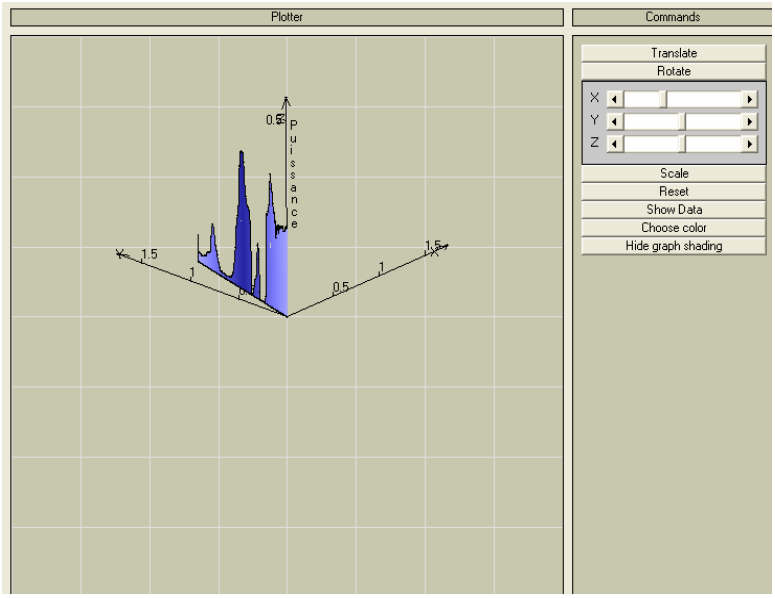

Fig. 15 Rotation du graphe selon l'axe X

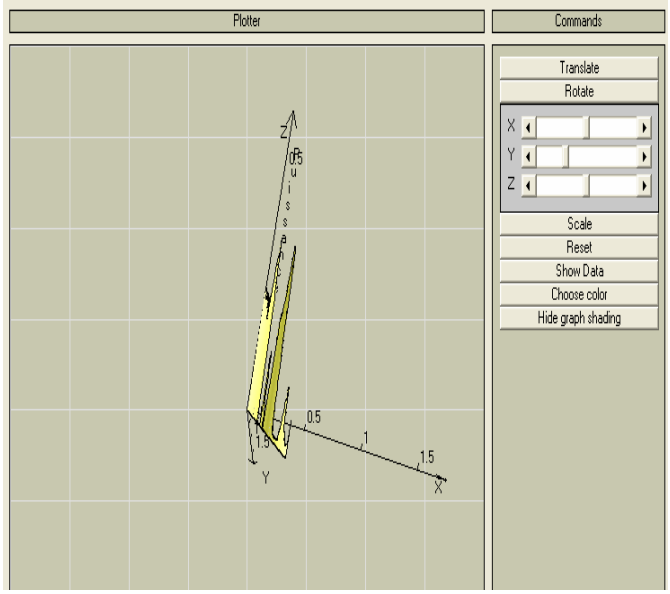

Fig. 16 Rotation du graphe selon l'axe Y

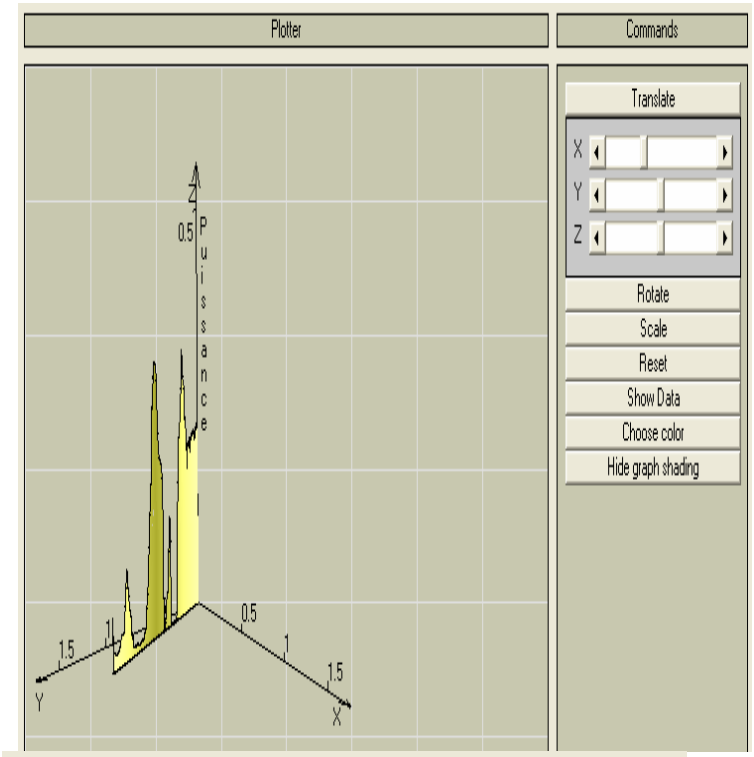

Fig. 17 Translation du graphe selon l'axe X

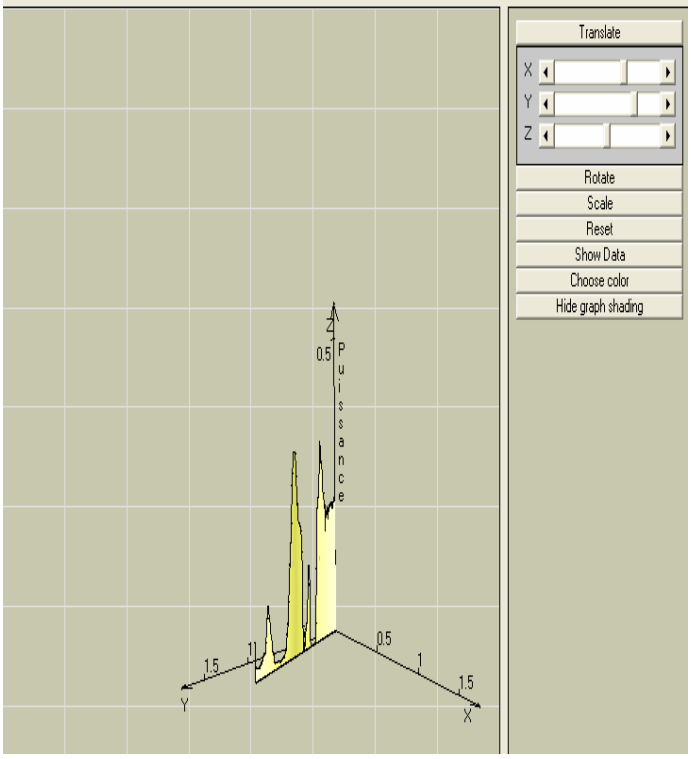

Fig. 18 Translation du graphe selon l'axe Y

3. VALIDATION EXPÉRIMENTALE

Afin de vérifier l'exactitude de mes résultats du graphique 3D, j'ai décidé d'entrer les données reçues de la PIAM dans Matlab et de comparer les deux graphiques, soit celui de Matlab et celui tracé par application JAVA existante et celui de l'application de Visuel Basic
Pour accomplir cela, il s'agissait simplement de prendre une prise de vue du graphique tracé et ensuite d'importer les données enregistrées par le logiciel dans Matlab.
Les résultats et la présentation du projet ont bien fascinés les deux anciens étudiants qui travaillaient sur le projet avant cette session.

La figure #19 prise de vue du Graphe tracé en Matlab, la figure # 20 la prise de vue du graphe tracé le logiciel JAVA) et finalement (la figure 21 qui est une prise de vue du graphe tracé le logiciel Visuel Basic.

On voit que ces trois graphiques concordent parfaitement à l'exception de l'échelle, car l'échelle de Matlab est automatique puis celle du graphique tracé par mon logiciel n'est pas modifiable pour le moment. Si ces deux graphiques concordent, le graphique que j'ai créé doit être correct, donc j'ai bien validé mes résultats.

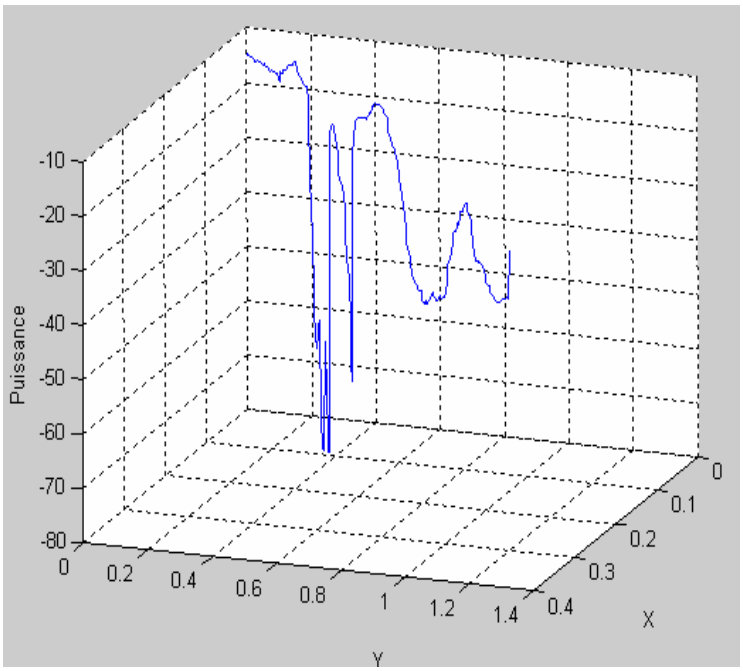

Fig. 19 Graphique tracé en Matlab avec les données de la machine

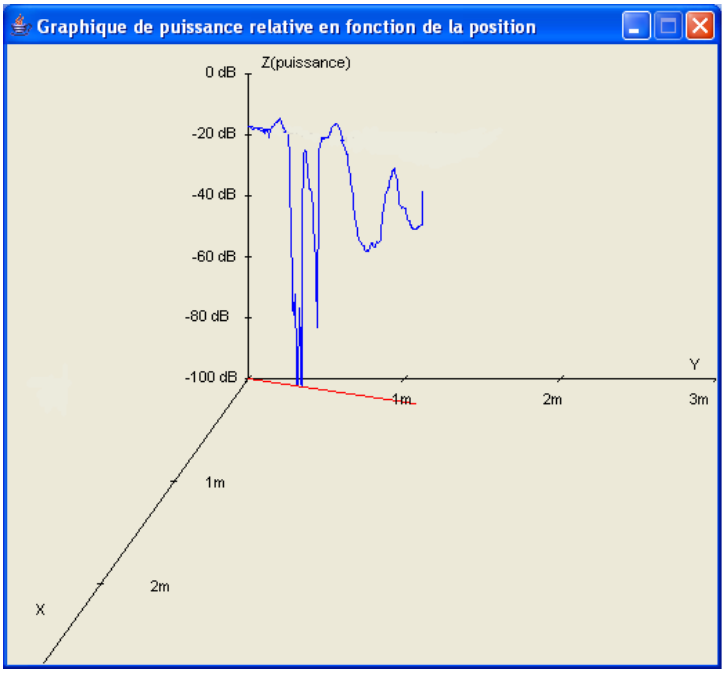

Fig. 20 Graphique tracé par le logiciel JAVA

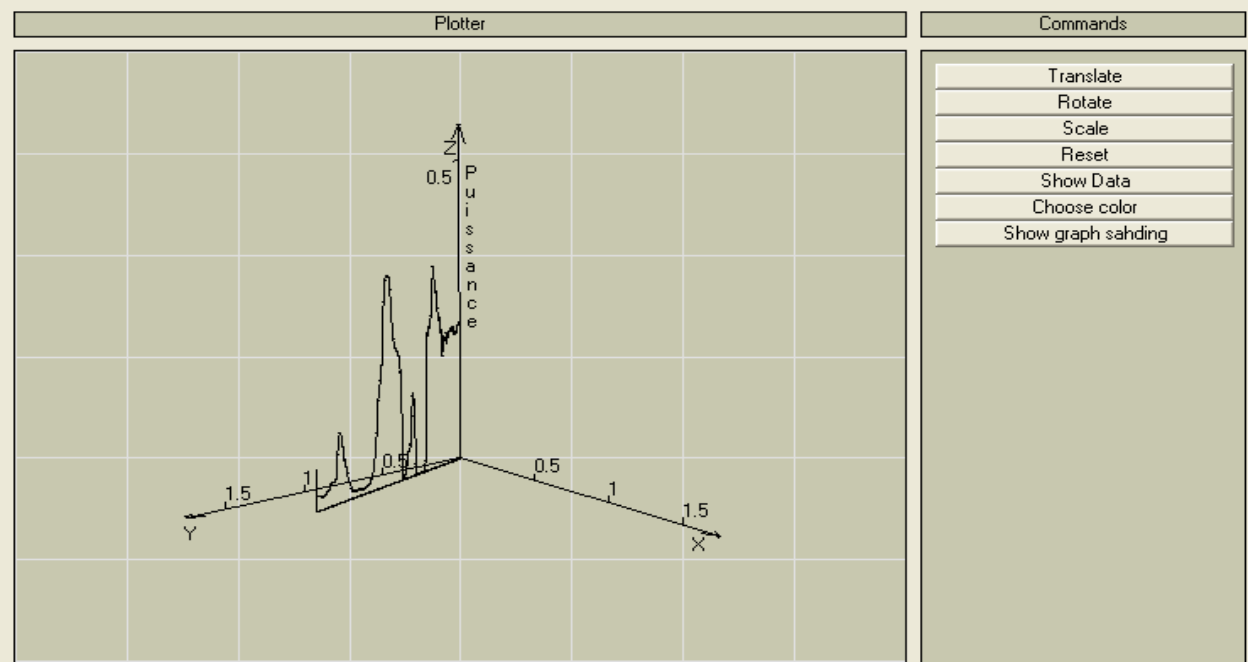

Fig. 21 Graphique 3D actuel par Visuel Basic

4. CONCLUSION ET RECOMMANDATIONS

4.1 Recommandations

Dans le but d'assurer la continuation de ce projet, la PIAM, je dois emmener des recommandations qui pourraient être entreprises par un ou plusieurs étudiants futurs dans le cadre de leur projet.

Actuellement l'application travaille sur des données existant sous formes de tableau Excel que j'ai convertis en format Access. Mais dans le future, vaut mieux que ça soit en temps real.

Ce qui n'est pas encor fait, l'application devra être intégrer de telle manière que le traçage soit déclencher, chaque fois que la machine se déplace et que l'application JAVA reçoit une triplet de données (x,y,P). Cette dernière soit assimilée par l'application Visuel Basic.

Mon majeur, est qui est vraiment triste est que j'ai jamais fait comment manipuler le matériel en utilisant l'application JAVA existante.

Si j'avait l'aide j'airai du prendre la liste chaîne de l'application existante et fur et à mesure que la machine bouge et les à chaque événement le traçage s'effectuera.

4.2 Discussion

Pour conclure, ce projet a été très enrichissant car j'ai appris beaucoup de choses mais aussi parce que les tâches que j'ai accomplies sont profitables. Par exemple, le graphique 3D avec interface et utile de control que j'ai fait facilite la visualisation des données lues ce qui rends la compréhension des résultats plus aisée et qui permet en conséquence d'économiser du temps lorsqu'on prends des mesures.

Ce projet m'a beaucoup aidé à comprendre les langages de programmation JAVA, et Visuel Basic. J'ai pu expérimenter avec chacun d'entre eux en accomplissant mes tâches primaires mais aussi en faisant de nombreux tests afin de mieux comprendre leur fonctionnement.

Cote personnel, ce projet était un grand défit d'organisation de ma vie, du fait que je travaillait dans deux jobs, plus le temps planifié et consacré pour la réalisation de ce projet. Du fait j'ai récolté une vaste expérience qui me sera utile tout au long de ma carrière en tant qu'informaticien. De plus, j'ai fait tout mon travail seul cet automne, ce qui m'a permis de me débrouiller par moi-même

Enfin, je juge que tout s'est déroulé comme il fallait car j'ai pu réaliser tous mes objectifs dans le temps alloués.

5. BIBLIOGRAPHIE

*** http://java.sun.com**

- Advance programing with JAVA.

- 3 D garphics and the animation

*** http://goforit.unk.edu/vb6/default.htm**

- Graphics and simple animation.

- Using graphics in Visual Basic

*** http://silkyroad.developpez.com/VBA/EvenementsGraphiques/**

- Utiliser les évènements dans les graphiques .

- Utile pour les menus dynamiques.

*** http://www.garybeene.com/3d/3d-math.htm (Nov 2006)**

- 3D graphics and math.

6. ANNEXE

6.1 Code Visuel Basic du graphique 3D)

```
Option Explicit

Public Sub GetTheData()
  Dim i As Double ' Local variable for looping purposes
  ' Global database connection object
  Set gCn = New ADODB.Connection

  ' Set ConnectionString for Connection Object.
  gCn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                  "Data Source = " & App.Path & "\data\db1.mdb"

  ' Open the database
  gCn.Open

  ' Set a global Recordset
  Set gRs = New ADODB.Recordset

  ' Open the table that contains the X, Y, Z coordinates
  gRs.Open "Test3D", gCn, adOpenKeyset, adLockOptimistic

  ' Copy data from the table into array of points and make room for the origin
  ' and for the X, Y, Z axis end points. Make room for the arrows at each
  ' positive end of the Axis. 2 points neede at each end to draw an arrow.
  ' So we added 10 entries :
  ' * 1 entry for the origin
  ' * 3 entries for the 3 positive ends of the axis
  ' * 6 entries for the three arrows at each positive end of the axis.
  ReDim Preserve a3DPoints(gRs.RecordCount + 49, 3)
  ReDim Preserve aXYPoints(gRs.RecordCount + 49, 3)

  i = 0
  While Not gRs.EOF
    i = i + 1
    a3DPoints(i, 1) = gRs.Fields(0) * X_Unit     ' This is the X coordinate
    a3DPoints(i, 2) = -1 * gRs.Fields(1) * Y_Unit ' This is the Y coordinate
    a3DPoints(i, 3) = gRs.Fields(2) * Z_Unit     ' This is the Z coordinate
```

```
    aXYPoints(i, 1) = gRs.Fields(0) * X_Unit      ' This is the X coordinate
    aXYPoints(i, 2) = -gRs.Fields(1) * Y_Unit     ' This is the Y coordinate
    aXYPoints(i, 3) = 0                           ' This is the Z coordinate

    gRs.MoveNext
   Wend

   gRs.Close
   gCn.Close
  End Sub

  Public Sub Append_Center_Ends(SW As Double)
   Dim cX As Double
   Dim cY As Double
   Dim cZ As Double

   ' Append an entry for the center
   indxO = UBound(a3DPoints) - 48
   Fill_Array indxO, 0, 0, 0

   ' Append an entry for the positive end point of the X-Axis
   indxX = UBound(a3DPoints) - 47
   cX = SW / 2 - dblIncrement
   Fill_Array indxX, cX, 0, 0
   ' Append 2 entries to draw an arrow at the positive end of the X-Axis
   Fill_Array indxX + 1, cX - 0.2, -0.05, 0
   Fill_Array indxX + 2, cX - 0.2, 0.05, 0
   Fill_Array indxX + 3, cX - 0.2, 0.2, 0 ' "X"

   Fill_Array indxX + 4, 0.5, 0, 0
   Fill_Array indxX + 5, 0.5, 0, 0.1
   Fill_Array indxX + 6, 0.5, 0, 0.2 ' "0.5"

   Fill_Array indxX + 7, 1, 0, 0
   Fill_Array indxX + 8, 1, 0, 0.1
   Fill_Array indxX + 9, 1, 0, 0.2 ' "1"

   Fill_Array indxX + 10, 1.5, 0, 0
   Fill_Array indxX + 11, 1.5, 0, 0.1
   Fill_Array indxX + 12, 1.5, 0, 0.2 ' "1.5"

   ' Append an entry for the positive end point of the Y-Axis
   indxY = UBound(a3DPoints) - 34
   cY = -SW / 2 + dblIncrement
   Fill_Array indxY, 0, cY, 0
   ' Append 2 entries to draw an arrow at the positive end of the Y-Axis
```

```
Fill_Array indxY + 1, -0.05, cY + 0.2, 0
Fill_Array indxY + 2, 0.05, cY + 0.2, 0
Fill_Array indxY + 3, 0.1, cY - 0.2, 0 ' "Y"

Fill_Array indxY + 4, 0, -0.5, 0
Fill_Array indxY + 5, 0, -0.5, 0.1
Fill_Array indxY + 6, 0, -0.5, 0.2

Fill_Array indxY + 7, 0, -1, 0
Fill_Array indxY + 8, 0, -1, 0.1
Fill_Array indxY + 9, 0, -1, 0.2

Fill_Array indxY + 10, 0, -1.5, 0
Fill_Array indxY + 11, 0, -1.5, 0.1
Fill_Array indxY + 12, 0, -1.5, 0.2

' Append an entry for the positive end point of the Z-Axis
indxZ = UBound(a3DPoints) - 21
cZ = SW / 2 - 2 'cX
Fill_Array indxZ, 0, 0, cZ
' Append 2 entries to draw an arrow at the positive end of the X-Axis
Fill_Array indxZ + 1, -0.1, 0, cZ - 0.2
Fill_Array indxZ + 2, 0.1, 0, cZ - 0.2
Fill_Array indxZ + 3, -0.2, 0, cZ + 0.1 ' "Z"

Fill_Array indxZ + 4, 0, 0, 0.5
Fill_Array indxZ + 5, 0, -0.1, 0.5
Fill_Array indxZ + 6, 0, -0.2, 0.5

Fill_Array indxZ + 7, 0, 0, 1
Fill_Array indxZ + 8, 0, -0.1, 1
Fill_Array indxZ + 9, 0, -0.2, 1

Fill_Array indxZ + 10, 0, 0, 1.5
Fill_Array indxZ + 11, 0, -0.1, 1.5
Fill_Array indxZ + 12, 0, -0.2, 1.5

Fill_Array indxZ + 13, 0.1, 0, cZ - 0.2 ' "P"
Fill_Array indxZ + 14, 0.1, 0, cZ - 0.3 ' "u"
Fill_Array indxZ + 15, 0.1, 0, cZ - 0.4 ' "i"
Fill_Array indxZ + 16, 0.1, 0, cZ - 0.5 ' "s"
Fill_Array indxZ + 17, 0.1, 0, cZ - 0.6 ' "s"
Fill_Array indxZ + 18, 0.1, 0, cZ - 0.7 ' "a"
Fill_Array indxZ + 19, 0.1, 0, cZ - 0.8 ' "n"
Fill_Array indxZ + 20, 0.1, 0, cZ - 0.9 ' "c"
Fill_Array indxZ + 21, 0.1, 0, cZ - 1 ' "e"
```

```
End Sub

Private Sub Fill_Array(indx As Double, X As Double, Y As Double, Z As Double)
  a3DPoints(indx, 1) = X
  a3DPoints(indx, 2) = Y
  a3DPoints(indx, 3) = Z

  aXYPoints(indx, 1) = 0
  aXYPoints(indx, 2) = 0
  aXYPoints(indx, 3) = 0
End Sub
```

```
Option Explicit

' Declare variables for the database
Global gCn As ADODB.Connection
Global gRs As ADODB.Recordset

Global X_Unit As Double
Global Y_Unit As Double
Global Z_Unit As Double
Global dblIncrement As Double

' Pi
Global pi As Double

' The array of 3D points
Global a3DPoints() As Double

' This array serve as a projection of points in the XY Plane
Global aXYPoints() As Double

' The array of X, Y, Z Axis
Global a3DAxis(4, 3) As Double

' Inexes of the origin and of the XYZ-Axis
Global indxO As Double
Global indxX As Double
Global indxY As Double
Global indxZ As Double

' State array to save the state of the scroll bars
Type typState
```

```
  Scrl As Double
  oldScrl As Double
End Type

Global aTState(2) As typState
Global aRState(2) As typState
Global aSState(2) As typState

' This variable is to take care of the hscrolll_change
' event. Changing the value of the hscroll bar will triger
' the Hscroll_Change event handler. At this point of
' the program flow, we don't want that to happen.
' We want that to happen only when user clicks on the
' Hscroll bar control.
Global CalledFromProgram As Boolean
Global Loading As Boolean
Global FirstTime As Boolean

' Array for debugging and tracing
Global aDebug() As String
Global aDebugIndex As Double
```

```
Option Explicit

Dim strCmdTRS As String
Dim strScrlTRS As String
Dim strTRS As String

Dim TempColor As Long
Dim clrRed As Long
Dim clrGreen As Long
Dim clrBlue As Long

Dim ShowLines As Boolean

Dim aPuissance(9)

Private Sub Form_Initialize()
  aPuissance(1) = "P"
  aPuissance(2) = "u"
  aPuissance(3) = "i"
  aPuissance(4) = "s"
  aPuissance(5) = "s"
  aPuissance(6) = "a"
  aPuissance(7) = "n"
  aPuissance(8) = "c"
```

```
  aPuissance(9) = "e"

  DebugInfo "Form_Initialize()"
  Debug.Print "Form_Initialize()"
  aDebugIndex = 0
  Loading = True
  FirstTime = True
  DoInitializations
  Loading = False
End Sub

Private Sub Form_Resize()
  DebugInfo "Form_Resize()"
  Debug.Print "Form_Resize()"
  If Not Loading Then
    If Not FirstTime Then
      Debug.Print "Form_Resize() -- True"
      DrawTheGrid
    End If
  End If
End Sub

Private Sub Form_Activate()
  DebugInfo "Form_Activate()"
  Debug.Print "Form_Activate()"
  If Not Loading Then
    If Not FirstTime Then
      Debug.Print "Form_Activate() -- True"
      DrawTheGrid
    End If
  End If
End Sub

Private Sub Form_Paint()
  DebugInfo "Form_Paint()"
  Debug.Print "Form_Paint()"
  If Not Loading Then
    Debug.Print "Form_Paint() -- True"
    DrawTheGrid
    FirstTime = False
  End If
End Sub

Private Sub DebugInfo(s As String)
  aDebugIndex = aDebugIndex + 1
  ReDim Preserve aDebug(aDebugIndex)
```

```
  aDebug(aDebugIndex) = s
End Sub

Private Sub DoInitializations()
  Dim i As Integer

  TempColor = 8454143 '14671839
  clrRed = TempColor Mod &H100
  clrGreen = (TempColor \ &H100) Mod &H100
  clrBlue = (TempColor \ &H10000) Mod &H100
  ShowLines = True

  For i = 0 To cmdTRS.UBound
    cmdTRS(i).Enabled = False
  Next i


  With pic3DGraph
    .ScaleWidth = 8
    .ScaleHeight = 8
    .ScaleLeft = -4
    .ScaleTop = -4
  End With

  X_Unit = 2
  Y_Unit = 2
  Z_Unit = 2
  dblIncrement = 1.5

  GetTheData
  Append_Center_Ends pic3DGraph.ScaleWidth

  PositionTheGraph

  ' Arrage the controls on the form
  Arrange_Form
  Load_Commands
  Arrange_TRS

  ResetScrolls
End Sub

Private Sub PositionTheGraph()
  Dim i As Integer
  ResetScrolls
```

```
  For i = 1 To 50
    subTRS 0, i, 50, "RX"
  Next i
  For i = 1 To 28
    subTRS 1, i, 50, "RY"
  Next i
  For i = 1 To 10
    subTRS 2, i, 50, "RZ"
  Next i

  For i = 1 To 14
    subTRS 0, i, 50, "SX"
  Next i
  For i = 1 To 13
    subTRS 1, i, 50, "SY"
  Next i

End Sub

Private Sub ResetScrolls()
  Dim i As Integer
  For i = 0 To hscrlXYZ.UBound
     aTState(i).Scrl = 0
     aTState(i).oldScrl = 0
     aRState(i).Scrl = 0
     aRState(i).oldScrl = 0
     aSState(i).Scrl = 0
     aSState(i).oldScrl = 0
  Next i

End Sub

Private Sub Arrange_Form()
  With lblPlotter
    .Left = 120
    .Top = 120
    .Width = 9015 '6015
    .Height = 255
    .BackColor = RGB(200, 200, 175) 'RGB(200, 200, 200)
  End With

  With lblCommands
    .Left = lblPlotter.Left + lblPlotter.Width + 120
    .Top = lblPlotter.Top
    .Width = 2895
    .Height = lblPlotter.Height
```

```
    .BackColor = lblPlotter.BackColor 'RGB(200, 200, 200)
  End With

  With pic3DGraph
    .Left = lblPlotter.Left
    .Top = lblPlotter.Top + lblPlotter.Height + 120
    .BackColor = lblPlotter.BackColor 'RGB(200, 200, 200)
  End With

  With picCommands
    .Left = lblCommands.Left
    .Top = lblCommands.Top + lblCommands.Height + 120
    .Height = pic3DGraph.Height
    .BackColor = lblPlotter.BackColor 'RGB(200, 200, 200)
  End With

  With Me
    .Width = 120 + lblPlotter.Width + 120 + lblCommands.Width + 240
    .Height = 120 + lblPlotter.Height + 120 + pic3DGraph.Height + 120 + 360 + 120
    .Left = (Screen.Width - .Width) / 2
    .Top = (Screen.Height - .Height) / 2
  End With
End Sub

Private Sub Load_Commands()
  Dim i As Integer

  With cmdTRS(0)
    .Caption = "Translate"
    .Left = 120
    .Top = 120
  End With

  For i = 1 To cmdTRS.UBound
    Unload cmdTRS(i)
  Next i

  For i = 1 To 6
    Load cmdTRS(i)
    With cmdTRS(i)
      .Left = cmdTRS(i - 1).Left
      .Top = cmdTRS(i - 1).Top + cmdTRS(i - 1).Height
      Select Case i
        Case 1: .Caption = "Rotate"
        Case 2: .Caption = "Scale"
        Case 3: .Caption = "Reset"
```

```
        Case 4: .Caption = "Show Data"
        Case 5: .Caption = "Choose color"
        Case 6: .Caption = "Hide graph shading"
      End Select
      .ZOrder 0
      .Visible = True
    End With
  Next i
End Sub

Private Sub Arrange_TRS()
  With picTRS
    .Left = cmdTRS(0).Left
    .Top = cmdTRS(0).Top + cmdTRS(0).Height
    .Width = cmdTRS(0).Width
    .ZOrder 0
    .BackColor = RGB(200, 200, 200)
  End With

  Load_Labels
  Load_Scrolls

  picTRS.Height = 120 + 3 * lblXYZ(0).Height + 3 * 120 + 120
  picTRS.Visible = False
End Sub

Private Sub Load_Labels()
  Dim i As Integer

  With lblXYZ(0)
    .Left = 120
    .Top = 120
    .AutoSize = True
    .Caption = "X"
    .Visible = True
  End With

  For i = 1 To lblXYZ.UBound
    Unload lblXYZ(i)
  Next i

  For i = 1 To 2
    Load lblXYZ(i)
    With lblXYZ(i)
      .Left = lblXYZ(i - 1).Left
      .Top = lblXYZ(i - 1).Top + lblXYZ(i - 1).Height + 120
```

```
      .AutoSize = True
      Select Case i
        Case 1: .Caption = "Y"
        Case 2: .Caption = "Z"
      End Select
      .Visible = True
    End With
  Next i
End Sub

Private Sub Load_Scrolls()
  Dim i As Integer

  With hscrlXYZ(0)
    .Left = lblXYZ(0).Left + lblXYZ(0).Width + 120
    .Top = lblXYZ(0).Top
  End With

  For i = 1 To hscrlXYZ.UBound
    Unload hscrlXYZ(i)
  Next i

  For i = 1 To 2
    Load hscrlXYZ(i)
    With hscrlXYZ(i)
      .Left = hscrlXYZ(i - 1).Left
      .Top = lblXYZ(i).Top
      .Visible = True
    End With
  Next i

  ' Initialize the state of each scroll
  For i = 0 To hscrlXYZ.UBound - 1
    aTState(i).Scrl = 0
    aTState(i).oldScrl = 0

    aRState(i).Scrl = 0
    aRState(i).oldScrl = 0

    aSState(i).Scrl = 0
    aSState(i).oldScrl = 0
  Next i

End Sub

Private Sub DrawTheGrid()
```

```
Dim i As Double
Dim j As Double

With pic3DGraph
  .Cls
  .FillStyle = 0

  .BackColor = lblPlotter.BackColor 'RGB(200, 200, 200)
  .ForeColor = vbBlack

  ' Draw the lines of the grid
  .ForeColor = &HE0E0E0
  For i = .ScaleLeft To .ScaleWidth / 2 Step 0.5
    pic3DGraph.Line (.ScaleLeft, i)-(-.ScaleLeft, i)
    pic3DGraph.Line (i, .ScaleHeight)-(i, -.ScaleHeight)
  Next i

  .ForeColor = vbBlack
  ' Draw the Origine point
  pic3DGraph.Circle (a3DPoints(indxO, 1), a3DPoints(indxO, 2)), 0.02
  ' Draw the axis
  DrawTheAxis indxO, indxX, 0.02, 0.2, 0.01, "X"
  DrawTheXYZUnits indxX, 4, "0.5"
  DrawTheXYZUnits indxX, 7, "1"
  DrawTheXYZUnits indxX, 10, "1.5"

  DrawTheAxis indxO, indxY, 0.02, 0.03, 0.01, "Y"
  DrawTheXYZUnits indxY, 4, "0.5"
  DrawTheXYZUnits indxY, 7, "1"
  DrawTheXYZUnits indxY, 10, "1.5"

  DrawTheAxis indxO, indxZ, 0.02, 0.2, 0.01, "Z"
  DrawTheXYZUnits indxZ, 4, "0.5"
  DrawTheXYZUnits indxZ, 7, "1"
  DrawTheXYZUnits indxZ, 10, "1.5"

  FillArrayPuissance indxZ, 13
End With

' Shade the graph
If Loading Then
  Timer1.Enabled = True
Else
  If FirstTime Then
    Timer1.Enabled = True
  Else
```

```
      DrawAllPoints
    End If
  End If
  Write_Author
End Sub

Private Sub DrawTheXYZUnits(indx, i, s)
  pic3DGraph.Line (a3DPoints(indx + i, 1), a3DPoints(indx + i, 2))- _
              (a3DPoints(indx + i + 1, 1), a3DPoints(indx + i + 1, 2))
  pic3DGraph.CurrentX = a3DPoints(indx + i + 2, 1)
  pic3DGraph.CurrentY = a3DPoints(indx + i + 2, 2)
  pic3DGraph.Print s
End Sub

Private Sub FillArrayPuissance(indx, i)
  Dim j As Integer
  For j = 1 To UBound(aPuissance)
    pic3DGraph.CurrentX = a3DPoints(indx + i, 1)
    pic3DGraph.CurrentY = a3DPoints(indx + i, 2)
    pic3DGraph.Print aPuissance(i - 12)
    i = i + 1
  Next j
End Sub

Private Sub DrawTheAxis(indxO, indx, rRadius, cX, cY, s As String)
  ' Draw the X-Axis
  pic3DGraph.Line (a3DPoints(indxO, 1), a3DPoints(indxO, 2))- _
              (a3DPoints(indx, 1), a3DPoints(indx, 2))
  pic3DGraph.Circle (a3DPoints(indx, 1), a3DPoints(indx, 2)), rRadius
  ' Draw an arrow at the positive end of the X-Axis
  pic3DGraph.Line (a3DPoints(indx, 1), a3DPoints(indx, 2))- _
              (a3DPoints(indx + 1, 1), a3DPoints(indx + 1, 2))
  pic3DGraph.Line (a3DPoints(indx, 1), a3DPoints(indx, 2))- _
              (a3DPoints(indx + 2, 1), a3DPoints(indx + 2, 2))
  pic3DGraph.CurrentX = a3DPoints(indx + 3, 1) ' - cX
  pic3DGraph.CurrentY = a3DPoints(indx + 3, 2) '+ cY
  pic3DGraph.Print s
End Sub

Private Sub DrawOnePoint(i As Double)
  'pic3DGraph.ForeColor = RGB(Abs(256 - i), Abs(256 - i), Abs(256 - i))
  If i <= 128 Then
    If clrRed - 1 < 0 Then clrRed = clrRed + 1 Else clrRed = clrRed - 1
    If clrGreen - 1 < 0 Then clrGreen = clrGreen + 1 Else clrGreen = clrGreen - 1
    If clrBlue - 1 < 0 Then clrBlue = clrBlue + 1 Else clrBlue = clrBlue - 1
  Else
```

```
    clrRed = clrRed + 1: clrGreen = clrGreen + 1: clrBlue = clrBlue + 1
  End If

  If ShowLines Then
    pic3DGraph.ForeColor = RGB(clrRed, clrGreen, clrBlue)
    pic3DGraph.Line (a3DPoints(i, 1), a3DPoints(i, 2))- _
                (aXYPoints(i, 1), aXYPoints(i, 2))
  End If

  pic3DGraph.ForeColor = vbBlack
  pic3DGraph.Circle (a3DPoints(i, 1), a3DPoints(i, 2)), 0.001
  pic3DGraph.Circle (aXYPoints(i, 1), aXYPoints(i, 2)), 0.001

  pic3DGraph.Line (a3DPoints(i, 1), a3DPoints(i, 2))- _
              (a3DPoints(i + 1, 1), a3DPoints(i + 1, 2))
  pic3DGraph.Line (aXYPoints(i, 1), aXYPoints(i, 2))- _
              (aXYPoints(i + 1, 1), aXYPoints(i + 1, 2))
End Sub

Private Sub DrawAllPoints()
  Dim i As Double

  For i = 1 To UBound(a3DPoints) - 49
    'pic3DGraph.ForeColor = RGB(Abs(256 - i), Abs(256 - i), Abs(256 - i))
    If i <= 128 Then
      If clrRed - 1 < 0 Then clrRed = clrRed + 1 Else clrRed = clrRed - 1
      If clrGreen - 1 < 0 Then clrGreen = clrGreen + 1 Else clrGreen = clrGreen - 1
      If clrBlue - 1 < 0 Then clrBlue = clrBlue + 1 Else clrBlue = clrBlue - 1
    Else
      clrRed = clrRed + 1: clrGreen = clrGreen + 1: clrBlue = clrBlue + 1
    End If
    If ShowLines Then
      pic3DGraph.ForeColor = RGB(clrRed, clrGreen, clrBlue)
      pic3DGraph.Line (a3DPoints(i, 1), a3DPoints(i, 2))- _
                  (aXYPoints(i, 1), aXYPoints(i, 2))
    End If
  Next i


  pic3DGraph.ForeColor = vbBlack
  ' Loop through all the 3D points and draw them
  For i = 1 To UBound(a3DPoints) - 49
    pic3DGraph.Circle (a3DPoints(i, 1), a3DPoints(i, 2)), 0.001
    pic3DGraph.Circle (aXYPoints(i, 1), aXYPoints(i, 2)), 0.001
  Next i
```

```
  ' Connect the points
  For i = 1 To UBound(a3DPoints) - 50
    pic3DGraph.Line (a3DPoints(i, 1), a3DPoints(i, 2))- _
                (a3DPoints(i + 1, 1), a3DPoints(i + 1, 2))
    pic3DGraph.Line (aXYPoints(i, 1), aXYPoints(i, 2))- _
                (aXYPoints(i + 1, 1), aXYPoints(i + 1, 2))
  Next i

  For i = 0 To cmdTRS.UBound
    cmdTRS(i).Enabled = True
  Next i
End Sub

Private Sub Write_Author()
  ' Print Author name
  lblAuthor.Left = -3.5
  lblAuthor.Top = 3.5
  lblAuthor.Visible = True
  'With pic3DGraph
  '  .ForeColor = RGB(1, 1, 1)
  '  .FontName = "Times new roman"
  '  .FontSize = 12
  '  .FontBold = True
  '  .FontTransparent = True
  '  .Font = 5
  '  .CurrentX = -3.5
  '  .CurrentY = 3.5
  '  pic3DGraph.Print "Programmed By: Youssef Jaber"
  'End With
End Sub

Private Sub cmdTRS_Click(Index As Integer)
  Dim i As Integer ' Local variable for looping purposes

  ' Program flow will take ownership of execution of the Hscroll_Change
  CalledFromProgram = True
  pic3DGraph.SetFocus

  Select Case Index                              ' HscrollBar          ' Array
    Case 0: For i = 0 To hscrlXYZ.UBound          '================      =========
              hscrlXYZ(i).Value = aTState(i).Scrl    '|<|   ||    |>| <-----|-  |   |
            Next i                              '================      |   |   |
            strCmdTRS = "T"                        '                   |-------|
    Case 1: For i = 0 To hscrlXYZ.UBound          '================      |   |   |
              hscrlXYZ(i).Value = aRState(i).Scrl    '|<|   ||    |>| <-----|-  |   |
            Next i                              '================      |   |   |
```

```
        strCmdTRS = "R"                    '                  |-------|
    Case 2: For i = 0 To hscrlXYZ.UBound          '================      |  |  |
        hscrlXYZ(i).Value = aSState(i).Scrl    '|<|  ||    |>| <-----|-  |  |
        Next i                           '================     =========
        strCmdTRS = "S"                        '                  ^   ^
                                        '                  |  |
                                        '                Value  |
                                        '                   Old
                                        '                   Value
    Case 3: pic3DGraph.Cls
        DoInitializations
        GoTo Exitsub
    Case 4: Load frmData: frmData.Show: Exit Sub
    Case 5: CommonDialog1.ShowColor
        TempColor = CommonDialog1.Color
        clrRed = TempColor Mod &H100
        clrGreen = (TempColor \ &H100) Mod &H100
        clrBlue = (TempColor \ &H10000) Mod &H100
        GoTo Exitsub
    Case 6:
        If ShowLines Then
          cmdTRS(Index).Caption = "Show graph sahding"
          ShowLines = False
        Else
          cmdTRS(Index).Caption = "Hide graph sahding"
          ShowLines = True
        End If
        GoTo Exitsub
  End Select

  ' Release ownership of execution of the Hscroll_Change to the user
  CalledFromProgram = False

  Resize_Commands_pic Index
  Exit Sub
Exitsub:
  pic3DGraph.Cls
  For i = 0 To cmdTRS.UBound
    cmdTRS(i).Enabled = False
  Next i

  Loading = True: FirstTime = True
  Timer1.Enabled = True
  DrawTheGrid
  Loading = False: FirstTime = False
End Sub
```

```
Private Sub Resize_Commands_pic(indx As Integer)
  Dim i As Integer

  For i = 0 To cmdTRS.UBound
    With cmdTRS(i)
      .Left = 120
      .Top = 120 + i * cmdTRS(i).Height
    End With
  Next i

  With picTRS
    .Left = cmdTRS(indx).Left
    .Top = cmdTRS(indx).Top + cmdTRS(indx).Height
    .Visible = True
  End With

  For i = indx + 1 To cmdTRS.UBound
    With cmdTRS(i)
      .Left = 120
      .Top = picTRS.Top + picTRS.Height + (i - (indx + 1)) * .Height
    End With
  Next i
  hscrlXYZ(0).SetFocus
End Sub

Private Sub hscrlXYZ_Change(Index As Integer)
  If Not CalledFromProgram Then
    Select Case Index
      Case 0: strScrlTRS = "X"
      Case 1: strScrlTRS = "Y"
      Case 2: strScrlTRS = "Z"
    End Select
    strTRS = strCmdTRS & strScrlTRS
    subTRS Index, hscrlXYZ(Index).Value, hscrlXYZ(Index).Max, strTRS
    DrawTheGrid
    'Timer1.Enabled = True
  End If
End Sub

Private Sub subTRS(indx As Integer, v, m, strTRS As String)
  'Static oldx As Double
  'Static oldY As Double
  'Static oldZ As Double

  Dim Scrl As Double
```

```
Dim scrlX As Double
Dim scrlY As Double
Dim scrlZ As Double

Scrl = v / (m / 2)
'lblStatus.Caption = v & ", " & Scrl
Select Case strTRS
  Case "TX": aTState(indx).Scrl = v
             scrlX = Scrl - aTState(indx).oldScrl
             TranslateXYZ scrlX, 0, 0
             aTState(indx).oldScrl = Scrl
  Case "TY": aTState(indx).Scrl = v
             scrlY = Scrl - aTState(indx).oldScrl
             TranslateXYZ 0, scrlY, 0
             aTState(indx).oldScrl = Scrl
  Case "TZ": aTState(indx).Scrl = v
             scrlZ = Scrl - aTState(indx).oldScrl
             TranslateXYZ 0, 0, scrlZ
             aTState(indx).oldScrl = Scrl

  Case "RX": aRState(indx).Scrl = v
             scrlX = Scrl - aRState(indx).oldScrl
             RotateXYZ a3DPoints, scrlX, 0, 0
             aRState(indx).oldScrl = Scrl
  Case "RY": aRState(indx).Scrl = v
             scrlY = Scrl - aRState(indx).oldScrl
             RotateXYZ a3DPoints, 0, scrlY, 0
             aRState(indx).oldScrl = Scrl
  Case "RZ": aRState(indx).Scrl = v
             scrlZ = Scrl - aRState(indx).oldScrl
             RotateXYZ a3DPoints, 0, 0, scrlZ
             aRState(indx).oldScrl = Scrl

  Case "SX": aSState(indx).Scrl = v
             scrlX = Scrl - aSState(indx).oldScrl
             If scrlX < 0 Then
               ScaleXYZ 1 / (1 - scrlX), 1, 1
             Else
               ScaleXYZ 1 + scrlX, 1, 1
             End If
             aSState(indx).oldScrl = Scrl
  Case "SY": aSState(indx).Scrl = v
             scrlY = Scrl - aSState(indx).oldScrl
             If scrlY < 0 Then
               ScaleXYZ 1, 1 / (1 - scrlY), 1
             Else
```

```
            ScaleXYZ 1, 1 + scrlY, 1
          End If
          aSState(indx).oldScrl = Scrl
    Case "SZ": aSState(indx).Scrl = v
          scrlZ = Scrl - aSState(indx).oldScrl
          If scrlZ < 0 Then
            ScaleXYZ 1, 1, 1 / (1 - scrlZ)
          Else
            ScaleXYZ 1, 1, 1 + scrlZ
          End If
          aSState(indx).oldScrl = Scrl
  End Select
End Sub

Private Sub TranslateXYZ(X As Double, Y As Double, Z As Double)
  Dim i As Double

  For i = 1 To UBound(a3DPoints)
    a3DPoints(i, 1) = a3DPoints(i, 1) + X
    a3DPoints(i, 2) = a3DPoints(i, 2) + Y
    a3DPoints(i, 3) = a3DPoints(i, 3) + Z

    aXYPoints(i, 1) = aXYPoints(i, 1) + X
    aXYPoints(i, 2) = aXYPoints(i, 2) + Y
    aXYPoints(i, 3) = aXYPoints(i, 3)
  Next i
End Sub

Private Sub RotateXYZ(a() As Double, X As Double, Y As Double, Z As Double)
  Dim MatrixX(3, 3) As Double
  Dim MatrixY(3, 3) As Double
  Dim MatrixZ(3, 3) As Double

  Dim nSum As Double
  Dim nSum1 As Double

  Dim i As Double
  Dim j As Double
  Dim k As Double

  ' create rotation matrix for x-axis
  MatrixX(1, 1) = 1: MatrixX(2, 1) = 0:      MatrixX(3, 1) = 0
  MatrixX(1, 2) = 0: MatrixX(2, 2) = Cos(X): MatrixX(3, 2) = -Sin(X)
  MatrixX(1, 3) = 0: MatrixX(2, 3) = Sin(X): MatrixX(3, 3) = Cos(X)

  ' create rotation matrix for y-axis
```

```
  MatrixY(1, 1) = Cos(Y):  MatrixY(2, 1) = 0: MatrixY(3, 1) = Sin(Y)
  MatrixY(1, 2) = 0:       MatrixY(2, 2) = 1: MatrixY(3, 2) = 0
  MatrixY(1, 3) = -Sin(Y): MatrixY(2, 3) = 0: MatrixY(3, 3) = Cos(Y)

  ' create rotation matrix for z-axis
  MatrixZ(1, 1) = Cos(Z): MatrixZ(2, 1) = -Sin(Z): MatrixZ(3, 1) = 0
  MatrixZ(1, 2) = Sin(Z): MatrixZ(2, 2) = Cos(Z):  MatrixZ(3, 2) = 0
  MatrixZ(1, 3) = 0:      MatrixZ(2, 3) = 0:       MatrixZ(3, 3) = 1

  MatrixMultiply a, MatrixX, X, Y, Z
  MatrixMultiply a, MatrixY, X, Y, Z
  MatrixMultiply a, MatrixZ, X, Y, Z

  'DrawTheGrid
End Sub

Private Sub ScaleXYZ(X As Double, Y As Double, Z As Double)
  Dim MatrixX(3, 3) As Double

  Dim nSum As Double

  Dim i As Double
  Dim j As Double
  Dim k As Double

  ' create rotation matrix for x-axis
  MatrixX(1, 1) = X: MatrixX(2, 1) = 0: MatrixX(3, 1) = 0
  MatrixX(1, 2) = 0: MatrixX(2, 2) = Y: MatrixX(3, 2) = 0
  MatrixX(1, 3) = 0: MatrixX(2, 3) = 0: MatrixX(3, 3) = Z

  MatrixMultiply a3DPoints, MatrixX, X, Y, Z
End Sub

Private Sub MatrixMultiply(a() As Double, _
                           b() As Double, _
                           X As Double, _
                           Y As Double, _
                           Z As Double)
  Dim i As Double
  Dim j As Double
  Dim k As Double
  Dim nSum As Double
  Dim nSum1 As Double

  nSum = 0
  nSum1 = 0
```

```
  For i = 1 To UBound(a, 1)
    For j = 1 To UBound(b, 2)
      For k = 1 To UBound(b, 1)
        nSum = nSum + a(i, k) * b(k, j)
        nSum1 = nSum1 + aXYPoints(i, k) * b(k, j)
      Next k
      a(i, j) = nSum
      aXYPoints(i, j) = nSum1
      nSum = 0
      nSum1 = 0
    Next j
  Next i
End Sub

Private Sub cmdReset_Click()
  DoInitializations
End Sub

Private Sub mnuFileExit_Click()
  End
End Sub

Private Sub Timer1_Timer()
  Static i As Double
  Dim j As Integer
  If i < UBound(a3DPoints) - 49 Then
    DrawOnePoint i
  Else
    Timer1.Enabled = False
    i = 0
    For j = 0 To cmdTRS.UBound
      cmdTRS(j).Enabled = True
    Next j
  End If
  i = i + 1
End Sub
```

```
Private Sub Form_Load()
  Dim itmX As ListItem

  With Me
    .Left = (Screen.Width - .Width) / 2
    .Top = (Screen.Height - .Height) / 2
  End With

  ListView1.ColumnHeaders.Add , , "Position X", ListView1.Width / 3
  ListView1.ColumnHeaders.Add , , "Position Y", ListView1.Width / 3
```

```
  ListView1.ColumnHeaders.Add , , "Position Z", ListView1.Width / 3
  ListView1.View = lvwReport
  For i = 1 To UBound(a3DPoints)
    Set itmX = ListView1.ListItems.Add(i, , a3DPoints(i, 1))
    itmX.SubItems(1) = a3DPoints(i, 2)
    itmX.SubItems(2) = a3DPoints(i, 3)
  Next
End Sub
```

```
Private Sub Form_Load()
  Dim itmX As ListItem

  With Me
    .Left = (Screen.Width - .Width) / 2
    .Top = (Screen.Height - .Height) / 2
  End With

  ListView1.ColumnHeaders.Add , , "Position X", ListView1.Width / 3
  ListView1.ColumnHeaders.Add , , "Position Y", ListView1.Width / 3
  ListView1.ColumnHeaders.Add , , "Position Z", ListView1.Width / 3
  ListView1.View = lvwReport
  For i = 1 To UBound(a3DPoints)
    Set itmX = ListView1.ListItems.Add(i, , a3DPoints(i, 1))
    itmX.SubItems(1) = a3DPoints(i, 2)
    itmX.SubItems(2) = a3DPoints(i, 3)
  Next
End Sub
```